Motion Mind 3

(Hardware Version 3 Only)

Motor Controller Data Sheet

Revision 8

April, 2016

Always visit www.solutions-cubed.com
for the most up-to-date documents and software

hardware made easy

Solutions Cubed

Phone: 530.891.8045
www.solutions-cubed.com

Copyright © 2016 Solutions Cubed, LLC

Table of Contents

List of Figures

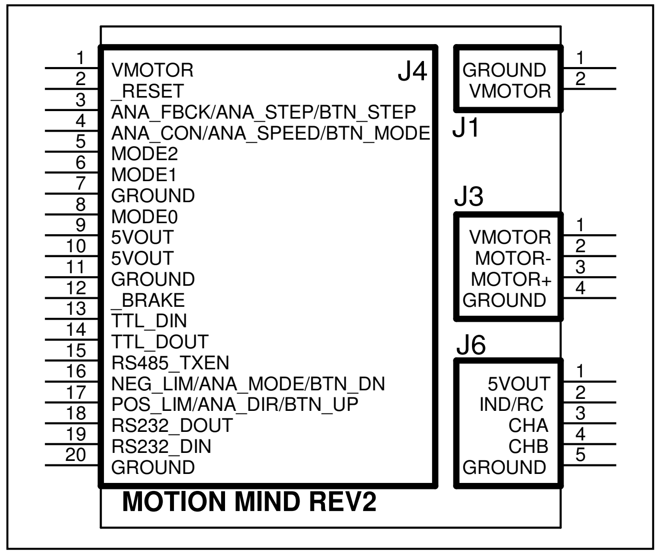
1.0 Revision Log

Date	Rev	Description
10-08	1	This revision relates to hardware revision 3.
03-10	2	Corrected broadcast command address, was described as “0”, but should be “99”
2-25-11	3	Included additional information about ASCII serial response timing fixed RC timing in Figure 19
4-25-11	4	Fixed some legacy typos referring to MOTM1,2 added pulse feedback mode
7-25-12	5	Added baud rate change in FUNCTION2 register and New Velocity Mode description
5-16-13	6	Added baud rate change in FUNCTION2 register and color logo. Modified communication timing.
2-1-15	7	Added bit13 FASTPID.FUNCTION2 bit included in rev A firmware.
4-13-16	8	Changed screen captures of serial bootloader software and instructions.

2.0 Introduction

Motion Mind
Open Loop / Closed Loop DC Motor Controller

Features



- Up to 9A continuous current (25A peak), 6-24VDC brush DC 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, 38.4KBPS, 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, and over-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 3 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 loop 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 (38.4KBPS, 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.

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 +32.0V
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-	32V transient spike
Motor Current Load	25A peak / 9A continuous @ 95% duty cycle

3.4 Electrical Characteristics and Settings

Characteristic	Min	Typ	Max	Unit	Notes
Supply Voltage	6		24	V	VMOTOR voltage
Supply Current		40		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
On Board Fuse Failure Time 11A	100			Hours	
On Board Fuse Failure Time 20A	.15		5	S	
Current measurement resolution		20		mA	0-1023 in ~20mA steps
Temperature measurement conversion					See section 3.8
Low Level Input RX pin			0.5	V	RX pin pulled to +5V with 10kΩ resistor
High Level Input RX pin	2.0			V	RX pin pulled to +5V with 10kΩ 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

Electrical Characteristics and settings **(continued)**

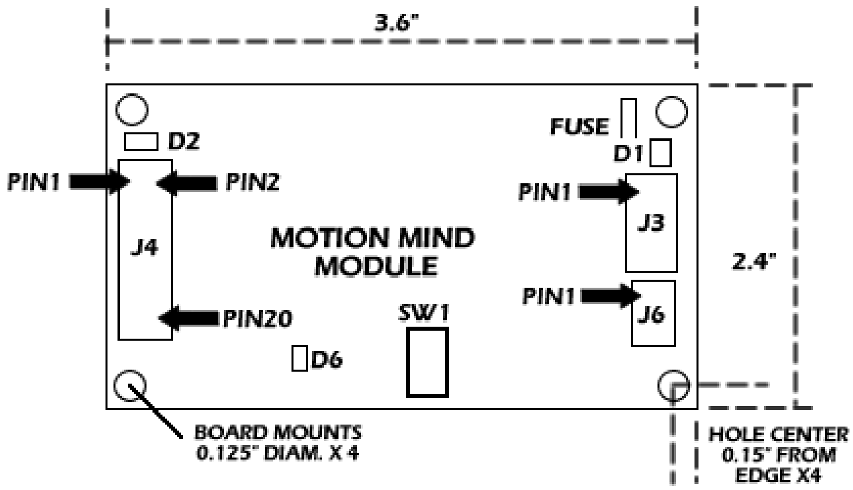
Characteristic	Min	Typ	Max	Unit	Notes
Low Level Input _RESET pin			0.5	V	_RESET pin pulled to +5V with 10kΩ resistor
High Level Input _RESET pin	2.0			V	_RESET pin pulled to +5V with 10kΩ resistor
Low Level Input CHA-B pins			0.8	V	CHA-B pins pulled to +5V with 100kΩ resistors
High Level Input CHA-B 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)
Time for a command to be responded to	1	2	50	mS	Commands that write to EEPROM take longer to implement (WRITE-STORE command). ASCII responses may take 5-10ms to respond.
V_M 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	9	18	36	KHz	PWM frequency can be doubled or divided using bits in the FUNCTION2 register.
Encoder Frequency		1.5	1.75	MHz	
Valid R/C Pulse Width	0.5		2.5	mS	RCMIN and RCMAX 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
Power up COMM LED blink time		400		mS	On power-up the communication LED blinks MODE+1 times. For each blink the LED is on for 300mS and off for 100mS. The FUNCTION.DISABLEBLINK bit can be used to disable this function and quicken power-up time.

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

3.3 Mechanical Dimensions

The following diagram may be used to develop PCB carrying boards or enclosures used to fit the controller into custom designs. The board requires approximately 0.8" height clearance on the component side.

Figure 1: Mechanical Dimensions



3.4 Mechanical Landmarks, Part Numbers, Mating Connectors

Landmark	Type	Description
D1	LED - red	Fuse blown indicator, lit when fuse is open and needs to be replaced
D2	LED – green	PWR- 5V regulator power indicator
D6	LED – green	COMM- Indicates mode on power-up and flashes when serial data is accepted
FUSE	10A	Automotive style mini-fuse Fuse - ***Littlefuse Inc. PN: 0297010.WXNV***
J3	1x4 screw terminal	DC motor connections for motor and motor supply Connector – ***OST PN: ED120/4DS accepts 14-22 AWG***
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 Connector - ***3M PN: 103308-5*** Mates with ribbon cable - ***Digi-Key PN: M3AAK-2006R-ND***
J6	1x5 0.1" locking header	Connects to quadrature encoders, or R/C signal input, may also be used as connection to 5VOUT and GND Connector – ***Molex PN: 22-23-2051*** Mates with – ***Molex PN: 22-01-2055 w/ crimp terminal 08-50-0114 accepts 22-30 AWG***
SW1	Mode setting switch	Switch used to adjust modes of operation and communication methods.

3.5 Connectivity Overview and Mode Switch (SW1) Settings

J3 Terminal Block Descriptions

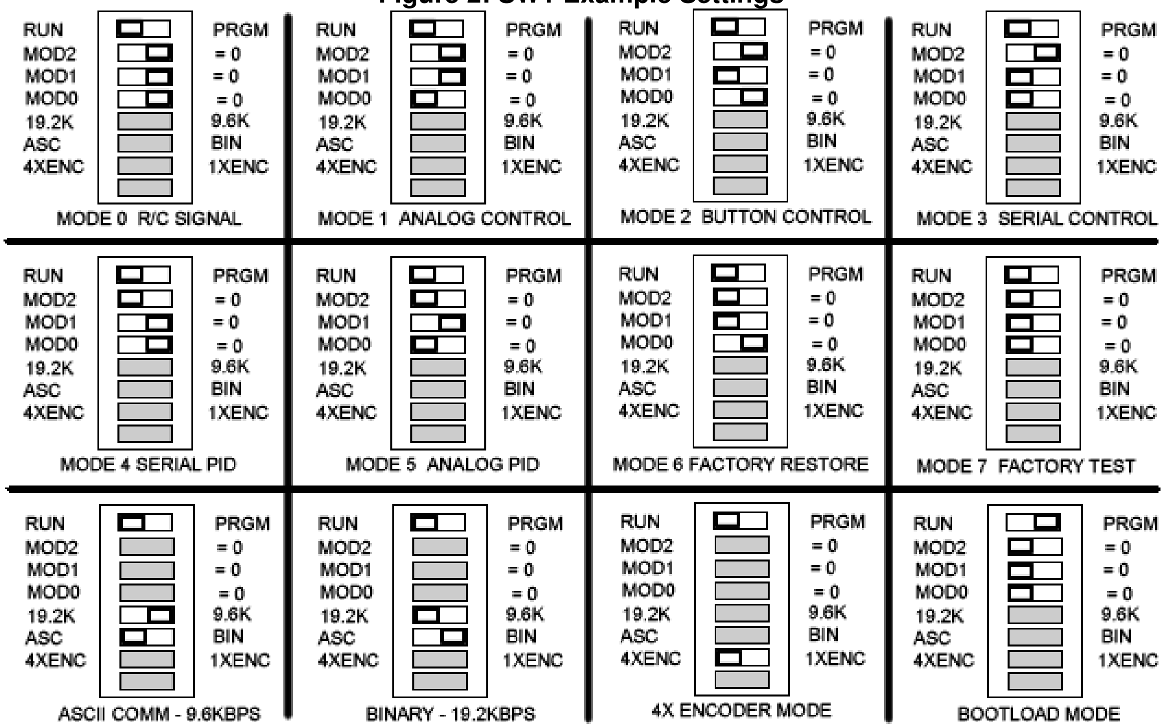
Terminal	Name	Type	Description
1	*VM*	POWER	The primary supply voltage connects to this terminal (36V maximum)
2	*M-*	POWER	The negative lead of the brushed DC motor, or other load, connects to this terminal
3	*M+*	POWER	The positive lead of the brushed DC motor, or other load, connects to this terminal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

J4 Master Control Interface Pin Descriptions

Pin	Name	Type	Description
1	*VMOTOR*	POWER	Connected to the motor voltage at P1 of J3 and P2 of J1, this connection is fused
2	*_RESET*	INPUT	Pulling _RESET 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	*ANA_FBCK/ ANA_STEP/ BTN_STEP*	INPUT	0-5V analog input, max 2.5KΩ input impedance, 3.4KHz low pass filter
4	*ANA_CON/ ANA_SPEED/ BTN_MODE*	INPUT	0-5V analog input, max 2.5KΩ input impedance, 3.4KHz low pass filter
5	*MODE2*	INPUT	Input used to select operating mode, pulled to 5V with 12.5kΩ (min) resistor
6	*MODE1*	INPUT	Input used to select operating mode, pulled to 5V with 12.5kΩ (min) resistor
7	*GROUND*	POWER	Ground return
8	*MODE0*	INPUT	Input used to select operating mode, pulled to 5V with 12.5kΩ (min) resistor
9	*5VOUT*	POWER	+5VDC output, can supply up to 200mA, should be left unconnected if not needed
10	*5VOUT*	POWER	+5VDC output, can supply up to 200mA, should be left unconnected if not needed
11	*GROUND*	POWER	Ground return
12	*_BRAKE*	INPUT	Pulling _BRAKE low forces the H-bridge into an “off” state, is pulled to +5VDC with a 100kΩ resistor
13	*RX*	INPUT	TTL level, 8N1, serial reception pin (data from the Master unit), is pulled to +5VDC with a 100kΩ resistor
14	*TX*	OUTPUT	TTL level, 8N1, serial transmission pin (data to the Master unit), is pulled to +5VDC with a 100kΩ resistor
15	*RS485_TXEN*	OUTPUT	TTL output with series 1 kΩ resistor, used to drive RS485 transceiver ICs
16	*NEG_LIM/ ANA_MODE/ BTN_DN*	INPUT	Negative going limit switch, when asserted movement in the negative direction is prevented, logic level for assertion is 0V, this input is pulled to +5VDC by 12.5kΩ (min) resistor
17	*POS_LIM/ ANA_DIR/ BTN_UP*	INPUT	Positive going limit switch, when asserted movement in the positive direction is prevented, logic level for assertion is 0V, this input is pulled to +5VDC by a 12.5kΩ (min) resistor
18	*RS232_TX*	OUTPUT	RS232 level serial output (for connection to RS232 serial port)
19	*RS232_RX*	INPUT	RS232 level serial input (for connection to RS232 serial port)
20	*GROUND*	POWER	Ground return

SW1 Operating modes: The various operating modes of the controller are enabled with SW1 (grayed out equals "don't care").

