February 1993

# Encoder

The Newsletter of the Seattle Robotics Society

# Motor Driver Basics, part 2

by Bob Nansel

Last time I introduced the basic concepts of controlling motor speed and direction with pulse-width-modulation (PWM) and H-bridge circuits. I showed two unidirectional solid-state and two bidirectional motor drivers, but both bidrectional drivers required relays to reverse the motor.

This time I will show two simple H-bridges that are completely solid-state, one using bipolar transistors, the other using power MOSFETs. Neither circuit can be considered high performance, but they both work

For both circuits, I used a 12V, 300 RPM Barber/Colman gearhead motor (#FYQF-63310-9) as the test motor with an RC snubber network with a 47 ohm resistor 0.0033 µF capacitor.

## Bipolar Transistor H-Bridge

Since a bipolar transistor conducts in only one direction (from collector to emitter) when saturated ON, it must have an anti-parallel connected "free-wheeling" diode to conduct recirculating current in the opposite direction. Sometimes this anti-parallel diode is referred to as a "protection" diode because it prevents large negative or positive voltage spikes from forming across the transistor.

Remember, those spikes are a natural consequence of the motor's inertia and inductance attempting to keep current flowing through the motor's windings when the transistor supplying that current shuts OFF. The low-side diodes keep the negative spikes from going any lower than one diode drop below ground, while the high-side diodes keep positive spikes from rising any higher than one diode drop above B+.

This combination of four free-wheeling diodes is used extensively elsewhere in electronics where its known by a more familiar name, the full-wave rectifier bridge. In fact, diodes D1 through D4 in figure 1 are actually part of a 4-pin DIP package full-wave rectifier available from Radio Shack (#276-1161). The positive pin is connected to B+, the negative to ground, and the two AC "input" pins are connected the positive and negative

(Please see **Driver** continued on p. 7)

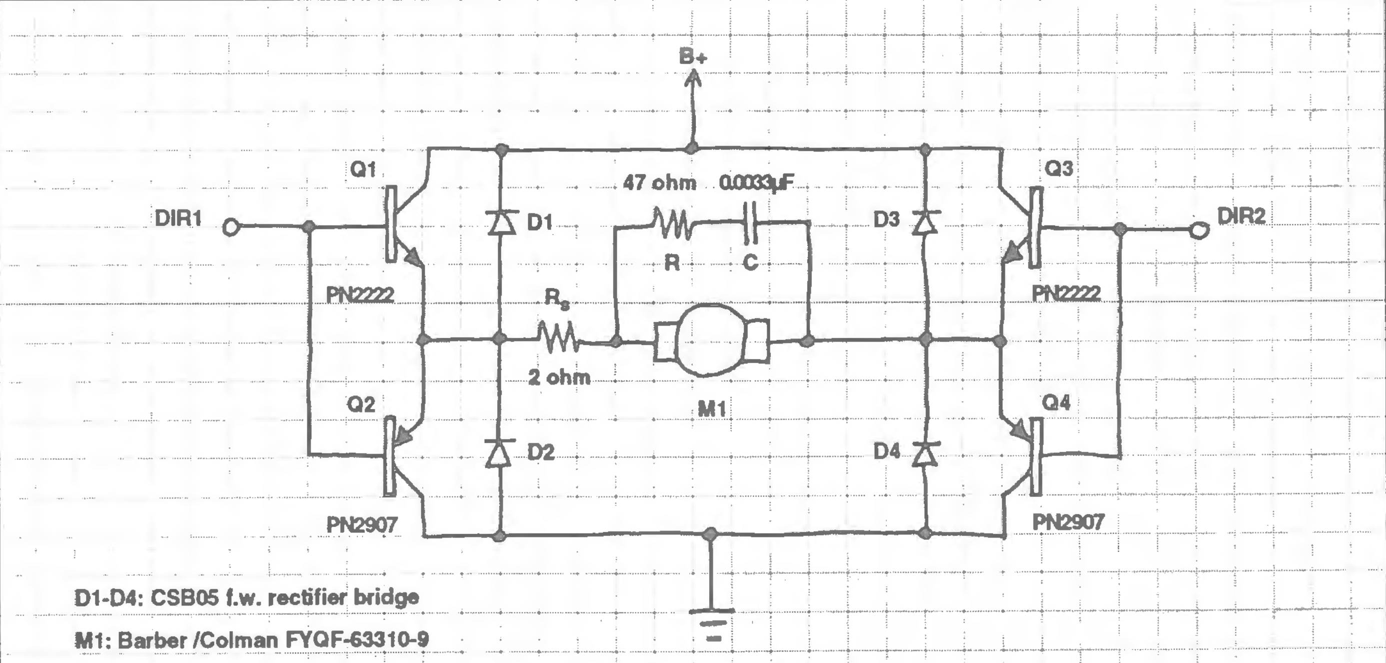



Figure 1: Bipolar H-Bridge

# Technical Articles I'd Like to See

The *Encoder* is driven by you, the technically minded robot builder. *Encoder* readers possess collectively an incredible array of technical skills and knowledge. I want to tap that knowledge for future issues of the *Encoder*.

