Adding automatic on-off router control to the E3/4 or Evo3/4

(updated 12-24-2020)

It's not terribly difficult to reach up and turn on my Bobscnc router before starting a program, but I've always thought it would be nice if the router would turn off automatically when the program is complete, in case I don't happen to be standing right there at the time. Then my son added automatic on-off functionality to his E3 based on information he found in the FB group and online, so when he later came for a visit I asked him to help me do the same to my E4 (and I have since done the same to my Evo4). It took us only about 20 minutes to add the feature, much of that was because I didn't have all of the materials ready in advance, and it works like a charm.

Because we weren't able to locate a single reference document on the procedure I thought I take a few minutes to document the process for those who might want to make use of it later.

The relay we used is the same IoT Relay that several others have mentioned in the FB group. For those who are comfortable with controls terminology, the IoT Relay is a DPST interposing relay that has been integrated into a very convenient "black box" package with the following features:

- ⇒ a main power switch,
- ⇒ two normally open (NO) outlets (which the IoT calls "normally OFF"),
- ⇒ one normally closed (NC) outlet (which the IoT calls "normally ON"),
- ⇒ one "always ON" outlet (i.e., "always" means whenever the main power switch is on), and
- ⇒ a control input that accepts a very wide range of input signals, 3-48VDC or 12-120VAC,

… all for less than $30. In short, you could probably save a couple of bucks by buying all of the components separately and assembling the package yourself, but I can't imagine why you would want to (unless perhaps electronics is a hobby of yours!)

Here's how it works:

a. Connect the relay to the E3/4 or Evo3/4 controller, and plug the router power cord into the relay.

b. At the beginning of the program, our gcode needs to include the "M3" command, which tells the controller to energize the the pins to which we connected the relay.

c. When the IoT senses that signal, it closes (i.e., turns on) the NO outlets and opens (i.e., turn off) the NC outlet.

d. The router (with its switch already turned on) is plugged into one of the NO outlets, so it starts running.

e. At the end of the program, our gcode needs to include the "M5" command, which tells the controller to de-energize the applicable pins.

f. In the absence of the control signal the IoT Relay returns to its initial state, and our router stops turning.

Simple huh? Okay, I realize that I'm not the best at explaining this sort of stuff to people, so let me walk through the process that I used, step by step.

Materials:

- IoT Relay (https://www.amazon.com/gp/product/B00WV7GMA2)
- Breadboard jumper wires similar to these that my son bought (https://www.amazon.com//dp/B07GD2BWPY). There are undoubtedly other options available, and I only used 4 of the 120 wires, but they worked great and made it very easy.
- Extension wire. This will be the wire that will be strung from the breadboard jumper wires to the IoT Relay. I recommend something flexible that has stranded copper wire, because it will be doing a lot of bending as the gantry moves. It doesn't need to be very large gage wire (but the jumper wires are 28 awg, so don't use anything smaller than that), and it needs to be long enough to comfortably reach the relay.
- Butt splices or other method to connect the breadboard jumper wires to the extension wire.
- Wire ties.

Caution: After adding this feature, the router will start and stop automatically. Use extra caution when working near the router … e.g., changing bits, securing the work piece, etc.!

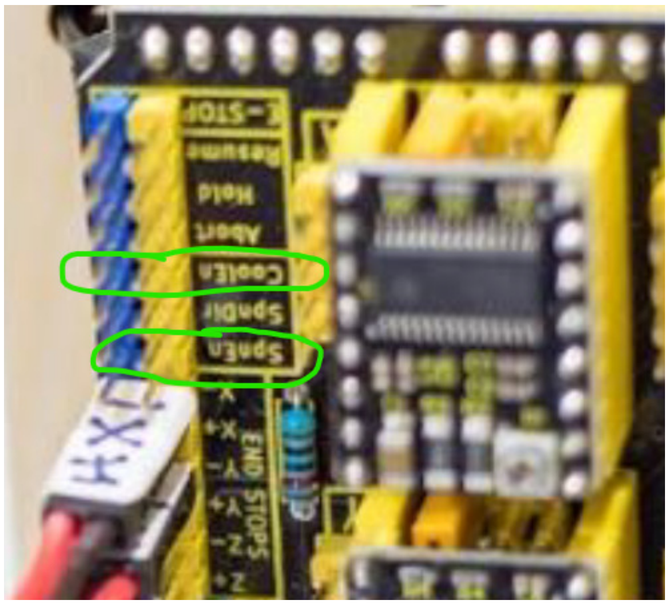
Once you've gathered up all the stuff you need, here are the installation steps:

