



Motion Mind  
Motor Controller Data Sheet  
Revision 6  
October 27<sup>th</sup>, 2005

**Always visit [www.solutions-cubed.com](http://www.solutions-cubed.com)  
for the most up-to-date documents and software**



**SOLUTIONS CUBED, LLC**  
256 East First Street Chico, CA 95928  
phone: 530.891.8045 fax: 530.891.1643  
[www.solutions-cubed.com](http://www.solutions-cubed.com)

Copyright © 2005 Solutions Cubed, LLC

## Table of Contents

<b>1.0</b>	<b>Revision Log</b>	<b>3</b>
<b>2.0</b>	<b>Introduction</b>	<b>4</b>
2.1	Description	4
<b>3.0</b>	<b>Electrical – Mechanical – Functional Descriptions</b>	<b>5</b>
3.1	Absolute Maximum Ratings	5
3.2	DC Electrical Characteristics	5
3.3	AC Electrical Characteristics	6
3.4	Mechanical Dimensions	7
3.5	Connectivity Overview	8
3.6	Indicator LEDs	9
3.7	Input Pin Considerations	10
3.8	Two's Compliment Numbering System	10
3.9	Quadrature Encoder Interface	11
3.10	The PID Filter	11
3.11	Tuning The PID Filter	13
3.12	Velocity Measurements and Closed-Loop Control	13
<b>4.0</b>	<b>Operating Modes</b>	<b>14</b>
4.1	Overview	14
4.2	Getting Started	14
4.3	Mode 0: R/C Signal Control	16
4.4	Mode 1: Analog Control	20
4.5	Mode 2: Button Control	21
4.6	Mode 3: Serial Control	23
4.7	Mode 4: Serial PID Control (Position)	25
4.8	Mode 5: Analog PID Control(Position)	27
4.9	Mode 6: Factory Restore	29
4.10	Mode 7: Factory Test	29
<b>5.0</b>	<b>Communication Protocol</b>	<b>30</b>
5.1	Overview	30
5.2	Response to Commands	31
5.3	CHANGE_SPEED Command	32
5.4	MOVETO_ABSOLUTE Command	33
5.5	MOVETO_RELATIVE Command	33
5.6	MOVEAT_VELOCITY Command	34
5.7	WRITE Command	35
5.8	WRITE_STORE Command	36
5.9	READ Command	37
5.10	RESTORE Command	40
5.11	RESET Command	40
5.12	SELF_TEST Command	40
<b>6.0</b>	<b>Register Definitions – Function/Status Bits</b>	<b>41</b>
6.1	Overview	41
6.2	Register Descriptions	42
6.3	FUNCTION Register Bits	44
6.4	STATUS Register Bits	45
<b>7.0</b>	<b>Disclaimer / Warrantee</b>	<b>46</b>
<b>8.0</b>	<b>Errata Sheet</b>	<b>47</b>

## List of Figures

Figure 1: Mechanical Dimensions	7
Figure 2: Baud and Communication Jumpers	8
Figure 3: Mode Jumpers	9
Figure 4: Analog and Encoder Input Circuits	10
Figure 5: Basic Serial Connections	15
Figure 6: R/C Pulse Timing	16
Figure 7: R/C Mode Connections	17
Figure 8: Analog Mode Connections	20
Figure 9: Button Mode Connections	22
Figure 10: Serial (Open-Loop) Mode Connections	24
Figure 11: Serial PID Control (Closed-Loop) Mode Connections	26
Figure 12: Analog PID Control (Closed-Loop) Mode Connections	29
Figure 13: Register Index and Format Table	41
Figure 14: Register Descriptions Table	42
Figure 15: FUNCTION Register Bits	44
Figure 16: STATUS Register Bits	45

## 1.0 Revision Log

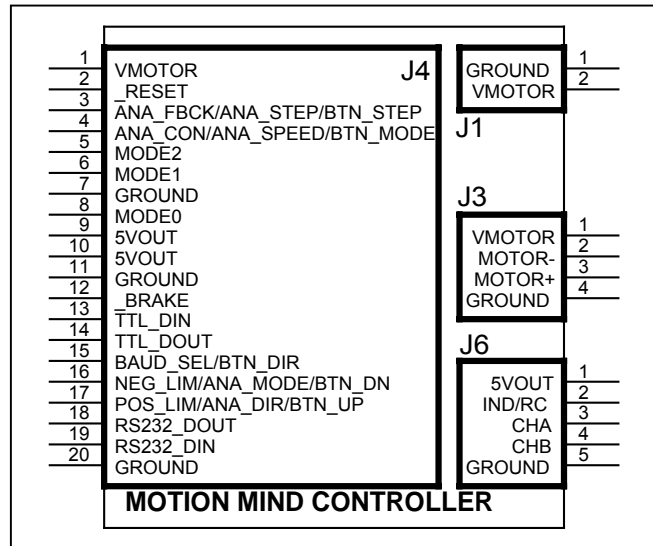
Date	Rev	Description	By
01-05	1	Original Implementation	L. Glazner
05-05	2	Changed reference to VELOCITTYLIMIT register values from 1 to 65535 to 1 to 1023. Added information about FUNCTION.AD SERIAL and FUNCTION.ENABLEDB modes added to REV2 firmware.	L. Glazner
06-05	3	Updated ERRATA to include bug related to mode 4 save position functions and the use of the velocity limit mode	L. Glazner
06-05	4	Updated ERRATA, to include limit switch function additions, and ASCII bad command response with bad address, added sections related to mode 0 RC position control with analog feedback, fixed document wide errors related to J4 P16 and J4 P17 functionality.	L. Glazner
08-05	5	Fixed analog settings based on ANA_MODE (J4 P16) state, reduced operating voltage	L. Glazner
10-05	6	Added comments related to REV5 firmware register and mode updates, including PWMLIMIT, VNLIMIT, VPLIMIT, DEADBAND, DESIREDPOSITION, FUNCTION.VIRTLIMIT, FUNCTION.RCPOS-ENCFDBCK, FUNCTION.ACTIVESTOP, and removal of FUNCTION.HEXMODE.	L. Glazner

## 2.0 Introduction

### Motion Mind Open Loop / Closed Loop DC Motor Controller

#### FEATURES

- ◆ Up to 9A continuous current (25A peak), 6-24VDC brushed motors
- ◆ Easy connectivity to motor, encoder, and master unit (if used)
- ◆ Open loop analog, button, serial, R/C control modes
- ◆ Binary or ASCII control interface, RS232 or TTL signal levels, 19.2KBPS or 9.6KBPS
- ◆ Analog PID based closed-loop position control mode (10 bit resolution – analog feedback)
- ◆ Serial PID based closed loop velocity control mode (16 bit – requires encoder)
- ◆ Serial PID based closed loop position control mode (32 bit – requires encoder)
- ◆ Brake, reset, negative limit switch, and positive limit switch inputs (limits on some modes only)
- ◆ Built in over-temperature, over-current, over-voltage, under-voltage protection
- ◆ Many operating parameters configured through external settings (jumpers, logic inputs, analog inputs)
- ◆ Freeware communication software available for evaluation and programming



#### 2.1 DESCRIPTION

The Motion Mind DC Motor Controller is capable of controlling one brushed DC motor. A variety of control methods are supported including open loop and closed loop control.

Open loop control methods do not require feedback from an encoder and can operate with serial data input, button presses, analog control signals (0-5V), and pulse control from a hobby R/C type control signal (1-2ms pulses, approximately every 20ms).

Closed control requires either a quadrature encoder (2 or 3 channel) or an analog feedback signal (0-5V). When operating in closed loop modes, position and/or velocity control are possible. Some limit switch functionality is supported through negative and positive limit switch inputs. The state of the index input from a three-channel encoder can be monitored and the approximate position of the last index pulse present is available.

In addition to a variety of control techniques, the Motion Mind module has numerous communication settings available to the user. A high-speed binary communication protocol is implemented with an external jumper determining the baud rate (19.2KBPS or 9.6KBPS). The binary communication protocol may be replaced with a simpler ASCII command set through an external jumper. The ASCII protocol ensures ease of programming with simple terminal programs and also operates at both baud rates.

Electrically the Motion Mind can communicate at RS232 levels through an on-board translator IC, or through a TTL serial interface with an open-collector output that allows multiple modules to share the same data bus.

The board measures 2.4" wide x 3.6" long x 0.8" tall. The Module is powered directly off of the motor voltage and may provide 200mA of regulated 5VDC voltage to other off board devices.

## 3.0 Electrical - Mechanical – Functional Descriptions

### 3.1 Absolute Maximum Ratings

*These are stress ratings only. Stresses above those listed below may cause permanent damage and/or affect device reliability. The operational ratings should be used to determine applicable ranges of operation.*

Storage Temperature	-50°C to +150°C
Operating Temperature	0°C to +70°C
Motor Voltage (VM)	+6V to +36.0V
Voltage on logic control pins	-0.3V to +5.5V
Voltage on CHA, CHB, pins	-0.3V to +5.3V
Voltage on VM, M+, M-	40V transient spike
Motor Current Load	25A peak / 9A continuous

### 3.2 DC Electrical Characteristics

Characteristic	Min	Typ	Max	Unit	Notes
Supply Voltage	6		24	V	VMOTOR voltage
Supply Current		25		mA	VMOTOR = 12V, no other connections
5VOUT Voltage	4.9		5.1	V	
5VOUT Current Source Capability		200		mA	Up to 200mA may be used to power external 5V circuits
ANALOG_IN voltage range	0		5	V	5VOUT is the full-scale input for the 10-bit ADC used in analog control modes
ADC resolution		4.88		mV	Per ADC bit
CHA-B voltage	-0.3		+5.3	V	CHA-B pins pulled to +5V with 100kΩ resistors
Peak load current			25	A	
Max continuous motor current	7	9	10	A	Continuous current rating at room temp. 95% duty-cycle, resistive load
Low Level Input RX pin			0.5	V	RX pin pulled to +5V with 100kΩ resistor
High Level Input RX pin	2.0			V	RX pin pulled to +5V with 100kΩ resistor
Low Level Input _BRAKE pin			0.5	V	_BRAKE pin pulled to +5V with 100kΩ resistor
High Level Input _BRAKE pin	2.0			V	_BRAKE pin pulled to +5V with 100kΩ resistor

note: "Typ" values are for design guidance only and are not guaranteed

**DC Electrical Characteristics (continued)**

Characteristic	Min	Typ	Max	Unit	Notes
Low Level Input _RESET pin			0.5	V	_RESET pin pulled to +5V with 10kΩ resistor
High Level Input _RESET pin	2.0			V	_RESET pin pulled to +5V with 10kΩ resistor
Low Level Input CHA-B pins			0.8	V	CHA-B pins pulled to +5V with 100kΩ resistors
High Level Input CHA-B pins	3.5			V	CHA-B pins pulled to +5V with 100kΩ resistors
Low Level Input LIMIT switch pins			0.5	V	Limit switch pins pulled to +5V with 12.5kΩ res.
High Level Input LIMIT switch pins	2.0			V	Limit switch pins pulled to +5V with 12.5kΩ res.
Low Level Output TX pin			0.6	V	TX pin pulled to +5V with 10kΩ resistor
High Level Output TX pin	3.8		4.8	V	TX pin pulled to +5V with 10kΩ resistor
Max Analog Source Impedance			2.5	kΩ	Buffer signals with greater than this impedance (such as a 10kΩ potentiometer)

note: "Typ" values are for design guidance only and are not guaranteed

**3.3 AC Electrical Characteristics**

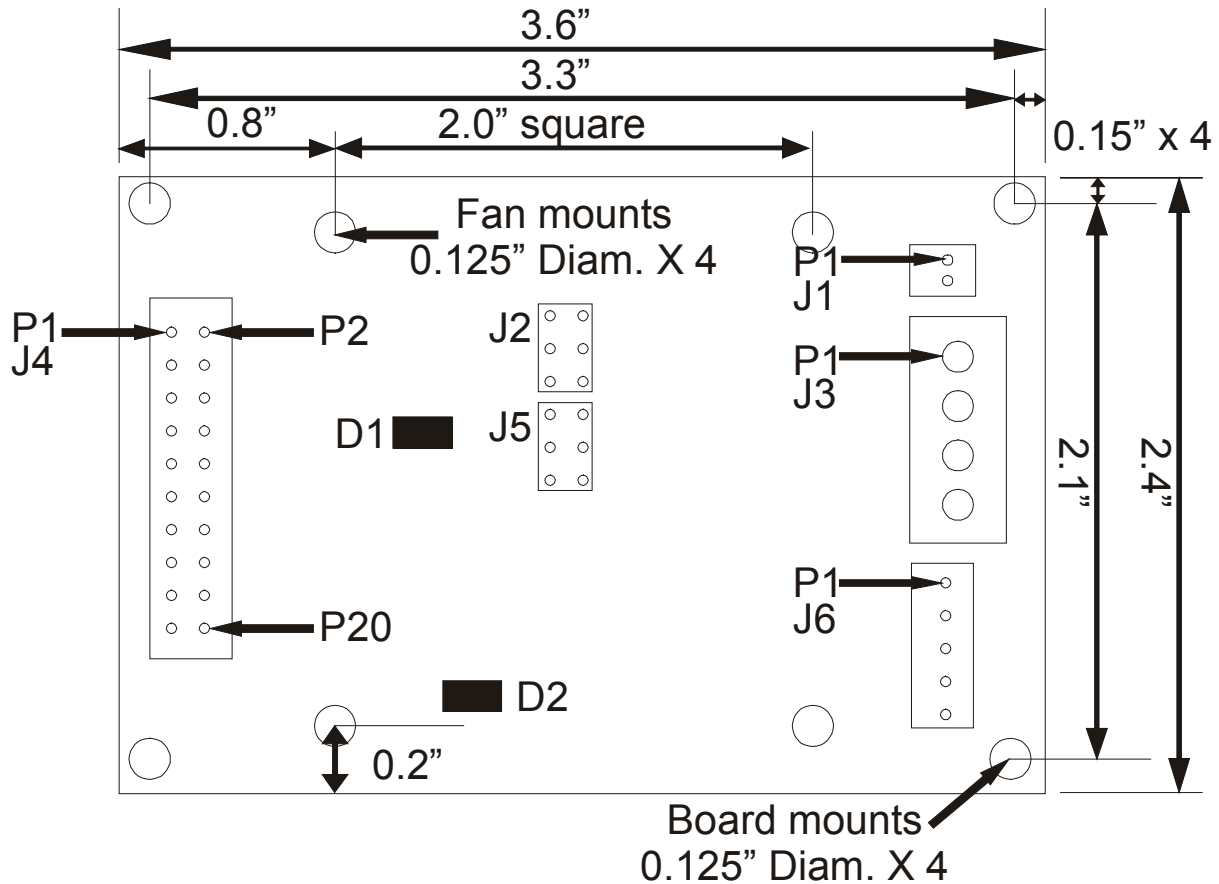
Characteristic	Min	Typ	Max	Unit	Notes
Time for a command to be responded to		5	40	mS	Commands that write to EEPROM take longer to implement
V <sub>m</sub> rise time to ensure good reset	0.05			V/mS	If this condition is not met then microcontroller may not power up correctly
PWM update rate		200		Updates/S	
PWM frequency		38.4		KHz	
Encoder Frequency		1.5	1.75	MHz	
Valid R/C Pulse Width	0.5		2.5	mS	RCMIN and RC MAX may be adjusted via the serial interface
R/C Pulse Period		50		mS	An R/C pulse must be received within 50ms of the previous pulse
Encoder Frequency		100		KHz	

note: "Typ" values are for design guidance only and are not guaranteed

### 3.4 Mechanical Dimensions

The following diagram may be used to develop PCB carrying boards or enclosures used to fit the controller into custom designs. Cooling fans with a 2"-square mounting footprint may be attached with the 4-40 (0.125" diameter) mounting holes using 0.75" stand-offs. Cooling fans may be powered through the connection at J1. The board requires approximately 0.8" height clearance.

