



Column #58, February 2000 by Lon Glazner:

Motor Control Made Easy

Back in August 1999, I wrote an article on the design of, and interface to, a high current H-bridge driver. The H-bridge is a configuration of transistors, which is often used for motor control applications. That design had a couple of drawbacks when used in conjunction with a BASIC Stamp. The most significant drawback was that the BASIC Stamp would have to devote all of its attention to controlling the H-bridge if varying motor speeds were desired.

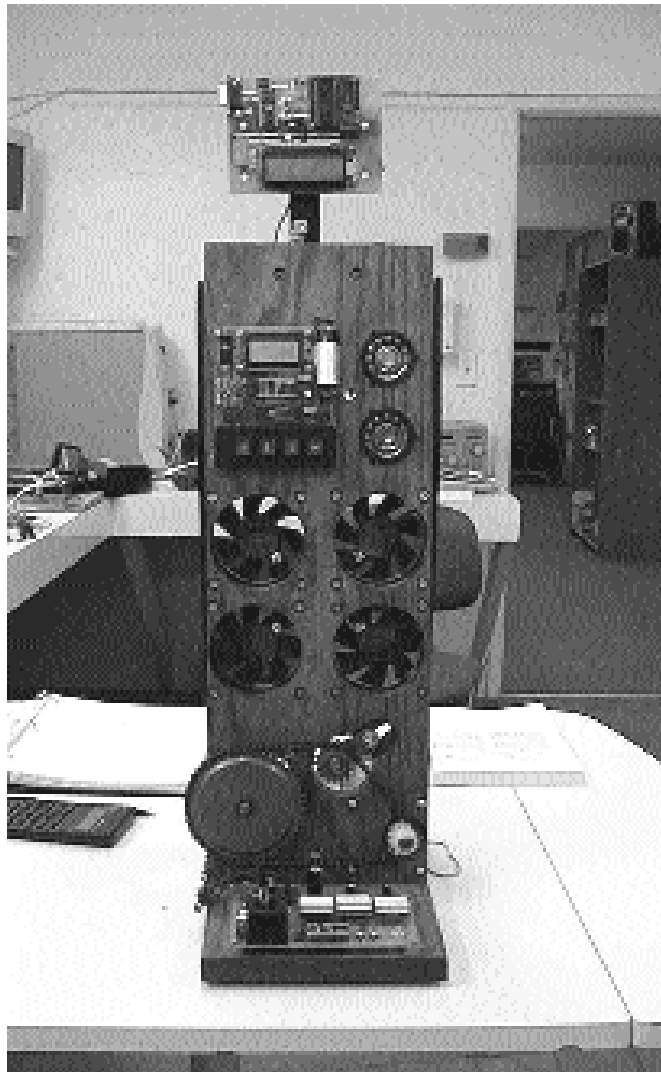
A pulse-width-modulated (PWM) signal is typically used to control the amount of power provided by an H-bridge to its load (in this case, a motor). The amount of time that the H-bridge is “on” can be related to the “duty-cycle” of the PWM signal. A 50% duty-cycle correlates to the H-bridge providing half-power to the load (0% relates to no power at the load, and 100% is full power). By adjusting the duty cycle up or down, you can adjust the speed of a motor.

Using a BASIC Stamp to vary the speed of the motor directly doesn’t allow your Stamp enough time to do anything else of much use, such as receive inputs from users to modify motor speed. For smaller motors (under 2A continuous), the Motor Mind B can fill in as both the H-bridge and the motor speed controller.

Column #58: Motor Control Made Easy

Prior to COMDEX in '98, I designed a BASIC Stamp-based motor control board as a demonstration of the Motor Mind B's capabilities. This system had one BASIC Stamp 2 (BS2) controlling three motor assemblies in conjunction with three Motor Mind B modules.

Figure 58.1: Gronkulator



Defining the Design

When I started thinking of a demo for the Motor Mind B, I wanted to touch on as many of the common motor control requirements as I could. I ended up creating three separate motor control functions that displayed some very common motor control applications which could be implemented with the BASIC Stamp and the Motor Mind B.