The *Encoder's* success is built on solid, nuts-and-bolts articles that guide robot builders through the complexities of motor control, sensor signal processing, software design, hardware and mechanics. We prefer projects that stretch the envelope, but that you can build now with *today's* tools and materials. Those of you who have contributed such articles in the past, I thank you.

To those of you who have never submitted articles, I want to offer this challenge: take a look through the following list of ideas. Somewhere in this list is an area of technology where *you* have unique expertise or interest. Write an article based on your experience. Tell us about your successes, but also tell us about your failures. Give us examples of circuits, drawings or photographs of your work. You will be recognized for your work, and others will learn from your experience.

So, in future articles, I'd like to present more material on...

**Robot Mechanics**
- Lead screws
- Practical hydraulics
- Practical pneumatics
- Using solenoids
- Shape memory alloy actuators
- Driving brushless DC motors
- Rewinding DC motors

**Robot Controllers**
- Miniboard projects
- Processor independent I/O interfaces
- PID control algorithms

**Advanced Computing Architectures**
- Genetic algorithms
- Neural networks
- Fuzzy logic

**Electronics, Basic & Advanced**
- Switching characteristics of power MOSFETs
- Signal processing
- DSP chips for dummies

**Navigation & Simulation**
- Path-planning
- Configuration space
- World modeling
- CROBOTS
- Robot motion simulation including friction, wheel slip and inertial effects

## *Encoder* readers posses collectively an incredible array of technical skills and knowledge. I want to tap that knowledge...

**Optical Obstacle Detection**
- Structured light obstacle detection systems
- Structured light projectors
- Converting camera autofocus mechanisms for robot use
- Smoke, fire & heat sensors for security robots
- beacon sensors

**Sensing Forces & Displacements**
- force sensitive resistors
- proximity detectors
- strain gages
- linear displacement sensors

## We prefer projects that stretch the envelope, but that you can build now with *today's* tools and materials...

**Sensing Robot Motion & Orientation**
- Solid state gyroscopes
- Solid-state Accelerometers
- Flux-gate compasses

**The Power Struggle**
- Photovoltaic power sources for robots
- Care and feeding of lead acid batteries
- Care and feeding of NiCd batteries
- Efficient DC/DC converters
- Power management

**Advanced Robot Systems**
- Solar-powered "moon rover" robot testbeds
- Modem line telepresence robots
- "Gopher" & "Follow me" robotic aids for the disabled
- Security robots
- Delivery robots
- Submersible robots
- Flying robots

Of course, if you have ideas for articles not in this list, drop me a line.

The Editor

# The President Says...

by Karl Lunt

I have been giving PADSDEMO a workout recently, and thought I would pass along some of my experiences.

For those not yet informed, PADSDEMO is a shareware version of the PADS schematic-capture and PCB-layout program offered by PADS Software, Inc. The full-up PADS program can handle schematics with over 550 ICs and thousands of connections; PADSDEMO has been "crippled" so it can only handle about 30 ICs and 150 connections. So far, none of my projects have come close to these limits.

PADSDEMO runs only on PCs and really needs a 80386 or better to work smoothly. It supports VGA and some versions of super-VGA, and you should have a mouse to get the best use of the program.

You can get a free copy of PADSDEMO by calling the PADS company (1-800-255-7814) and answering a few questions. They will send you a set of floppy discs that contain the PADS program and fairly complete documentation.

Even better, PADS Software provides a toll-free technical hotline for its PADSDEMO users. The staff is very knowledgeable, polite, and promptly answered all my questions. When was the last time you heard of such support for a shareware program?

The PADSDEMO program comes with a standard library of over 6000 parts. These include logic ICs as well as microcontrollers such as the 68HC705C8. It also covers connectors, caps, resistors; just about anything you need for your schematic and PCB design.

I needed to use a 40-pin DIP version of the 68HC705C8 in a project, but PADSDEMO only had the PLCC variant. I had to add the DIP version to the PADSDEMO library, but that section of the manual is very complete and it only took a couple of hours.

I will point out one operation that the manual does not cover well at all, but will cause you problems when you go to lay out a PCB of your schematic. The problem area concerns stringing networks of connections such as power and ground.

PADSDEMO will tie large networks in an arbitrary pattern. This pattern is guaranteed to make your routing difficult (it isn't intentional, but it is guaranteed). When you use PADS-PCB (the layout portion of PADS-DEMO), you will find yourself needing to change the network pattern in order to get a good routing of your PCB traces. Unfortunately, the manual doesn't tell you how to change the network patterns.

The trick involves using PADS-PCB's On-the-Fly (OTF) routing option. This feature lets you make or break connections at will, so you can restring them as necessary. You make a connection by simply selecting (with your mouse) the two circuit points involved. You break a connection by selecting a line joining two circuit points.