Figure 1: Mechanical Dimensions



Mechanical Landmark Descriptions

Landmark	Type	Description
D1	LED – green	PWR- 5V regulator power indicator
D2	LED – green	COMM- Indicates mode on power-up and flashes when serial data is accepted
J1	1x2 0.1" locking header	Fan or accessory connection provides access point for unfiltered motor voltage
J2	2x3 header	Baud select and ASCII/Binary protocol mode select jumpers
J3	1x4 screw terminal	DC motor connections for motor and motor supply
J4	20 pin 0.1" shrouded header	Master control unit interface connection, provides for insertion of serial data, brake, analog control, limit switches and/or button switches
J5	2x3 header	Mode select jumpers
J6	1x6 0.1" locking header	Connects to quadrature encoders, or R/C signal input, may also be used as connection to 5VOUT and GND

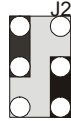
### 3.5 Connectivity Overview

#### J1 Locking Accessory Pin Descriptions

Pin	Name	Type	Description
1	<i>GND</i>	POWER	The supply voltage return connects to this pin, common to all other grounds
2	<i>VM</i>	POWER	The primary supply voltage connects to this pin

**J2 Baud and Communication Jumpers:** The baud jumper is also available J4 P15.

**Figure 2: Baud and Comm Jumpers**



9.6KBPS / ASCII  
Mode selected



19.2KBPS / TTL  
Mode selected

#### J3 Terminal Block Descriptions

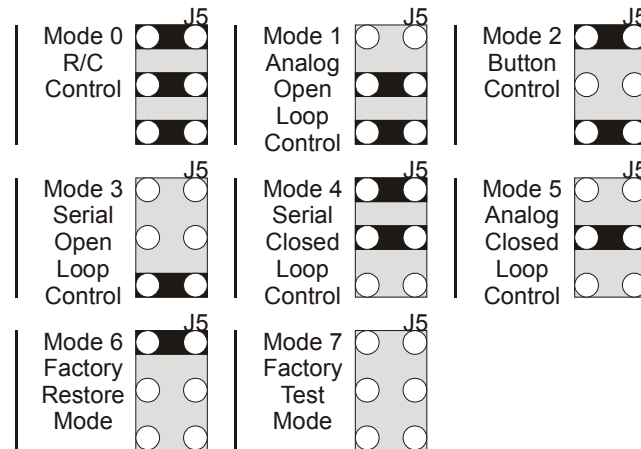
Terminal	Name	Type	Description
1	<i>VM</i>	POWER	The primary supply voltage connects to this terminal (36V maximum)
2	<i>M-</i>	POWER	The negative lead of the brushed DC motor, or other load, connects to this terminal
3	<i>M+</i>	POWER	The positive lead of the brushed DC motor, or other load, connects to this terminal
4	<i>GND</i>	POWER	The supply voltage return connects to this terminal

#### J4 Master Control Interface Pin Descriptions

Pin	Name	Type	Description
1	<i>VMOTOR</i>	POWER	Connected to the motor voltage at P1 of J3 and P2 of J1
2	<i>_RESET</i>	INPUT	Pulling <i>_RESET</i> low performs a hardware reset of controller this input is pulled to +5VDC with a 10kΩ resistor and tied to ground with a 0.1uF capacitor, leave unconnected if not used
3	<i>ANA_FBCK/ ANA_STEP/ BTN_STEP</i>	INPUT	0-5V analog input, max 2.5KΩ input impedance, 3.4KHz low pass filter
4	<i>ANA_CON/ ANA_SPEED/ BTN_MODE</i>	INPUT	0-5V analog input, max 2.5KΩ input impedance, 3.4KHz low pass filter
5	<i>MODE2</i>	INPUT	Input used to select operating mode, pulled to 5V with 12.5kΩ (min) resistor
6	<i>MODE1</i>	INPUT	Input used to select operating mode, pulled to 5V with 12.5kΩ (min) resistor
7	<i>GROUND</i>	POWER	Ground return
8	<i>MODE0</i>	INPUT	Input used to select operating mode, pulled to 5V with 12.5kΩ (min) resistor
9	<i>5VOUT</i>	POWER	+5VDC output, can supply up to 200mA, should be left unconnected if not needed
10	<i>5VOUT</i>	POWER	+5VDC output, can supply up to 200mA, should be left unconnected if not needed
11	<i>GROUND</i>	POWER	Ground return
12	<i>_BRAKE</i>	INPUT	Pulling <i>_BRAKE</i> low forces the H-bridge into an "off" state, is pulled to +5VDC with a 100kΩ resistor
13	<i>RX</i>	INPUT	TTL level, 8N1, serial reception pin (data from the Master unit), is pulled to +5VDC with a 100kΩ resistor
14	<i>TX</i>	OUTPUT	TTL level, 8N1, serial transmission pin (data to the Master unit), is pulled to +5VDC with a 100kΩ resistor
15	<i>BAUD_SEL/ BTN_DIR</i>	INPUT	Selects the baud rate for serial communication, is pulled to +5VDC with a 100kΩ resistor also determines direction motor spins in un-directional button mode
16	<i>NEG_LIM/ ANA_MODE/ BTN_DN</i>	INPUT	Negative going limit switch, when asserted movement in the negative direction is prevented, logic level for assertion is software configurable, this input is pulled to +5VDC by 12.5kΩ (min) resistor
17	<i>POS_LIM/ ANA_DIR/ BTN_UP</i>	INPUT	Positive going limit switch, when asserted movement in the positive direction is prevented, logic level for assertion is software configurable, this input is pulled to +5VDC by a 12.5kΩ (min) resistor
18	<i>RS232_TX</i>	OUTPUT	RS232 level serial output (for connection to RS232 serial port)
19	<i>RS232_RX</i>	INPUT	RS232 level serial input (for connection to RS232 serial port)
20	<i>GROUND</i>	POWER	Ground return

**J5 Operating mode hardware jumpers:** The J5 jumper settings configure the mode of operation. These connections can also be accessed at P5, P6, and P8 of J4. The mode settings can be changed “on-the-fly”. However, many modes of operation are not cross compatible. One example comes to mind; the R/C mode shares a timer with the closed loop serial control mode. If the user were to switch between these modes on the fly then the position and velocity data would be corrupted and the Motion Mind module would operate incorrectly.

**Figure 3: Mode Jumpers**



**J6 Encoder – R/C Receiver Connector:** A two or three channel quadrature encoder may be connected to J6 when operating in the closed loop serial control mode. In R/C mode this connector may be used to power an R/C receiver and the R/C pulse signal may be connected to IND/RC (J6 P2).

**J6 Locking Encoder Connector Pin Descriptions**

Pin	Name	Type	Description
1	5V	POWER	This terminal can be used to power an incremental encoder, the external circuitry should draw less than 200mA
2	IND / RC	INPUT	Index input signal from quadrature encoder, also used as R/C signal input, this input is pulled to +5VDC with a 100kΩ resistor, and is protected with a 5.6V zener diode
3	CHA	INPUT	Channel A input signal from quadrature encoder, this input is pulled to +5VDC with a 100kΩ resistor, and is protected with a 5.6V zener diode
4	CHB	INPUT	Channel B input signal from quadrature encoder, this input is pulled to +5VDC with a 100kΩ resistor, and is protected with a 5.6V zener diode
5	GND	POWER	The supply voltage ground return connects to this terminal

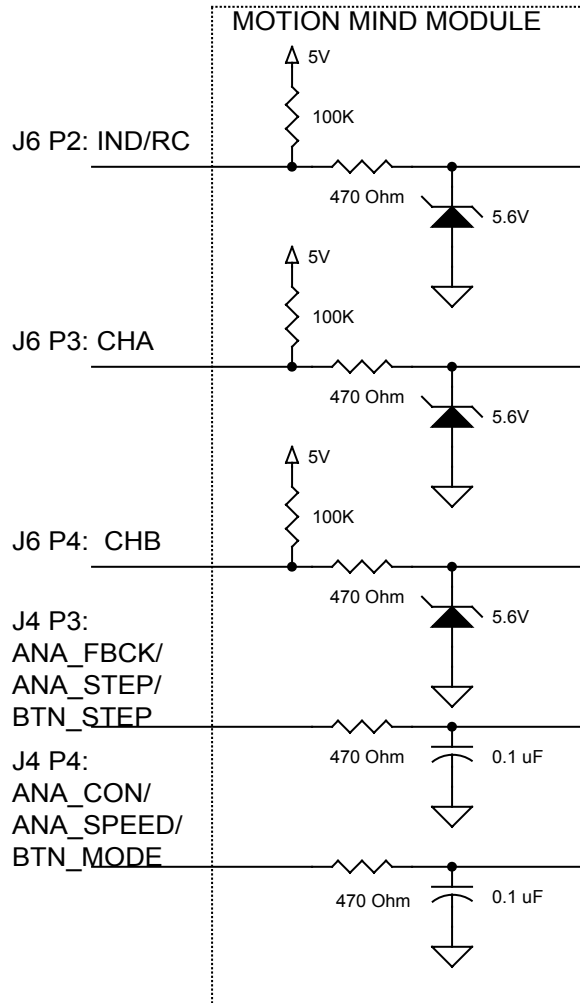
### 3.6 Indicator LEDs

Two indicator LEDs are present on the Motion Mind controller. D1 labeled PWR is a green LED that lights when power is applied to the VMOTOR connection and the 5V linear regulator powers up. D2 (also green) is labeled COMM and serves two functions. The first function is that on power up it will blink a number of times equal to the operating mode + 1 (example: mode 0 = 1 blink, mode 7 = 8 blinks). The second function of the COMM LED is that it will blink when serial communication is received and accepted.

### 3.7 Input Circuit Considerations

The protection and filtering circuits used by this controller could affect some attached circuits. The input protection circuits and filters are detailed here for reference.

**Figure 4: Analog and Encoder Input Circuits**



### 3.8 Two's Complement Number System

This controller can accept negative numbers for some settings. In order to generate a two's complement negative value take the binary or hexadecimal representation of the absolute value of the number, compliment it (every 1 becomes a 0 and every 0 becomes a 1) and add 1 to the result.

**Two's Complement Examples**

Number	Absolute Value	Hexadecimal	Compliment	Two's Complement
-1	1	H'00000001'	H'FFFFFFF'	H'FFFFFFF'
-32768	32768	H'00008000'	H'FFFF7FFF'	H'FFFF8000'
-100000	100000	H'000186A0'	H'FFFE795F'	H'FFFE7960'

### 3.9 Quadrature Encoder Interface

Correct encoder connections can vary based on manufacturer, encoder mounting technique, and the motor gearing. If the encoder connections are reversed, the position controller will run away from commanded positions. In other words, a command to move to a positive position will result in the motor running in reverse. If you encounter this problem, swap the *CHA*, *CHB* connections at J6, or the *M+* and *M-* connections at J3.

The Motion Mind controller makes use of a quadrature decoder circuit that decodes pulses at a 4:1 ratio. Therefore a 512 counts-per-revolution (CPR) encoder would be represented as a 2048CPR encoder by the Motion Mind circuit. This improves resolution and accuracy in position control, but needs to be accounted for by the user. When calculating position moves, you should account for the encoder resolution (CPR) and gear ratio. Accurate position control will make use of high CPR encoders and motor gearing that allows small motor shaft changes to create large position error signals. Position can be related to motor shaft movement with the following equation. The example shown is for a 30:1 gear ratio with a 512CPR encoder attached. For this example a 1° move would be approximately 85 counts.

$$\begin{aligned} \text{shaft\_revolution} &= \text{gear\_ratio} \times \text{encoder\_CPR} \times 4 \\ 61,440 &= 30 \times 512\text{CPR} \times 4 \end{aligned}$$

A tradeoff will be made between the ability to move to a position quickly, and the accuracy of the move. From a mechanical standpoint a high motor gear ratio is useful in increasing the error signal, and providing a stiffness to resist changes in the mechanical load. A high gear ratio also reduces the motor output shaft speed. The nature of the PID algorithm causes large position errors to be compensated for by the Proportional (P) part of the PID. Small errors are compensated for by the summation or Integral (I) portion of the PID. As the small errors build up over time the Integral term will eventually nudge the motor into the desired position. Therefore a low CPR encoder and/or a motor without gears will create a less accurate control system more dependant on the Integral portion of the PID. If your system appears to be driven primarily by the Integral portion of the PID try reducing the PIDSCALAR setting and retuning the PID.

The Motion Mind controller will work with 2 channel quadrature encoders. Many encoders provide a third “index” channel that indicates each revolution of the encoder. Often this index location is used to ensure a known start-up position for the mechanical system. The Motion Mind controller monitors this input and provides an approximate position for the index point in the INDEXPOS register. The value stored in this register is the last valid POSITION (actual position) measurement taken before the index pulse occurred. Therefore, the INDEXPOS value may be off by as much as the motor velocity when the measurement is taken. The STATUS.INDEXBIT is only set when the index input is asserted. Using the INDEXPOS register and STATUS.INDEXBIT it is possible to determine the exact position of an index point.

### 3.10 The PID Filter

In modes 4 and 5 (closed loop modes) the Motion Mind controller makes use of a proportional (“P”), integral (“I”), and derivative (“D”), or PID, filtering technique with a Velocity Feed-Forward factor used by some settings. This filter effects changes in the output PWM signal relative to the error signal generated by the feedback signal. The user defines a desired position and an error signal is calculated based on the difference between the desired position and the actual position.

A serial interface (ASCII or binary) is required to program the PID registers. The default PID settings are designed to work for 512CPR quadrature encoders. If operating in closed loop analog mode the PIDSCALAR will need to be reduced before tuning of the PID can occur.

The PID can be broken down into four stages that can each be described separately. Each section may be modified by the user, but once the PID Filter is “tuned” the user will typically just send the desired position to the controller.

- A) Error Signal:** An error signal is required by the PID filter to effect motor position control. The PID filter converts the error signal to a motor drive signal. It is this conversion that closes the loop between the feedback (encoder or analog input) and the motor drive signal. Typically a large position error results in a large output from the PID filter, a small position error creates small motor drive signals. In this closed loop control the master unit or user will provide the desired position.

$$Error(t) = DesiredPosition(t) - ActualPosition(t)$$

- B) PID Algorithm:** Each component of the PID has a different effect on the output of the filter. In addition, there are some configuration bits associated with the “I” term that the user can elect to enable. Briefly, the “P” term is used to create motor drive signals that compensate for large position errors. The “P” term is responsible for gross motor movements. The “I” term accumulates small position errors by summing them, and will compensate for small position errors after enough of them have occurred. The “I” term will nudge the motor into the desired position over-time and compensate for errors too small for the “P” term to address. The “D” term is the least effective term of the PID. It acts to minimize abrupt changes in motor speed and essentially provides drag on the motor.

$$Output = PTERM \times Error(t) + ITERM \times \int Error(t) + DTERM \times \partial(Error(t)) / \partial t$$

Setting any of the terms to 0 will remove that specific component of the PID from the filter. Typically only the “D” term is left unused. The “I” term described above shows an infinite summation of error signals. If this were to actually happen, even small error signals over time could swamp out the effect of the “P” term. This is commonly called integral wind-up. Restrictions on the size of the “I” sum are implemented on the Motion Mind to prevent wind-up. Additionally the user may set the FUNCTION.SATPROT bit and the summation of the “I” term is prevented when the motor output is at full-scale (saturated), and the sum of the error terms is limited to a small value.

- C) PID Output Scaling:** The position error term can vary widely from system to system. For example, a high CPR encoder would generate a very large error signal for a small move. Whereas a system using a potentiometer for feedback will always have very small position error (there’s only 1023 possible positions). After the “P”, “I”, and “D” terms generate the output, it must be scaled back to a value that roughly fits into the range of +/-1023 (the duty-cycle settings of the motor control PWM outputs). This scaling is accomplished by the PIDSCALAR register. Setting the PIDSCALAR too low will cause the motor to oscillate; setting it too high will cause position errors to be too small to generate motor movement.

$$Output = Output \div 2^{PIDSCALAR}$$

- D) Velocity Feed-Forward:** When position control is operated in a velocity controlled mode (FUNCTION.VELLIMIT bit is set) or a Move\_Velocity command is sent (see section 5 for communication protocol details) the Velocity Feed-Forward term is added to the output of the PID. In either of the previously mentioned cases, the Motion Mind controller generates a desired velocity for the move. This open-loop information is added to the output of the PID filter. A ratio is established by the VELOCITYFF register. Setting this value to 0 (or clearing the FUNCTION.VELLIMIT bit) removes this open-loop addition to the PID output.

$$Output = Output + DesiredVelocity \times \frac{VELOCITYFF}{255}$$

### 3.11 Tuning the PID Filter

There are a number of methods of tuning a PID filter. The method described here is one simple way of ensuring motor position control. You'll need to familiarize yourself with the communication protocol and make use of the serial interface to tune the PID.

1. Set the VELOCITYFF, ITERM, DTERM to 0 and make sure the FUNCTION.VELLIMIT and FUNCTION.SATPROT bits are clear.
2. Apply step movements of approximately 1 encoder shaft revolution (for a 512CPR encoder, this is  $4 \times 512 = 2,048$ ). Adjust the PTERM and PIDSCALAR settings until the motor moves approximately to this position. Keep in mind that reducing the PIDSCALAR increases the PID output, and that each change in the PIDSCALAR doubles or divides-by-two the PID output (for example decreasing the PIDSCALAR from 14 to 11 increases the PID filter output by 8 times).
3. Continue to apply step movements while increasing the ITERM. Having the ITERM large will cause motor oscillations. To stop the oscillations, reduce the ITERM, increase the PTERM, and/or set the FUNCTION.SATPROT bit. With the ITERM added the motor should move roughly to the desired position and then get nudged into the final position over time.
4. Continue applying step movements, and increase the DTERM.
5. Try different step sizes (larger and smaller) and experiment with the other settings.

For stiff motor position control, increase all of the PID settings and set the FUNCTION.VELLIMIT and FUNCTION.SATPROT. The following settings worked well for a 512CPR encoder 218:1 gear-head motor for counts as low as 10 and up to 1,000,000.

```
VELOCITYLIMIT = 25
VELOCITYFF = 200
PTERM = 30000
ITERM = 1000
DTERM = 60000
PIDSCALAR = 14
FUNCTION.SATPROT = enabled
FUNCTION.VELLIMIT = enabled
```

**3.12 Velocity Measurements and Closed-Loop Control:** In mode 4, position control can have velocity limits applied to it. The velocity measurements are in encoder-counts-per-PID loop, so some conversion may be required. The VELOCITY register contains the average of 64 velocity measurements (in closed-loop analog mode this measurement is not averaged, and will likely be too small to be of any use). In order for position control movements to be velocity limited the FUNCTION.VELLIMIT bit should be set. Once set, the contents of the VELOCITYLIMIT register will be used to determine motor speed.

$$\text{MotorShaftRotations / Second} = \frac{\text{VELOCITY} \times 50}{\text{EncoderCPR} \times \text{GearRatio}}$$

## 4.0 Operating Modes

### 4.1 Overview

The Motion Mind motor controller has eight modes of operation. Many of these modes of operation have sub-modes or settings that can be adjusted through a variety of external connections, or internal register settings. Each of the modes is described briefly in this overview, and more detailed descriptions, including sample electrical connections, follow under the section describing each operating mode.

There are four open-loop modes (no feedback signals required) that accept a variety of inputs and adjust the voltage applied to the motor accordingly. The first three open-loop modes were designed so that they could be configured externally (without the serial interface). But the serial interface may still be used to fine-tune some of the settings. The two additional closed-loop control modes implement PID position control algorithms and require motor position feedback (either from a quadrature encoder or analog signal depending on the mode). Both of the closed-loop modes will require programming of at least some of the registers to achieve optimum position control.

Many of the connections (specifically at J4) have multiple functions, each associated with a specific operating mode. To make this datasheet easier to read the multiuse pins are referred to by only the function being described in each section. For example, J4 P16 is the *NEG\_LIM/ANA\_MODE/BTN\_DN* pin. When referenced under open loop analog mode this pin is described as *ANA\_MODE* (J4 P16), but under button mode this pin is referred to as *BTN\_DN* (J4 P16).

### 4.2 Getting Started

The Motion Mind module is highly configurable through a variety of external settings. Before programming and operating the Motion Mind controller these steps should be taken.