The entire demonstration came to be affectionately called the “Gronkulator” (Figure 58.1) For those of you interested in controlling motors with the BASIC Stamp, there may be some information of use to you in this article.

Function 1: Geared 24VDC Motor

The largest motor on the Gronkulator is a 24VDC 1.5A heavily geared motor. Motors of this size, and especially those with gearing, can easily exceed the Motor Mind B’s maximum current rating (2A continuous, 3.5A spikes for very short periods). The primary danger to the Motor Mind B, or any other H-bridge device, occurs during fast stops, starts, or during reversing. Geared motors with heavy loads such as the one discussed here can easily zap controlling electronics into silicon slag.

There are a few simple techniques that can be used to prevent damage from occurring to your Motor Mind B based system. For instance, by utilizing the /BRAKE pin on the Motor Mind B, or slowing responses to speed changes introduced by a user, the voltage/current spikes caused by geared motors can be reduced to levels which do not present a danger to the controlling electronics.

This motor control example provides a road map to controlling a heavily geared motor — with a substantial load — via simple joystick input. Motor speed and direction are easily controlled by user input.

Function 2: Self-Regulated Motor Speed

There are some instances where you would like to have your motor speed controlled by an outside stimulus. This can be accomplished with the SPDCON (speed control) command available in the Motor Mind B. In this example, the BS2 tells the Motor Mind B to adjust the motor speed based on a frequency at the Motor Mind B’s TACH (tachometer input). The BS2 sends a desired tachometer frequency to the Motor Mind B during program initialization. The Motor Mind B then adjusts the power to the motor up or down until the frequency input at the tachometer pin matches that of the desired frequency as set by the BS2.

Column #58: Motor Control Made Easy

This control method can be implemented with motors that have tachometer, or encoder, output signals. In this instance, however, a somewhat different concept has been implemented. This design has the Motor Mind B controlling four 24VDC fans wired in parallel. A light sensor that outputs a frequency feeds the tachometer input. Adjusting the amount of light reaching the sensor adjusts the speed of the fans. Blocking the light with your finger, or causing a shadow to cover the sensor, adjusts the fan speed. This simulates a temperature-to-frequency conversion circuit that might be used to control fans in an enclosure used by the BS2. Very little program space of the BS2 is required to implement this method of control.

Function 3: Reed Switches and Speed Control

The third motor control function displayed by the Gronkulator introduces position feedback and speed control through the BS2 to the Motor Mind B. This hardware and software combination was used to raise and lower a LCD display located at the top of the Gronkulator. The reed switches determine the upper and lower stop positions for the arm that the LCD display is attached to. A magnet mounted on a stationary portion of this motion system closes each reed switch when it comes into close proximity to the magnet. The switch closure is detected by the BS2 and the motor direction is reversed. There are additionally two user-input signals that allow the user to speed up or slow down the movement of the arm carrying the LCD.

The Hardware

There was quite a bit of customized hardware used in the Gronkulator. All of this hardware was ordered from the surplus company C&H Sales (www.candhsales.com, 1-800-325-9465). I would recommend the C&H catalog to all Stamp hobbyists who are interested in interfacing to mechanical systems. They offer a wide variety of surplus motors, gears, and pulleys.

The schematic for the motion control aspect of the Gronkulator shows the simplicity that the Motor Mind B contributes to motor control.

Data sheets for the Motor Mind B can be downloaded at www.solutions-cubed.com. The Motor Mind B is ideal for simple proportional motor control applications. While it could be used for PID systems, the response time of the serial communication interface, the number of speed steps (255), and the slow PWM frequency, really limit it to motor systems which do not need fast response times. In other words, the Motor Mind B is inadequate for use as a DC motor position controller where a lot of resolution is required (precise positioning is not possible).

