# Building (H-)Bridges

## — Part 1 —

by Peter Best

*Today's robotic creations use various methods — which are based on the physics of our little planet — to enable the motion of their main body or subordinate appendages. The various parts and pieces of these intelligent "things mechanical" use hydraulics, air pressure, muscle wire, and even gravity to invoke a mechanical displacement from Point A to Point B.*

Despite the increasing use of the aforementioned physical methods in robotic equipment, the major component involved in making robotic things move is still the motor. Take a look at the advertisements and columns in *SERVO*. The majority of them have some sort of motor at their root. And, in most cases, if a motor is not in the column's or advertisement's mix, the electronics that drive or control a motor are.

Motors are the main motivation of this column, as well. However, instead of delving into the nuances of how motors use magnetic fields to create motion, I'm going to show you how to build electronic circuitry that controls the activation, deactivation, and direction of the electronic movement within a motor's magnetic domain.

## Driving a Brushed DC Motor

Small brushed DC motors are fascinating. Their internal complexity is overshadowed by their ease-of-use. If the motor's minimum operating voltage is low enough, connecting the brushed motor's two power leads across a battery is all that is needed to get the motor's shaft to rotate. The fun comes in when you reverse the battery's polarity with respect to the motor's power leads. The motor shaft will then spin in the opposite direction.

If the brushed motor has exceptional bearings supporting the shaft, disconnecting the battery will result in the shaft coasting to a stop. The ability of a motor to provide forward motion, reverse motion, and to stop are physical properties used by every robotic device that employs the services of a traditional electromagnetic motor. However, to take robotic advantage of the work done by a motor, the motor's forward, reverse, and stop properties must be able to be controlled.

The circuit I've devised in Figure 1a is the most basic of brushed DC motor control designs. A positive voltage that is sufficient enough to turn on the MOSFET applied to the MOSFET's gate will provide a ground path for the motor through the MOSFET which, in turn, will put the motor's shaft into motion.

The Schottky diode that parallels the motor is there to allow a conduction path for the back EMF that is created by the motor coil when the motor shaft stops spinning. This type of motor control is great if all you want to do is drive the brushed DC motor full tilt in a single direction. If rotating the motor shaft in a single direction is fine, but you don't want your robotic device moving at warp speed all of the time, you must consider controlling the speed of the brushed DC motor.

With the circuit shown in Figure 1a, speed control is easily achieved by simply applying a PWM (Pulse Width Modulation) signal to the MOSFET's gate. The higher the on time (logical high) of the PWM signal, the faster the motor's shaft will spin.

What if your little robotic device had to use a brushed DC motor and controller in the Figure 1a configuration to move one of its mechanical parts from Point A to Point B and then return to Point A? I'm thinking about some really nasty things that have to do with DPDT mechanical switches to switch the brushed DC motor's