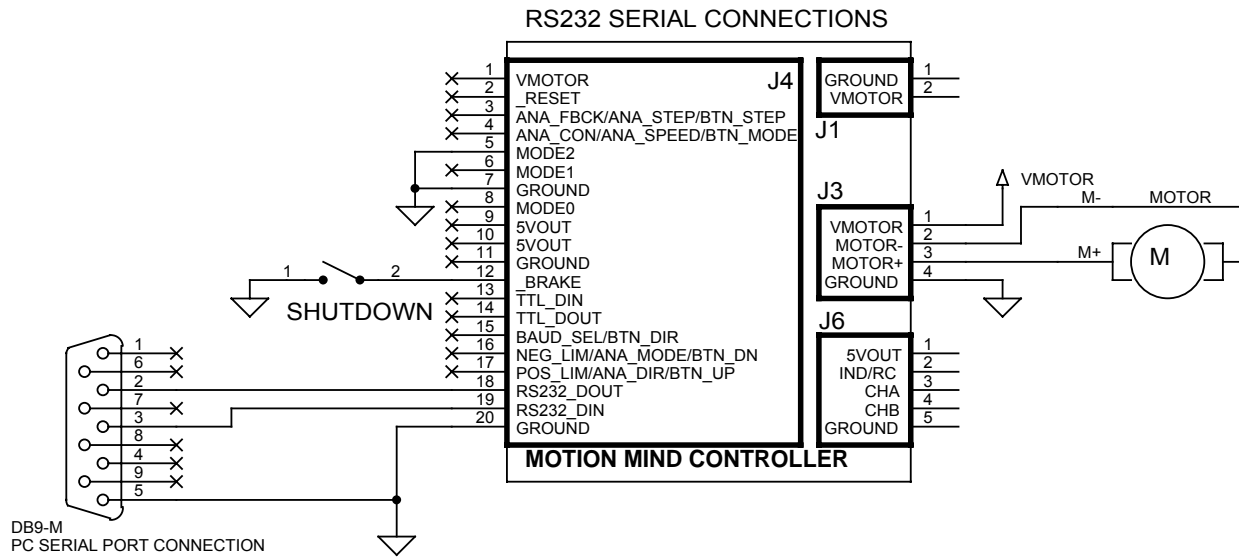
**Step1:** Determine the mode of operation you plan on using, then set the jumpers at J5 for that mode (see section 3.5). Alternatively, logic levels may be applied to J4-P5, P6, and P8 to select the operating mode.

**Step2:** Select the baud rate (19.2KBPS or 9.6KBPS) and mode of communication (binary or ASCII) by setting the jumpers at J2 (again see section 3.5 for details). Please note that in Mode 2 (button mode) when operating in uni-directional mode, the *BAUD\_SEL* (J4 P15) input pin is also used to determine the direction the motor spins (forward or reverse). If you're using this mode, and implementing serial communication, you'll be required to use the baud rate associated with the desired motor direction.

**Step3:** If a serial interface is required for the controller, then determine whether you'll be using the RS232 connections or the TTL level connections at J4. RS232 is the electrical specification common to PC serial ports. The TTL level connections might be used in a system where a separate controlling unit is attached that operates off of 5V logic levels (such as a dedicated microcontroller). When operating in TTL mode multiple modules can share the TTL lines as long as they have been pre-programmed with different address values. See section 5 for details on the communication protocol.

**Step4:** Connect the motor and motor voltage and apply power

**Figure 5: Basic Serial Connections**



**Summary of Operating Modes**

Mode	Submode	Description
<b>Mode 0</b> R/C Signal Control		This is an open-loop control mode that measured a 1-2mS pulse provided by a hobby type radio control receiver. The pulse duration determines motor speed and direction. The user can fine-tune the acceptable pulse signal parameters through the serial interface.
	R/C Position Control	In REV4+ firmware. This is a closed loop control mode that uses an analog signal for position feedback. R/C signals are converted from a 1-2ms pulse timing value to a desired position value that may range from 0 to 1023. The analog signal is converted to a value from 0 to 1023 and constitutes the actual position feedback signal.
<b>Mode 1</b> Analog Control		An open-loop control mode, an analog input controls the output voltage to the motor. For example, if the input is 50% of full scale, then the motor has an average of 50% of the motor voltage applied to it. Uni-directional or bi-directional mode is determined by a logic level input.
	Uni-Directional	0-5V input controls the motor from stopped to full speed. A separate logic level input determines whether the motor runs forward or reverse.
	Bi-Directional	0-5V input controls the motor from full reverse (0V) to full forward (5V). A dead band around 2.5V is used to stop the motor.
<b>Mode 2</b> Button Control		This open-loop control mode allows 2 external buttons to either increase or decrease motor speed. The button presses are debounced based on a timer setting that may be adjusted via the serial interface. The step size in motor control voltage is determined by an analog input, or alternatively through loading a register and setting a flag via the serial interface. Uni-directional or bi-directional mode is determined by a logic input.
	Uni-Directional	Button presses will increase or decrease the motor drive voltage between stopped and full speed. Motor direction (forward/reverse) is determined by a logic level input.
	Bi-Directional	Button presses adjust the motor drive voltage from full reverse to full forward. A delay is added at the stop position to allow a "stop" to occur even when the motor step setting is high.

**Summary of Operating Modes (continued)**

<b>Mode 3</b> Serial Control		This open-loop mode accepts serial motor speed and direction commands.
<b>Mode 4</b> Serial PID Control		This closed-loop mode requires a 2 or 3 channel quadrature encoder to be connected to J6. The feedback from this encoder is used to make motor speed/direction adjustments to ensure the motor tracks user supplied position or velocity commands. The detailed serial communication protocol is required to fully implement this operating mode.
<b>Mode 5</b> Analog PID Control		A 0-5V analog signal (typically from a potentiometer) is used to feed motor position back to the controller. The user supplies an analog signal, also 0-5V, that represents the desired position. A PID algorithm is used to make adjustments to motor speed and direction based on the desired motor position and the actual motor position. Many of the PID settings will need to be fine-tuned through the serial interface.
	Analog Feedback with serial control	In REV2+ firmware setting the FUNCTION.ADSERIAL bit allows you to provide the desired position through the serial communication interface.
	Analog Control with encoder feedback	In REV5+ firmware setting the FUNCTION.ENCFDBCK bit allows you to use a 0-5V signal as the position control (0-5VDC, 0-1023 positions) with a quadrature encoder as the actual position feedback signal.
<b>Mode 6</b> Factory Restore		When placed into mode 6 and the <i>POS_LIM</i> and <i>NEG_LIM</i> input lines (J4 P17, J4 P16) are pulled to ground the internal EEPROM of the Motion Mind motor controller is loaded with the factory default settings.
<b>Mode 7</b> Factory Test		This mode is used after production as a self-test for the unit. Customers are not expected to use this mode.

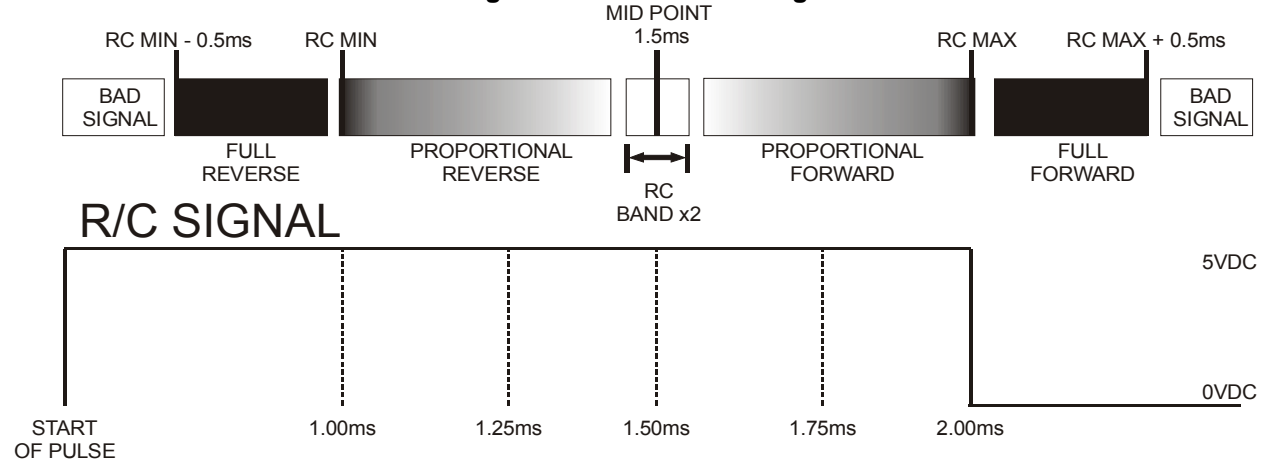
**4.3 Mode 0: Radio Control (R/C) Speed Control:**

**How it Works:** In R/C control mode the Motion Mind controller monitors a radio control signal present at the *IND/RC* pin (J6 P2). The signal is typically from 1-2ms in duration, and repeats roughly every 20ms. Signals 1.5ms in duration represent a stop condition, 1ms represents full speed reverse, and 2ms represents full speed forward. Motor speed control is proportional in between the previously mentioned points, and the end-points themselves may be adjusted via the serial interface. R/C pulse measurements and associated registers are in increments of 814ns. For example the default setting for *RCMAX* is 2ms, this is derived from the register value 2457 multiplied by 814ns.

$$RCMAX = 2457 \times 814ns = 1.99ms$$

Signals with a duration less than *RCMIN* – 0.5ms or greater than *RCMAX* + 0.5ms are bad signals and typically result in a stopped motor. If no signal is measured within 50ms then it is also considered a bad signal. As mentioned previously, a bad signal typically results in the motor being stopped. However by setting the *FUNCTION.LASTRC* bit you can force the Motion Mind controller to use it's last valid signal measurement in place of a bad signal. It's also useful to point out that transient bad signals (those that appear briefly, such as 1 bad signal out of 20 measurements) might cause motor jitter but probably wouldn't actually stop the motor.

**Figure 6: R/C Pulse Timing**

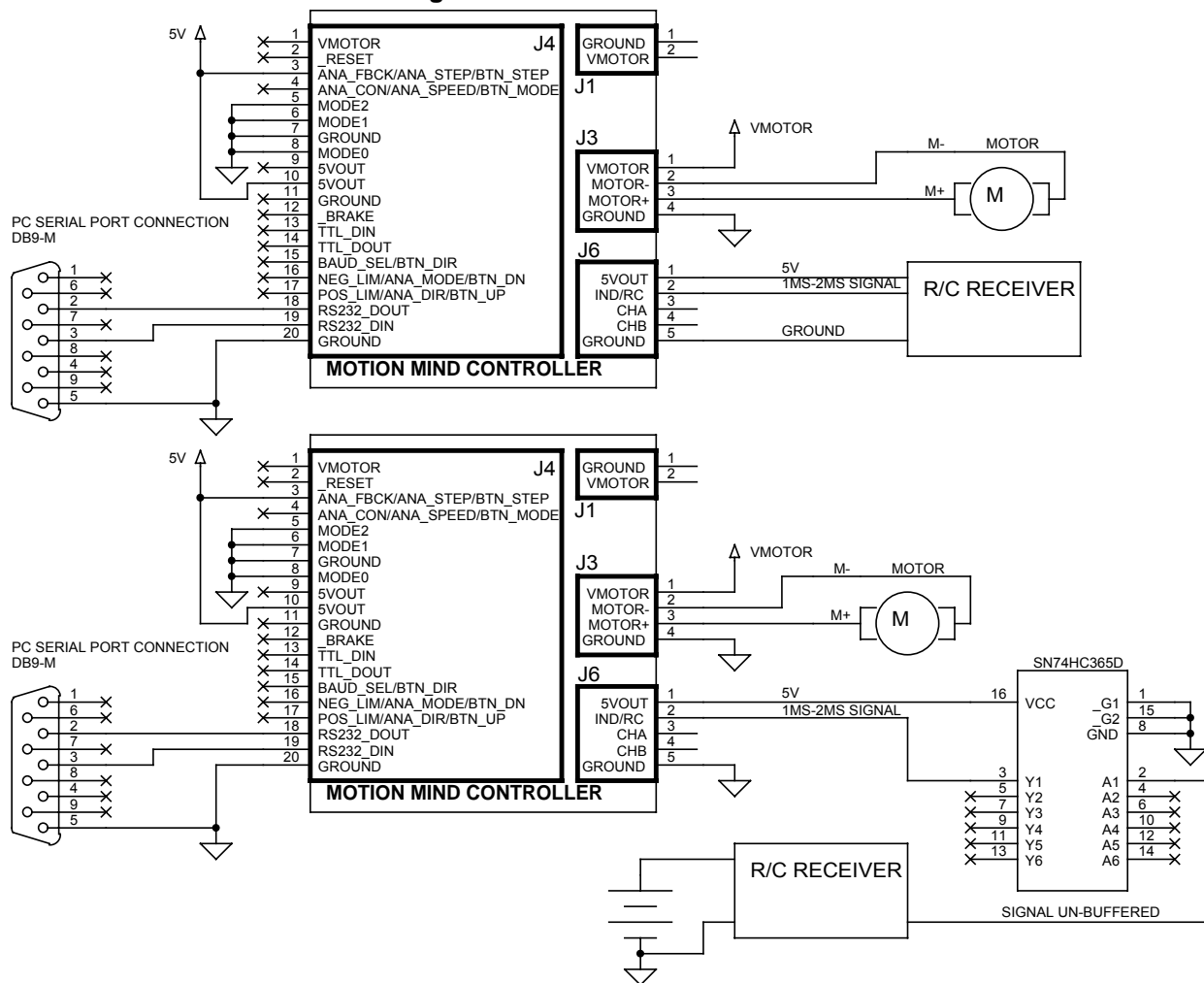


**Fine Tuning to Your R/C Receiver:** The Motion Mind controller should work with your R/C receiver straight out of the box. But you may want to fine-tune your receiver/transmitter to the controller. In order to do this you'll need to make use of the serial interface. Then follow these steps.

1. Adjust the trim-pot on your transmitter until the RCCOUNT register is equal to 1.5ms
2. Push the joystick to full forward and read the RCCOUNT register
3. Store this value in EEPROM as the new RCMAX value
4. Push the joystick to full reverse and read the RCCOUNT register
5. Store this value in EEPROM as the new RCMIN value

Some R/C receivers will not put out a 0-5V signal. If your system is jittery this may be the case. Providing a logic level buffer may help eliminate this problem. The second example shown below is a simple circuit that may help if you run into this problem.

**Figure 7: R/C Mode Connections**



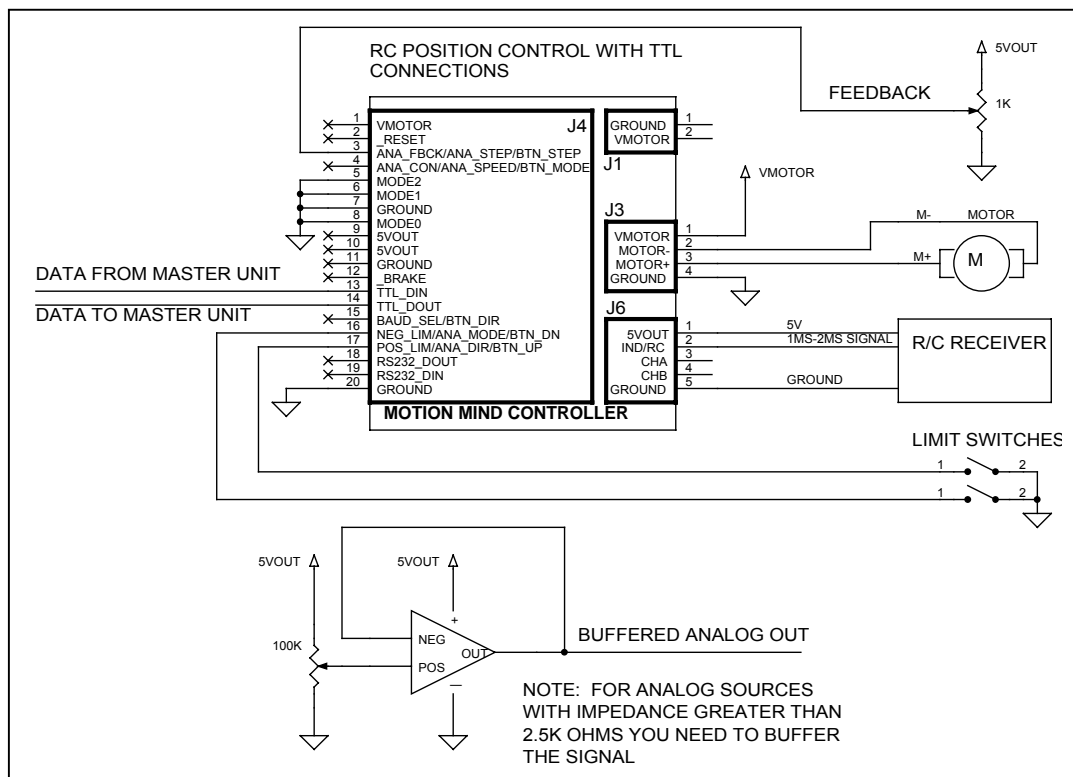
**Mode 0 Sub Mode: Radio Control (R/C) Position Control:**

**How it Works:** Firmware revision 4+ only. Setting the FUNCTION.RCPOS bit enables R/C Position Control mode. In this mode an analog signal (0-5VDC) is used to provide a motor feedback signal. The analog value is converted to a position of 0 (0VDC) to 1023 (5VDC). The RC signal is then converted from a timing count (shown in the RCCOUNT register) to a value between 0 and 1023. The RCMIN and RCMAx registers play a part in the conversion. In order to have a full scale of positions the actual maximum RC signal should have a timing count slightly higher than the RCMAx register value, and the minimum RC signal should be slightly less than the RCMIN register value. Before fine-tuning the RCMAx and RCMIN values make sure they are set to values outside of the expected signal duration (for example set RCMIN to 0.75ms (921) and RCMAx to 2.25ms (2764). Otherwise the RCCOUNT value is internally limited to RCMAx and RCMIN and you won't be able to read the full range of timer counts.