Figure 58.3: A close-up of the geared motor assembly

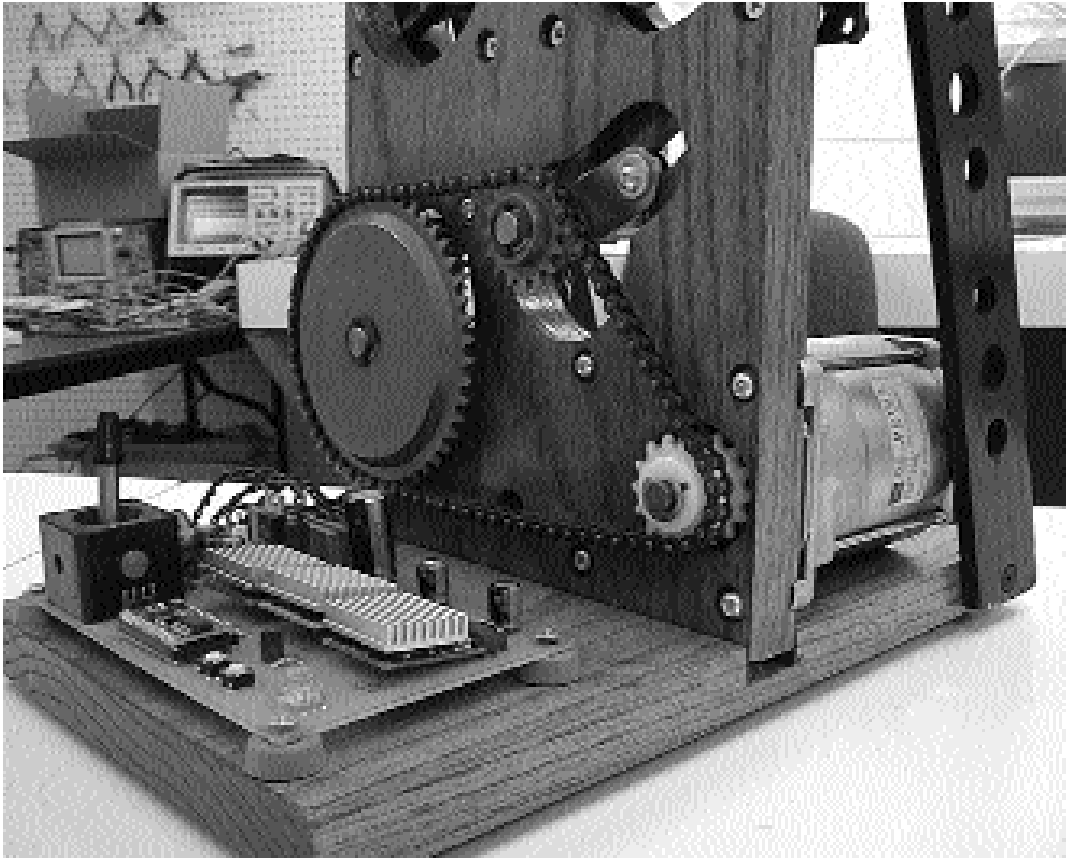
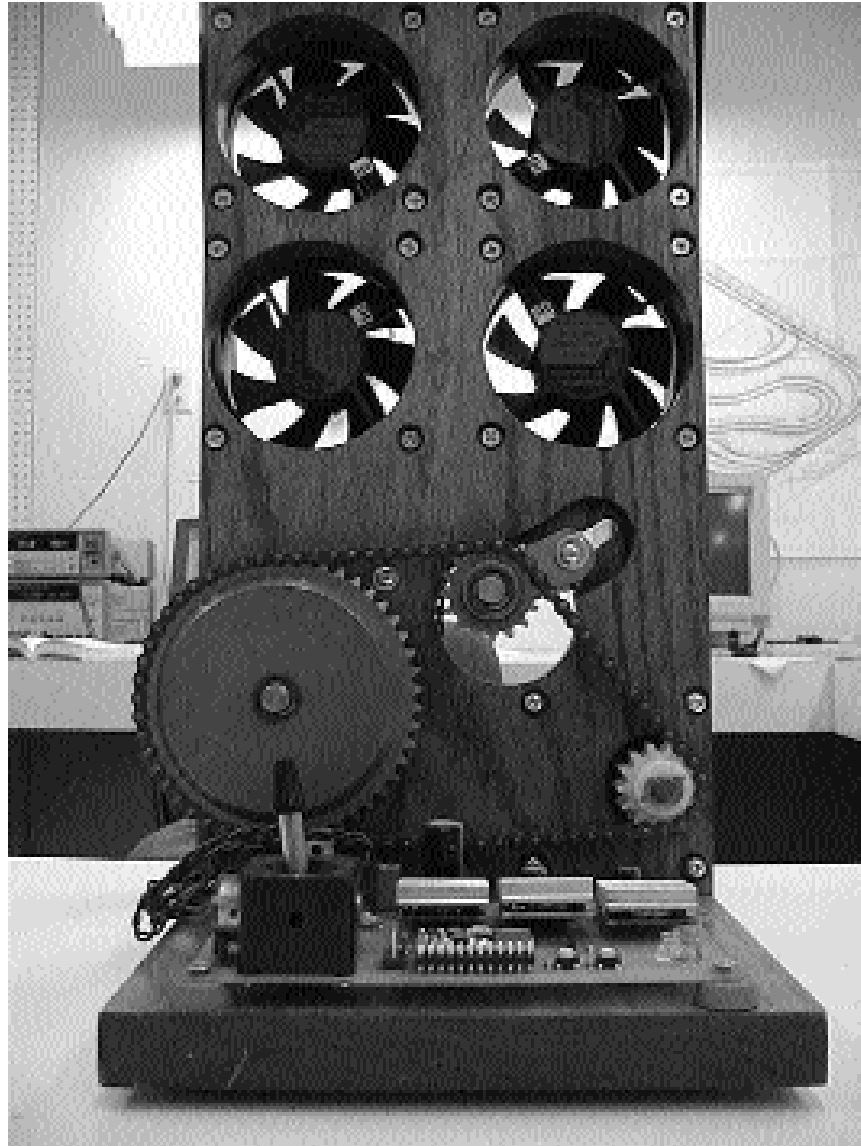