When you break a connection, PADS-PCB automatically tests to see if you have also broken the network. This means it checks to see if you have electrically isolated any parts of the original network. If so, it warns you and forces you to acknowledge what you have done before it actually makes the change.

This last feature is very important, and is the key to restringing networks in PADS-PCB. For example, suppose PADS-LOGIC (the schematic-capture portion of PADSDEMO) generated the following network pattern:

```
C1 ------- R6 ------- Q3
```

and you really have to have the network look like:

```
R6 ------- C1 ------- Q3
```

in order to route your traces properly. Using the OTF feature, you would first add a connection between C1 and Q3, to give:

```
C1 ------- R6 ------- Q3
|                     |
+---------------------+
```

then you would break the connection between R6 and Q3, to give:

```
C1 ------- R6         Q3
|                     |
+---------------------+
```

which, as you can see, is electrically equivalent to our original goal. The key is that you first add a redundant connection, then break the original one. This sequence leaves the network intact throughout, assuring you that your changes were only in order, not in logic.

If you have not yet tried PADS-DEMO, I urge you to give PADS Software a call and get a copy. This program offers plenty of power for doing most robotics projects. Best of all, schematics and board layouts can be passed between PADS users as ASCII files. This means you can share your circuits and PCB patterns with others, via the SRS BBS.

Karl

# The Sumo Blend

by William Harrison, Sine Robotics

Sumo Robots are a blend of mechanics and logic.

An attraction of Sumo Wrestling is that no one strength can win. it takes a good balance of many strengths. The fastest car doesn't always win the race. Sometimes a good driver can win with a less-than-perfect car, or the lead car can lose, due to mechanical problems.

A Sumo robot's performance can be attributed to two types of strengths: Mechanics and Logic. Mechanics involve the physical robot, the way it's built, moves, uses power. The logic governs its movements and actions, the way it senses its environment, processes information, sends control commands. Each of these two strnegths can be farther broken down into smaller strength catagories such as traction or speed.

Electronics are split between the two main strengths. The physics of how a sensor works is Mechanics, but the information that it produces or why the sensor is used is Logic.

Anyone that builds a robot with

*(Please see* **Sumo** *cont. on p. 9)*

# Spontanenus


by Lance Keizer

Subsumption modules "learn." A new sensor module would monitor the bus, checking data codes. It would then select an unused data code which will preceed each of the data it places on the bus.

A new actuator module would execute test actuations to betray which signal codes have the highest correlations, selecting them for feedback.

Physical and mechanical parameters can now be dynamic. Your robot can change, wobble, be crooked, and the modules will adapt.

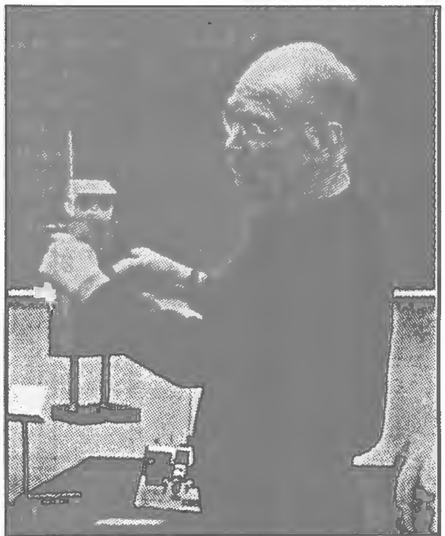
With several sensors, your robot will determine which to pay attention to, and could change its mind at any instant.

Dynamic software? Adapting, changing? No predictability? Suddenly it's not simple. Sounds exaggerated? Wait until you have to troubleshoot what it has arranged.

Sophisticated? No. Simple rules for each module produce complicated emergent behavior. Keep each module as simple as possible. Behavior emerges from module interactions

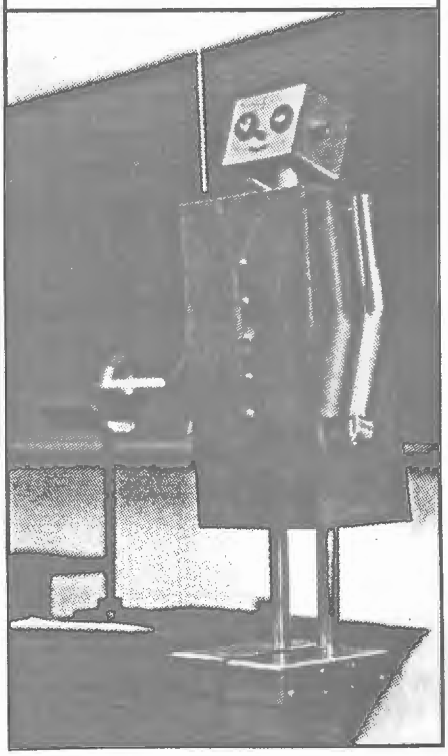
If it were three-dimensional, we could see changes. Software tags, flags, and maps may reveal routing. An entirely new kind of diagnostics will be required. It is a new frontier.