Figure 2: SW1 Example Settings



SW1 Switch Descriptions

Switch	Name	Setting Description
1	*BOOTLOADER*	RUN (switch off): Normal program operation PRGM (switch on): Used only for firmware upgrades
2	*MODE2*	MOD2 (switch off): Pin is pulled to +5V with weak pull-up, equals logic "1" =0 (switch on): Pin is tied to ground, equals logic "0". This J4 mode pin connection will also be tied to ground
3	*MODE1*	MOD1 (switch off): Pin is pulled to +5V with weak pull-up, equals logic "1" =0 (switch on): Pin is tied to ground, equals logic "0". This J4 mode pin connection will also be tied to ground
4	*MODE0*	MOD0 (switch off): Pin is pulled to +5V with weak pull-up, equals logic "1" =0 (switch on): Pin is tied to ground, equals logic "0". This J4 mode pin connection will also be tied to ground
5	*BAUD RATE*	19.2K (switch off): 19.2K baud serial communication 9.6K (switch on): 9.6K baud serial communication Button Mode: In unidirectional Button Mode (operating mode2) placing this switch in the "off" (19.2K) position will run the motor forward. Placing it in the "on" (9.6K) position will run the motor in reverse.
6	*COMM MODE*	ASC (switch off): ASCII communication protocol is in use BIN (switch on): Binary communication protocol is in use
7	*ENCODER RATIO*	4XENC (switch off): Encoder ratio of 4:1 1XENC (switch on): Encoder ratio of 1:1
8	*UNUSED*	not used

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

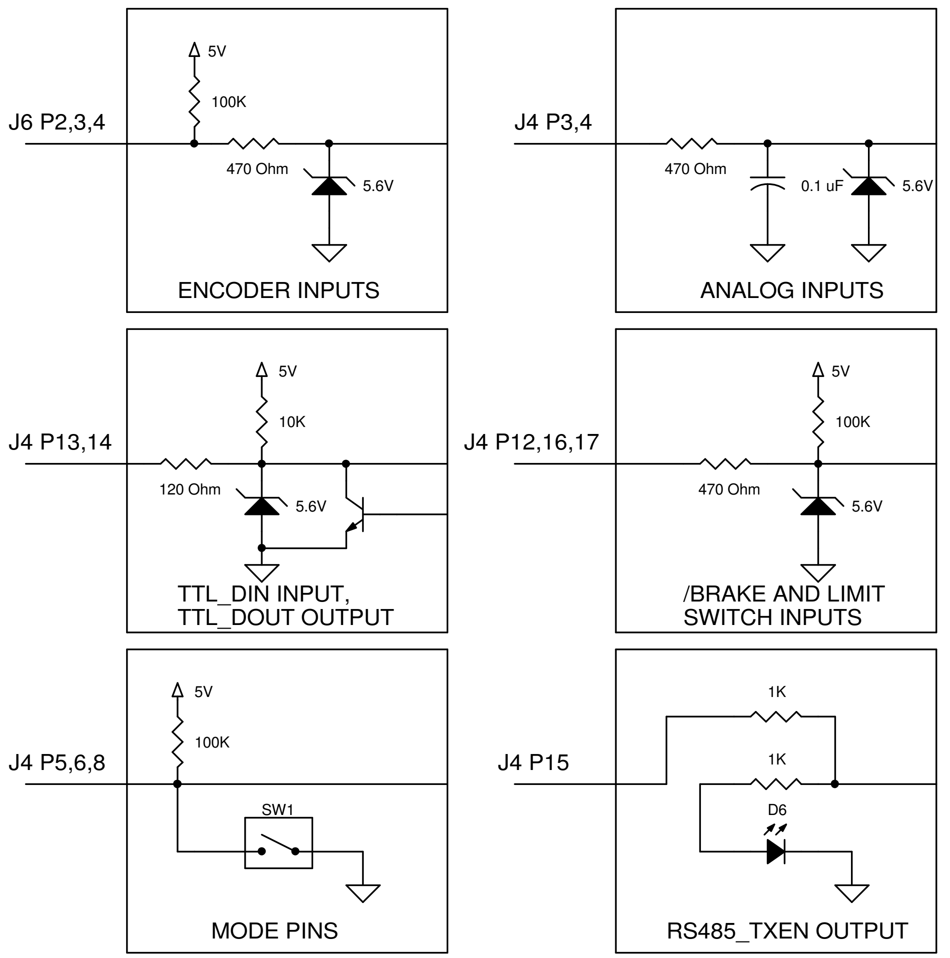
3.6 Indicator LEDs

D1, red = fuse blown when lit, replace fuse
D2, power = 5V onboard supply is functioning when lit
D6, communication-heartbeat = This LED 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. Setting and storing the FUNCTION.DISABLEBLINK bit in EEPROM will disable the blink on power up. This may be done to reduce power-up timing.

3.7 Input/Output 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 3: Input/Output Circuits

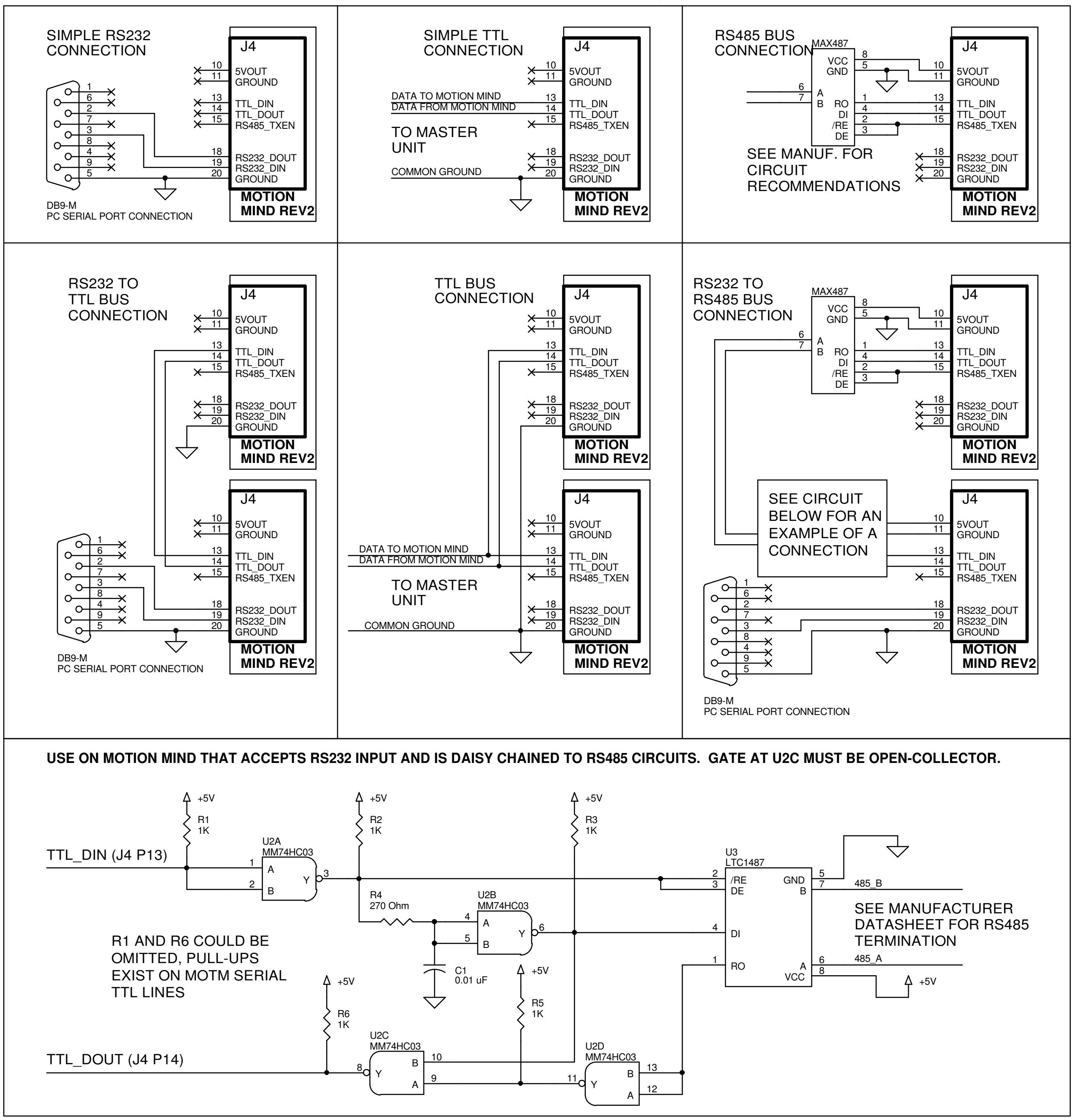


3.8 Serial Bus Construction

This controller has been designed to facilitate versatile serial bus construction. Each device has an RS232 converter for connection to a PC serial port. The TTL level connections to this converter are open-collector. This allows the devices to be paralleled for short distances. The RS485_TXEN pin works in conjunction with the controller's operating system drive RS485 circuitry should a larger network, or longer cables be required. Motion Minds must be individually programmed to a different address before connecting to a multi-unit serial bus.

When connecting more than 4 Motion Minds the designer should consider implementing an RS485 bus. An RS485 bus is also recommended for systems where the connections between multiple Motion Minds are greater than several feet.