1. Select 2 pairs of breadboard jumper wires, each with the same two colors for consistency (I chose purple and gray), and each with a female connector on one end and a male connector on the other.
2. Attach one pair of jumper wires to the two pins on the controller as follows:

   E3/4 (stock) —> "SpnEn" pins

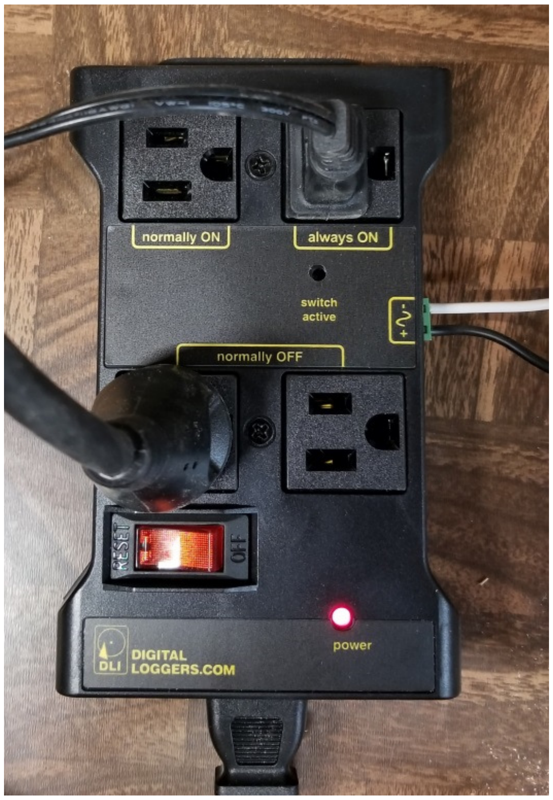
   E3/4 (self-squaring gantry) —> "CoolEn" pins

   Evo3/4 —> "CoolEn" pins



3. Pay close attention to which color jumper is connected to the yellow pin (+). (Image at right is from the Evo4 manual. Your controller may look slightly different, but the pins will be labeled similarly.)
4. Optional: Attach the second pair of jumper wires to the ends of the first pair to get farther away from the controller before splicing on the wires.. I wrapped electrical tape around the connection to reduce the chance of it coming loose due to vibration.
5. Attach the extension wires to the ends on the second pair of jumper wires using butt splices.
6. Use wire ties to secure the wiring to the gantry frame.
7. Remove the green plug from the side of the IoT Relay by pulling it outward, noting which terminal is + and which is –.
8. Loosen the screws to open the wire insertion ports.

9. Strip the end of the + wire (as noted in step 3), insert into the port, and tighten the screw to secure. In my case, the purple jumper was on the + pin, which I then connected to the extension wire with the black stripe, and has been connected to the + terminal of the green plug in the photo at right.
10. Repeat the process with the other (-) wire.
11. Reinsert the green plug into the IoT Relay.
12. Plug the router into one of the outlets marked "normally OFF".
13. If desired, you can plug the power supply for the E3/4 controller into the "always ON" outlet. This is convenient for turning it off with the main power switch when not in use.
14. Plug the IoT Relay into a 120VAC outlet.
15. Turn on the IoT Relay's power switch.
16. Turn on your router's switch.
17. Ensure that your software is providing the "M3" command near the beginning of the gcode file, and "M5" near the end. (See a couple of examples below.)