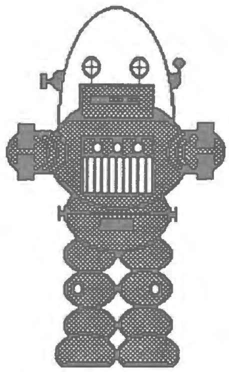
Brainstorm.



Attendees of the February 20th meeting of the S.R.S. were treated by a charming demonstration of *RC*, a radio controlled toy robot built by retired Boeing engineer Francis Reynolds (pictured above).

Reynolds built *RC* for fun, but *RC* was surprisingly sophisticated, especially considering he included over ten separate functions. Thanks, Frank.





P.A.R.T.S
Portland Area Robotics Society

Issue #05 By. Marvin Green (503) 656-8367.

# The $15.00 Wonder Computer.

The MC68HC11A1 is one of the most popular microcontrollers today. This chip can be found in everything from airplanes to automobiles. This article will discuss the 68HC11 in one its most overlooked (and most useful) modes. The single chip mode. The single chip mode is ideal when space is limited and a small program is needed to perform a task. Single chip mode makes all the chip resources available:

**68HC11 Features**
- 5 parallel ports
- 8 A/D Converter
- 2 serial ports
- Event Counter
- 16-bit timer
- COP Monitor
- 10 ma current
- 256 bytes RAM
- 512 bytes EEPROM

All of these features, and more, in a chip the size of a postage stamp. This chip has no built in programming language, but it does have one special feature that makes it very powerful, and easy to program. The 68HC11 contains a special BOOT loader ROM.

The Boot loader program runs when the chip is first powered up, and tries to load/run a program coming into it's serial port. If no communication signal is present at the serial port, the boot loader then tries to run a program it's internal EEPROM.

**It's the EEPROM**, that is really interesting. A small program can be loaded into the EEPROM and made to autostart. The program won't be lost, even at power down. EEPROM is like EPROM, but it can rewritten without extra equipment. (No EPROM erasers or programmers needed). Granted 512 byte EEPROM is not very big, but with all the chips unique features, and a powerful instruction set, much can be accomplished with a small amount of memory.

This chip is easy to program because Motorola has done a great job of supporting this chip. They have a FREE BBS (512) 891-3733 that has many programs to support the 68HC11. You can find small 'C' compilers, assemblers, emulators, and down loaders. Here are some suggested programs that you should download:

| | |
|---|---|
| pcbugb.arc | Interactive programming assemble/disassembler that runs even with limited resources. |
| sim6841.arc | A newer version of my SIM68 Version 0.41 of my 68HC11 simulator. |
| as11.exe | Cross assembler for the MC68HC11 |
| assembler.doc | Documentation for assemblers |
| sc_11.zip | BETA version C compiler for the HC11. |
| wipe.arc | Utility to program 68HC11 EEPROM from PC |
| loadall.arc | Programs various MC68HC11 MCUs directly from a PC using the internal Boot loader (on chip). |

**The PCBUG program** runs on your PC (while connected to 68HC11) and allows easy interactive programming of the 68HC11. It has on screen help, an assembler / disassemble, trace and breakpoint functions. You can develop and test your programs right on the chip. The manual is also free:

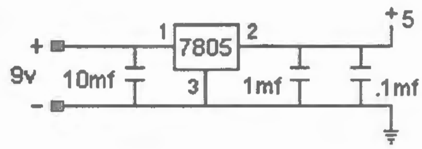
**PCBUG11**

To receive your free copy of the User's Manual, send a large (6" X 9") stamped (2ea. 1st class stamps) self addressed envelope to :

PCBUG11 Manual
Mail Stop #OE319
6501 William Cannon Dr. W.
Austin, Texas, 78735

**The Schematic of the 68HC11 single chip microcontroller** is very easy to build, and consists of four major parts. Part one is the RS232 interface that connects the chip to a PC serial port. Secondly is the clock circuit using an 8Mz crystal, next the reset circuit using a MC34064 and then the power supply. The jumpers are for auto starting a program in EEPROM. When the Tx and Rx lines are tied together, this forces the boot loader to jump to the program located in EEPROM.

The MAX232 chip is used to convert the PD1 (Tx) and PD0 (Rx) signals to voltage levels of the PC, and should connect to a standard 9 pin or 25 pin serial connector on your PC. The chip should be mounted in a socket so it can be removed if not needed when all programming is completed.




The 8Mz crystal can be misleading because the 68HC11 divides the crystal frequency by four, and the chips clock speed is actually 2Mz.

The MC34064 should be used to give the 68HC11 chip a good 'firm' reset signal. The 8054HN can also be used.