Figure 4: Serial Data Bus Configurations

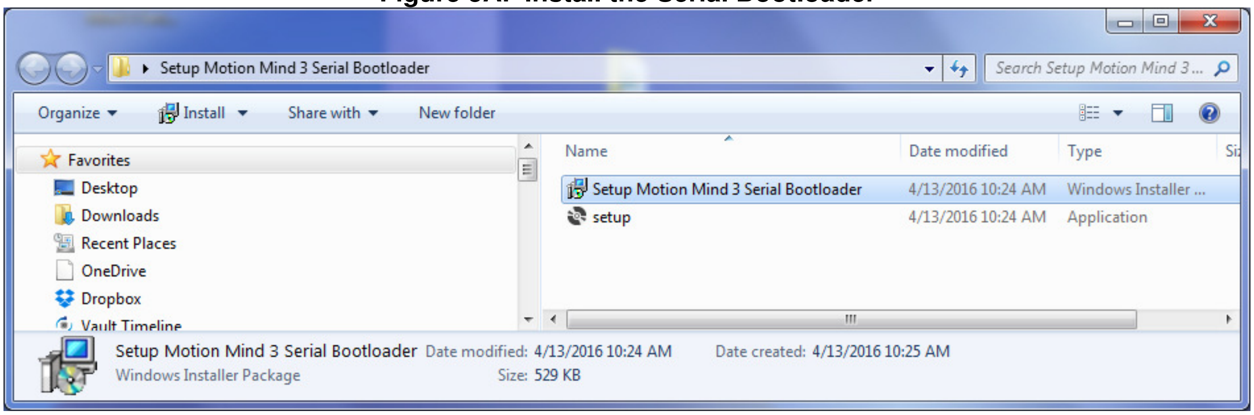


3.4 3.9 Firmware Upgrades

This controller is shipped with a serial bootloader. The bootloader resides in the system firmware and allows the user to upgrade firmware to the latest revision when it is available. To upgrade the firmware you'll need the Solutions Cubed bootloader software and the desired firmware text file from the company's website.

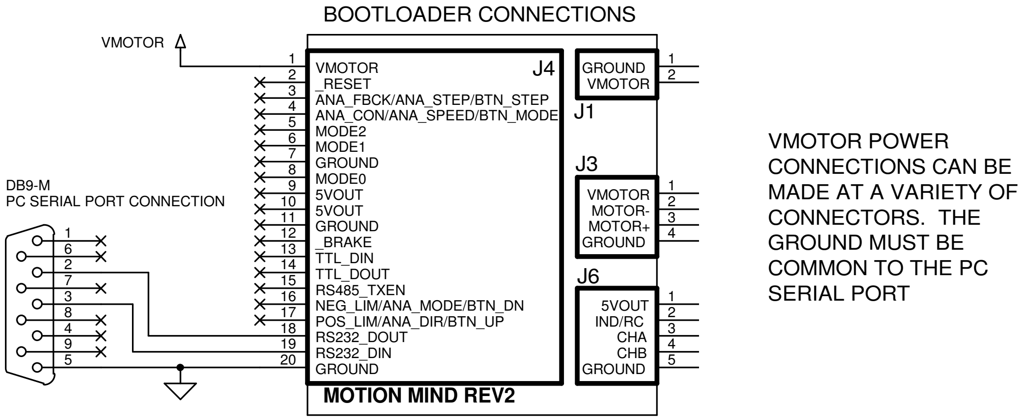
Step1: Install the bootloader software on a PC. Place the firmware upgrade file to a known directory.

Figure 5A: Install the Serial Bootloader



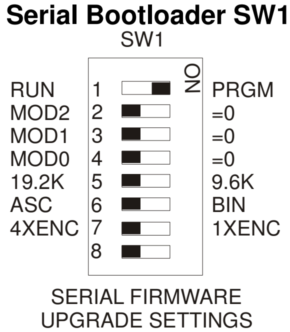
Step2: Connect a PC serial port to the controller, and provide the connections for the *VMOTOR* and *GROUND* (J4 P1 and P20, J3 P1 and P4, or J1 P2 and P1). Do not power the controller yet.

Figure 5B: Serial Bootloader Connections



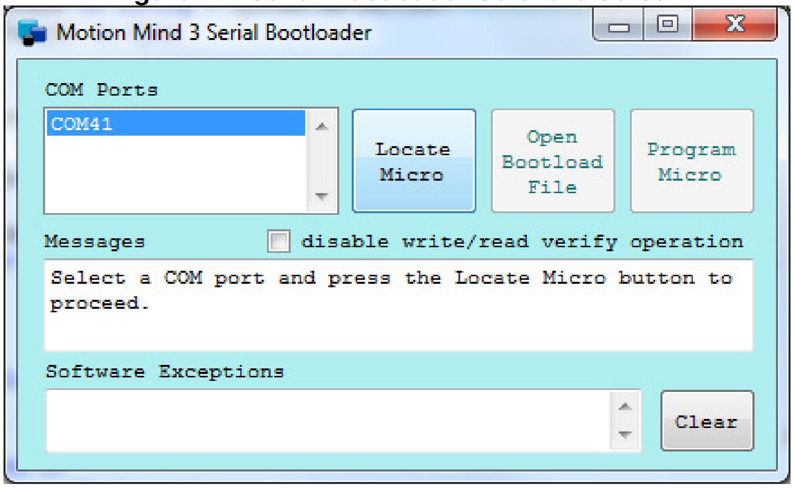
Step3: Set SW1 switch 1 to PRGM. Apply power to the controller. No LEDs will blink on power up, however D6 should remain lit until serial communication occurs.

Figure 6: Serial Bootloader SW1 Settings



Step4: Launch the bootloader program. Select a COM port from the list box, and press the "Locate Micro" button. If the microcontroller responds the "Open Bootload File" button will be enabled.

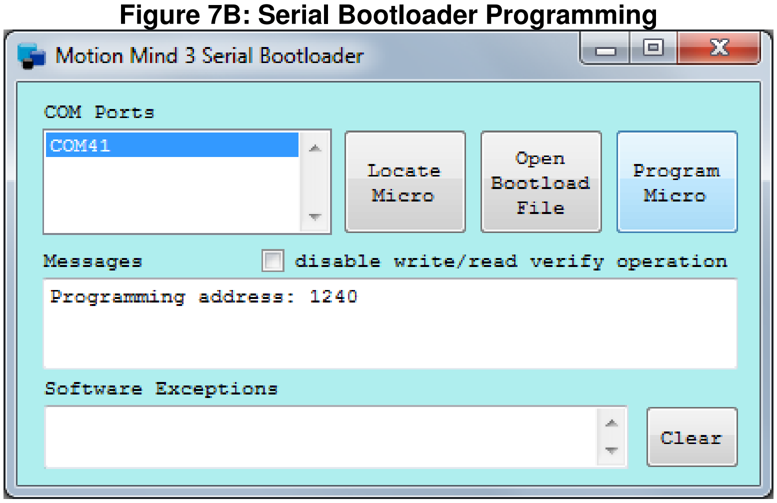
Figure 7A: Serial Bootloader Software Screen



Step5: Click the "Open Bootload File" button and select the appropriate Motion Mind firmware bootloader file (example "motm3_0a_bootloader.hex". The "Program Micro" button will be enabled after the file has been loaded.

Step6: Press the "Program Micro" button and the firmware will begin uploading to the controller. To reprogram additional controllers you may skip steps 4 and 5. Selecting the disable write/read verify operation check box will make bootloading faster.

Figure 7B: Serial Bootloader Programming



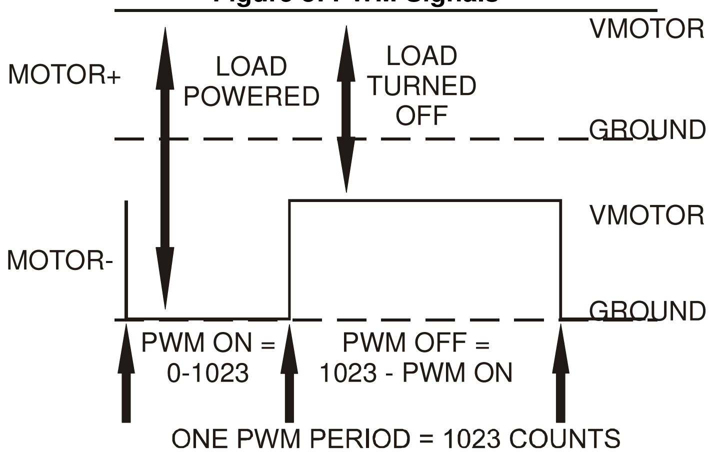
Step7: Remove power. Change the SW1 settings to the desired operating mode. Re-apply power. The LED D6 should blink indicating that you are no longer in programming/booloader mode.

Step 8: You may need to restore factory default settings if your upgrade includes new firmware registers that didn't exist in previous firmware versions. You can restore the factory default settings by placing the Motion Mind in mode six and grounding the limit switch connections at J4 (P16 and P17)

3.10 PWM And Current Limit Functions

Active voltage and current limiting functions are built into the controller through the PWMLIMIT and AMPSLIMIT registers. Four MOSFET switches control motor speed and direction. The low side switches (connected between the motor and *GROUND*) allow current to flow through the load when they are turned on. Using pulse-width-modulation (PWM) the power applied to the load can be controlled.

Figure 8: PWM Signals



For each PWM period the load is turned on for a chunk of time. The total PWM period is 1023 "time-slices" long. The PWM ON time can be from 0 to 1023, 0 being off all of the time and 1023 being on all of the time. The example above shows the motor running forward with roughly a 33% positive duty cycle (a PWM setting of ~341 out of 1023).

The PWMLIMIT register sets a maximum PWM ON value for the controller. A value from 0 to 1023 may be selected. The PWMLIMIT can be useful to reduce average currents and voltages in closed loop control modes that might become unstable.

The AMPSLIMIT register is used to set desired current limits. The AMPS register contains a value of 0-1023 that represents system current in 20mA increments (512 = 10.24A). If the AMPS register exceeds the AMPSLIMIT value the PWMLIMIT is internally reduced based the difference between the two current registers.

3.11 Two's Compliment Number System

This controller can accept negative numbers for some settings. In order to generate a two's compliment 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 Compliment Examples

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

3.12 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 or 1:1 ratio. This is selectable at SW1. A 512 counts-per-revolution (CPR) encoder would be represented as a 2048CPR encoder by the Motion Mind circuit with SW1 in 4XENC mode. 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 170 counts.

$$shaft\_revolution = gear\_ratio \times encoder\_CPR \times 4*$$
$$61{,}440 = 30 \times 512CPR \times 4*$$

*The x4 comes from the 4:1 decoding. If using 1:1 decoding (selected at SW1) then don't multiply the result by 4.

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 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.

3.13 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.

The PID can be broken down into four stages that can each be described separately. Each section may be modified by the user with the serial interface, 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.

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

3.14 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. 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.

3.15 Velocity Measurements and Closed-Loop Control: 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.

Use this equation for x4ENC SW1 setting:

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

Use this equation for x1ENC SW1 setting:

$$MotorShaftRotations / Second = \frac{VELOCITY \times 200}{EncoderCPR \times GearRatio}$$

3.16 Changing PWM Frequency: The FUNCTION2 register (register 30) has function bits that allow the user to modify the PWM signal that drives the motor. When setting FUNCTION2.FREQ/2 the 18KHz PWM signal is divided by 2 resulting in a frequency of 9KHz. 9KHz is an audible frequency. The FUNCTION2.FREQX2 bit changes the PWM frequency to 36KHz. These FUNCTION2 bits are tested sequentially so the highest bit set is the one enabled. For example, if both bits are set the FREQX2 function is enabled.