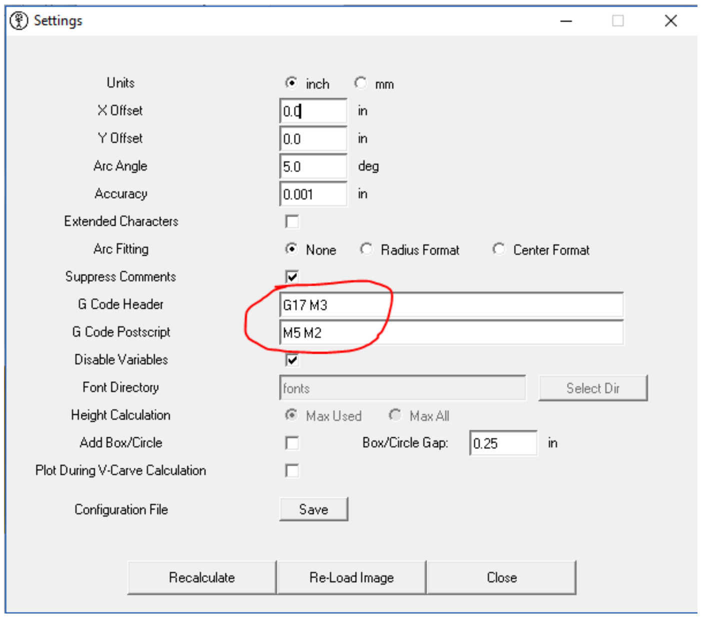


That's all there is to it!

Examples of setting up your software to include M3 and M5:

If you use **F-Engrave**, you've probably already seen the tip to remove extraneous commands from the header and post-script, in which case you may currently have G17 (arc plane selection) in the header and M2 (end program) in the post-script.

To add M3 and M5, select "General Settings" from the Settings menu, and insert M3 <u>after</u> G17 and M5 <u>before</u> M2. Then for all future gcode that you create, the M3 and M5 commands will be included.



Units	[✓ inch] mm
X Offset	0.0 in
Y Offset	0.0 in
Arc Angle	5.0 deg
Accuracy	0.001 in
Extended Characters	☐
Arc Fitting	[✓ None] Radius Format Center Format
Suppress Comments	✓
G Code Header	G17 M3
G Code Postscript	M5 M2
Disable Variables	✓
Font Directory	fonts
Height Calculation	[✓ Max Used] Max All
Add Box/Circle	☐ Box/Circle Gap: 0.25 in
Plot During V-Carve Calculation	☐
Configuration File	Save

In **VCarve Desktop**, I needed to edit the post processor file to insert the M3 and M5 commands.

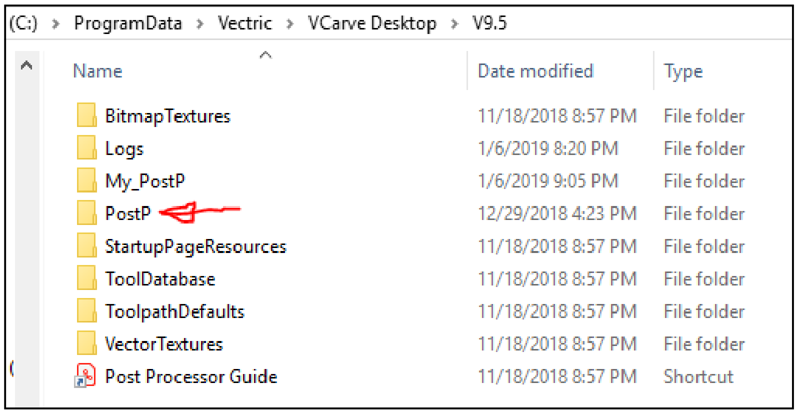
I'm no expert on this stuff, but after doing a little reading, here is how I went about it and it is working for me.

First, I located the post processor files (see top image at right).