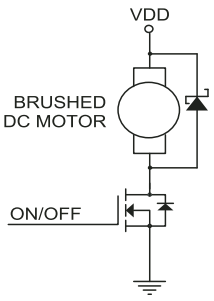



FIGURE 1A

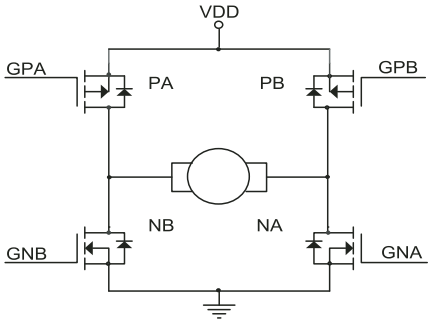
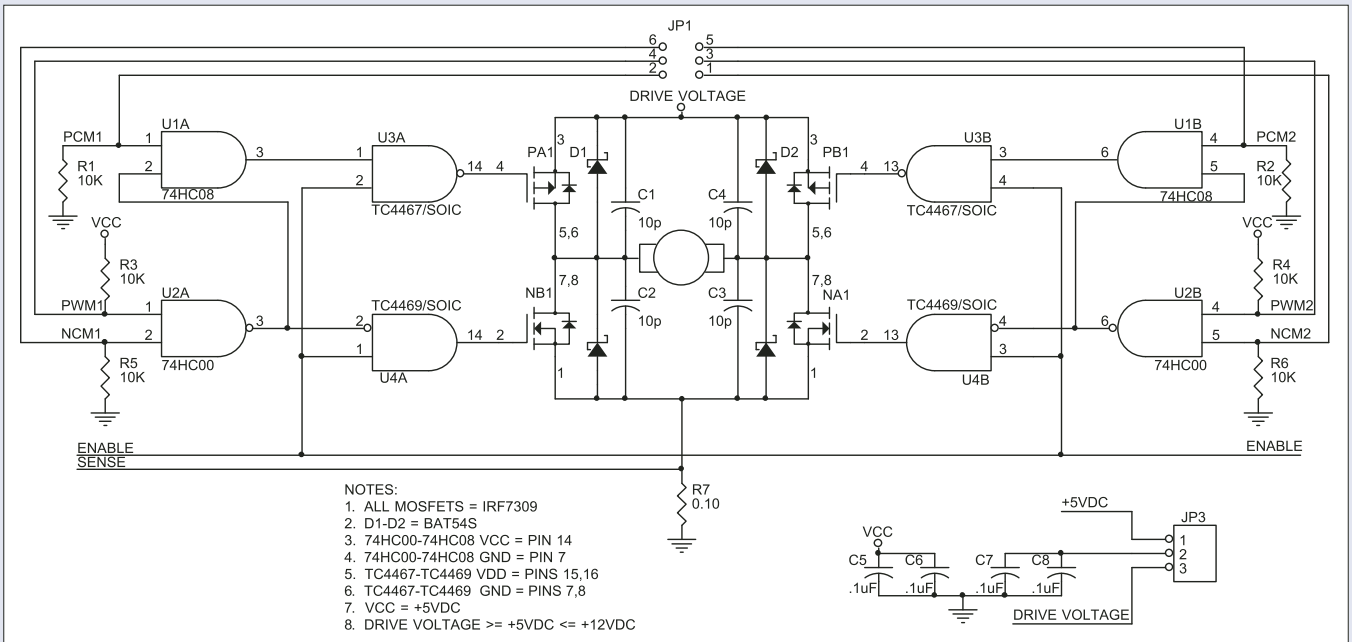



FIGURE 1B

| GPA | GPB | GNA | GNB | FUNCTION |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | OFF (FREEWHEEL) |
| 0 | 1 | 1 | 0 | FORWARD |
| 1 | 0 | 0 | 1 | REVERSE |
| 0 | 1 | 0 | 1 | SMOKE |
| 1 | 0 | 1 | 0 | SMOKE |

FIGURE 1. (A) This is Electronics 101 stuff. If you want to see the magic smoke, just leave the Schottky diode out of this little circuit. Also, if you want to experiment with this circuit, be sure to place a 100 ohm resistor in series with the MOSFET gate. (B) Don't try this at home! This circuit is bare bones and is for illustration purposes only. Although the MOSFETs will switch and drive the motor per the truth table, there is no protection for the MOSFETs in this circuit other than their internal diodes.




SCHEMATIC 1
DRIVES 1 AND 2

SCHEMATIC 1. If you take it one piece at a time, this is a very simple circuit to analyze logically. Once you understand how a half-bridge works here, you have the key to understanding what happens when you combine half-bridges to form full H-Bridges.

power terminals into reverse mode that we really don't want to discuss. So, let's add a reverse gear to our brushed DC motor electronically.

In order to switch the brushed DC motor's power leads between being sourced and sinked, you need the circuit in Figure 1b, which adds a pair of P-Channel MOSFETs (PA and PB) to provide the sourcing of power to each of the brushed DC motor's power leads. The sinking function is provided by a pair of N-Channel MOSFETs — NA and NB — which are tied to each of the brushed DC motor's power leads, as well.

Note that one N-Channel MOSFET drain and one P-Channel MOSFET drain are connected to each of the brushed DC motor's power leads. Power to the brushed DC motor enters at the P-Channel MOSFETs' source pins. The brushed DC motor's power path to ground is provided by the sinking N-Channel MOSFETs — NA and NB.