A five volt regulator is all that is needed to convert a nine volt battery to power the chip. The 68HC11 can't push much current, and the outputs should be connected to some kind of driver. ( such as a transistor if driving a relay. )

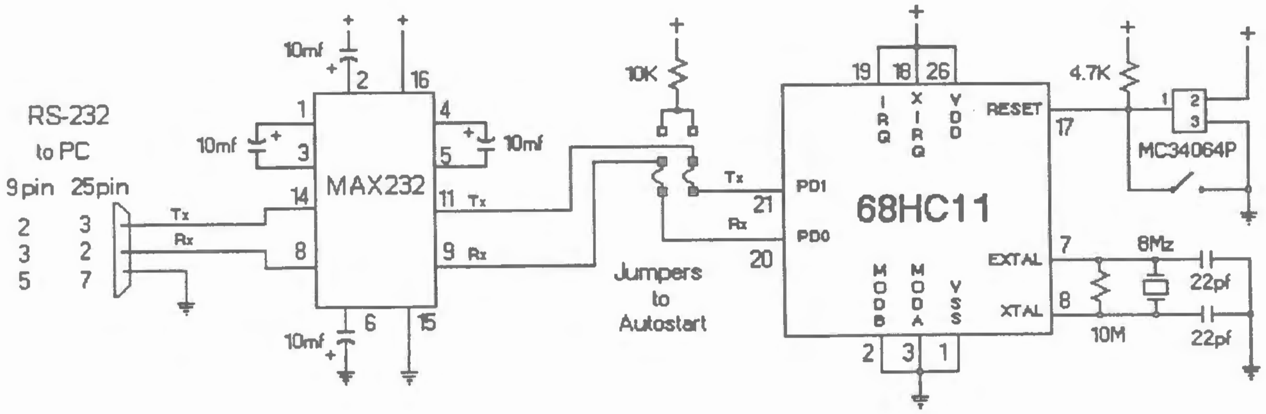
With all the power of the 68HC11 has to offer, a very tiny microcontroller can be built with ease. The 68HC11 can be networked together, with the SPI port, to form a very complex parallel processing system. For example in a robot, a single chip can be the brain for a decent line following robot. ( only 512 bytes needed. ) A series of controllers can be networked together, one chip controlling all motor speed control and current sensing, and another handling bumpers and monitoring battery voltage.

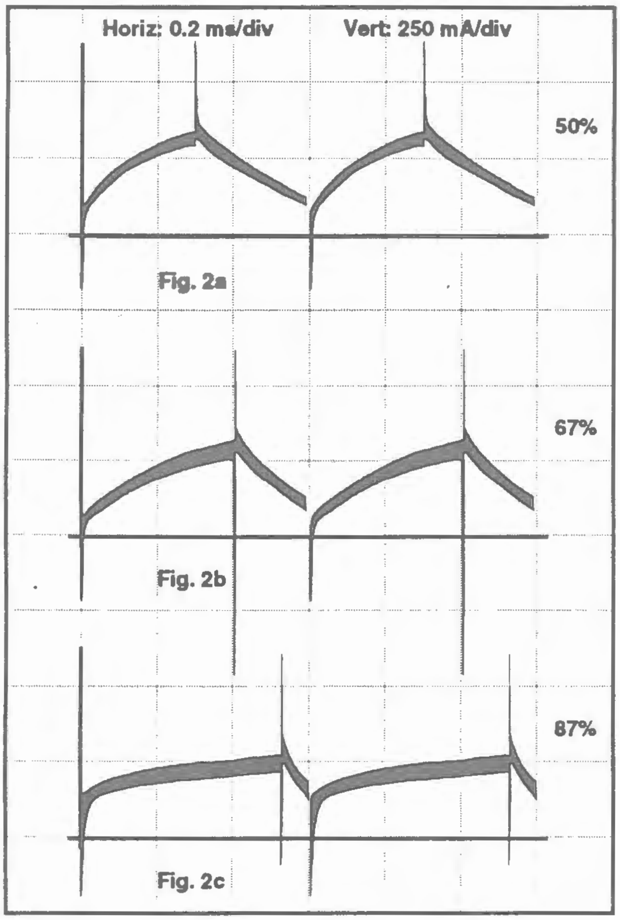
**This microcontroller was put together for about $15.** The programming software was free, the documentation was free... Who could ask for more? Contact your local Motorola Rep. or Motorola for the MC68HC11 technical summary and document 68HC11RM/AD.

# PARTS

| | | | | |
|---|---|---|---|---|
| 68HC11A1 MCU chip | $5.95 | 52 pin PLCC socket | $1.59 | B.G. MICRO |
| 8Mhz crystal | $1.49 | MAX232 RS232 chip | $2.30 | P.O. Box 280298 |
| 16 pin socket | $ .10 | 7805 voltage reg. | $ .35 | Dallas, Texas 75228 |
| | | | | (214) 271-5546 |
| MC34064P | $1.20 | ACTIVE call 1-800-677-8899 for free catalog. | | |

Misc. resistors, capacitors and circuit board can be found at local Radio Shack, or ACTIVE. Take care when building this project, and use a serial cable to connect the MCU to your PC. If you need any help with this project, give me a call.






(**Driver** *continued from p. 1)*

leads of the motor. The 4-pin DIP package is particularly convenient to use because it requires half the connections of a circuit using individual diodes.