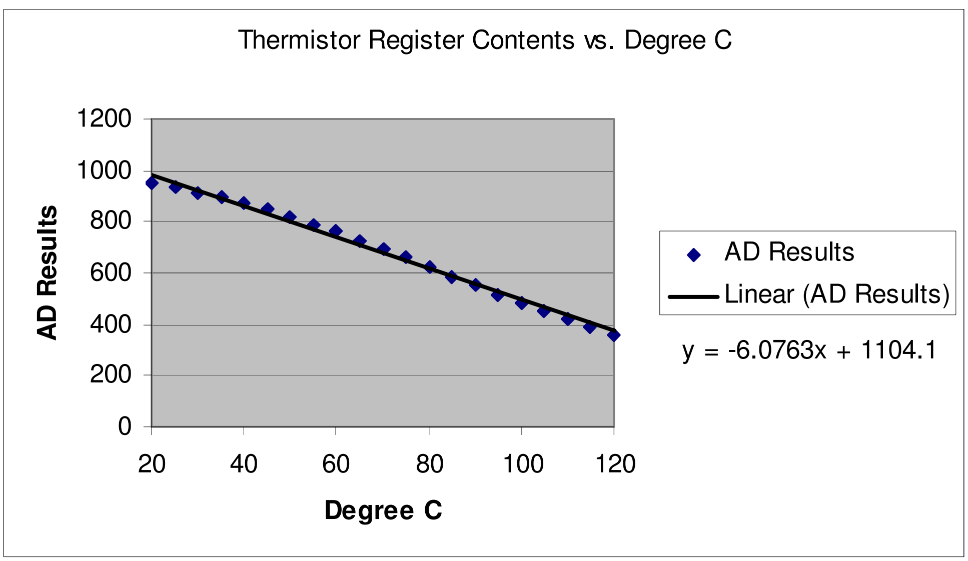
For both 9KHz and 36KHz the effective PWM resolution is lowered from 10 bits to 9 bits. This change doesn't require any changes on the part of the user, but may result in less fine control at lower or higher PWM settings.

3.17 Expanding the Analog/Position Relationship: The FUNCTION2 register (register 30) has function bits that effect mode 5, the Analog PID closed loop control mode. If the FUNCTION.RCPOS-ENCFBCK bit is set in mode 5 the feedback signal does not come from the analog input at J4 P3, but instead comes from an encoder attached to J6. The control signal still comes from a 0-5V analog signal at J4 P4.

In normal operation the analog signal at J4 P4 is converted to a value from 0-1023 and this value is used as the DESIRED POSITION. When using an encoder as the position feedback source the range of movement available is limited by the analog measurement. Bits 2 though 7 of the FUNCTION2 register are used to expand the DESIRED POSITION measurement. For example, setting the FUNCTION2.ADX8 bit will allow a 0-5V signal at J4 P4 to command a position of 0-8184 (0-1023*8). ADX2, ADX4, ADX8, ADX16, ADX32, and ADX64 bits are available. These FUNCTION2 bits are tested sequentially so the highest bit set is the one enabled. For example, if all bits are set the ADX64 function is enabled.

Resolution is limited by the multiplier and the fact that the analog control signal only provides 10 bits of position information.

3.18 Thermistor Values and Over-Temperature Protection: The Motion Mind has an on onboard thermistor that is used to shut down the H-bridge when the board gets above 90˚C. When the board cools to below 80˚C the H-bridge will be automatically re-enabled. This protective feature can be disabled by setting the FUNCTION2.OVERTEMP bit. Raw thermistor values can be read from the THERMISTOR register and relate to the module's temperature as described in the graph below.



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 the steps below should be taken. The schematics that follow show electrical connections to the J4 mode pins. These connections can be made with the SW1 mode setting switch. They do not need to be made both at SW1 and J4.

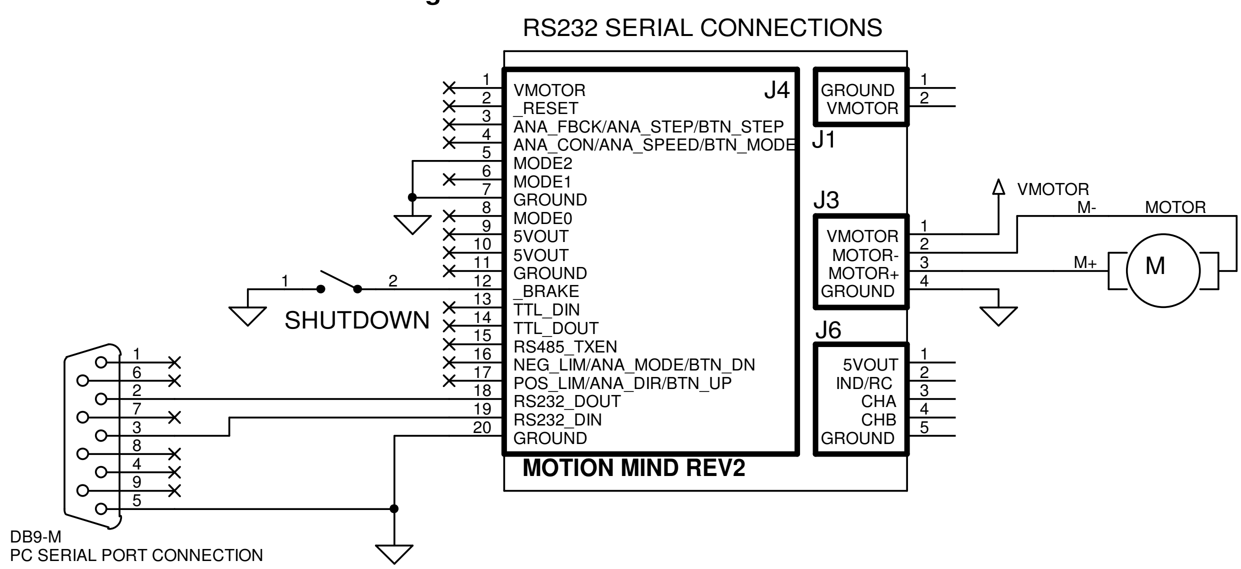
Step1: Determine the mode of operation you plan on using, then set the switches at SW1 (see section 3.5 for more information). Alternatively, logic levels may be applied to J4-P5, P6, and P8 to select the operating mode. SW1 mode select switches (MOD0-2) should be "off" if J4 connections are used.

Step2: Select the baud rate (19.2KBPS or 9.6KBPS) and mode of communication (binary or ASCII) by setting the appropriate switches on SW1. Please note that in Mode 2 (button mode) when operating in uni-directional mode, the baud select setting 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 at 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 addresses. See section 5 for details on the communication protocol.

Step4: Connect the motor and motor voltage and apply power

Figure 9: Basic Serial Connections



Summary of Operating Modes

Mode	Submode	Description
Mode 0 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	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.
Mode 1 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.
Mode 2 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.
Mode 3 Serial Control		This open-loop mode accepts serial motor speed and direction commands.
Mode 4 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.
Mode 5 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	Setting the FUNCTION.ADSERIAL bit allows you to provide the desired position through the serial communication interface.
	Analog Control with encoder feedback	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. The DESIRED POSITION may be expanded with bits 2 through 6 of the FUNCTION2 register.
Mode 6 Factory Restore		When placed into mode 6 and the *POS_LIM* and *NEG_LIM* 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.
Mode 7 Factory Test		This mode is used after production as a self-test for the unit. Customers are not expected to use this mode.

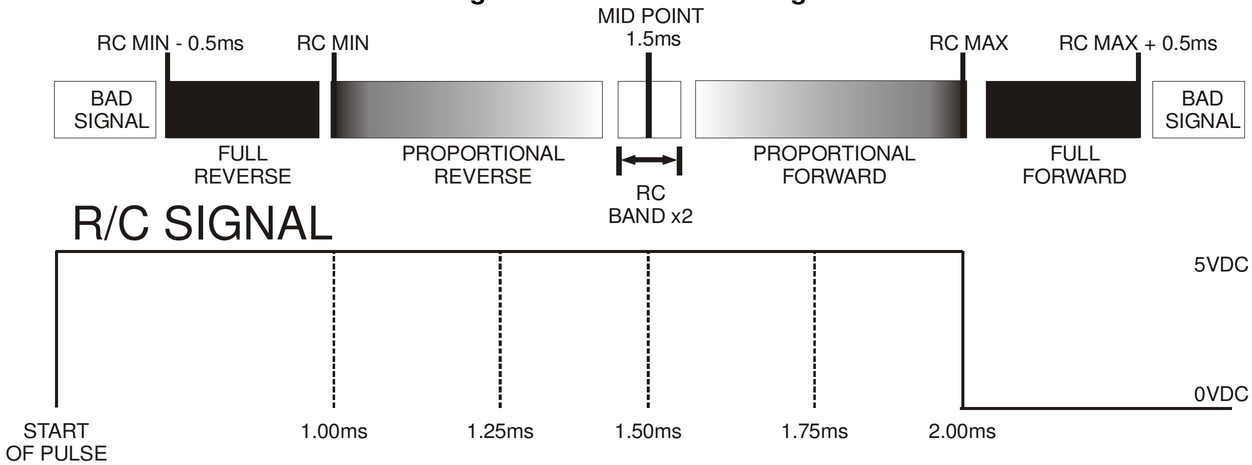
4.3 Mode 0: Radio Control (R/C) Speed Control:

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 1736ns. For example the default setting for RCMAX is 2ms, this is derived from the register value 1152 multiplied by 1736ns.