To ensure full scale position control use the serial interface (programming software is provided at our web site) to read the RCCOUNT register with the R/C joystick pushed all of the way forward. Then program the RCMAx register with a value just below the RCCOUNT. Pull the joystick back to the minimum signal position, read the RCCOUNT value, and program the RCMIN register with a value slightly more than the RCCOUNT.

Since this is a PID position control mode the PID related registers should be adjusted to fine tune the controller to track the error signal. If FUNCTION.ENABLEDB is set then the RCBAND register is used both to set a dead band around 1.5ms and will cause the motor drive signal to turn off if the actual position is within the desired position +/- RCBAND register value.

**Figure 7 (continued)**



**Interface/Programming Specific to Mode 0**

<b>Input Pins</b>	<b>Description</b>
<i>IND/RC</i> (J6 P2)	Apply the R/C control signal here
<i>ANA_STEP</i> (J4 P3)	The analog signal applied to this pin determines the motor speed change allowed between motor speed updates. 0V corresponds to a step of 1 (it would take 1023 motor speed updates to go from stopped to full speed. 5V relates to a step of 1023 (it would take one motor speed update to go from 0 to full speed).
<i>POS_LIM</i> (J4 P17)	Positive limit switch, after asserted (0V) the motor is prevented from moving in a positive direction. R/C position mode only.
<i>NEG_LIM</i> (J4 P16)	Negative limit switch, after asserted (0V) the motor is prevented from moving in a negative direction. R/C position mode only.
<b>Programmable Registers</b>	<b>Description</b>
RCMAX	This value represents the pulse width that equals full-forward speed. The default setting is 2ms. Change this setting to fine-tune the Motion Mind controller to your receiver.
RCMIN	This value represents the pulse width that equals full-reverse speed. The default setting is 1ms. Change this setting to fine-tune the Motion Mind controller to your receiver. When operated in the RC Position Control mode this register helps define the
RCBAND	This value is the time period around 1.5ms that results in a stopped motor condition.
VELOCITYLIMIT	Normally R/C mode will use the analog signal present on <i>ANA_STEP</i> (J4 P3) to set the allowable change in motor speed between signal measurements. However, if you set the FUNCTION.ADSTEP bit, the allowable change in motor speed is determined by the contents of the VELOCITYLIMIT register, and not the analog input. Please note that after first setting the FUNCTION.ADSTEP bit the processor should be reset to allow this new setting to take effect.
PWMLIMIT	Default is 1023 setting to a lower value limits the duty cycle of the PWM signal driving the H-Bridge. For example 512 limits the duty-cycle to 50% (+/-). Can be used to current limit closed loop applications
<b>Programmable Registers (RC Position Mode only)</b>	<b>Description</b>
POSITION	2's compliment 32-bit position value from the analog feedback potentiometer (limited to 0 to 1023 due to A/D 10-bit conversion)
PTERM	The proportional "P" constant associated with the PID filter, this register contains positive values from 0 to 65535
ITERM	The integral "I" constant associated with the PID filter, this register contains positive values from 0 to 65535
DTERM	The derivative "D" constant associated with the PID filter, this register contains positive values from 0 to 65535
PIDSCALAR	The output of the PID filter is divided by 2^PIDSCALAR
TIMER	The timer register is used to debounce limit switch inputs, defaults to 50ms
VNLIMIT	Sets a virtual negative position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set. (Rev5+ firmware only)
VPLIMIT	Sets a virtual positive position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set. (Rev5+ firmware only)
DEADBAND	When used in conjunction with the FUNCTION.ENABLEDB bit a dead band around the desired position is created. If the actual position moves within the desired position +/- DEADBAND the PWMOUT signal is driven to 0 (Rev5+ firmware only).
<b>Function bit settings</b>	<b>Description</b>
FUNCTION.LASTRC	If this bit is set then upon receiving a bad R/C signal the controller will insert the average of the last 32 good R/C signals it received. If this bit is cleared then a bad R/C signal will result in the motor being stopped. Sometimes erroneous signals can still fall within the valid signal range. For example, if you were to shut off your R/C transmitter, the last signal sent might get clipped at 1ms. A 1ms signal is a valid full speed reverse signal (using the default system settings).
FUNCTION.ADSTEP	Set this bit if you want to prevent the analog input at <i>ANA_STEP</i> (J4 P3) from being used to set the motor speed step limit. When set the step limit is determined by the contents of the VELOCITYLIMIT register. Please note that after first setting the FUNCTION.ADSTEP bit the processor should be reset to allow this new setting to take effect.
FUNCTION.RCPOS	Set this bit to enable RC Position Control mode (Rev4+ firmware only)
FUNCTION.VIRTLIMIT	Enables virtual software limits defined by VNLIMIT and VPLIMIT registers (Rev5+ firmware only)

#### 4.4 Mode 1: Analog Control

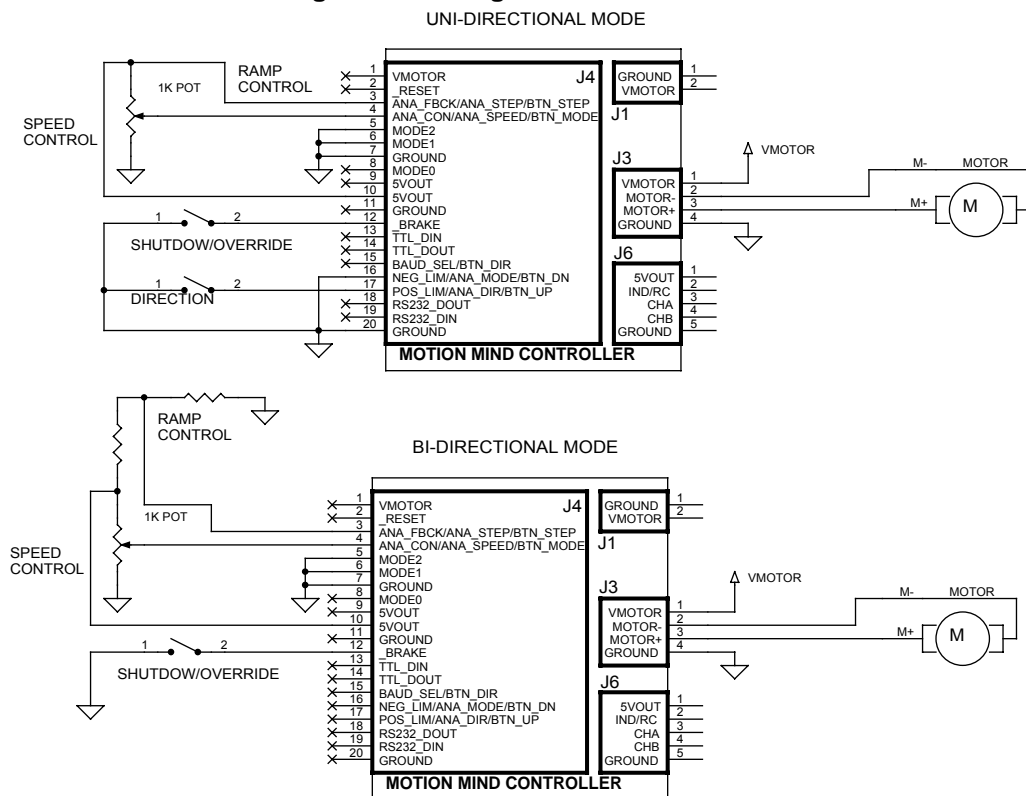
**How it Works:** In this mode the 0-5V signal present at *ANA\_SPEED* (J4 P4) is translated into the motor speed. Motor speed is updated periodically based on the value of the *TIMER* register. The default update rate is every 50ms, and may be adjusted in 5ms increments. In addition to the update rate the change in motor speed is limited by the 0-5V signal at *ANA\_STEP* (J4 P3), or alternatively by the value in the *VELOCITYLIMIT* register (if the *FUNCTION.ADSTEP* bit is set). For your initial testing you can tie *ANA\_STEP* to 5V. In bi-directional mode a “dead-band” centered at 2.5V exists (2.43V-2.57V). If the analog input falls within this range then the motor is stopped.

Analog inputs should have a source impedance of less than 2.5KΩ. For larger impedances (such as potentiometers with greater than 2.5KΩ full scale resistance) you should buffer the signal with an op-amp configured as a unity-gain voltage follower. Using larger impedances will reduce the accuracy of the analog-to-digital measurements.

**Sub Mode Pin Settings for Mode 1**

Pins	Bi-directional Mode	Unidirectional Forward	Unidirectional Reverse
<i>ANA_SPEED</i> (J4 P4)	0V = full reverse 5V = full forward 2.5V = stopped	0V = stopped 5V = full forward	0V = stopped 5V = full reverse
<i>ANA_DIR</i> (J4 P17)	N/A	5V	0V
<i>ANA_MODE</i> (J4 P16)	No connect or 5V	0V	0V

**Figure 8: Analog Mode Connections**



### Interface/Programming Specific to Mode 1

Input Pins	Description
<i>ANA_SPEED</i> (J4 P4)	This 0-5V analog input defines the desired motor speed. When operated in bi-directional mode 0V is full reverse, 2.5V stopped, and 5V is full forward. In unidirectional mode 0V is stopped, 5V is full forward, and the logic level at J4 P17 determines the direction the motor spins.
<i>ANA_STEP</i> (J4 P3)	The analog signal applied to this pin determines the motor speed change allowed between motor speed updates. 0V corresponds to a step of 1 (it would take 1023 motor speed updates to go from stopped to full speed. 5V relates to a step of 1023 (it would take one motor speed update to go from 0 to full speed).
<i>ANA_DIR</i> (J4 P17)	When operating in unidirectional mode this pin determines the direction the motor is spinning (0V for reverse, 5V for forward)
<i>ANA_MODE</i> (J4 P16)	Unidirectional mode is enabled if this pin is at 0V. If at 5V or not connected then bi-directional mode is running.
Programmable Registers	Description
TIMER	The timer register is used in analog mode as a motor speed update timer. The default setting is 10 (50ms). The allowed value for this register is 0-255 where the units are 5ms time blocks (example: a 200ms update period would occur if TIMER was set to 40).
VELOCITYLIMIT	Normally open-loop analog mode will use the analog signal present on <i>ANA_STEP</i> (J4 P3) to set the allowable change in motor speed between signal measurements. However, if you set the FUNCTION.ADSTEP bit, the allowable change in motor speed is determined by the contents of the VELOCITYLIMIT register, and not the analog input. Please note that after first setting the FUNCTION.ADSTEP bit the processor should be reset to allow this new setting to take effect. (Rev5+ firmware only)
PWMLIMIT	Default is 1023 setting to a lower value limits the duty cycle of the PWM signal driving the H-Bridge. For example 512 limits the duty-cycle to 50% (+/-). Can be used to current limit closed loop applications
Function bit settings	
FUNCTION.ADSTEP	Set this bit if you want to prevent the analog input at <i>ANA_STEP</i> (J4 P3) from being used to set the motor speed step limit. When set the step limit is determined by the contents of the VELOCITYLIMIT register. Please note that after first setting the FUNCTION.ADSTEP bit the processor should be reset to allow this new setting to take effect.

#### 4.5 Mode 2: Button Control

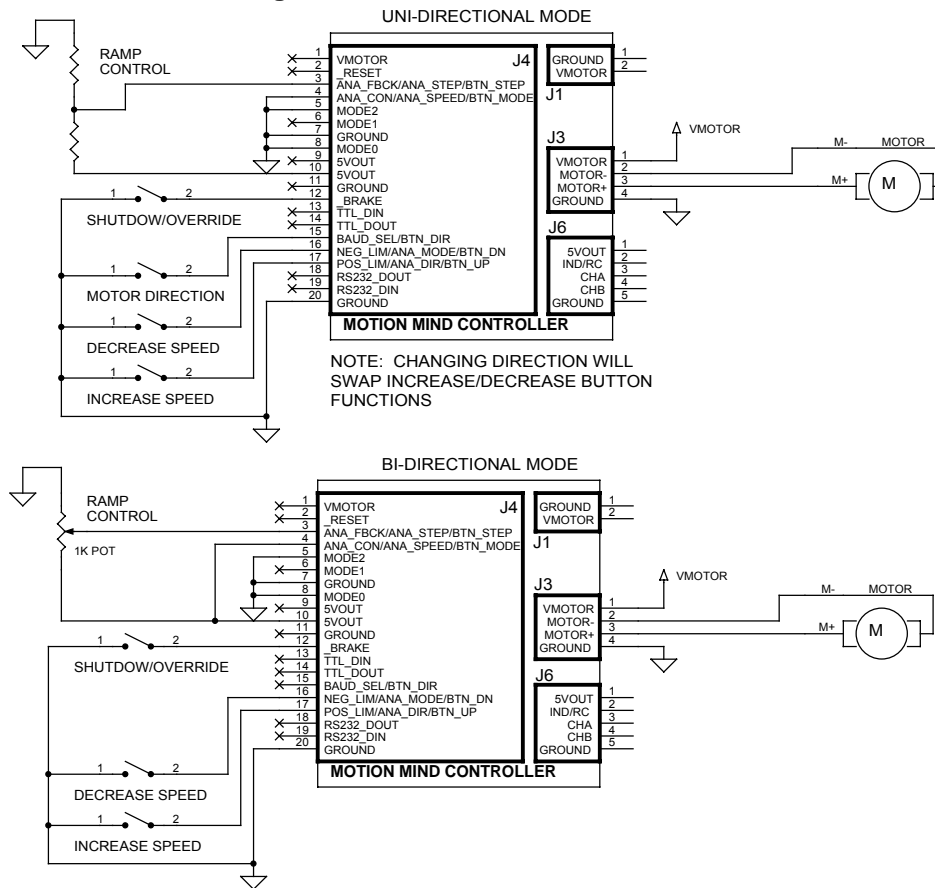
**How it Works:** By connecting buttons to *BTN\_UP* (J4 P17) and *BTN\_DN* (J4 P16) motor speed may be controlled with button presses. The buttons are debounced for a period of time of 5ms times the contents of the TIMER register (default is 50ms). The button does not need to be released for a new button press to be registered. Holding a button down will cause the motor speed to slew upwards or downwards depending on the button pressed. The rate of change of the motor speed is based on the debounce time (TIMER register) and the analog value present at the *BTN\_STEP* (J4 P3) pin (or if the FUNCTION.ADSTEP bit is set, the value in the VELOCITYLIMIT register is used).

When in bi-directional mode, a motor speed transition through 0 will cause the motor to be stopped for a period equal to 4 times the debounce time (default is 200ms). This extended stop time allows the user to stop the motor even if the rate of change in motor speed is large.

#### Sub Mode Pin Settings for Mode 2

Pins	Bi-directional Mode	Unidirectional Forward	Unidirectional Reverse
<i>BTN_MODE</i> (J4 P4)	5V	0V	0V
<i>BTN_DIR</i> (J4 P15)	N/A	5V	0V
<i>BTN_STEP</i> (J4 P3)	0-5V analog sets motor speed change per button press	0-5V analog sets motor speed change per button press	0-5V analog sets motor speed change per button press
<i>BTN_UP</i> (J4 P17)	Press modifies motor speed up to full speed forward	Press modifies motor speed up to full speed forward	Press modifies motor speed up to stopped
<i>BTN_DN</i> (J4 P16)	Press modifies motor speed down to full reverse	Press modifies motor speed down to stopped	Press modifies motor speed down to full reverse

**Figure 9: Button Mode Connections**



**Interface/Programming Specific to Mode 2**

Input Pins	Description
<i>BTN_MODE</i> (J4 P4)	5V configures the controller for bi-directional motor control; 0V configures the controller for unidirectional motor control (with <i>BTN_DIR</i> (J4 P15) determining forward or reverse).
<i>BTN_STEP</i> (J4 P3)	The analog signal applied to this pin determines the motor speed change allowed between motor speed updates. 0V corresponds to a step of 1 (it would take 1023 motor speed updates to go from stopped to full speed. 5V relates to a step of 1023 (it would take one motor speed update to go from 0 to full speed).
<i>BTN_UP</i> (J4 P17)	Pressing this button increases the motor speed if the motor is running forward, or decreases the motor speed if running in reverse
<i>BTN_DN</i> (J4 P16)	Pressing this button decreases the motor speed if the motor is running forward, or increases the motor speed if running in reverse
Programmable Registers	Description
TIMER	The timer register is used to debounce the button presses. The default setting is 50ms. Debounce is an electrical term that describes how long a button must be in it's current state before it is accepted as valid.
VELOCITYLIMIT	Normally button mode will use the analog signal present on <i>BTN_STEP</i> (J4 P3) to set the allowable change in motor speed between signal measurements. However, if you set the <i>FUNCTION.ADSTEP</i> bit, the allowable change in motor speed is determined by the contents of the <i>VELOCITYLIMIT</i> register, and not the analog input. Please note that after first setting the <i>FUNCTION.ADSTEP</i> bit the processor should be reset to allow this new setting to take effect.
PWMLIMIT	Default is 1023 setting to a lower value limits the duty cycle of the PWM signal driving the H-Bridge. For example 512 limits the duty-cycle to 50% (+/-). Can be used to current limit closed loop applications (Rev5+ firmware only)

Function bit settings	Description
FUNCTION.ADSTEP	Set this bit if you want to prevent the analog input at <i>BTN_STEP</i> (J4 P3) from being used to set the motor speed step limit. When set the step limit is determined by the contents of the VELOCITYLIMIT register. Please note that after first setting the FUNCTION.ADSTEP bit the processor should be reset to allow this new setting to take effect.

#### 4.6 Mode 3: Serial Control

**How it Works:** Open-loop serial control allows the user to control motor speed and direction through a serial interface. An ASCII or binary protocol may be used. The user can choose between a TTL (logic level) and an RS232 compatible communication link. Communication rates of 9.6KBPS and 19.2KBPS are supported. The TTL TX pin (J4 P14) is open collector and may be used to connect multiple units that have individually had their addresses programmed to different values. For communication protocols see section 5.