Figure 3 shows a simple 2-channel PWM signal generator I used to drive the DIR1 input of the H-Bridge. The PWM frequency remains constant at a value determined by R1, R2 and C1 of figure 3, (about 2 kHz in this case). A high level on DIR1 biases Q1 ON, and Q2 OFF. With DIR2 tied to ground, Q4 will always be ON and Q3 will always be OFF. The current follows the path through Q1, the motor and Q4.

Since both the high-side NPN and low-side PNP transistors are operated in Common Collector ("Emitter Follower") mode, as long as the PWM driving signal remains between B+ and ground no base current limiting resistors are needed, which makes the circuit quite simple. Common Emitter mode would be more efficient (lower $V_{ce}$ saturation voltage, thus more voltage left over for the motor), but you can't just tie the high-side and low-side transistor bases together in a Common Emitter circuit (but if you do, be sure to wear eye protection...).

Common Emitter H-bridges require base resistors for each transistor and separate high-side and low-side drive circuits to prevent the base current of the high-side PNP transistor from turning on the low-side NPN transistor of the same branch. I opted for the simpler Common Collector circuit.

In order to show the behavior of both normal and recirculating currents, I inserted $R_S$, a 2 ohm resistor, in the motor current path. By measuring the voltage across $R_S$ with an oscilloscope and applying Ohm's law (I=V/R), I determined the dynamic current flowing through the motor windings for various PWM duty cycles. The results are shown in figure 2.

With a PWM 50% duty cycle on DIR1 (fig. 2a), the gearmotor barely moved and could be easily stalled. Whenever this happened, the overall voltage seen across Rs rose because the back EMF of the motor was no longer "fighting" the current, in effect decreasing the impedance of the motor. At a 67% duty cycle (fig. 2b), the motor ran only slightly faster, but much more solidly and with less inclination to stall. At 87% (fig. 2c) the motor ran at nearly full speed.

The particular transistors I used for this test circuit weren't really

(**Driver** *continued next page)*

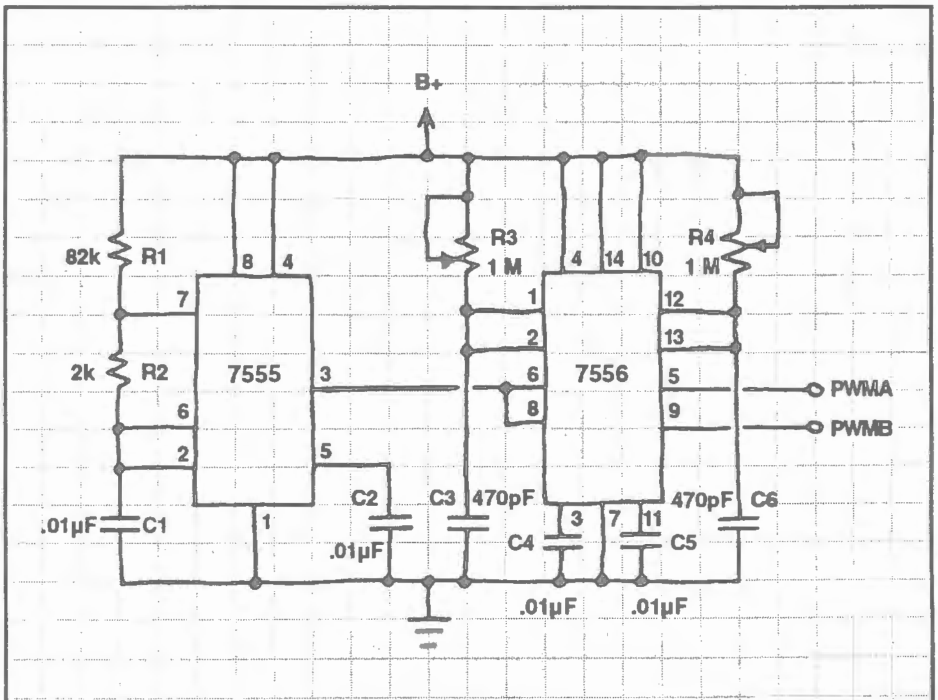



Figure 3: PWM Generator

(**Driver** *continued from previous page*)

suited to handle the currents demanded by the Barber/Colman gearhead. They ran very hot to the touch (I got a blister on one finger as proof) and I fried a half dozen transistors in the course of my experiments. For lower current motors, such as the tape recorder style motors used on Karl Brown's Albert, the circuit would probably work fine.

## Complementary Power MOSFET H-Bridge

Bipolar transistors are cheap and easy to apply for *low* current circuits, but for higher currents, nothing can beat the ruggedness and simplicity of power MOSFETs. Figure 5 shows an H-bridge made from complementary P-channel and N-channel MOSFETs.

Here, both high and low-side MOSFETs are connected Common Source, the equivalent of Common Emitter in the bipolar world.

