This is Revision F. ***If you downloaded this from Chief Delphi, please check to see if there is a more recent revision available.*** *The later revisions are toward the* ***bottom*** *of the list* 2/10/2013 Ether

Bang-Bang Speed Controller for Shooter Wheel

- ✔ Simpler and more robust than PID (for this application); requires no tuning; provides fastest spin-up and recovery time.
- ✔ Works with Victor (and probably Talon[1]) motor controller. Jag should be avoided for this application[2].
- ✔ Motor Controller should be jumpered for **coast** mode (**not** brake mode).
- ✔ A wheel speed sensor is required. You can use any device which provides a usable speed signal
- ✔ This method requires that the wheel has sufficient moment of inertia (many shooter wheels do)

1 I believe the Talon should work, but its locked antiphase switching method has not yet been tested (to my knowledge as of this date) in this particular application.

2 See Appendix A for explanation

This is Revision F. ***If you downloaded this from Chief Delphi, please check to see if there is a more recent revision available.*** *The later revisions are toward the* ***bottom*** *of the list* 2/10/2013 Ether

PseudoCode for the Bang-Bang Controller

It's really this simple:

```
if (measured_RPM > target_RPM) motor_command = 0;

else motor_command = 1;
```

… where:

```
"measured_RPM" is the decoded RPM signal from the wheel speed sensor.  Don't
filter this signal going to the bang-bang control.

"target_RPM" is the desired speed (setpoint).

"motor_command" is the output from the bang-bang control and is to be sent to
the motor controller:  0 means neutral (with coast mode, not brake); 1 means
100% forward voltage.  Don't voltage-ramp or filter this motor command.
```

For best control, run the bang-bang controller at 20ms... or faster if you can afford to do so.

This is Revision F. ***If you downloaded this from Chief Delphi, please check to see if there is a more recent revision available.*** *The later revisions are toward the* ***bottom*** *of the list* 2/10/2013 Ether

Speed Sensor and Decoding

- ✔ Use an encoder or a one-per-rev home-brew optical or magnetic sensor.
- ✔ If using an encoder, connect only one channel of the encoder to the Digital Sidecar. Configure FPGA to read one channel only (no quadrature)
- ✔ Configure the FPGA to read rising edges only
- ✔ Let the FPGA compute the period in hardware with its 153KHz polling rate and 1MHz clock.
- ✔ Do a quick calculation to determine how many samples you should configure the FPGA to use for its sample ring buffer, or experiment to find the value which gives the best tradeoff between noise and phase lag (whatever gives you fast and stable operation at your setpoints).

Ask on CD if any of the above bullet points are unclear or you need further instructions how to do them.

This is Revision F. ***If you downloaded this from Chief Delphi, please check to see if there is a more recent revision available.*** *The later revisions are toward the* ***bottom*** *of the list* 2/10/2013 Ether

Appendix A: Special Steps for Using the Jag

You can make the Jag work, but you have to fuss with it a bit. Here's the problem:

> The Jag overcurrent protection may activate on initial startup due to high command at low rpm. This can be worked around by adding a little additional code and doing some tuning, but for this application it's more convenient and less fuss to use a motor controller that doesn't have this limitation.

If a Jag is used:

> Disable the Jag's "automatic voltage ramping". The ramping interferes with the bang-bang action at the setpoint. Instead limit the motor command when the motor speed is below a certain threshold, as shown below.
>
> Don't use a low-pass filter or a slew-rate limiter on the bang-bang output command.
>
> Don't plug the encoder into the Jag: the CAN bus limits the speed at which you can read the speed and send the new command to the motors. Bang-bang does not like such delays.

Add an extra line of code to the bang-bang:

```
if (measured_RPM >= target_RPM) motor_command = 0.0;
else if (measured_RPM >= spinup_RPM) motor_command = 1.0;
else motor_command = low_command;
```

… and then tune the two parameters:

"**low_command**" should be tuned to the ***highest*** value (0.0 to 1.0) that can be commanded at zero rpm without causing the motor controller to shut down.

"**spinup_RPM**" should be tuned to the ***lowest*** speed above which full power can be commanded without causing the Jag to shut down.