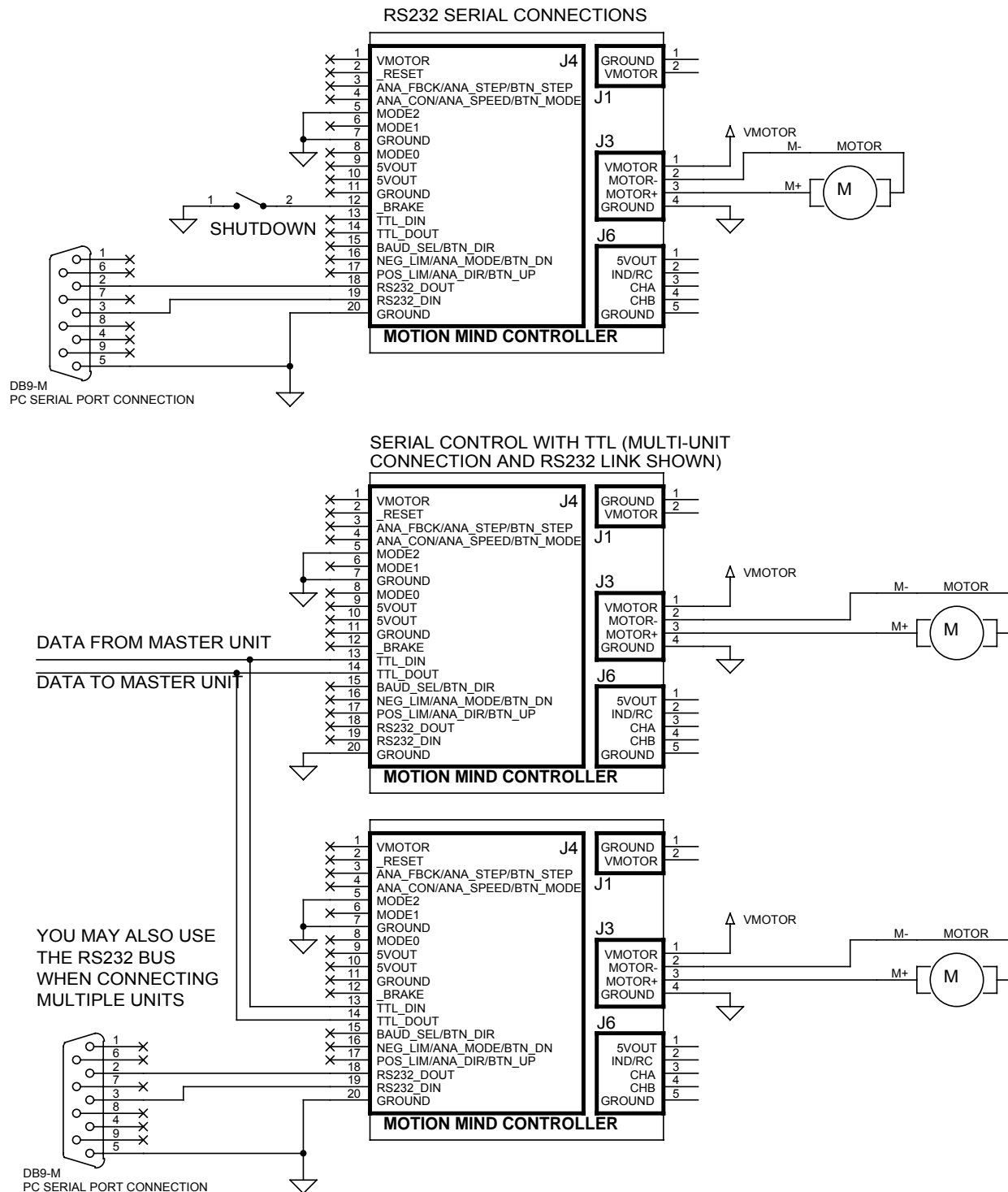
#### Summary of Serial Commands Used by Mode 3

Command	Description
CHANGE_SPEED	Adjust motor speed and direction
WRITE	Writes value to internal register
WRITE_STORE	Writes value to internal register and stores value in EEPROM
READ	Reads one or more of the internal registers (note: there are additional READ commands in ASCII mode)
RESTORE	Restores factory default values to internal registers and EEPROM
RESET	Performs a software reset

#### Interface/Programming Specific to Mode 3

Input Pins	Description
<i>RX</i> (J4 P13)	TTL data (logic 0 = 0V; logic 1 = 5V) sent to the controller
<i>TX</i> (J4 P14)	TTL data (logic 0 = 0V; logic 1 = 5V) sent from the controller, open collector output
<i>RS232_TX</i> (J4 P18)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent from the controller
<i>RS232_RX</i> (J4 P19)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent to the controller
<i>BAUD_SEL</i> (J4 P15 or J2 jumper)	5V = 19.2KBPS, 0V = 9.6KBPS
<i>ASCII</i> (J2 jumper)	Selects between binary and ASCII mode of communication
Programmable Registers	Description
TIMER	The timer register is used to determine the time frame between motor speed updates. The default setting is 50ms.
VELOCITYLIMIT	The allowable change in motor speed is determined by the contents of the VELOCITYLIMIT register.
PWMLIMIT	Default is 1023 setting to a lower value limits the duty cycle of the PWM signal driving the H-Bridge. For example 512 limits the duty-cycle to 50% (+/-). Can be used to current limit closed loop applications
Function Bit Settings	Description
FUNCTION.HEXMODE	If this bit is set and ASCII communication mode is being used, then all data returned from the controller will be returned as hexadecimal encoded ASCII.

**Figure 10: Serial (Open-Loop) Mode Connections**



#### 4.7 Mode 4: Serial PID Control

**How it Works:** This mode uses the serial data interface for commanding the controller to move to specific locations or velocities. Tuning of the PID is required for this mode, and a 2 or 3 channel quadrature encoder is required for feedback. For the communication protocol see section 5.

#### Summary of Serial Commands Used by Mode 4

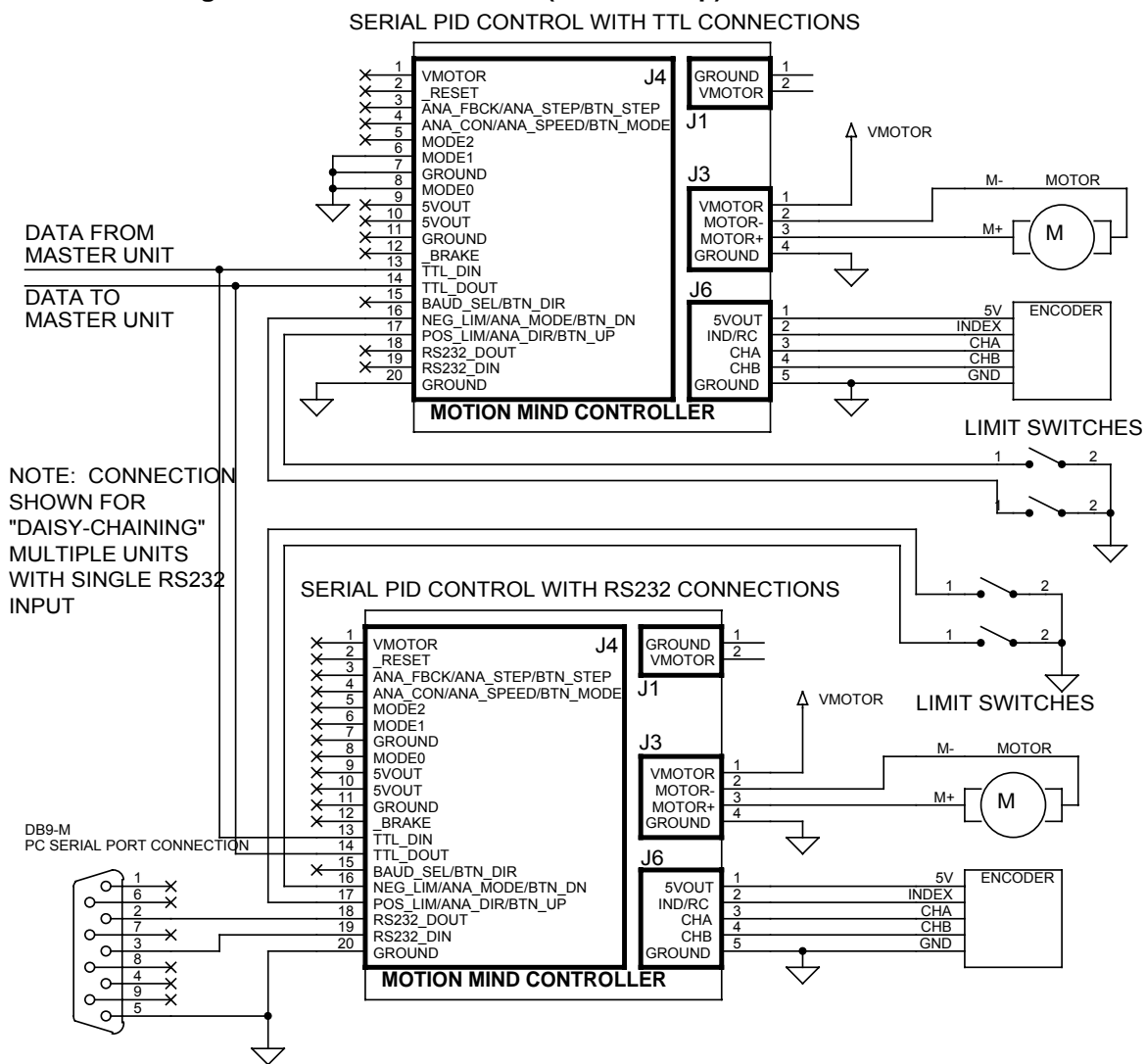
Command	Description
MOVETO_ABSOLUTE	Moves to desired position provided (example: you're at position +10,000 and you send a desired position of -1500, the motor moves to position -1500)
MOVETO_RELATIVE	Moves motors to desired position provided relative to current position (example: you're at position +10,000 and you send a desired position of -1500, the motor moves to position +8500)
MOVEAT_VELOCITY	Moves motor at desired velocity with no regard to position
WRITE	Writes value to internal register
WRITE_STORE	Writes value to internal register and stores value in EEPROM
READ	Reads one or more of the internal registers (note: there are additional READ commands in ASCII mode)
RESTORE	Restores factory default values to internal registers and EEPROM
RESET	Performs a software reset

#### Interface/Programming Specific to Mode 4

Input Pins	Description
RX (J4 P13)	TTL data (logic 0 = 0V; logic 1 = 5V) sent to the controller
TX (J4 P14)	TTL data (logic 0 = 0V; logic 1 = 5V) sent from the controller, open collector output
RS232_TX (J4 P18)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent from the controller
RS232_RX (J4 P19)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent to the controller
BAUD_SEL (J4 P15 or J2 jumper)	5V = 19.2KBPS, 0V = 9.6KBPS
ASCII (J2 jumper)	Selects between binary and ASCII mode of communication
POS_LIM (J4 P17)	Positive limit switch, after asserted (0V) the motor is prevented from moving in a positive direction
NEG_LIM (J4 P16)	Negative limit switch, after asserted (0V) the motor is prevented from moving in a negative direction
Programmable Registers	Description
POSITION	2's compliment 32-bit position value from the quadrature encoder, the master unit will typically read this register, but it may also be written to
VELOCITYLIMIT	When the FUNCTION.VELLIMIT bit is set the contents of this register are used to determine the allowable motor speed, this register contains positive values from 1 to 1023
VELOCITYFF	When the FUNCTION.VELLIMIT bit is set the contents of this register are used as the numerator of a Velocity Feed Forward constant that adds the desired motor speed to the output of the PID filter, this register contains positive values from 0 to 255
PTERM	The proportional "P" constant associated with the PID filter, this register contains positive values from 0 to 65535
ITERM	The integral "I" constant associated with the PID filter, this register contains positive values from 0 to 65535
DTERM	The derivative "D" constant associated with the PID filter, this register contains positive values from 0 to 65535
PIDSCALAR	The output of the PID filter is divided by 2 <sup>PIDSCALAR</sup>
TIMER	The timer register is used to debounce limit switch inputs, defaults to 50ms
DEADBAND (RCBAND in firmware 2,3,4)	When used in conjunction with the FUNCTION.ENABLEDEB bit a dead band around the desired position is created. If the actual position moves within the desired position +/- DEADBAND the PWMOUT signal is driven to 0.
VNLIMIT	Sets a virtual negative position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set. (Rev5+ firmware only)
VPLIMIT	Sets a virtual positive position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set. (Rev5+ firmware only)
PWMLIMIT	Default is 1023 setting to a lower value limits the duty cycle of the PWM signal driving the H-Bridge. For example 512 limits the duty-cycle to 50% (+/-). Can be used to current limit closed loop applications (Rev5+ firmware only)

Function Bit Settings	Description
FUNCTION.LOADPOS	Setting this bit causes the position value stored in EEPROM (see FUNCTION.SAVEPOS below) to be loaded from EEPROM on power up.
FUNCTION.SATPROT	If this bit is set the summation of errors used by the integral portion of the PID is halted when the motor drive signal is at full scale, and the sum of error terms is limited in overall size, this allows for the use of much larger ITERM settings and prevents integral wind-up
FUNCTION.SAVEPOS	When set the contents of the APOSITION register will be stored in EEPROM if the motor remains motionless for 10 minutes.
FUNCTION.VELLIMIT	When set position movements are velocity limited to the velocity of the +/-VELOCITYLIMIT register
FUNCTION.HEXMODE	If this bit is set and ASCII communication mode is being used, then all data returned from the controller will be returned as hexadecimal encoded ASCII.
FUNCTION.ENABLEDB	When set the contents of the RCBAND register are used to create a dead band around the desired position. <b>Not for use with MOVEAT_VELOCITY command.</b>
FUNCTION.VIRTLIMIT	Enables virtual software limits defined by VNLIMIT and VPLIMIT registers (Rev5+ firmware only)

Figure 11: Serial PID Control (Closed-Loop) Mode Connections



#### 4.8 Mode 5: Analog PID Control

**How it Works:** The Motion Mind controller can use an analog signal, like that from a potentiometer, as a motor feedback signal. When operated in this mode the control signal (desired position) is an analog signal, or in REV2 and above firmware, may be provided via the serial interface (the source of the desired position is determined by the FUNCTION.ADSERIAL bit). The PID will need to be “tuned” to the analog and mechanical system, which by its nature, lacks resolution. Because the analog inputs are read by a 10-bit analog-to-digital converter there are only 1023 possible positions. This means the PIDSCALAR should be reduced from its default setting, and the PTERM should be increased. No velocity control is allowed in closed-loop analog mode (there aren’t enough positions to make velocity control meaningful). The following PID settings may be a useful starting point for tuning your system.

```

VELOCITYLIMIT = 1023
VELOCITYFF = 0
PTERM = 30000
ITERM = 10
DTERM = 60000
PIDSCALAR = 11
FUNCTION.SATPROT = enabled
FUNCTION.VELLIMIT = disabled
    
```

#### Summary of Serial Commands Used by Mode 5

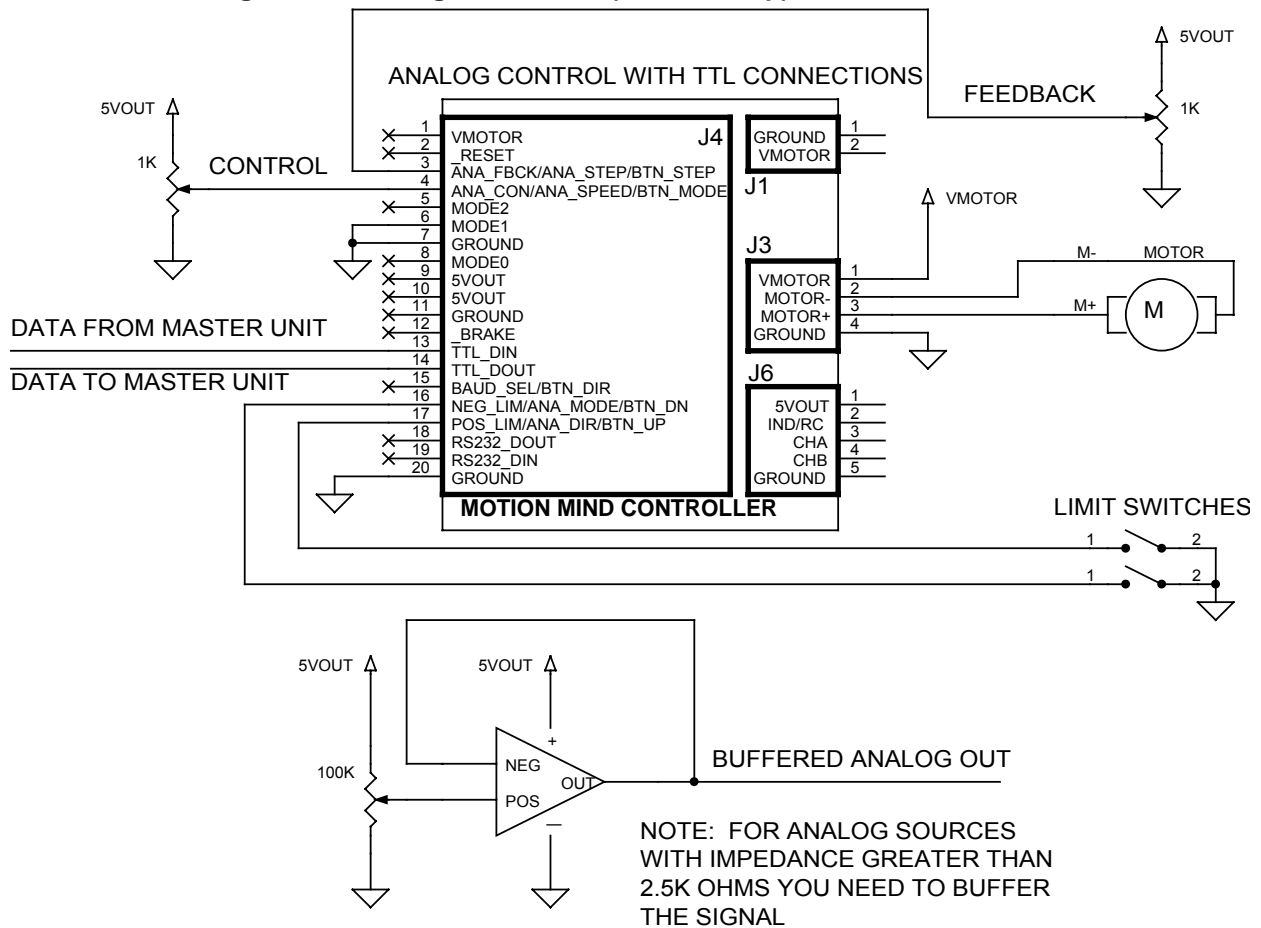
Command	Description
MOVETO_ABSOLUTE	When the FUNCTION.ADSERIAL bit is set (REV2 and above firmware only) this command may be used to provide the desired analog position. Position value should be limited to 0 to 1023.
MOVETO_RELATIVE	When the FUNCTION.ADSERIAL bit is set (REV2 and above firmware only) this command may be used to modify the desired analog position. Position value should be limited to 0 to 1023.
WRITE	Writes value to internal register
WRITE_STORE	Writes value to internal register and stores value in EEPROM
READ	Reads one or more of the internal registers (note: there are additional READ commands in ASCII mode)
RESTORE	Restores factory default values to internal registers and EEPROM
RESET	Performs a software reset