$$RCMAX = 1152 \times 1736ns = 2ms$$

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 10: R/C Pulse Timing

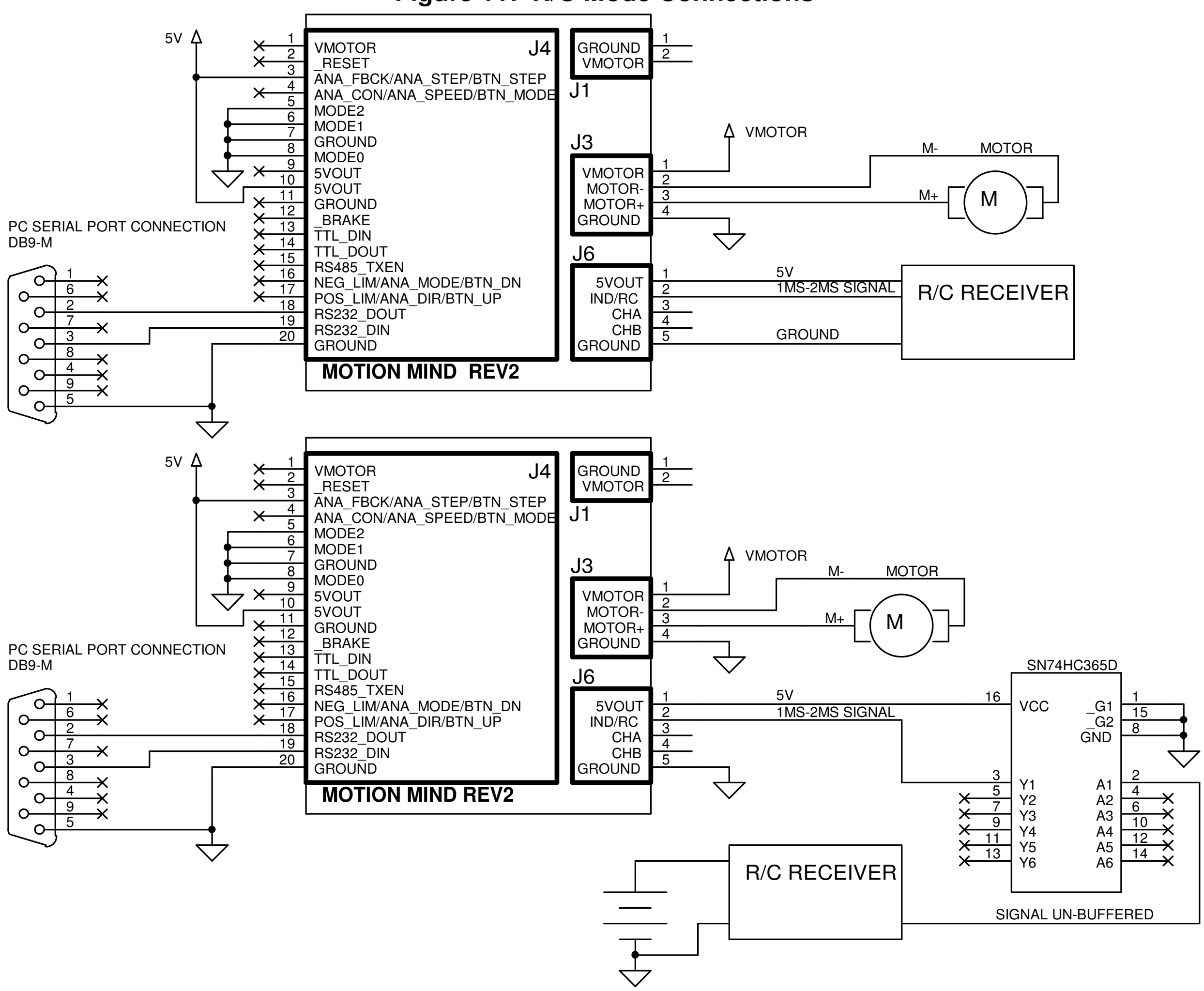


Fine Tuning to Your R/C Receiver: The Motion Mind controller should work with your R/C receiver straight out of the box. But you may want to fine-tune your receiver/transmitter to the controller. In order to do this you'll need to make use of the serial interface. Then follow these steps.

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 11: R/C Mode Connections

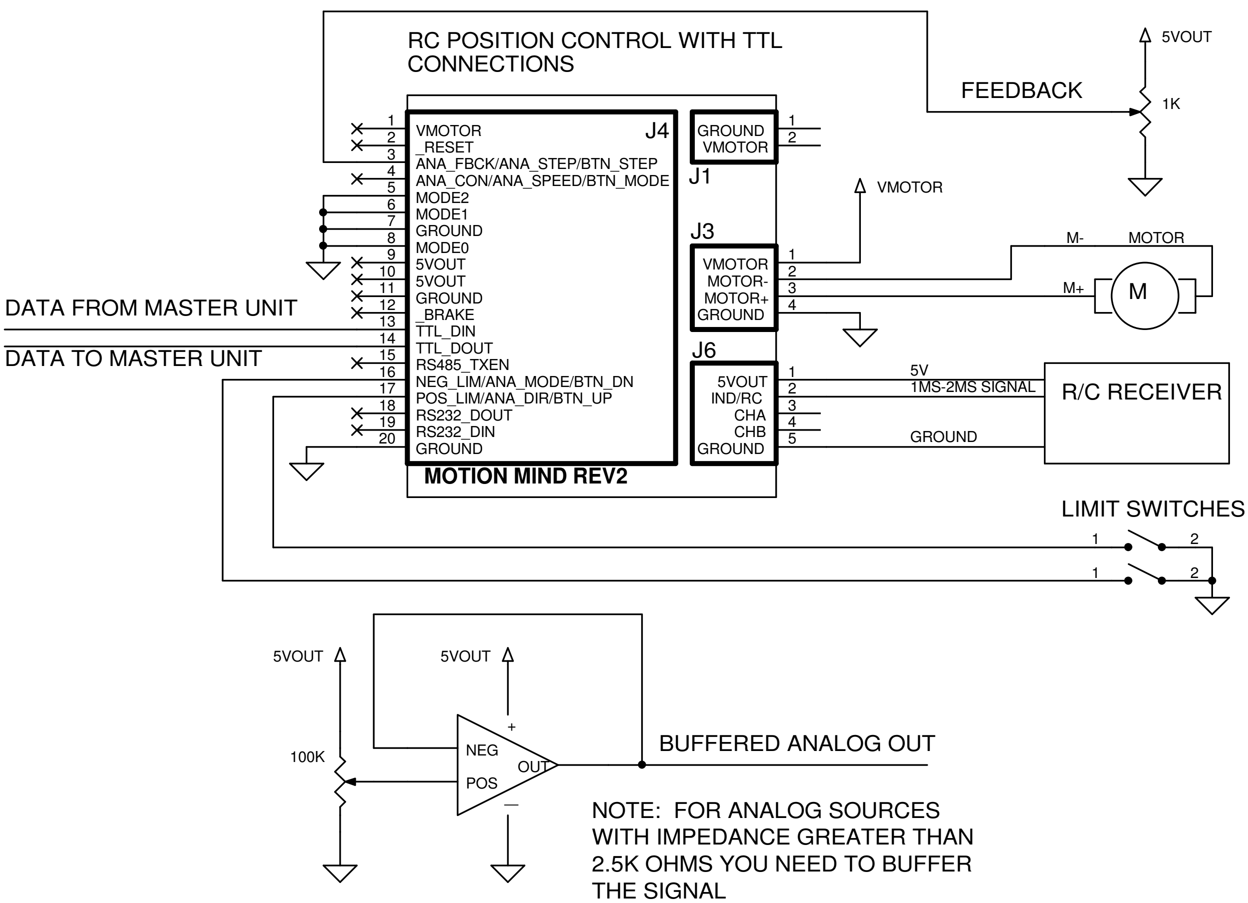


Mode 0 Sub Mode: Radio Control (R/C) Position Control:
How it Works: Setting the FUNCTION.RCPOS bit enables R/C Position Control mode. In this mode an analog signal (0-5VDC) is used to provide a motor feedback signal. The analog value is converted to a position of 0 (0VDC) to 1023 (5VDC). The RC signal 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 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 (432) and RCMAX to 2.25ms (1296). 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 12: R/C Closed Loop Mode



Interface/Programming Specific to Mode 0

Input Pins	**Description**
IND/RC (J6 P2)	Apply the R/C control signal here
ANA_STEP (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.
POS_LIM (J4 P17)	Positive limit switch, after asserted (0V) the motor is prevented from moving in a positive direction. R/C position mode only.
NEG_LIM (J4 P16)	Negative limit switch, after asserted (0V) the motor is prevented from moving in a negative direction. R/C position mode only.
Programmable Registers	**Description**
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 *ANA_STEP* (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 applications
AMPSLIMIT	Default is 512 (10.24A), this value sets the active current limiting the threshold. Each count corresponds to approximately 20mA
Programmable Registers (RC Position Mode only)	**Description**
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.
VPLIMIT	Sets a virtual positive position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set.
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.
Function bit settings	**Description**
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 *ANA_STEP* (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
FUNCTION.VIRTLIMIT	Enables virtual software limits defined by VNLIMIT and VPLIMIT registers
FUNCTION.DISABLEPID	Turns off PID output (no drive signal to motor).

4.4 Mode 1: Analog Control

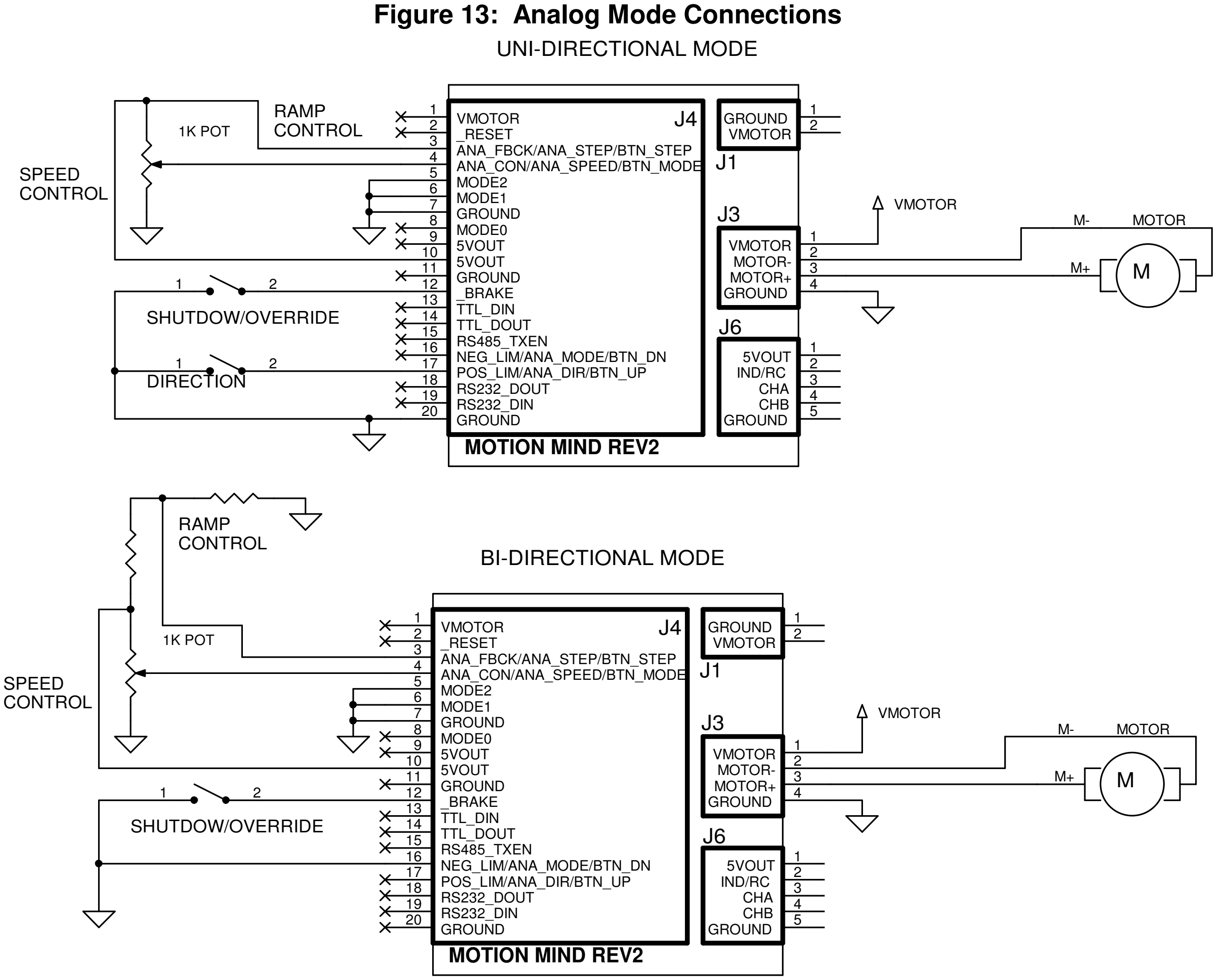
How it Works: In this mode the 0-5V signal present at *ANA_SPEED* (J4 P4) is translated into the motor speed. Motor speed is updated periodically based on the value of the TIMER register. The default update rate is every 50ms, and may be adjusted in 5ms increments. In addition to the update rate the 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
ANA_SPEED (J4 P4)	0V = full reverse 5V = full forward 2.5V = stopped	0V = stopped 5V = full forward	0V = stopped 5V = full reverse
ANA_DIR (J4 P17)	N/A	5V	0V
ANA_MODE (J4 P16)	0V	No connect or 5V	No connect or 5V