For the sake of discussion, let's assume that MOSFETs PA and NA associate with clockwise rotation of the brushed DC motor shaft and PB and NB associate with counter-clockwise rotation of the brushed DC motor shaft. With that, to spin the motor shaft in a clockwise direction, we must provide (source) power to the motor using the PA MOSFET. The other motor power lead must somehow get to ground. The N-Channel MOSFET — NA — provides for a ground path for the motor and performs the sinking, or grounding, function upon its activation. We can apply the same logic for counter-clockwise rotation of the motor shaft by activating MOSFETs PB and NB. I've assembled a motor shaft direction truth table with Figures 1a and1b.

Notice in the Figure 1 truth table that a couple of combinations of activated MOSFETs result in SMOKE. Never do we want to activate MOSFET pairs PA and NB or PB and NA at the same time. It's pretty obvious in Figure 1b that activating these pairs in any combination will produce a path from the source voltage to ground through the MOSFETs, which have very low drain-to-source resistances.

In other words, activating the PA/NB and/or PB/NA MOSFET pairs will produce a virtual short circuit from the power source to ground. If your coding prowess is exceptional and you feel that you can handle switching the MOSFET pairs correctly using only your firmware, go for it. However, some simple hardware placed in front of the MOSFET pairs will assure that your "perfect" firmware won't take the basic H-Bridge shown in Figure 1b into SMOKE mode.

## Controlling the H-Bridge

Our main objective here is to provide a fool-proof control mechanism for the MOSFET pairs that make up our H-Bridge. Rather than depend on "perfect coding," let's employ the services of a couple of MOSFET driver ICs that bring a bit more to the table than just being able to drive a MOSFET gate.

The Microchip TC4467 and TC4469 are four-output CMOS buffers/MOSFET drivers that can, by themselves, deliver up to 1.2A of peak drive current. In fact, you can actually drive motors that require less than 250 mA of current directly from the TC4467 or TC4469 output pins.

The difference in the TC4467 and TC4469, when compared to other MOSFET drivers, is their inclusion of integral logic gates to complement the MOSFET drivers. Each of the four output drivers in both the TC4467 and TC4469 is front-ended by a two-input logic gate. The TC4467's input pair

feeds a standard NAND gate while the TC4469 logic gates are configured as AND with an inverted input.

I've added a 74HC00 and 74HC08 to the TC4467 and TC4469 H-Bridge controller mix in Schematic 1. Be aware that although the components in Schematic 1 look to be in a full H-Bridge configuration, they are not. What you actually see in Schematic 1 is a pair of half-bridges. PA1 and NB1 make up one of the half-bridges and PB1 and NA1 comprise the other half-bridge. Placing jumpers across JP1 will combine the pair of half-bridges into a full H-Bridge. For now, we'll forego the JP1 jumpers and keep the configuration in half-bridge mode as I walk us through the bridge control logic.

The idea is to not allow the vertical pairs of MOSFETs to be activated simultaneously. So, let's logically analyze the logic gates to see if our protection circuitry works. PCM (P-Channel MOSFET) 1/2, PWM (Pulse Width Modulation) 1/2, and NCM (N-Channel MOSFET) 1/2 are all bridge control inputs. Let's look at what the PA1 and NB1 MOSFET gates look like logically with no input stimulus on the PCM1, PWM1, and NCM1 inputs. Assume the ENABLE line to be active, or logically high.

Since pin 1 of U1A is pulled low, the output of AND gate U1A will be logically low regardless of the logic level applied to U1A pin 2. With the ENABLE line held at a logically high level, the inputs currently being applied to the TC4467 NAND driver will produce a high level on the output of U3A, which turns PA1 off. With one input pulled high and the other pulled low, U2A's output pin will be logically high.

The high level on the output of U2A feeds the invert input pin on the TC4469 MOSFET driver. Thus, pin 2 of TC4469 is effectively a logical low input. Since U4A is an AND gate with an inverted input, the output of U4A will be low, turning off MOSFET NB1. The PB1/NA1 half-bridge circuitry is identical and so are the results of logic levels applied to the gates of PB1 and NA1 with relation to input levels applied to PCM2 and NCM2.