#### Interface/Programming Specific to Mode 5

Input Pins	Description
RX (J4 P13)	TTL data (logic 0 = 0V; logic 1 = 5V) sent to the controller
TX (J4 P14)	TTL data (logic 0 = 0V; logic 1 = 5V) sent from the controller, open collector output
RS232_TX (J4 P18)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent from the controller
RS232_RX (J4 P19)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent to the controller
BAUD_SEL (J4 P15 or J2 jumper)	5V = 19.2KBPS, 0V = 9.6KBPS
ASCII (J2 jumper)	Selects between binary and ASCII mode of communication
POS_LIM (J4 P17)	Positive limit switch, after asserted (0V) the motor is prevented from moving in a positive direction
NEG_LIM (J4 P16)	Negative limit switch, after asserted (0V) the motor is prevented from moving in a negative direction
ANA_CON (J4 P4)	0-5VDC analog position control signal, must have a source impedance of less than 2.5KΩ
ANA_FBCK (J4 P3)	0-5VDC analog feedback signal, must have a source impedance of less than 2.5KΩ

<b>Programmable Registers</b>	<b>Description</b>
POSITION	2's compliment 32-bit position value from the analog feedback potentiometer (limited to 0 to 1023 due to A/D 10-bit conversion)
PTERM	The proportional "P" constant associated with the PID filter, this register contains positive values from 0 to 65535
ITERM	The integral "I" constant associated with the PID filter, this register contains positive values from 0 to 65535
DTERM	The derivative "D" constant associated with the PID filter, this register contains positive values from 0 to 65535
PIDSCALAR	The output of the PID filter is divided by 2^PIDSCALAR
TIMER	The timer register is used to debounce limit switch inputs, defaults to 50ms
DEADBAND (RCBAND in firmware 2,3,4)	When used in conjunction with the FUNCTION.ENABLEDB bit a dead band around the desired position is created. If the actual position moves within the desired position +/- DEADBAND the PWMOUT signal is driven to 0.
VNLIMIT	Sets a virtual negative position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set. (Rev5+ firmware only)
VPLIMIT	Sets a virtual positive position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set. (Rev5+ firmware only)
PWMLIMIT	Default is 1023 setting to a lower value limits the duty cycle of the PWM signal driving the H-Bridge. For example 512 limits the duty-cycle to 50% (+/-). Can be used to current limit closed loop applications (Rev5+ firmware only)
<b>Function Bit Settings</b>	<b>Description</b>
FUNCTION.LOADPOS	Setting this bit causes the position value stored in EEPROM (see FUNCTION.SAVEPOS below) to be loaded from EEPROM on power up.
FUNCTION.SATPROT	If this bit is set the summation of errors used by the integral portion of the PID is halted when the motor drive signal is at full scale, and the sum of error terms is limited in overall size, this allows for the use of much larger ITERM settings and prevents integral wind-up
FUNCTION.SAVEPOS	When set the contents of the APOSITION register will be stored in EEPROM if the motor remains motionless for 10 minutes.
FUNCTION.VELLIMIT	When set position movements are velocity limited to the velocity of the +/- VELOCITYLIMIT register
FUNCTION.HEXMODE	If this bit is set and ASCII communication mode is being used, then all data returned from the controller will be returned as hexadecimal encoded ASCII.
FUNCTION.ADSERIAL	When set the controller does not use the analog control input to determine the desired position. Instead serial commands may be used (MOVETO_ABSOLUTE and MOVETO_RELATIVE). Care should be taken to keep the position values between 0 and 1023. This option is provided in REV2 and above firmware only.
FUNCTION.ENABLEDB	When set the contents of the DEADBAND register are used to create a dead band around the desired position. This option is provided in REV2 and above firmware only.
FUNCTION.RCPOS-ENCFDBCK	When set the feedback position signal is taken from an encoder connected to J6. This option is provided in REV5 and above firmware only.
FUNCTION.VIRTLIMIT	Enables virtual software limits defined by VNLIMIT and VPLIMIT registers (Rev5+ firmware only)

**Figure 12: Analog PID Control (Closed-Loop) Mode Connections**



Note: When connecting a feedback potentiometer you must ensure that the wiper voltage increases as the motor drives forward. This is a mechanical issue and will depend on your wiring and mounting method. Connecting the potentiometer in reverse will cause the motor to run in the wrong direction. For your initial tests, try to arrange your mechanical system so no damage can occur if the system is connected incorrectly.

#### 4.9 Mode 6: Factory Restore

**How it Works:** Placing the Motion Mind controller in mode 6 and pulling both limit switch pins to ground (J4 P17 and J4 P16) will cause the factory default register settings to be reloaded into EEPROM.

#### 4.10 Mode 7: Factory Test

**How it Works:** This mode isn't designed for use by the customer.

## 5.0 Communication Protocol

### 5.1 Overview

The communication settings and protocols contained within the Motion Mind controller allow for the system to work for a variety of applications. See section 3.5 for setting the communication method (ASCII or binary), setting the baud rate (9.6KBS or 19.2KBPS), or connecting to a TTL (logic) or RS232 (PC serial port) electrical levels. When operating in TTL mode the TTL\_DOUT (J4 P14) is open-collector. Therefore multiple Motion Mind controllers can share the TTL serial bus. Figure 10 under the Serial (Open-Loop) Control Mode shows a shared TTL bus as well as an RS232 connection scheme.

The serial data receive buffer can accept no more than 30 bytes regardless of the operating mode.

The serial communication protocol may be implemented in any mode of operation. Some commands may not have an effect in some of the operating modes. For example, using the MOVETO\_ABSOLUTE command while operating in serial open-loop control mode would not have any effect on motor control.

#### 5.1.1 Binary Protocol Specifics

The binary protocol is designed for more efficient exchange of information, and has built in requirements to limit the possibility of bad commands from being accepted by the controller. Each byte of data in a command must be received within 5ms of the previous byte. After 5ms have elapsed with no new data, any data received will be processed to see if it is a valid command. Valid commands are responded to within 10ms of processing (typically within 5-7.5ms). The WRITE\_STORE command that accesses EEPROM may take longer to process.

Each binary command has four specific components that must be correct in order for the message to be accepted. They are Command Byte, Address Byte, Length of Message, and Checksum. The Command Byte and Length of Message are determined by the command being sent. The Address Byte defaults to a value of "1" but may be programmed to a different value. Motion Mind controllers will only accept commands whose Address Byte matches the on board ADDRESS register. The Checksum is the sum of all bytes in the command prior to the Checksum. This value is limited to the lowest byte of the sum. For example, if summing the bytes results in 1256 (H'4E8'), the Checksum would be sent as 232 (H'E8').

Many commands will require multiple byte values as the data. For example a MOVETO\_ABSOLUTE command requires a 32 bit, 2's compliment, position value. The 32-bit value is broken down into 4 bytes and sent least significant byte first. Keep in mind that the Checksum must be calculated from each byte-sized piece of the command string (not by summing the Command Byte, Address Byte, and 32-bit position value). The example below shows a MOVETO\_ABSOLUTE command with the 2's compliment representation of -10,000 being sent (see section 3.8 for 2's compliment format).

**Example of Binary MOVETO\_ABSOLUTE Command**

	Command	Address	Data0	Data1	Data2	Data3	Checksum
<b>Decimal</b>	21	1	240	216	255	255	220
<b>Hex</b>	H'15'	H'01'	H'F0'	H'D8'	H'FF'	H'FF'	H'DC'

The data string (with commas shown for clarity but not part of the data string) would be sent in the following order.

Decimal: 21,1,240,216,255,255,220  
Hexadecimal: H'15', H'01', H'F0', H'D8', H'FF', H'FF', H'DC'

### 5.1.2 ASCII Protocol Specifics

The ASCII protocol allows the Motion Mind controller to be programmed or controlled with off-the-shelf terminal programs as opposed to custom software. When the RS232 circuitry is used, a serial control system may be implemented by connecting directly to a PC serial port.

The ASCII protocol is less strict than the binary protocol. Data is expected to be sent to the controller via typing and therefore there is no timing requirement between data bytes. When using the ASCII interface, commands are processed after a carriage return (H'0D' or <CR>) and line feed (H'0A' or <LF>) are received. Most terminal programs may be set up to send these characters when the enter key is pressed. The ASCII interface will also account for backspaces, by removing the character received prior to the backspace.

The ASCII protocol has five components required for processing. The first is the Command Character. The second is the Motion Mind address expressed as two characters, with a leading zero for number less than 10 (example: "01"). The third is a space character <SP> between the address and data characters. The fourth is the data, which varies in length (negative numbers are expressed with a minus "-" character in front of them). The fifth is the carriage return <CR>, followed by a linefeed (<LF>) that tells the controller to process the command. Note that the READ, WRITE, and WRITE\_STORE commands require an additional set of characters related to the index value of the register being accessed.

The example below shows a MOVETO\_ABSOLUTE command to position -10,000.

#### Examples of ASCII MOVETO\_ABSOLUTE and WRITE Command

```
P01<SP>-10000<CR><LF>
W01<SP>01<SP>120<CR><LF>
```

### 5.2 Response to Commands:

Typically all commands received will initiate an acknowledge response. The exception is the READ command that will always return data from the register(s) requested. However three bits in the FUNCTION register can be used to modify the typical response to a non-READ command.

#### 5.2.1 Binary Acknowledge Response:

A typical acknowledge response for the binary protocol is a decimal 06 (hexadecimal H'06'). The user may set the FUNCTION.RETPOS, FUNCTION.RETVEL, and/or FUNCTION.RETTIME bits to have the Motion Mind controller return the contents of the POSITION, VELOCITY, and/or TIME registers. Any combination of the three bits may be set, and the data will be returned in the order listed above. POSITION is returned as 4 bytes, MSB first, in 2's compliment format. VELOCITY is returned as 2 bytes, MSB first, in 2's compliment format. TIME is returned as 4 bytes, MSB first, always positive. When any of these function bits is set, the response will start with the address of the controller, followed by the data requested, and finish with a checksum (the lowest 8 bits of the sum of all data bytes in the response).

**Binary Response Examples**

CHANGE_SPEED (speed = 128) with no related FUNCTION bits set	H'14 ,	H'01'	H'80'	H'00 ,	H'95'		
Response	H'06 ,						
MOVETO_ABSOLUTE (position=1000) with FUNCTION.RETPOS bit set	H'15 ,	H'01'	H'E8 ,	H'03 ,	H'00'	H'00'	H'01 ,
Response	H'01 ,	H'E8 ,	H'03'	H'00 ,	H'00'	H'EC'	

### 5.2.2 ASCII Acknowledge Response

A typical response to an ASCII command is "OK<CR><LF>". If the command is not seen to be valid the Motion Mind controller will respond with a "BAD<SP>COMMAND<CR><LF>". This is true of all commands except the READ command. The READ command will return a response based on the register(s) read from. Setting the FUNCTION.RETPOS, FUNCTION.RETVEL, and/or FUNCTION.RETTIME bits will modify the typical response and return the requested data instead of "OK".

#### ASCII Response Examples

Function bit set	ASCII Response Example
FUNCTION.RETPOS	POSITION=100000<CR><LF>
FUNCTION.RETVEL	VELOCITY=121<CR><LF>
FUNCTION.RETTIME	TIME=23986<CR><LF>

### 5.3 CHANGE\_SPEED Command:

CHANGE\_SPEED is used to modify the speed and direction of a motor when the Motion Mind controller is operated in mode 3; serial open-loop control. The duty cycle of the motor drive signal is determined by the absolute value sent. The sign of the value determines the direction the motor turns. For example, sending a speed of -512 results in a motor driven in reverse with a 50% duty cycle. Sending 768 runs the motor forward at a 75% duty cycle.

The VELOCITYLIMIT and TIMER registers can be used to ramp motor speeds up and down. The contents of the TIMER register determine how often motor speed is updated, and the VELOCITYLIMIT register sets the allowable change per update. As an example, assume the motor is stopped, the VELOCITYLIMIT = 10, TIMER = 10 (50ms), and you send a CHANGE\_SPEED command of +1000. The motor drive signal will increase in increments of 10 every 50ms until it reaches 1000. This would take 5 seconds. To modify the VELOCITYLIMIT and TIMER registers use the WRITE\_STORE (sets the new value as the default value) or WRITE command.

#### 5.3.1 Binary Protocol

The components of the CHANGE\_SPEED command are the command byte, address byte, speed-direction bytes (2), and the checksum. The speed and direction data is sent as a value from -1023 to +1023, and is sent LSB first. -1023 corresponds to full speed reverse, 0 is stopped, and 1023 to full speed forward. The example below shows a speed change to 763 (H'2FB').

#### CHANGE\_SPEED Command Binary Example

	Command	Address	Data0	Data1	Checksum
Decimal	20	1	251	2	12
Response	6				
Hex	H'14'	H'01'	H'FB'	H'02'	H'12'
Response	H'06'				

### 5.3.2 ASCII Protocol

The following example shows a speed change to 763 using the ASCII protocol.

Command: C01<SP>763<CR><LF>  
Response: OK<CR><LF>  
How it appears on screen:  
C01 763  
OK

### 5.4 MOVETO\_ABSOLUTE Command:

In mode 4 (serial closed-loop) this command moves the motor to the position commanded. Be sure to tune your PID filter for your mechanical system, if you don't you won't get accurate, responsive position control. Also keep in mind that in mode 4 the encoder is decoded on a 4:1 ratio. So if you had a 500CPR encoder (500 pulses = 1 revolution) you would send 2000 to move the shaft 1 rotation. This command sends the absolute position as a 32 bit, 2's complement number. Therefore, if you command a position change of -10,000 it will move to position -10,000.

Setting the FUNCTION.VELLIMIT bit and loading the VELOCITYLIMIT register with a desired value will limit the average velocity during the movement.

#### 5.4.1 Binary Protocol

The components of the MOVETO\_ABSOLUTE command are the command byte, address byte, desired position bytes (4), and the checksum. The position data is 2's complement and is sent LSB first. The example below shows a move to position -100,046 (H'FFFE7932'). Note that in mode 5, closed-loop analog control mode the position value is restricted to 0-1023.

**MOVETO\_ABSOLUTE Command Binary Example**

	Command	Address	Data0	Data1	Data2	Data3	Checksum
Decimal	21	1	50	121	254	255	190
Response	6						
Hex	H'15'	H'01'	H'32'	H'79'	H'FE'	H'FF'	H'BE'
Response	H'06'						

#### 5.4.2 ASCII Protocol

The following example shows a MOVETO\_ABSOLUTE position command of -100,046 using the ASCII protocol.

Command: P01<SP>-100046<CR><LF>  
Response: OK<CR><LF>  
How it appears on screen:  
P01 -100046  
OK

### 5.5 MOVETO\_RELATIVE Command:

In mode 4 (serial closed-loop) this command moves the motor from the current position to a position relative to the starting position. For example if you are starting at position +10,000, and you execute a MOVETO\_RELATIVE position of -2,000, you will end up at position +8,000. Be sure to tune your PID filter for your mechanical system, if you don't you won't get accurate, responsive position control. Also keep in mind that in mode 4 the encoder is decoded on a 4:1 ratio. So if you had a 500CPR encoder (500 pulses = 1 revolution) you would send 2,000 to move the shaft 1 rotation. This command sends the relative position as a 32 bit, 2's complement number.

Setting the FUNCTION.VELLIMIT bit and loading the VELOCITYLIMIT register with a desired value will limit the average velocity during the movement.

### 5.5.1 Binary Protocol

The components of the MOVETO\_RELATIVE command are the command byte, address byte, desired position bytes (4), and the checksum. The position data is 2's compliment and is sent LSB first. The example below shows a move to position +100,000 (H'186A0').

**MOVETO\_RELATIVE Command Binary Example**

	Command	Address	Data0	Data1	Data2	Data3	Checksum
Decimal	22	1	160	134	1	0	62
Response	6						
Hex	H'16'	H'01'	H'A0'	H'86'	H'01'	H'00'	H'3E'
Response	H'06'						

### 5.5.2 ASCII Protocol

The following example shows a MOVETO\_RELATIVE position command of +100,000 using the ASCII protocol.

Command: M01<SP>100000<CR><LF>  
 Response: OK<CR><LF>  
 How it appears on screen:  
 M01 100000  
 OK

### 5.6 MOVEAT\_VELOCITY Command:

This command is used in mode 4 (serial closed-loop). The MOVEAT\_VELOCITY command uses the feedback from the quadrature encoder to move the motor at a constant speed. You will need to make adjustments to the PID filter settings to get the best control results. The desired velocity is 2 bytes sent as a 2's compliment number. The value ranges from -32,768 to +32767, where negative numbers represent moving the motor in reverse. You'll need to account for the fact that the quadrature encoder is decoded at a 4:1 ratio, and the velocity measurement occurs over a 5ms period (a single loop through the PID).

If you know the desired rotation rate of your output shaft, you may use the following equation to determine the desired velocity setting. If you want to move the motor in reverse simply send a negative value.

$$VELOCITY = \frac{MotorShaftRotations / Second \times CPR \times GearRatio}{50}$$

#### 5.6.1 Binary Protocol

The components of the MOVEAT\_VELOCITY command are the command byte, address byte, velocity bytes (2), and checksum. The velocity data is sent as a 2's compliment number, LSB first. The example below shows a commanded velocity of -200.