Figure 13: Analog Mode Connections



Interface/Programming Specific to Mode 1

Input Pins	**Description**
ANA_SPEED (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.
ANA_STEP (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.
ANA_DIR (J4 P17)	When operating in unidirectional mode this pin determines the direction the motor is spinning (0V for reverse, 5V for forward)
ANA_MODE (J4 P16)	Bi-directional mode is enabled if this pin is at 0V. If at 5V or not connected then unidirectional 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 *ANA_STEP* (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
AMPSLIMIT	Default is 512 (10.24A), this value sets the active current limiting the threshold. Each count corresponds to approximately 20mA
Function bit settings	
FUNCTION.ADSTEP	Set this bit if you want to prevent the analog input at *ANA_STEP* (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.ACTIVESTOP	Whenever the PWM output is 0 both low side MOSFETs are tied to ground

4.5 Mode 2: Button Control

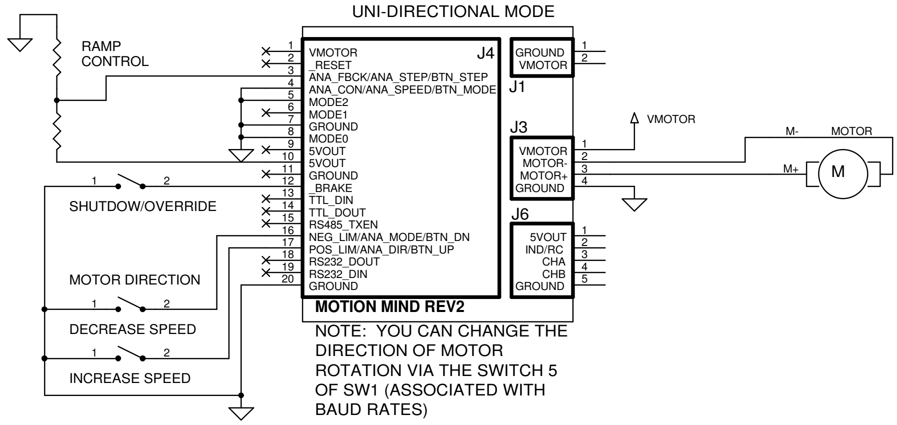
How it Works: By connecting buttons to *BTN_UP* (J4 P17) and *BTN_DN* (J4 P16) motor speed may be controlled with button presses. The buttons are debounced for a period of time of 5ms times the contents of the TIMER register (default is 50ms). The button does not need to be released for a new button 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).

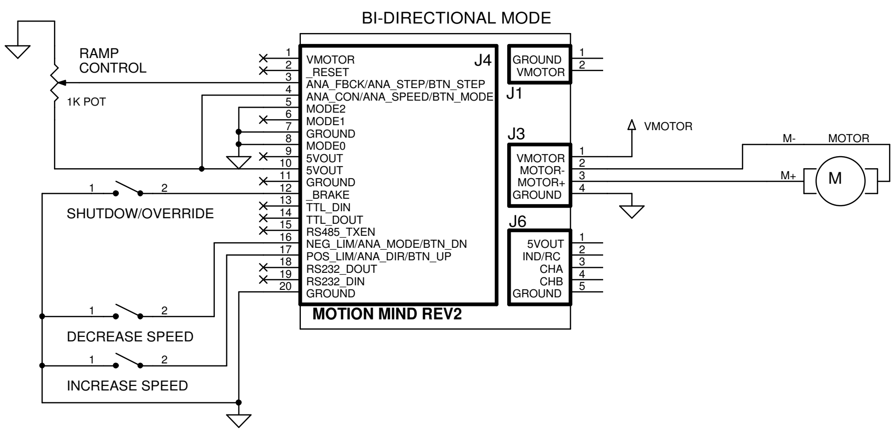
When in bi-directional mode, a motor speed transition through 0 will cause the motor to be stopped for a period equal to 4 times the debounce time (default is 200ms). This extended stop time allows the user to stop the motor even if the rate of change in motor speed is large.

Sub Mode Pin Settings for Mode 2

Pins	Bi-directional Mode	Unidirectional Forward	Unidirectional Reverse
BTN_MODE (J4 P4)	5V	0V	0V
BTN_DIR (SW1 switch 5; baud select on the DIP switch)	N/A	"off" or 19.2K	"on" or 9.6K
BTN_STEP (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
BTN_UP (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
BTN_DN (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 14: Button Mode Connections





Interface/Programming Specific to Mode 2

Input Pins	**Description**
BTN_MODE (J4 P4)	5V configures the controller for bi-directional motor control; 0V configures the controller for unidirectional motor control (with *BTN_DIR* (J4 P15) determining forward or reverse).
BTN_STEP (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.
BTN_UP (J4 P17)	Pressing this button increases the motor speed if the motor is running forward, or decreases the motor speed if running in reverse
BTN_DN (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 *BTN_STEP* (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.
AMPSLIMIT	Default is 512 (10.24A), this value sets the active current limiting the threshold. Each count corresponds to approximately 20mA

Function bit settings	Description
FUNCTION.ADSTEP	Set this bit if you want to prevent the analog input at *BTN_STEP* (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.ACTIVESTOP	Whenever the PWM output is 0 both low side MOSFETs are tied to ground

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

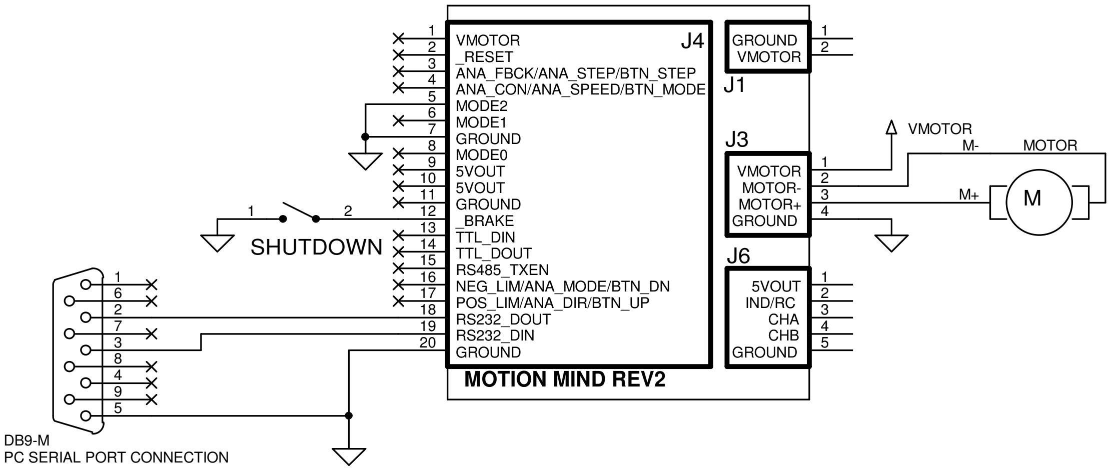
Command	Description
CHANGE_SPEED	Adjust motor speed and direction
WRITE	Writes value to internal register
WRITE_STORE	Writes value to internal register and stores value in EEPROM
READ	Reads one or more of the internal registers (note: there are additional READ commands in ASCII mode)
RESTORE	Restores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

Interface/Programming Specific to Mode 3

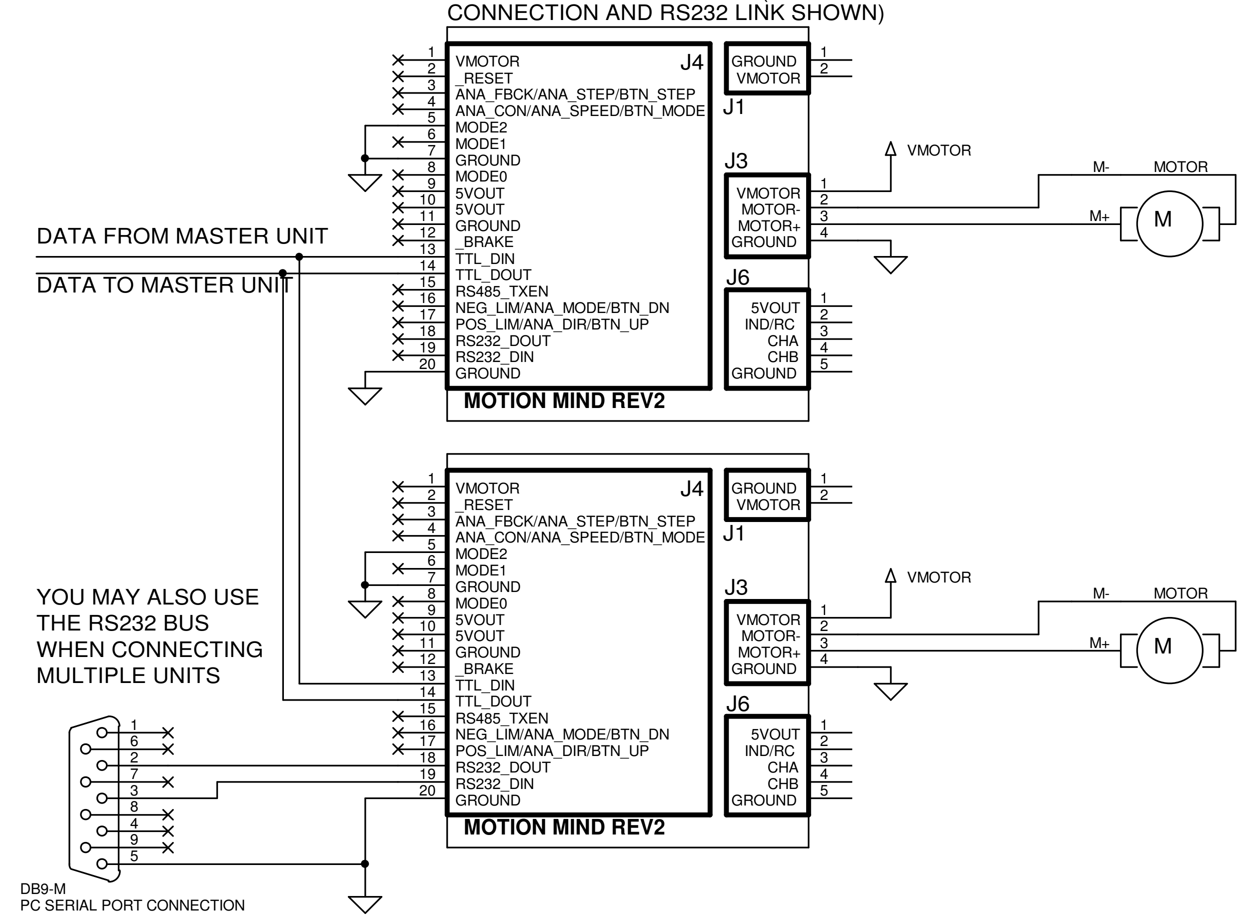
Input Pins	Description
RX (J4 P13)	TTL data (logic 0 = 0V; logic 1 = 5V) sent to the controller
TX (J4 P14)	TTL data (logic 0 = 0V; logic 1 = 5V) sent from the controller, open collector output
RS232_TX (J4 P18)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent from the controller
RS232_RX (J4 P19)	RS232 data (logic 0 = +12V; logic 1 = -12V) sent to the controller
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
AMPSLIMIT	Default is 512 (10.24A), this value sets the active current limiting the threshold. Each count corresponds to approximately 20mA
Function Bit Settings	**Description**
FUNCTION.ACTIVESTOP	Whenever the PWM output is 0 both low side MOSFETs are tied to ground

Figure 15: Serial (Open-Loop) Mode Connections

RS232 SERIAL CONNECTIONS



SERIAL CONTROL WITH TTL (MULTI-UNIT CONNECTION AND RS232 LINK SHOWN)



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 unless operating in pulse feedback sub ode (rev 4 firmware and above). For the communication protocol see section 5.