PCM1, in conjunction with the ENABLE input, is used to turn on MOSFET PA1. Introducing a logically high level to pin 1 of U1A with no external input stimulus applied to U2A results in both of U1A's input pins presenting a high logic level to the 74HC08 AND gate, which produces a high at pin 1 of U3A. With the ENABLE providing a high input level at pin 2 of U3A, the TC4467 NAND gate's output goes logically low to turn on MOSFET PA1.

Here's where things get interesting. NCM1 is used to turn on MOSFET NB1. Let's assume PA1 is on and we apply a logical high to the NCM1 input to turn on NB1. Again, ENABLE is active and is held in a logical high state. Taking NCM1 logically high produces a logical low on the output pin of the NAND gate U2A, which feeds a low to the input of AND gate U1A, which drives the output pin of U1A low, which drives the output pin of NAND gate U3A high and turns off PA1.

Meanwhile, the low level on the output pin of NAND gate U2A feeds the inverted input pin of the TC4469 AND gate, which results in a high being fed to the gate of NB1 turning the N-Channel MOSFET on. Once again, the control input logic that works for PA1 and NB1 works identically for PB1 and NA1.

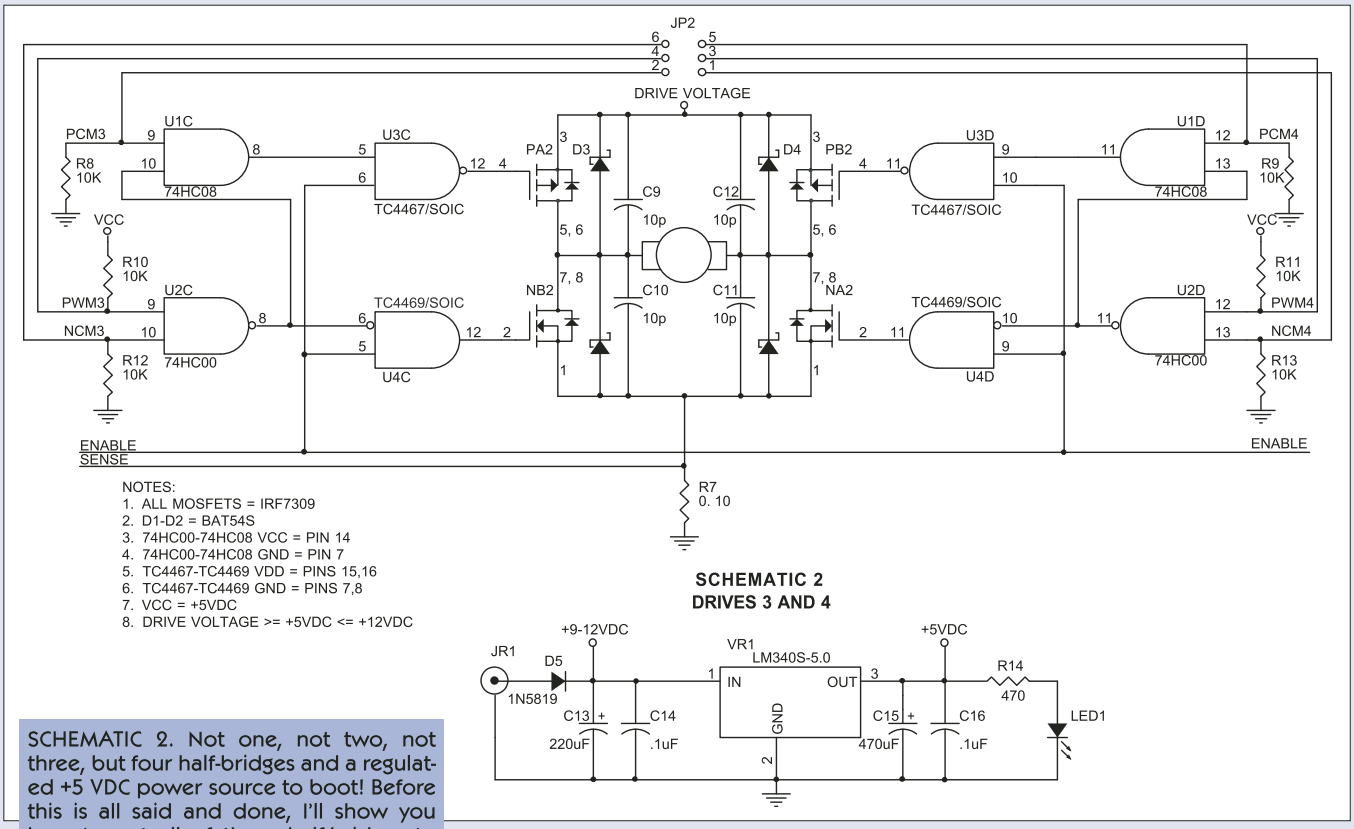
Okay, our MOSFET protection circuitry works great on paper. Let's check out the logic again and make sure we can actually turn the brushed DC motor shaft in both directions using half-bridge mode. Earlier, we associated PA1 and NA1 with clockwise rotation. So, to initiate clockwise motor shaft operation, we must energize PA1 and NA1. Applying a logical high to the PCM1 input will turn on PA1, which will allow the DRIVE VOLTAGE to flow to one of the brushed DC motor power leads.