**MOVEAT\_VELOCITY Command Binary Example**

	Command	Address	Data0	Data1	Checksum
Decimal	23	1	56	255	79
Response	6				
Hex	H'17'	H'01'	H'38'	H'FF'	H'4F'
Response	H'06'				

### 5.6.2 ASCII Protocol

The following example shows a motor velocity of -200 using the ASCII protocol.

Command: V01<SP>-200<CR><LF>  
 Response: OK<CR><LF>  
 How it appears on screen:  
     V01 -200  
     OK

### 5.7 WRITE Command:

The WRITE command is used to modify internal registers in the Motion Mind controller. It can be used in any mode of operation. Some registers are read/write registers and some are read only. The WRITE command stores the data you send in the register that you select, but does not write the data to EEPROM (data in EEPROM is loaded on power up into the internal registers). This allows you to modify registers without changing the default settings. If you'd like to change the default settings as well, use the WRITE\_STORE command. Each register has an index number associated with it, and has specific requirements for its contents. See section 6.0 for register index numbers, register descriptions, and register content requirements.

#### 5.7.1 Binary Protocol

The components of the WRITE command are the command byte, address byte, register index number, data (1,2, or 4 bytes), and the checksum.

**WRITE POSITION Register (-10,000) Binary Example**

	Command	Address	Index	Data0	Data1	Data2	Data3	Check sum
Decimal	24	1	0	240	216	255	255	223
Response	6							
Hex	H'18'	H'01'	H00'	H'F0'	H'D8'	H'FF'	H'FF'	H'DF'
Response	H'06'							

**WRITE VELOCITYLIMIT Register (100) Binary Example**

	Command	Address	Index	Data0	Data1	Check sum
Decimal	24	1	1	100	0	126
Response	6					
Hex	H'18'	H'01'	H01'	H'64'	H'00'	H'7E'
Response	H'06'					

**WRITE VELOCITYFF Register (128) Binary Example**

	Command	Address	Index	Data0	Check sum
Decimal	24	1	2	128	155
Response	6				
Hex	H'18'	H'01'	H02'	H'80'	H'9B'
Response	H'06'				

### 5.7.2 ASCII Protocol

The ASCII protocol requires the command and address character, a space character, two characters for the index value of the register you're writing to, another space character, the data you are writing, and a carriage return and line feed.

Command: W01<SP>00<SP>-10000<CR><LF>  
Response: OK<CR><LF>  
How it appears on screen:  
W01 00 -10000  
OK

Command: W01<SP>01<SP>100<CR><LF>  
Response: OK<CR><LF>  
How it appears on screen:  
W01 01 100  
OK

Command: W01<SP>02<SP>128<CR><LF>  
Response: OK<CR><LF>  
How it appears on screen:  
W01 02 128  
OK

### 5.8 WRITE\_STORE Command:

The WRITE\_STORE command is used to modify internal registers in the Motion Mind controller. It can be used in any mode of operation. Some registers are read/write registers and some are read only. The WRITE\_STORE command stores the data you send in the register that you select and copies data to EEPROM (data in EEPROM is loaded on power up into the internal registers). This allows you to modify the default settings. Each register has an index number associated with it, and has specific requirements for its contents. See section 6.0 for register index numbers, register descriptions, and register content requirements.

Writing to EEPROM may take up to 40ms. The motor is automatically stopped when the command is deemed valid. When operating in this mode you should avoid writing to EEPROM (using the WRITE\_STORE command) until the motor is stopped.

The EEPROM in the Motion Mind controller has a rating of 1,000,000 write cycles at room temperature. Avoid developing control systems that continuously write to EEPROM.

#### 5.8.1 Binary Protocol

The components of the WRITE\_STORE command are the command byte, address byte, register index number, data (1,2, or 4 bytes), and the checksum.

**WRITE POSITION Register (-10,000) Binary Example**

	Command	Address	Index	Data0	Data1	Data2	Data3	Check sum
Decimal	25	1	0	240	216	255	255	224
Response	6							
Hex	H'19'	H'01'	H00'	H'F0'	H'D8'	H'FF'	H'FF'	H'E0'
Response	H'06'							

**WRITE VELOCITYLIMIT Register (100) Binary Example**

	Command	Address	Index	Data0	Data1	Check sum
Decimal	25	1	1	100	0	127
Response	6					
Hex	H'19'	H'01'	H01'	H'64'	H'00'	H'7F'
Response	H'06'					

**WRITE VELOCITYFF Register (128) Binary Example**

	Command	Address	Index	Data0	Check sum
Decimal	25	1	2	128	156
Response	6				
Hex	H'19'	H'01'	H02'	H'80'	H'9C'
Response	H'06'				

**5.8.2 ASCII Protocol**

The ASCII protocol requires the command and address character, a space character, two characters for the index value of the register you're writing to, another space character, the data you are writing, and a carriage return and line feed.

Command: S01<SP>00<SP>-10000<CR><LF>

Response: OK<CR><LF>

How it appears on screen:

S01 00 -10000

OK

Command: S01<SP>01<SP>100<CR><LF>

Response: OK<CR><LF>

How it appears on screen:

S01 01 100

OK

Command: S01<SP>02<SP>128<CR><LF>

Response: OK<CR><LF>

How it appears on screen:

S01 02 128

OK

**5.9 READ Command:**

Utilizing the READ command allows you to read the internal registers of the Motion Mind controller. You may read all or some of the registers using this command. See section 6.0 for register index numbers, register descriptions, and register content.

### 5.9.1 Binary Protocol

The READ command in the binary protocol was designed to be both efficient and versatile. There are four components to the READ command. They are the command byte, address byte, read data, and the checksum. The read data is a 32-bit value where each bit location is related to an index value for a specific register (the highest 8-bits are left for expansion). If a bit is set then the contents of that register are returned as the response. You may set one bit, some of the bits, or all of the bits. This allows you to tailor the READ command to receive only the data you're interested in. The read data is sent LSB first. Data is returned in the order of the bits set. For example, if you set bit 0 and bit 1, the contents of register 0 (POSITION – 4 bytes) and register 1 (VELOCITYLIMIT – 2 bytes) would be returned in that order.

For example, if you wanted to read the contents of index 2 (VELOCITYFF), index 8 (PIDSCALAR), index 19 (ANALOGCON, and index 20 (ANALOGFBCK) your read data would look like this...

Read Data3 MSB (registers 24-31)	B'0000 0000'(binary)	H'00'(hexadecimal)	0(decimal)
Read Data2 (registers 16-23)	B'0001 1000'(binary)	H'18'(hexadecimal)	24(decimal)
Read Data1 (registers 8-15)	B'0000 0001'(binary)	H'01'(hexadecimal)	1(decimal)
Read Data0 LSB (registers 0-7)	B'0000 0100'(binary)	H'04'(hexadecimal)	4(decimal)

There are 32 bits (0 through 31) located in the 4 read data bytes, and bits 2,8,19, and 20 are set. This would cause the controller to respond with the contents of registers 2,8,19, and 20, in that order. The data string in decimal would be sent as follows (commas are shown for clarity and are not part of the data).

26(command), 1(address), 4(data0), 1(data1), 24(data2), 0(data3), 56(checksum)

The response to a READ command in binary mode consists of the address, the data requested (LSB first), and a checksum byte.

**READ POSITION register (bit 0 contents are 20,000) Example**

	Command	Address	Data0	Data1	Data2	Data3	Checksum
READ command	26	1	1	0	0	0	28
	Address	Data0	Data1	Data2	Data3	Checksum	
READ response	1	32	78	0	0	111	

### 5.9.2 ASCII Protocol

The ASCII protocol version of the READ command differs from the binary protocol. In ASCII mode the READ command mirrors the WRITE command. The user sends the command character, followed by two address characters (01-99), a space character, and then a two-character index value. As with other ASCII commands the command is processed after a carriage return (H'0D' or <CR>) and linefeed (H'0A' or <LF>) characters are sent. The index value determines which register contents are returned. The response generally consists of an ASCII label for the register returned. Here's an example of reading the PTERM register.

```
Command: R01<SP>04<CR><LF>
Response: PTERM=6000<CR><LF>
How it appears on screen:
R01 04
PTERM=6000
```

If the FUNCTION.HEXMODE bit is set then the data is returned as ASCII encoded hex data. It takes the Motion Mind controller less time to convert the data to this format, so a quicker response to READ commands can be accomplished while in hex mode.

```
Command: R01<SP>04<CR><LF>
```

Response: PTERM=\$1770<CR><LF>  
How it appears on screen:  
R01 04  
PTERM=\$1770

### 5.9.2.1 Special Forms of ASCII Mode READ Command

The ASCII READ command has three special read commands that are accessed by sending 97,98, or 99 as the register index value.

**READ 97-STATUS register bits:** Sending a READ index of 97 returns an ASCII string that shows the state of the various bits of the STATUS register. An ASCII "ON" is returned if the bit is set, and "OFF" is sent if the bit is clear. Comments to the right of the response are not returned.

Command: R01<SP>97<CR><LF>  
Response: S0OFF<CR><LF> 'Positive limit switch is not asserted  
S1OFF<CR><LF> 'Negative limit switch is not asserted  
S2OFF<CR><LF> 'Brake input is not asserted  
S3ON<CR><LF> 'Shows Index input bit is asserted  
S4OFF<CR><LF> 'Bad RC measurement has not been detected

How it appears on screen:  
R01 97  
S0OFF  
S1OFF  
S2OFF  
S3ON  
S4OFF

**READ 98-FUNCTION register bits:** Sending a READ index of 98 returns an ASCII string that shows the state of the various bits of the FUNCTION register. An ASCII "ON" is returned if the bit is set, and "OFF" is sent if the bit is clear. Comments to the right of the response are not returned.

Command: R01<SP>98<CR><LF>  
Response: F0OFF<CR><LF> 'Load position on power up is not enabled  
F1OFF<CR><LF> 'Return position as part of response is not enabled  
F2OFF<CR><LF> 'Return velocity as part of response is not enabled  
F3OFF<CR><LF> 'Return time elapsed as part of response is not enabled  
F4ON<CR><LF> 'Saturation limit of ITERM summation is enabled  
F5OFF<CR><LF> 'Save position (EEPROM) if no move is not enabled  
F6ON<CR><LF> 'Velocity limit mode is enabled  
F7OFF<CR><LF> 'ASCII hex response mode is not enabled  
F8OFF<CR><LF> 'Use last valid RC signal is not enabled  
F9OFF<CR><LF> 'Ignore analog input for velocity limit is not enabled  
F10OFF<CR><LF> 'Mode5 won't receive serial data position commands  
F11OFF<CR><LF> 'Mode4 and 5 not using a dead band  
F12OFF<CR><LF> 'RC position control mode is not enabled  
F13ON<CR><LF> 'Virtual position limits are enabled

How it appears on screen:  
R01 98  
F0OFF  
F1OFF  
F2OFF  
F3OFF  
F4ON  
F5OFF  
F6ON  
F7OFF  
F8OFF  
F9OFF  
F10OFF  
F11OFF  
F12OFF  
F13ON

**READ 99-Read all registers:** The READ all registers command returns the contents of all internal registers in a comma-delimited format. Below is an example of the command and response.

Command: R01<SP>99<CR><LF>

Response: 0,9,208,80,6342,48,0,1,13,50,4403,2661,100,0,0,473561,0,9,4,580,577,0,0,<CR><LF>

How it appears on screen:

R01 99

0,9,208,80,6342,48,0,1,13,50,4403,2661,100,0,0,473561,0,9,4,580,577,0,0,

### 5.10 RESTORE Command:

The restore command restores the factory default values to EEPROM. Since this command writes to EEPROM, the motor is stopped after the command is deemed valid. See section 6.0 for register definitions and default values.

#### 5.10.1 Binary Protocol

The binary command consists of the command byte, address byte, and a checksum.

**RESTORE Command Binary Example**

	Command	Address	Checksum
Decimal	27	1	28
Response	6		
Hex	H'1B'	H'01'	H'1C'
Response	H'06'		

#### 5.10.2 ASCII Protocol

Command: X01<CR><LF>

Response: OK<CR><LF>

How it appears on screen:

X01

OK

### 5.11 RESET Command:

Sending the RESET command causes the Motion Mind Controller to stop the motor and software reset.

#### 5.11.1 Binary Protocol

The binary command consists of the command byte, address byte, and a checksum.

**RESET Command Binary Example**

	Command	Address	Checksum
Decimal	28	1	29
Response	6		
Hex	H'1C'	H'01'	H'1D'
Response	H'06'		

#### 5.11.2 ASCII Protocol

Command: Y01<CR><LF>

Response: OK<CR><LF>

How it appears on screen:

Y01

OK

### 5.12 SELF-TEST Command:

This command is not to be used by the customer. If this command is sent you will probably need to remove power from the controller and restart the system. The self-test mode is only accessed in ASCII mode. The command is given here to prevent users from inadvertently sending this data to the controller.

Command: Z01<CR><LF>

## 6.0 Register Definitions – Function/Status Bits

### 6.1 Overview:

The Motion Mind controller maintains quite a bit of information related to its operational state in registers that may be read by a master controller unit, or used for testing. Many of these registers can also be written to, allowing the user to modify functionality. The registers are 1, 2, or 4 bytes in length. Some expect 2's compliment values (those with a negative value, see section 3.8). All registers have an associated index value that allows the user to access that register through serial commands (such as READ and WRITE commands). The FUNCTION and STATUS register contain bit-fields, where each bit can change functionality or provide information regarding your system.

**Figure13: Register Index and Format Table**

Index	Register	R/W	Size (Bytes)	Range	Default
0	POSITION	R/W	4	-2,147,483,648 to +2,147,483,647	0
1	VELOCITYLIMIT	R/W	2	+1 to +1,023	1023
2	VELOCITYFF	R/W	1	0 to +255	128
3	FUNCTION	R/W	2	0 to +65,535	0
4	PTERM	R/W	2	0 to +65,535	6000
5	ITERM	R/W	2	0 to +65,535	35
6	DTERM	R/W	2	0 to +65,535	0
7	ADDRESS	R/W	1	0 to +255	1
8	PIDSCALAR	R/W	1	0 to +32	14
9	TIMER	R/W	1	+1 to +255	10
10	RCMAX	R/W	2	0 to +65,535	2457
11	RCMIN	R/W	2	0 to +65,535	1229
12	RCBAND	R/W	2	0 to +65,535	40
13	RCCOUNT	R	2	0 to +65,535	N/A
14	VELOCITY	R	2	-32,768 to +32,767	N/A
15	TIME	R	4	0 to + 4,294,967,295	N/A
16	STATUS	R	2	0 to +65,535	N/A
17	REVISION	R	1	0 to +255	N/A
18	MODE	R	1	0 to +7	N/A
19	ANALOGCON	R	2	0 to +1023	N/A
20	ANALOGFBCK	R	2	0 to +1023	N/A
21	PWMOUT	R	2	-1023 to +1023	N/A
22	INDEXPOS	R	4	-2,147,483,648 to +2,147,483,647	N/A
23	VNLIMIT	R/W	4	-2,147,483,648 to +2,147,483,647	-100000
24	VPLIMIT	R/W	4	-2,147,483,648 to +2,147,483,647	+100000
25	PWMLIMIT	R/W	2	1 to +1023	1023
26	DEADBAND	R/W	0	0 to +1023	15
27	DESIREDPOSITIO N	R	4	-2,147,483,648 to +2,147,483,647	N/A

6.2 Register Descriptions:

Figure14: Register Descriptions Table

Index	Register	Description
0	POSITION	This register contains the current position count from the quadrature encoder x4. A single revolution by a 500CPR encoder would result in a POSITION of 2000. The mechanical system should prevent the position count from exceeding the positive or negative limits.
1	VELOCITYLIMIT	This register functions in a variety of ways based on the mode of operation. In modes 0,1, and 2, (RC, Button, Analog open-loop) the contents of this register may be used in place of the analog input to place a step limit on motor speed changes (setting FUNCTION.ADSTEP does this). In mode 3 (serial open-loop), this register defines the step limit allowed for motor speed changes. In mode 4 (serial closed-loop) the contents of this register are used to limit motor position changes per PID update if the FUNCTION.VELLIMIT bit is set.
2	VELOCITYFF	In mode 4 (serial closed-loop mode) If FUNCTION.VELLIMIT is set then this register provides a proportional gain outside of the PID loop. The extra proportional gain is based on the desired velocity times the VELOCITYFF register divided by 255. This is a velocity feed-forward loop.
3	FUNCTION	The FUNCTION register contains a number of bits that each can enable or disable specific functions. This is a 16 bit register and not all bits are used. See section 6.3 for details.
4	PTERM	The proportional term of the PID provides the brute force in motor movement. Setting this value to 0 removes the P portion from the PID.
5	ITERM	The integral term of the PID is summed over time to provide to small errors to be corrected. This value is typically small. Hi integral values will force the control to oscillate. Enabled the FUNCTION.SATPROT function can allow for increased ITERM settings. Setting this value to 0 removes the I portion from the PID.
6	DTERM	The derivative term of the PID provides acts as a drag on motor movement reducing the impact of step commands. The derivative setting has the least impact on the PID and will often be set to 0. Setting this value to 0 removes the D portion from the PID.
7	ADDRESS	The address value that the Motion Mind controller will accept communication to. To place multiple controllers on the same communication bus, you'll need to program each with a different address value.
8	PIDSCALAR	The output of the PID filter will typically be a large number. The PIDSCALAR is the number of "divide-by-two's" the output goes through before being used to determine the output drive signal (PWMOU). For small count encoder you would want to reduce the PIDSCALAR (making the error signal output larger).
9	TIMER	This is the debounce and motor speed update timer. It is in increments of 5ms. To increase motor responsiveness to commands (in open-loop modes) reduce this number and increase the VELOCITYLIMIT (or increase the analog signal being used to control motor speed changes). To slow motor speed changes (create ramping of motor changes) do the opposite. This debounce rate is also used to monitor the POS_LIM, and NEG_LIM input pins. The default setting is 50ms (10 x 5ms) and could be reduced to cause faster response to limit switches.
10	RCMAX	The R/C pulse width associated with maximum forward speed in increments of 814ns. The default setting is 2ms and pulse widths from RCMAX to RCMAX + 0.5ms are treated as RCMAX in width. Pulse widths greater than RCMAX + 0.5ms are considered bad signals.
11	RCMIN	The R/C pulse width associated with maximum reverse speed in increments of 814ns. The default setting is 1ms and pulse widths from RCMIN to RCMIN - 0.5ms are treated as RCMIN in width. Pulse widths less than RCMIN - 0.5ms are considered bad signals.
12	RCBAND	Pulse widths of 1.5ms +/- RCBAND (in 814ns increments) are treated as 1.5ms signals (stopped). The default puts pulse widths from 1.467ms to 1.533ms are treated as 1.5ms signals.
13	RCCOUNT	This is the raw R/C pulse width measured while in mode 0. It is expressed in increments of 814ns.