Summary of Serial Commands Used by Mode 4

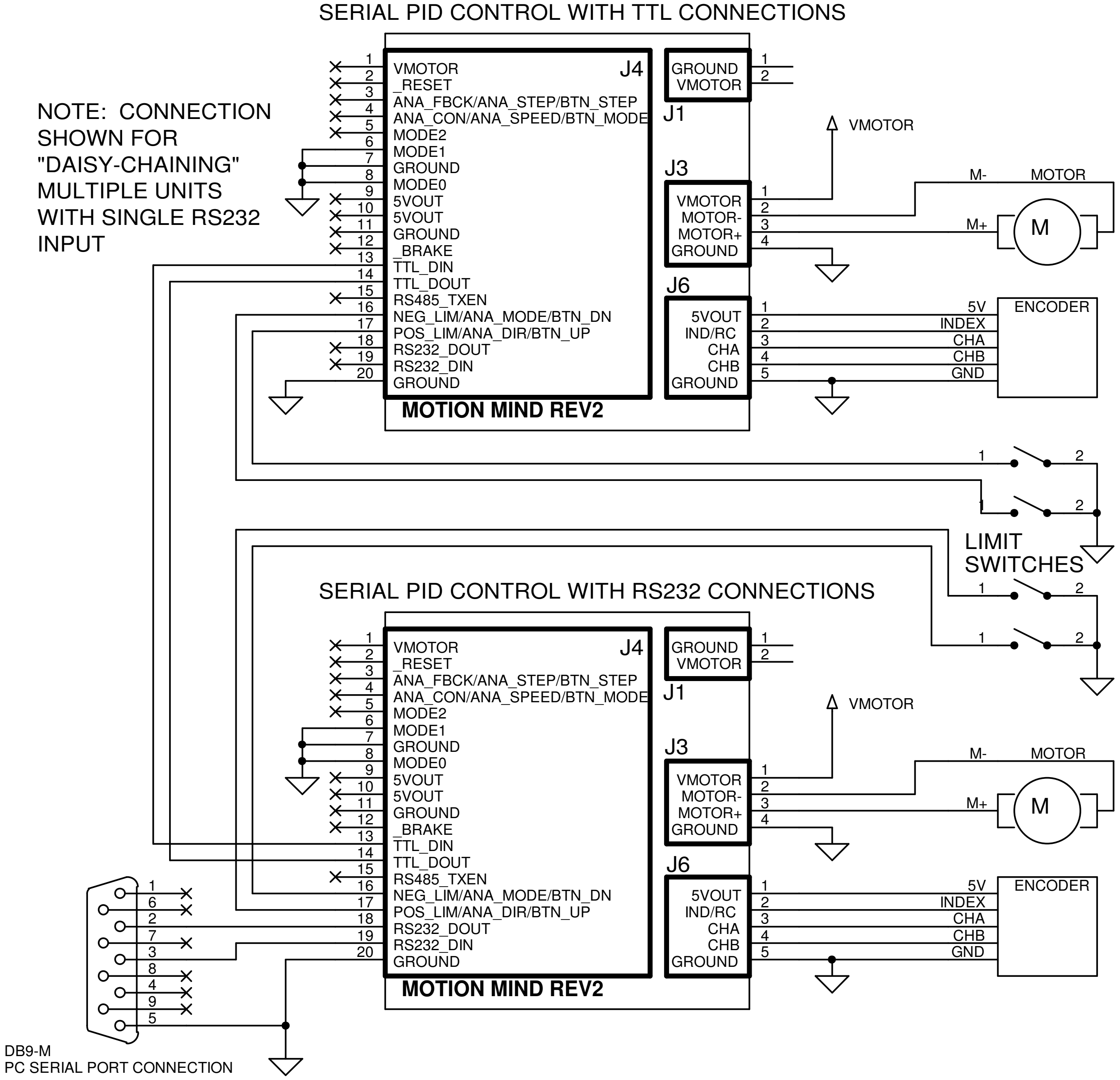
Command	Description
MOVETO_ABSOLUTE	Moves to desired position provided (example: you're at position +10,000 and you send a desired position of -1500, the motor moves to position –1500)
MOVETO_RELATIVE	Moves motors to desired position provided relative to current position (example: you're at position +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
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^PIDSCALAR
TIMER	The timer register is used to debounce limit switch inputs, defaults to 50ms
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.
VNLIMIT	Sets a virtual negative position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set.
VPLIMIT	Sets a virtual positive position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set.
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.
AMPSLIMIT	Default is 512 (10.24A), this value sets the active current limiting the threshold. Each count corresponds to approximately 20mA

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.ACTIVESTOP	Whenever the PWM output is 0 both low side MOSFETs are tied to ground
FUNCTION.ENABLEDB	When set the contents of the DEADBAND register are used to create a dead band around the desired position. **Not for use with MOVEAT_VELOCITY command.**
FUNCTION.VIRTLIMIT	Enables virtual software limits defined by VNLIMIT and VPLIMIT registers.
FUNCTION.DISABLEPID	Turns off PID output (no drive signal to motor).
FUNCTION2.PULSEFB	Can be set to enable the pulse feedback sub mode of serial PID control

Figure 16A: Serial PID Control (Closed-Loop) Mode Connections



Pulse Feedback sub mode (Rev4 firmware and above): In serial control mode the Motion Mind may be configured to use a 0-4.096mS pulse as a feedback signal. Setting the FUNCTION2.PULSEFB bit enables this function. This mode is designed to interface to a US Digital MA3-P12-250-D absolute position encoder. The feedback signal may be fed into J6 of the Motion Mind.

When using the pulse feedback setting The POSITION register will then be loaded with the pulse duration present at P2 of J6. The Position value is in increments of 0.217uS. So for a full range of 4.096mS the position would be 18,875. There are a few important notes to consider when employing this feedback method. The MA3-P12 is a 12-bit absolute encoder. So even though the Motion Mind provides resolution of 360 degrees / 18,875 counts (0.019 degrees/count) the MA3-P12 is only a 12-bit encoder (360 degrees / 4097uS (0.089 degrees / uS). Additionally, the Motion Mind will have difficulty measuring within 50uS of either the maximum or minimum pulse width. Also, mechanical limits should be employed. If the encoder rotates beyond either the minimum or maximum point the feedback signal will cause the motor speed to ramp up causing a runaway condition.

Figure 16B. MA3-P12 Encoder Pulse Diagram

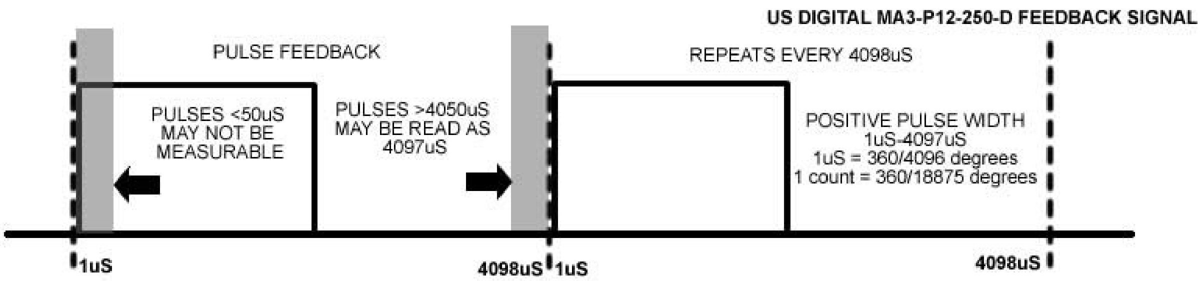
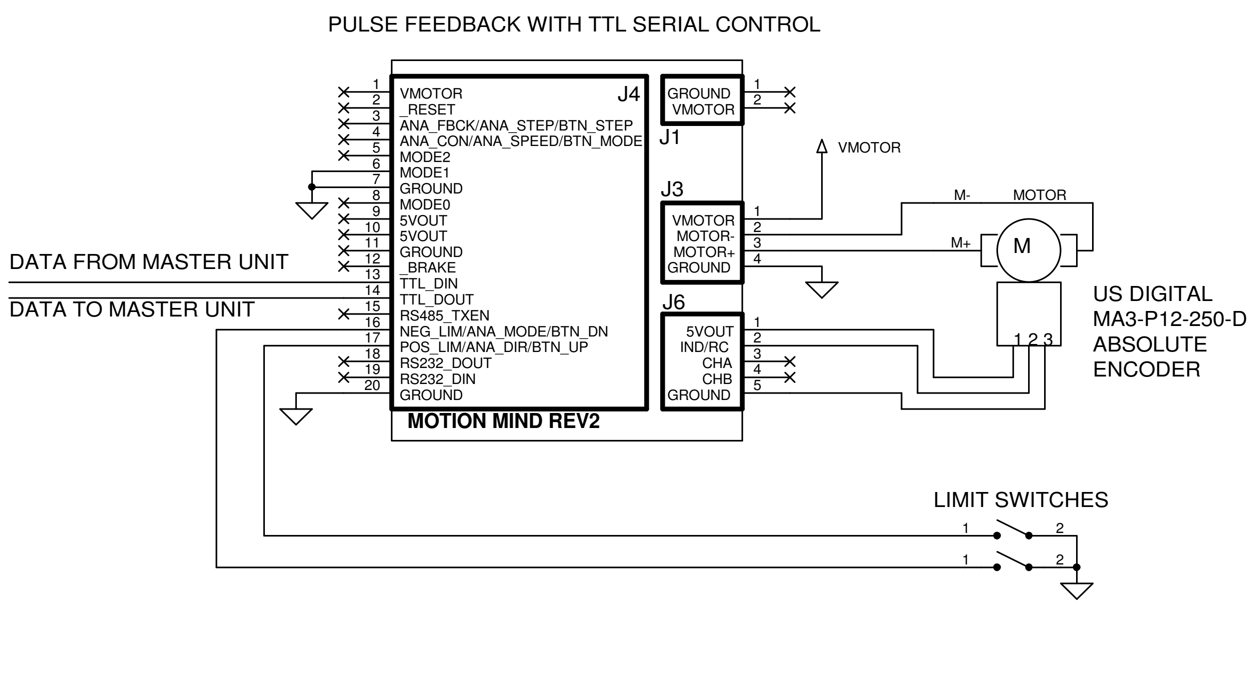


Figure 16C: Encoder Pulse Feedback Schematic



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, 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. The following PID settings may be a useful starting point for tuning your system.