NA1 is activated by issuing a logical high level to the NCM2 input. Once NA1 is on, the brushed DC motor's ground path is established causing the brushed DC motor's shaft to turn in a clockwise direction. To shift the brushed DC motor's shaft into reverse, we must remove the input stimulus from PCM1 and NCM2 and apply logical high levels to PCM2 and NCM1. The removal of clockwise stimulus before applying the counter-clockwise logic levels is necessary because we are actually controlling an independent pair of half-bridges.

If we were to jumper pins 1 to 2, 3 to 4, and 5 to 6, applying a logical high to PMC1 or NCM2 would turn on PA1 and NA1 and result in clockwise rotation of the brushed DC motor's shaft. Applying a logical high to the PCM2 input with the JP1 jumpers in place would result in turning on PB1 and NB1 and counter-clockwise rotation of the brushed DC motor's shaft. Thus, with the JP1 jumpers populated, we combine the pair of half-bridges into a full H-Bridge with all of the safety features we designed still intact.

In our simplified half-bridge brushed DC motor driver scenario, the PWM signal is applied to the NCM1 or NCM2 inputs. Since PWM1 and PWM2 are both tied logically high, the alternating PWM signal presented to the NCM1 or NCM2 inputs will force the MOSFET gates of NA1 or NB1 to chop the brushed DC motor's ground path relative to the duty cycle of the incoming PWM signal and thus,

**"Today's robotic creations use various methods — which are based on the physics of our little planet — to enable the motion of their main body or subordinate appendages."**




SCHEMATIC 2. Not one, not two, not three, but four half-bridges and a regulated +5 VDC power source to boot! Before this is all said and done, I'll show you how to put all of these half-bridges to work in various configurations.

regulate the motor shaft's speed.

In most real-world cases, the PWM signal emanated by the microcontroller will be continuous as the programmer will simply kick off the microcontroller's PWM engine and only manipulate the PWM duty cycle as needed. With the PWM running continuously, most likely a full H-Bridge configuration will be used and the NCM1 or NCM2 inputs are then used as gates to allow or disallow the PWM signal from passing on to the N-Channel MOSFET gates.

Schematic 2 utilizes the rest of the gate logic inside all of the ICs that make up the four half-bridges. The TC4467 and TC4469 can be powered by the standard logic supply of +5 VDC or by the DRIVE VOLTAGE, which can span from +5 VDC to +18 VDC. A jumper on JP3 determines which supply the TC4467 and TC4469 draw their power from. If you're scratching your head as to why I haven't mentioned the 0.10 ohm sense resistor, don't worry. I've got plans for that guy.