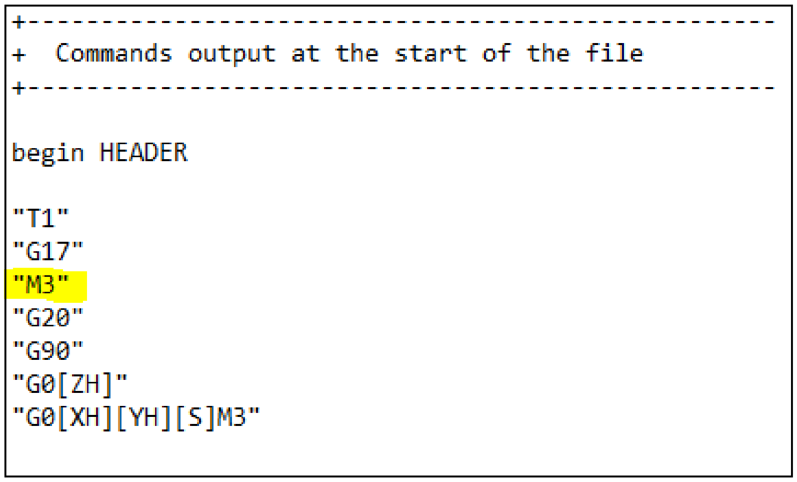
Then I scrolled down in that folder until I found the file called "Grbl_inch", and I *copied* that file to the "My_PostP" folder (the folder just above the PostP folder in the image at right).

As the readme file in My_PostP explains, any post processors that are placed in that folder will be the only options displayed in the software. Since I only plan to use grbl inch, this sounded like a great plan to avoid accidentally grabbing the wrong file.

I opened the newly-copied Grbl_inch file in Notepad and added the M3 command to the header and the M5 command to the footer as shown in the middle and lower images at right.

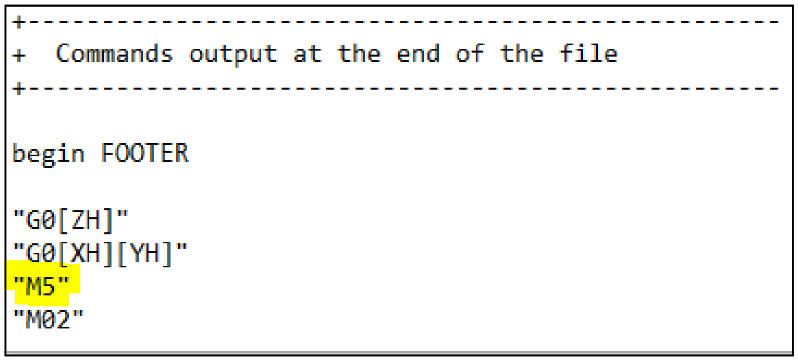

(C:) › ProgramData › Vectric › VCarve Desktop › V9.5

Name	Date modified	Type
BitmapTextures	11/18/2018 8:57 PM	File folder
Logs	1/6/2019 8:20 PM	File folder
My_PostP	1/6/2019 9:05 PM	File folder
PostP	12/29/2018 4:23 PM	File folder
StartupPageResources	11/18/2018 8:57 PM	File folder
ToolDatabase	11/18/2018 8:57 PM	File folder
ToolpathDefaults	11/18/2018 8:57 PM	File folder
VectorTextures	11/18/2018 8:57 PM	File folder
Post Processor Guide	11/18/2018 8:57 PM	Shortcut


```
+---------------------------------------------------
+  Commands output at the start of the file
+---------------------------------------------------

begin HEADER

"T1"
"G17"
"M3"
"G20"
"G90"
"G0[ZH]"
"G0[XH][YH][S]M3"
```


```
+---------------------------------------------------
+  Commands output at the end of the file
+---------------------------------------------------

begin FOOTER

"G0[ZH]"
"G0[XH][YH]"
"M5"
"M02"
```

File created by Greg Grote, with assistance and input from Geremy Grote and Kjell Evensen. Please let Greg know of errors or omissions.

Original (posted to FB group Jan 7, 2019)

Feb 21, 2020 … updated to include self-squaring gantry feature

Dec 24, 2020 … updated to include Evo3/4