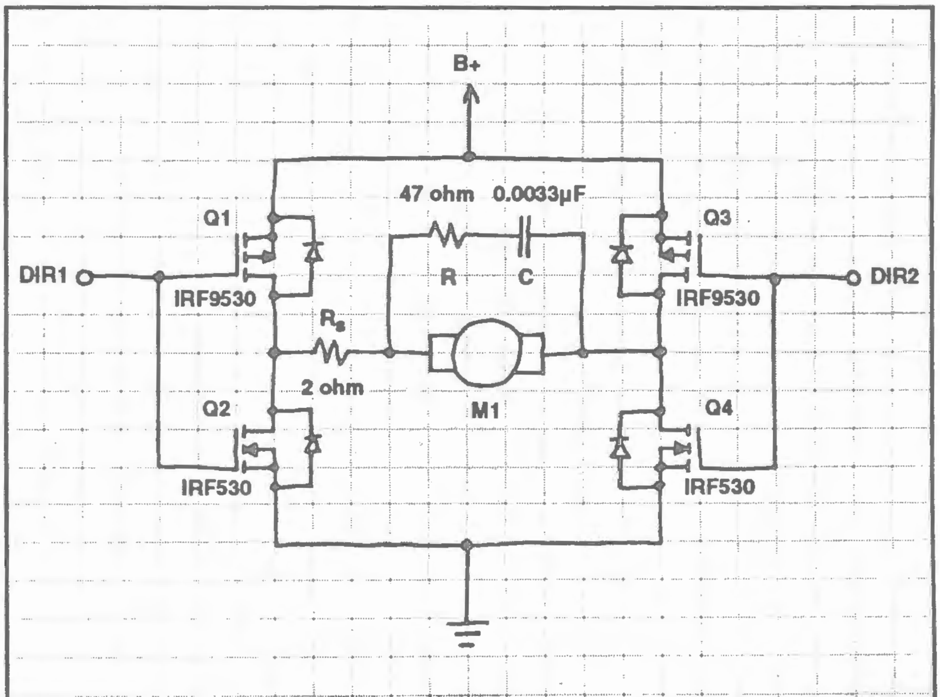



**Figure 5: Complementary MOSFET H-Bridge**

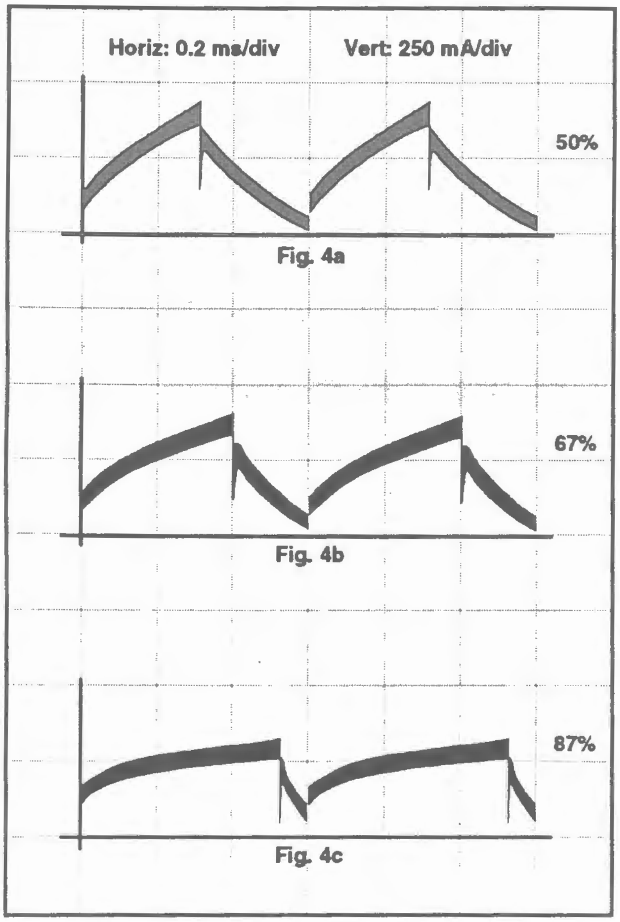
In the case of MOSFETs, though, Common Source topology *does* allow you to tie the high and low-side gates together without fear for the high-side switch somehow turning the low-side switch ON.

Power MOSFETS have intrinsic anti-parallel diodes built in, so separate free-wheeling diodes aren't automatically needed. Finally, one of the more endearing characteristics of MOSFETs is that they conduct equally well in *both* directions when biased ON. This means that recirculating current can flow "backward" through a MOSFET with potentially lower voltage drop than if forced to flow through an anti-parallel diode.

I tested this circuit using the PWM generator of figure 3 to drive DIR1, but in this case when DIR1 is high, the low-side MOSFET, Q2, turns *ON* and the high-side turns *OFF*. With DIR2 tied high, motor current goes through Q3, the motor, then Q2, opposite the case of the first circuit. To keep my current measurements positive, I simply reversed my scope probes from the previous setup. The results are shown for 50%, 67% and 87% duty cycles in figure 4.