Index	Register	Description
14	VELOCITY	Used in mode 4 (serial closed-loop mode) this register contains an average of the last 64 velocity measurements. Each measurement occurs over a 5ms period and equals 4X the actual encoder count. Therefore an average velocity of 200 relates to 50 encoder counts during a 5ms period.
15	TIME	Elapsed time since power up in 5ms increments. This 32 bit positive number can count for 248 days before rolling over.
16	STATUS	A 16 bit register containing bits related to input pins or other status conditions. Not all bits are used. See section 6.4 for details.
17	REVISION	The firmware revision of the operating system. This may be compared to online errata sheets if known bugs exist in the firmware.
18	MODE	Current operating mode
19	ANALOGCON	Raw analog measurement at J4 P4. The value will range from 0 to 1023 in increments of 5VOUT/1023 (typically 4.89mV)
20	ANALOGFBCK	Raw analog measurement at J4 P3. The value will range from 0 to 1023 in increments of 5VOUT/1023 (typically 4.89mV)
21	PWMOUT	The actual PWM drive signal where the sign determines motor direction. Ranges from -1023 to +1023 with 0 being stopped.
22	INDEXPOS	This 32-bit 2's compliment number relates to the last position measured before the index input is asserted. Using this value the user can locate the approximate index position. Then using the STATUS.INDEX bit a controller can home in on the actual index position. The index pulse comes from a 3 <sup>rd</sup> channel on many quadrature encoders, and occurs with each encoder revolution.
23	VNLIMIT (Rev5+ firmware only)	This 32-bit 2's compliment number is used with the FUNCTION.VIRTLIMIT setting to establish a virtual negative stop limit for closed loop position control modes of operation. The VNLIMIT represents a lower boundary for negative moving positions. For example if the VNLIMIT is set for -10000, and the controller is directed to move to -20000, it will be forced to stop at -10000. The STATUS.VNLIMIT bit will be set to "1" when this boundary is reached.
24	VPLIMIT (Rev5+ firmware only)	This 32-bit 2's compliment number is used with the FUNCTION.VIRTLIMIT setting to establish a virtual positive stop limit for closed loop position control modes of operation. The VPLIMIT represents an upper boundary for positive moving positions. For example if the VPLIMIT is set for +10000, and the controller is directed to move to +20000, it will be forced to stop at +10000. The STATUS.VPLIMIT bit will be set to "1" when this boundary is reached.
25	PWMLIMIT (Rev5+ firmware only)	This 16 register (limited to +1 to +1023) restricts the PWM h-bridge drive signal. At the default value (1023) the PWM output can range from -1023 to +1023 (-100% and + 100% duty cycle respectively). Setting the PWMLIMIT to 512 would keep the drive signal to the h-bridge between -50% and +50%. For closed loop modes this register can be used to current limit the h-bridge by preventing higher duty cycles from occurring.
26	DEADBAND (Rev5+ firmware only)	This 16-bit register (limited 0 to +1023) is used to establish a "dead band" (stopped motor) around 2.5V DC for bi-directional analog control mode. Each bit represents a 4.88mV step. So in order to create a dead band from 2.4V to 2.6V this register would be set to 20 (4.88mV*20 = 0.98V).  When FUNCTION.ENABLEDB is set this register will also be applied to closed loop modes to create a dead band around the commanded position.
27	DESIRED POSITION (Rev5+ firmware only)	This 32-bit 2's compliment register holds the commanded position internally used by the controller. This is a read only register. In closed-loop modes this is the position the controller has been commanded to move to. In some modes of operation this register may be useful to have access (specifically MODE0 RC Position Control Mode).

### 6.3 FUNCTION Register Bits:

The FUNCTION register contains a number of bits that are used to enable or disable certain functions.

**Figure15: FUNCTION Register Bits**

Bit	Name	Description
0	POSPWRUP	When set: The POSITION register will be loaded from EEPROM on power up. This can be used in conjunction with the FUNCTION.SAVEPOS mode to restore the last position to the system after power is removed and then reapplied. An external controller could also use the WRITE_STORE command to store current position data to EEPROM and then setting this bit would restore that position on power-up. When clear: The POSITION register defaults to 0 on power up.
1	RETPOS	When set: The typical acknowledge response to non-READ commands is replaced with the contents of the POSITION register. This can be used with the FUNCTION.RETVEL and FUNCTION.RETTIME modes. When clear: No effect.
2	RETVEL	When set: The typical acknowledge response to non-READ commands is replaced with the contents of the VELOCITY register. This can be used with the FUNCTION.RETPOS and FUNCTION.RETTIME modes. When clear: No effect.
3	RETTIME	When set: The typical acknowledge response to non-READ commands is replaced with the contents of the TIME register. This can be used with the FUNCTION.RETVEL and FUNCTION.RETPOS modes. When clear: No effect.
4	SATPROT	When set: The integral summation is limited to +/-4096, this severely limits the ability of the integral to build up over time to account for small errors. But it may also prevent large errors from accumulating and swamping out the proportional part of the PID. When clear: The integral summation can build up to +/-120,000.
5	SAVEPOS	When set: If operating in mode 4 (serial closed-loop), the POSITION register will be stored in EEPROM (non-volatile memory) if the motor velocity stays at 0 continuously for 5 minutes. When clear: no effect.
6	VELLIMIT	When set: If operating in mode 4 (serial closed-loop), and using position control commands (not MOVEAT_VELOCITY) the absolute value of the motor velocity is limited to the value in VELOCITYLIMIT register. When clear: Motor movements are at the highest speed possible.
7	HEXMODE (firmware 1,2,3, and 4) ACTIVESTOP (REV5 and above firmware only)	Firmware 1,2,3, and 4, removed in firmware 5+ When set: ASCII READ commands (or FUNCTION.RETxxxx modes) will return data as hexadecimal ASCII data. For example, in a 2 byte register +10,000 would be returned as \$2710. There are a couple of reasons to use this mode. The first is that the conversion of binary to ASCII takes some time, and you may have faster responses to serial commands in hex mode. The second is that the READ command with an index of 99 (read all) may cause glitches in mode 4 (serial closed-loop) because of the time required to convert all of the registers to ASCII characters. If this is the case then hex mode will reduce processing time. When clear: ASCII data is returned without conversion to hex format. Firmware 5+ When set: When motor control reaches a stop condition both leads of the motor are tied to ground. When clear: Motor leads float when stop occurs
8	LASTRC	When set: In mode 0 (R/C open loop) if a bad R/C pulse is received (no pulse, pulse too long, or pulse too short) the average of the last 64 valid pulses will be used to determine motor speed. Turning on/off an RC transmitter can still cause erroneous signals of valid duration to be received. This function bit reduces but does not remove this possibility. When clear: An invalid R/C pulse will be treated as a 1.5ms (stopped) signal.

Bit	Name	Description
9	ADSTEP	When set: Modes 0,1, and 2 use an analog input to select a motor speed change limit. It may be desirable to do away with this analog input. if that's the case setting this bit forces the controller to take it's step limit from the contents of the VELOCITYLIMIT register. When clear: Modes 0,1, and 2 use an analog input to determine motor step changes allowed from one update to the next.
10	ADSERIAL (REV2 and above firmware only)	When set: In Mode 5 the source of the desired position is serial commands (MOVETO_ABSOLUTE or MOVETO_RELATIVE). Care should be taken to ensure that the commanded position is a value between 0 and 1023. When clear: In Mode 5 the source of the desired position is the analog control signal (0-5V) provided at ANA_CON (J4 P4).
11	ENABLEDB (REV2 and above firmware only)	When set: In modes 4 and 5 the contents of the DEADBAND** register are used to set a dead band around the desired position. If (POSITION – DEADBAND) <= Desired Position <= (POSITION + DEADBAND) then the PWMOUT signal is forced to 0. . Not for use with MOVEAT_VELOCITY command. When clear: No dead band exists in modes 4 and 5. <b>**Note-</b> In firmware 2,3,4 the register RCBAND was used to set the dead band, and was shared with the RCBAND setting.
12	RCPOS (REV4 and above firmware only) RCPOS- ENCFDBCK (REV5 and above firmware only)	MODE0 When set: Uses the analog feedback input as the actual position signal and converts the 1-2ms R/C signal to a desired position value. When clear: Mode 0 operates in speed control mode. MODE5 When set: The desired position is derived from the analog input (J4 P4) 0-5V 0-1023 positions, but the feedback comes from an encoder attached to J6. When clear: The desired position is derived from the analog control input (J4 P4) and the position feedback is derived from the analog feedback input (J4 P3).
13	VIRTLIMIT (REV5 and above firmware only)	When set: Virtual limit settings are used to restrict movement in closed loop modes. VNLIMIT and VPLIMIT registers determine the position limits. When clear: Virtual position limits are not used

#### 6.4 STATUS Register Bits:

The STATUS register contains a number of bits that indicate i/o states or other status related information.

Figure16: STATUS Register Bits

Bit	Name	Description
0	NEGLIMIT	Set when the input <i>NEG_LIM</i> (J4 P16) is at 0V
1	POSLIMIT	Set when the input <i>POS_LIM</i> (J4 P17) is at 0V
2	BRAKE	Set when the input <i>_BRAKE</i> (J4 P12) is at 0V
3	INDEX	Set when the input <i>IND/RC</i> (J6 P2) is at 0V
4	BADRC	Set when the R/C pulse is not received within 50ms, is shorter than RCMIN – 0.5ms, or is longer than RCMAX + 0.5ms
5	VNLIMIT	Set when the negative virtual limit position is reached
6	VPLIMIT	Set when the positive virtual limit position is reached

## 7.0 Disclaimer / Warrantee

**Disclaimer of Liability and Accuracy:** Information provided by Solutions Cubed is believed to be accurate and reliable. However, Solutions Cubed assumes no responsibility for inaccuracies or omissions. Solutions Cubed assumes no responsibility for the use of this information and all use of such information shall be entirely at the user's own risk.

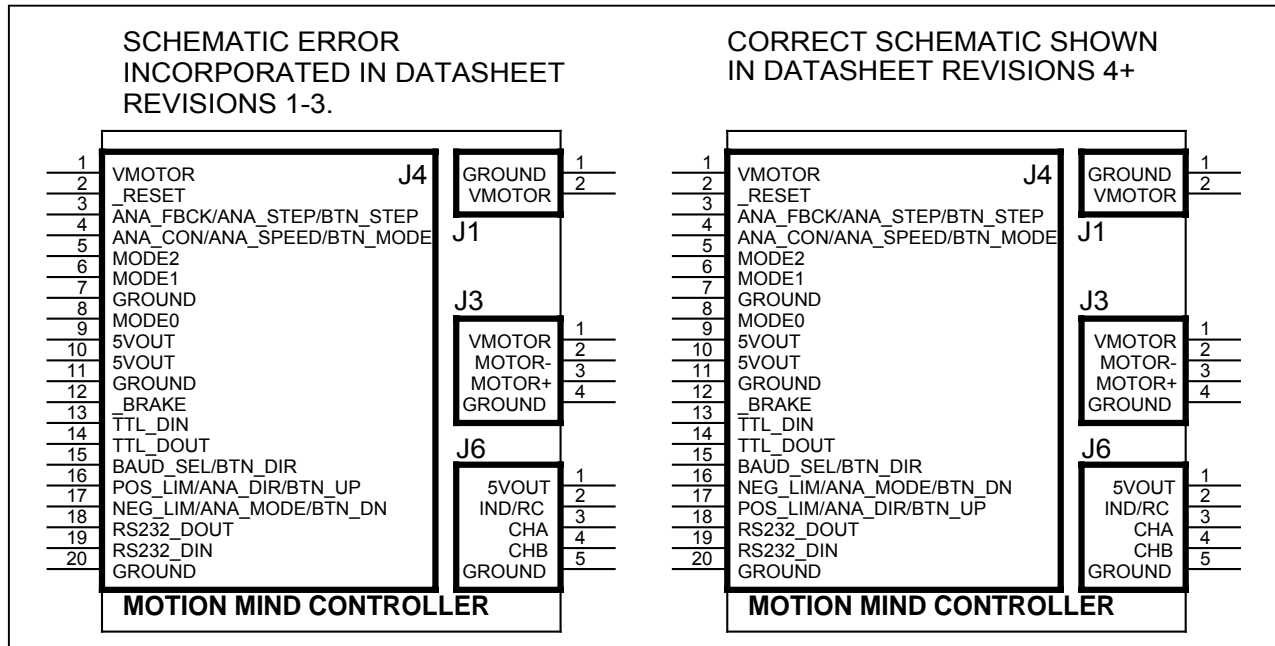
**Life Support Policy:** Solutions Cubed does not authorize any Solutions Cubed product for use in life support devices and/or systems without express written approval from Solutions Cubed.

**Warrantee:** Solutions Cubed warrants all Motion Mind motor control modules against defects in materials and workmanship for a period of 90 days. If you discover a defect, we will, at our option, repair or replace your product or refund your purchase price. This warrantee does not cover products that have been physically abused or misused in any way.

## 8.0 Errata Sheet

The ERRATA sheet tracks known bugs that we have been made aware of and whether or not they have been fixed as firmware changes are made. Other information may also be included if it appears to meet the needs of our customers. From time to time we may also incorporate functional changes in the Motion Mind. When these occur they are noted as a “Notice of change or addition”.

**Datasheet Error: Datasheet revisions 1-3.** The pin functions for J4 P16 and J4 P17 were swapped on earlier datasheet revisions. J4 P17 is the positive limit switch, analog direction, and button up input, J4 P16 is the negative limit switch, analog mode, and button down input.



### Firmware Revision 1:

- A)** When using the ASCII communication protocol writing values greater than 32,768 or less than 0 to the VELOCITYLIMIT register may cause the motor to run. Values written to this register should be limited to between +1 and +1023. Fixed in firmware revision 2.

### Firmware Revision 2:

- A)** When operating in mode 4 (serial PID position control mode) with the FUNCTION.POSPWRUP, FUNCTION.SAVEPOS, and FUNCTION.VELLIMIT bits set, the controller will power up thinking it is at position 0, and will run to the saved position. Clearing the FUNCTION.VELLIMIT bit prevents this from happening. Fixed in firmware revision 3.
- B)** Notice of change or addition: Operating in ANALOG position control mode with a serial position command (serial control / analog feedback) was not available in firmware revision 1. Setting the FUNCTION.ADSERIAL bit and operating in mode 5 allows this control method to be used.
- C)** Notice of change or addition: By setting the FUNCTION.ENABLEDB bit you may use the value in the RCBAND register so set a dead band around the desired position (modes 4 and 5). When the actual position value falls within the desired position +/- RCBAND the motor drive signal is forced off. This can prevent a motor from heating up when it is essentially in position but still being driven due to the PID settings.

**Firmware Revision 3:**

- A) When operating in ASCII serial mode, and with multiple units on the serial bus, serial responses are stepped on when the un-addressed unit(s) replies with a “BAD COMMAND<CR><LF>” response. There is no work around barring running separate DOUT connections from the Motion Mind units to individual pins on the controlling device. You can operate in binary mode and avoid this issue. Fixed in firmware revision 4.

**Firmware Revision 4:**

- A) Notice of change or addition: Limit switch functionality in ANALOG position control mode was not provided in previous firmware revisions. Since the schematic related to ANALOG position control mode showed limit switches we have incorporated the limit switch functionality in firmware revision 4. Otherwise limit switches do not work in ANALOG position control mode for firmware revisions 3 and below.
- B) Notice of change or addition: An RC position control mode can be accessed by placing the Motion Mind into mode 0 and setting the FUNCTION.RCPOS bit. In this mode the Motion Mind uses the analog feedback as the actual position and the RC signal as the desired position. Adjustment of the RCMIN and RCMAX registers will be necessary to get full-scale control. Limit switch functionality has been included in this mode. RC position control is available in firmware revision 4.

**Firmware Revision 5:**

- A) Notice of change or addition: Virtual limit switch functionality was added with the FUNCTION.VIRTLIMIT bit and VPLIMIT, VNLIMIT registers. The user may now program limit points into the device to prevent closed loop control modes from exceeding the desired points. Additional STATUS bits were included, and are asserted (set to 1) if a virtual limit is reached.
- B) Notice of change or addition: PWMLIMIT register was added that restricts the drive signal to the H-bridge to the PWMLIMIT setting. In systems where a full drive signal might cause excessive currents the PWM value can now be limited to a reduced value.
- C) Notice of change or addition: DEADBAND register (index 26) was added and is used to set the dead band (forces stopped motor) around 2.5V in bi-directional analog mode. This register may also be used to set a dead band around commanded positions in the closed loop modes (by setting the FUNCTION.ENABLERC bit). In firmware revisions 2,3,4 this was done with the RCBAND register.
- D) Notice of change or addition: FUNCTION.HEXMODE functionality was removed from ASCII READ commands.
- E) Notice of change or addition: FUNCTION.HEXMODE bit was replaced with FUNCTION.ACTIVESTOP bit. When set a stop condition ties the motor leads to ground.
- F) Notice of change or addition: FUNCTION.RCPOS is changed to FUNCTION.RCPOS-ENCFDBCK. In MODE0 setting this bit enables RC Position Control. In MODE5 setting this bit allows Analog Position Control to use an incremental encoder as the position feedback source.
- G) Notice of change or addition: DESIREDPOSITION (index27) the internal position value that the controller is attempting to reach can now be read by the user.