Figure 58.4: The Gronkulator wind tunnel and gear system



The Software

The software for the motor control applications that were used for this demonstration project is pretty simple. The only aspect of the software that requires much in the way of explanation is the MotorControl1 subroutine.

The hardware was designed to allow connections to both of the potentiometers available in the joystick that was purchased from C&H Sales. In the actual application, I only make use of one of these joysticks for speed and direction control of the geared motor (Motor 1). The RCTIME function was used to measure the position of the joystick. Any measurement greater than \$0850 (decimal 2128) meant that the motor was supposed to be running in the forward direction. If the RCTIME joystick value is less than \$0750 (decimal 1872), then the motor is supposed to be turning backwards. Any value in between \$0750-\$0850 implements a STOP command.

The motor speed is also calculated from the joystick values received from the RCTIME measurements. Prior to sending the new speed to the Motor Mind B, the BS2 requests a STATUS update from the Motor Mind B. The STATUS command causes the Motor Mind B to return its current speed and direction to the Master device (BS2, in this case). The STATUS direction information is used to determine if the current speed and direction calculated from the joystick measurement is the opposite of that being implemented by the Motor Mind B. If this is the case, then instead of implementing the speed and direction from the joystick, the BS2 implements a direction reversal.

When reversing the motor, the BS2 first pulls the /BRAKE pin of the Motor Mind B low, which causes the H-bridge on the Motor Mind B to shut down. It is not necessary to use the /BRAKE when stopping or reversing all motors. But asserting the /BRAKE pin low is the safest way to stop a geared motor and prevent damage to the Motor Mind B or the BS2.