The motor ran noticibly faster at all three PWM duty cycles, which I attribute to the lower voltage drops across the MOSFETs as compared to the bipolars. Also, the motor could be run at duty cycles lower than 50%; the motor I tested operated satisfactorily as low as 33%.

Notice the waveforms are also relatively free from current spikes




(**Driver** *continued next page)*

as compared with the bipolar waveforms. I noticed the "singing" of the gearmotor wasn't as loud, which may be a result of the cleaner waveform.

### Summary

Which circuit would I choose? That would depend on the size and current required and cost constraints. The MOSFET circuit will outperform any comparable current capacity bipolar circuit, but P-channel MOSFETs are three times the cost of their N-channel counterparts, so MOSFET H-bridges can be more expensive on the basis of the power transistor elements. The MOSFET circuit presented here could be used for up to, say, 10 amps with suitable heatsinking. Scaling up the bipolar circuit to such levels quickly becomes a headache since a bipolar power transistor capable of handling 10 amps needs upto 1 amp just for base current, making base drive circuitry mandatory.

The MOSFET circuit can be driven directly by CMOS. In the end, the costs tilt a little towards the MOSFET circuit. If P-channel MOSFETs were cheaper (something not likely to happen soon), it would be no contest.

Since N-Channels MOSFETs are so much cheaper, why not just using N-channel MOSFETs for an H-Bridge? Next time, I'll show a circuit that does just that.

(Sumo *continued from p. 4*)

one of these strengths without the other will lose (unless the opponent has a worse weakness). It does no good to have a wonderful control program if your robot has too little traction or a wheel that falls off. Contrariwise, a robot that is mechanically strong but unable to locate its opponent might drive off the edge of the Dohyo.

So don't forget to put effort into all of the strengths of your robot, and don't get too tied up with just one. Good luck with your Sumo robots.

## About the Seattle Robotics Society

The Seattle Robotics Society was formed in 1982 to serve those interested in learning about and building robots. We are a diverse group of professionals and amateurs, highschool students and college professors, engineers and tinkerers. Our passion is the creation of cybernetic creatures that challenge the old definitions of life, intelligence and practicality. We meet 10:00 a.m. to 12:00 noon the third Saturday of every month at North Seattle Community College in room 1652. If you are building a robot or just planning one, come down and meet the gang. We are on an exciting journey and welcome you to join us.

## Upcoming Events

| | |
|---|---|
| **March 20, 1993** | **SRS Meeting at NSCC, rm 1652, 10:00 am.**<br>**First SRS Sumo Scrimmage**<br>**Nominations for SRS Executive Council.** |
| **April 6-8** | **7th International Service Robot Congress**<br>Cobo Center, Detroit, MI;<br>Contact: NSRA at (313) 994-6088 |
| **April 17** | **SRS Meeting. Executive Council elections.** |
| **April 22-25** | **Second International BEAM Robot Olympics**<br>Ontario Science Center, Toronto, Ontario, Canada |
| **May 15** | **Second SRS Sumo Scrimmage** |
| **July 22-25** | **Robothon Northwest 1993**<br>Mobile Robot Competition and Symposium<br>Contact: Karen Nansel<br>Robothon Northwest, Dept. E<br>816 N. 105<br>Seattle, WA 98133<br>(206) 782-5989, 8am-5pm PST |


## Contributors

*Editor*
Bob Nansel

*Circulation*
Jeff Sandys

*Contributors*
Marvin Green, Lance Keizer
WilliamHarrison, Karl Lunt



## Contacts

**Membership $12 per year, February to February.** Backissues are $2 for one, $1.50 each additional issue in US & Canada, $3 for one, $2 each additional for international orders. Make your check in US funds payable to:

**Jeff Sandys, Treasurer**
**Seattle Robotics Society**
**P.O. Box 30668**
**Seattle, WA 98103-0668**

The Editor can be reached at:

**Robert Nansel @ SRS BBS**
**206-362-5267, 1200/2400 8N1, 24 Hr**

To submit hardcopy or disks:

**Bob Nansel, Editor**
**816 N. 105**
**Seattle, WA 98133**

Seattle Robotics Society
PO Box 30668
Seattle, WA 98103-0668

Place
Stamp
Here

**What's Inside #18:**



# In Upcoming Encoders...

While we are always looking for general robotics related articles, schematics, mechanical drawings and code examples, the following issues we'll try to focus on the specific topics listed below. Please note the deadlines for submission. If you have article ideas, drop the editor a line.

**March '93**
Deadline: 2/20/93
Hardware: Ultrasonic sensors
Software: Range mapping

**April '93**
Deadline: 3/20/93
Hardware: ROBI Controllers
Software: Path-Planning

**May '93**
Deadline: 4/17/93
Hardware: Shaft Encoders
Software: Dead Reckoning

**June '93**
Deadline: 5/15/93
Hardware: Stepper Motors
Software: Stepper algorithms