## The H-Bridge Hardware

Enough theory ... let's build something! I've assembled all of the components in Schematics 1 and 2 into the dual H-Bridge hardware you see in Photo 1. The printed circuit board itself is an inexpensive double-sided board with a ground plane covering the entire non-component side. There are so few components that comprise the dual H-Bridge circuit that you can actually build this whole thing up by looking at the component placement in Photo 1.

The H-Bridge circuitry you see in Photo 1 can be more easily understood when broken down into its component parts. Photo 2 details the anti-smoke logic and the TC4467/TC4469 MOSFET drivers. From left top to right

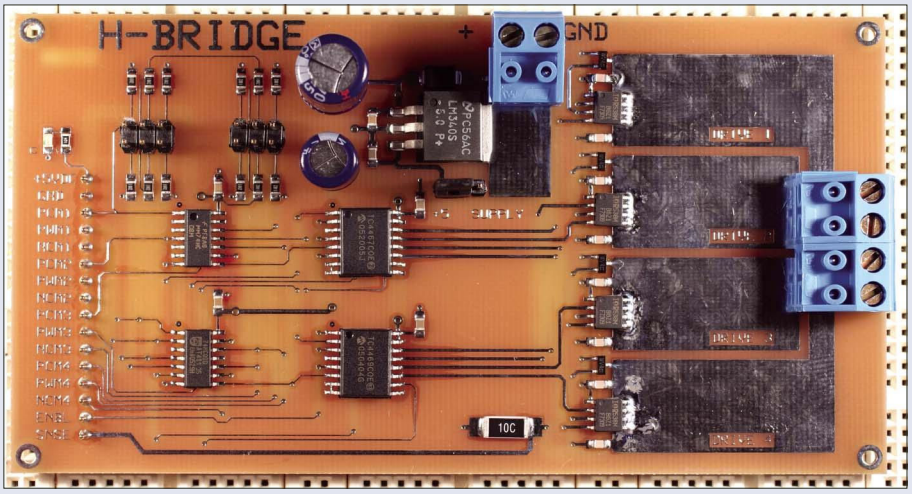



PHOTO 1. There are actually four half-bridges buried within this mix of logic and MOSFET drivers. You can run this baby as four independent half-bridges or a pair of full H-Bridges. In addition to being able to drive small motors with ease, the circuitry boasts some simple logic that prevents you from accidentally letting the magic smoke out of the MOSFETs and their drivers.

top, you see the power indicator LED and its respective current limiting resistor. Directly to the right of the power indicator LED/resistor combination is JP1 which, when jumpered, puts half-bridges 1 and 2 (or Drives 1 and 2) into a single, full H-Bridge configuration. JP2 — whose jumpers put Drives 3 and 4 into full H-Bridge mode — is positioned to the right of JP1.

Photo 3 is a bird's-eye-view of the H-Bridge +5 VDC logic power supply. There's nothing fancy here as the dual H-Bridge logic power supply is constructed around an LM340S-5.0 fixed voltage regulator. The logic power supply always powers the 74HC00 and the 74HC08 and can be jumpered to supply +5 VDC power to the MOSFETs, TC4467 and TC4469. The Drive Voltage jumper — which is just below the logic power supply — can also be positioned to provide the voltage at the input of the LM340S-5.0 to the MOSFETs, TC4467 and TC4469. The Drive Voltage jumper is set across the +5 VDC position in Photo 3.

The 10 pF EMI (Electromagnetic Interference) capacitors, the BAT54S made up of a pair of back-EMF steering diodes, and the IRF7309 MOSFET pair are all packed into the electronic components you're close up on in Photo 4. All of the IRF7309 MOSFET drains (pins 5, 6, 7, and 8) are tied together and bonded to a one-square-inch heatsink/drive output pad. The heatsink size is directly proportional to the amount of current you want to pull through the bridge's MOSFETs.