While the /BRAKE pin stops the H-bridge from driving the motor, it does not stop the Motor Mind B from generating a PWM signal to drive the H-bridge (the Motor Mind B, which is a module, consists of both a controller chip, and an H-bridge IC, on a PCB). For this reason, after asserting the /BRAKE pin, the BS2 sends a STOP command to the Motor Mind B which minimizes the duty-cycle generated by the on-board controller to drive the H-bridge IC. When the /BRAKE pin is driven high again, the motor will essentially be stopped.

In addition to the manipulation of the /BRAKE pin and sending the STOP command, the BS2 adds about 200ms of delay time to allow the geared motor to completely stop before the program drives the /BRAKE pin high, and then proceeds. This allows enough time for the mechanical motor to slow down and stop.

The second motor control module in this software occurs in the initialization portion of the program. This occurs when the BS2 sends a SPDCON command to a second Motor Mind B at the start of the program. The SPDCON command sets the desired frequency at the tachometer pin to 18,000Hz.

In addition to setting the desired frequency, the SPDCON command is used to set the “gate-time” of the tachometer function in the Motor Mind B. The gate-time allows the user to optimize the update rate of the speed control function versus the resolution of the tachometer. The slower the update rate of the speed control, the better the resolution of the tachometer. A more detailed account of this functionality can be found in the Motor Mind B data sheet.

Finally, the linear motion device that raises and lowers the LCD display is controlled by a third Motor Mind B via the BS2. The MotorControl3 subroutine tests the reed switch inputs (which close when they are in the proximity of a magnet) and two push button switches which can be used to increase or decrease the motor speed.

If a reed switch is closed, then the BS2 program stops the motor, reverses the motor, and sends the previous speed setting to the Motor Mind B. The program then waits until the reed switch has opened again before exiting the subroutine. Waiting for the reed switch to open prevents the BS2 from re-entering the subroutine and reversing the motor over-and-over again without ever allowing the reed switch to clear the magnet.

If the user-accessible speed control push buttons are pressed, the BS2 will increase or decrease the speed that will be sent to the Motor Mind B. This increases or decreases the rate at which a LCD is raised and lowered. Minimum and maximum speed values are also established to prevent the user from rolling over the byte-sized value that is used to control motor speed.

All things considered, the small amount of software used in this demonstration design adequately display the versatility in BS2 motion control when used in conjunction with a Motor Mind B.

In Closing

There were a few reasons I wanted to write this article on the Gronkulator. But first and foremost is the creativity required, and enjoyment received, in interfacing electronics to mechanical systems. I think that the enjoyment Stamp enthusiasts receive from “making stuff move” shows in the number of Stamp-based robotics projects available from Parallax, and other companies today (like Mondo-tronics; www.RobotStore.com).

Even though the Gronkulator is a couple of years old, I still like to power it up for visitors and allow them to control the various motors and fans, as well as use the talking clock (which wasn't discussed here). I still spend plenty of time thumbing through surplus catalogs searching out funky mechanical systems that beg for a BASIC Stamp controller.

Whether it's via the Motor Mind B or your average RC hobby servo, making mechanical systems has allowed me to explore a side of engineering that I don't typically get involved in. These excursions into mechanical systems have been real eye-openers and have provided me with the kind of fun that can be a catalyst for self-education. If you have the opportunity, find an old box of electro-mechanical junk and make something move. Your Stamp projects will never be the same afterwards.

```
'Program Listing 58.1: MMBDemo.BS2

'MMBDemo.BS2
'
Input          0
Output         2
Input          11
Input          11
Input          12
Input          14
Input          15
'
MotorPin       VAR    Byte
SpeedMot1     VAR    Byte
SpeedMot3     VAR    Byte
Function      VAR    Byte
Speed         VAR    Byte
PotPin        VAR    Byte
Direction     VAR    Byte
Temp          VAR    Byte
PotResult     VAR    Word
'
StopMotor     CON    $00
ReverseMotor  CON    $01
StatusCheck   CON    $05
Motor1        CON    10
Motor2        CON    9
Motor3        CON    8
BrakeMot1     CON    2
PotA          CON    12
PotB          CON    11
FastSW        CON    15
SlowSW        CON    14
MaxSW         CON    0
MinSW         CON    1
'*****
Initialize:
    HIGH      BrakeMot1
    PAUSE     1000
    SpeedMot3 = $80
    LOW       Motor2
    PAUSE     500
    HIGH      Motor2
    PAUSE     100
    SEROUT   Motor2,396,[$55,$04,$46,$50,$01]
    GOTO     MainProgram
'*****
MotorSpeed:
    DEBUG    "Speed = ",ISHEX2 Speed,CR
    SEROUT   MotorPin,396,[$55,$03,Speed]
```