## Crossing the Bridge

I've provided the dual H-Bridge ExpressPCB layout file — which is available on the *SERVO* website **(www.servomagazine.com**) — for those of you that may wish to customize the H-Bridge design I've presented or simply order your own set of dual H-Bridge boards from ExpressPCB directly. For those of you that are surface mount challenged, you can also get all of the SMT components that make up the dual H-Bridge I've described as through-hole and DIP packaging, with the exception of the IRF7309 MOSFETs. There's nothing to stop you from using MOSFETs that are packaged differently than the IRF7309s.

If you haven't skipped through the pages to get here, you've made it to the end of this part of our H-Bridge discussion. You now know how to control each and every one of the MOSFETs on all four half-bridges and, once you build up the H-Bridge circuit I've offered to you, you can attach a brushed DC motor and jumper in the bridge control logic levels necessary to make the motor shaft spin in a clockwise or counter-clockwise direction. If you already have the means and knowledge to do so, you can also present a PWM signal to the proper bridge control inputs and control the speed of your motor's shaft.

Although we've crossed the "bridge," we're not done. Next time, I'll introduce you to the brand new PIC16FHV616 and show you how to use simple PIC assembler code to exploit all of the PIC16HV616's special functionality to drive stepper and brushed DC motors with the H-Bridge circuitry you've just read about. I'll also produce some circuitry and firmware to put that 0.10 ohm sense resistor to work as a vital component of a safety net and motor current monitor station for the H-Bridge MOSFETs. SV

**Peter Best can be contacted via email at peterbest@cfl.rr.com**

PHOTO 2. There are 16 logic gates and eight MOSFET drivers in this shot. Note the bridge control inputs pinned out to the left and the bridge configuration jumper blocks, which are bare indicating half-bridge mode. The dozen 10K resistors surrounding the bridge configuration jumper blocks guarantee that the MOSFETs are turned off in the absence of input stimulus at the bridge control inputs.

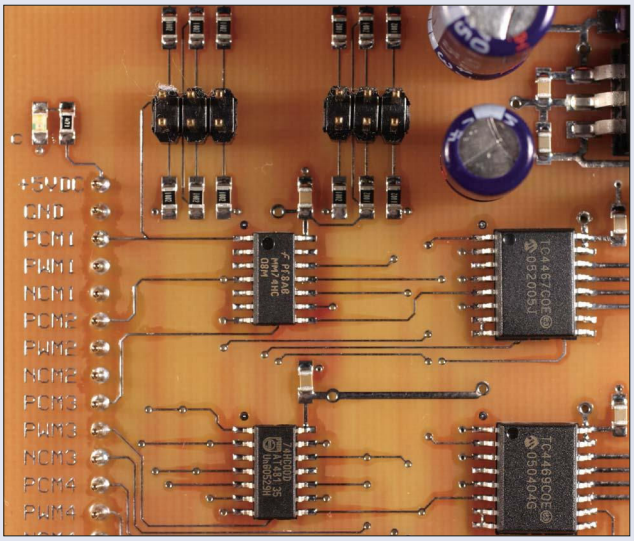



PHOTO 3. Imagine that ... a National Semiconductor fixed voltage regulator providing regulated power to the logic. The screw terminal in this shot is the portal for incoming power for the dual H-Bridge logic and motor.

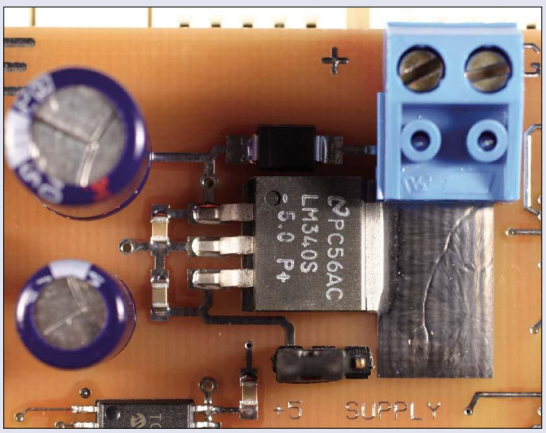



PHOTO 4. Here's a complete half-bridge, including the back-EMF steering diodes and the EMI capacitors. The IRF7309 was designed to switch things about in laptop computers and other electronic devices that require higher switching currents from smaller footprint components.