Column #58: Motor Control Made Easy

```
RETURN
'*****
MotorFunction:
    SEROUT MotorPin,396,[$55,Function]
    RETURN
'*****
MeasurePot:
    HIGH    PotPin
    PAUSE   1
    RCTIME  PotPin,1,PotResult
    RETURN
'*****
MotorControll:
    MotorPin = Motor1
    PotPin = PotA
    GOSUB MeasurePot
    '
    debug "potA ",ISHEX4 PotResult,cr
    If PotResult > $0850 then Forward
    If PotResult < $0750 then Backward
    LOW    BrakeMot1
    Function = StopMotor
    GOSUB MotorFunction
    RETURN
Forward:
    HIGH    BrakeMot1
    SEROUT MotorPin,396,[$55,$05]
    SERIN  MotorPin,396,100,Forward,[Direction,Temp]
    If Direction <> 0 Then ReverseDirection
        PotResult = (PotResult - $0850)/3
        If PotResult > $FF then UpperLimit
        Speed = PotResult
        Gosub MotorSpeed
    RETURN
Backward:
    HIGH    BrakeMot1
    SEROUT MotorPin,396,[$55,$05]
    SERIN  MotorPin,396,100,Backward,[Direction,Temp]
    If Direction <> 1 Then ReverseDirection
        PotResult = ($750 - PotResult)/3
        If PotResult > $FF then UpperLimit
        Speed = PotResult
        Gosub MotorSpeed
    RETURN
UpperLimit:
    Speed = $FF
    GOSUB MotorSpeed
    RETURN
ReverseDirection:
    LOW    BrakeMot1
    Function = StopMotor
    GOSUB MotorFunction
```

```

    PAUSE 100
    Function = ReverseMotor
    GOSUB MotorFunction
    PAUSE 100
    HIGH BrakeMot1
    RETURN
'*****
MotorControl3:
    MotorPin = Motor3
    If IN0 = 0 then ChangeDirection0
    IF IN1 = 0 then ChangeDirection1
    IF IN14 = 0 then ReduceMot3
    IF IN15 = 0 then IncreaseMot3
    GOTO UpdateMot3
ReduceMot3:
    If SpeedMot3 < $0A then UpdateMot3
    SpeedMot3 = SpeedMot3 - 10
    GOTO UpdateMot3
IncreaseMot3:
    If SpeedMot3 > $F6 then UpdateMot3
    SpeedMot3 = SpeedMot3 + 10
UpdateMot3:
    Speed = SpeedMot3
    GOSUB MotorSpeed
    RETURN
ChangeDirection0:
    Function = StopMotor
    GOSUB MotorFunction
    PAUSE 50
    Function = ReverseMotor
    GOSUB MotorFunction
    Speed = SpeedMot3
    GOSUB MotorSpeed
WaitForHigh0:
    If IN0 = 0 then WaitForHigh0
    RETURN
ChangeDirection1:
    Function = StopMotor
    GOSUB MotorFunction
    PAUSE 50
    Function = ReverseMotor
    GOSUB MotorFunction
    Speed = SpeedMot3
    GOSUB MotorSpeed
WaitForHigh1:
    If IN1 = 0 then WaitForHigh1
    RETURN
'*****
MainProgram:
    GOSUB MotorControl1

```

Column #58: Motor Control Made Easy

```
GOSUB MotorControl3  
goto MainProgram
```