VELOCITYLIMIT = 1023
VELOCITYFF = 0
PTERM = 10000
ITERM = 10
DTERM = 6000
PIDSCALAR = 12
FUNCTION.SATPROT = enabled
FUNCTION.VELLIMIT = disabled

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 you mechanical system so no damage can occur if the system is connected incorrectly.

Summary of Serial Commands Used by Mode 5

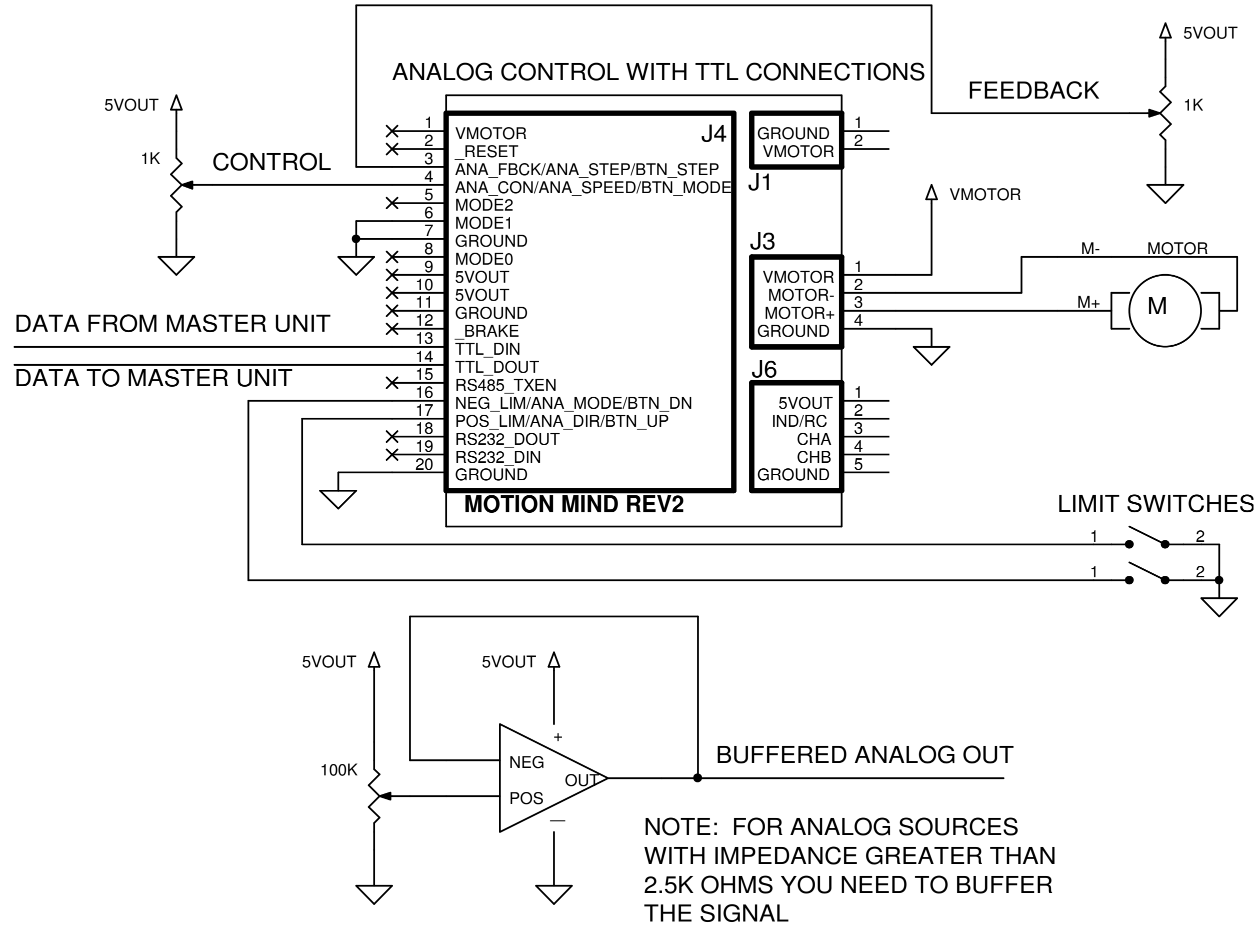
Command	**Description**
MOVETO_ABSOLUTE	When the FUNCTION.ADSERIAL bit is set this command may be used to provide the desired analog position. Position value should be limited to 0 to 1023.
MOVETO_RELATIVE	When the FUNCTION.ADSERIAL bit is set this command may be used to modify the desired analog position. 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
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

Programmable Registers	Description
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	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.
VPLIMIT	Sets a virtual positive position limit (a software limit switch) when FUNCTION.VIRTLIMIT is also set.
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
AMPSLIMIT	Default is 512 (10.24A), this value sets the active current limiting the threshold. Each count corresponds to approximately 20mA
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.ACTIVESTOP	Whenever the PWM output is 0 both low side MOSFETs are tied to ground
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.
FUNCTION.ENABLEDB	When set the contents of the DEADBAND register are used to create a dead band around the desired position.
FUNCTION.RCPOS-ENCFDBCK	When set the feedback position signal is taken from an encoder connected to J6.
FUNCTION.VIRTLIMIT	Enables virtual software limits defined by VNLIMIT and VPLIMIT registers.
FUNCTION.DISABLEPID	Turns off PID output (no drive signal to motor).
FUNCTION2.ADX2	Only used when FUNCTION.RCPOS-ENCFDBK is set. DESIRED POSITION is analog conversion from J4 P4 (0-1023) x 2 (0-4046). For example, an analog signal of 2.5V that converts to 512 bits would command the motor to position the encoder at 1024 counts (512 x 2). This allows wider range of encoder values to be used but still limits resolution of control to 10 bits.
FUNCTION2.ADX4	As above but DESIRED POSITION is analog conversion from J4 P4 (0-1023) x 4.
FUNCTION2.ADX8	As above but DESIRED POSITION is analog conversion from J4 P4 (0-1023) x 8.
FUNCTION2.ADX16	As above but DESIRED POSITION is analog conversion from J4 P4 (0-1023) x 16.
FUNCTION2.ADX32	As above but DESIRED POSITION is analog conversion from J4 P4 (0-1023) x 32.
FUNCTION2.ADX64	As above but DESIRED POSITION is analog conversion from J4 P4 (0-1023) x 64.

Figure 17: 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 you mechanical system so no damage can occur if the system is connected incorrectly.

4.8 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.

4.9 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 (19.2KBPS or 9.6KBS), or connecting to a TTL (logic) or RS232 (PC serial port) electrical levels. Note that a baud rates of 38.4KBPS and 115.2KBPS may be used by setting FUNCTION2 bits in some firmware revisions. When operating in TTL mode the TTL_DOUT (J4 P14) is open-collector. Multiple Motion Mind controllers can share the TTL serial bus. Figures under the Serial (Open-Loop) Control Mode show a shared TTL bus as well as an RS232 connection scheme. The serial data receive buffer can accept no more than 50 bytes regardless of the operating mode.

Sending an address of “99” in ASCII mode, or 99-255 in binary mode, constitutes a broadcast command. These commands will be accepted and no response will be returned. This allows multiple units on a communication bus to execute a command at the same time.

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 has 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. The WRITE_STORE command may require 50ms before responding.

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 (excepting broadcast commands, see above). 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 3.11 for 2’s compliment format).

Example of Binary MOVETO_ABSOLUTE Command

	Command	Address	Data0	Data1	Data2	Data3	Checksum
Decimal	21	1	240	216	255	255	220
Hex	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. **Note that responses to the ASCII Read command may exceed 5ms if the contents of the register being read represent a large number.** Responses to ASCII commands will my take up to 10ms.

The ASCII protocol is less strict than the binary protocol. Data is expected to be 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>

READ commands will return a 4 character ASCII code and an equal sign prior to the registers contents. This is to ease reading of the registers. For example reading register index 0 the ACTUAL POSITION register will return a string similar to this "APOS=-10000"<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'	

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	APOS=100000<CR><LF>
FUNCTION.RETVEL	VELO=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 value 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 wont 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 compliment 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 compliment 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 wont 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 compliment 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.

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

***Use 200 for 1X encoder mode**

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_STORE 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 Command Example
(reading register 0 holding 20,000 decimal)

	Command	Address	Data0	Data1	Data2	Data3	Checksum
decimal	26	1	1	0	0	0	28
	Address	**Data0**	**Data1**	**Data2**	**Data3**	**Checksum**	
response	1	32	78	0	0	111	
	Command	**Address**	**Data0**	**Data1**	**Data2**	**Data3**	**Checksum**
hex	H'1A'	H'01'	H'01'	H'00'	H'00'	H'00'	H'1C'
	Address	**Data0**	**Data1**	**Data2**	**Data3**	**Checksum**	
response	H'01'	H'20'	H'4E'	H'00'	H'00'	H'6F'	

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: PTRM=6000<CR><LF>
How it appears on screen:
R01 04
PTRM=6000

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. These commands may require take as long as 50ms for the response to be generated. **You should not use these commands if operating in a closed-loop velocity control or position control mode (modes 4 or 5 typically).**

READ 97-STATUS register bits: Sending a READ index of 97 returns and 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
	S5ON<CR><LF>	‘Virtual negative limit position has been reached
	S6OFF<CR><LF>	‘Virtual positive limit position has not been reached
	S7ON<CR><LF>	‘Amps limit has been reached
	S8ON<CR><LF>	‘PWM output signal is being limited
	S9OFF<CR><LF>	‘Controller has not reached the DESIRED POSITION yet

READ 98-FUNCTION register bits: Sending a READ index of 98 returns and 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>	‘Active stop 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
	F14OFF<CR>LF<>	‘PID output in closed loop mode is not disabled
	F15ON<CR><LF>	‘LED blink on power up is disabled

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,(…all registers contents displayed…),<CR><LF>

5.10 READ REGISTER Command:

The READ REGISTER command was included to be easier to implement than the legacy READ command. This command allows you to read the contents of a single register from the Motion Mind. Additionally, any registers added through firmware upgrades with an index value higher than 31 won't be accessible through the legacy READ command, but will be accessible via the READ REGISTER command.

5.10.1 Binary Protocol

This command is issued by sending the command byte, address, index value of the register you want to read, and a checksum. The index value for each register can be found in section 6.0 of the datasheet. The response to this command will consist of the off the address of the responding unit, the data associated with the register that is read, and a checksum. Here is an example of reading the VELOCITY LIMIT register containing a value of 1023 (in decimal).

29(command), 1(address), 1(register index), 31(checksum)

Since the VELOCITY LIMIT registers is a 2 byte wide value, the response would be
1(address), 255(register contents LSB), 3(register contents MSB), 3(checksum)

READ REGISTER Command Binary Example
(reading register 0 holding 20,000 decimal)

	Command	**Address**	**Register**	**Checksum**			
decimal	29	1	0	30			
	Address	**Data0**	**Data1**	**Data2**	**Data3**	**Checksum**	
response	1	32	78	0	0	111	
	Command	**Address**	**Data0**	**Checksum**			
hex	H'1D'	H'01'	H'00'	H'1E'			
	Address	**Data0**	**Data1**	**Data2**	**Data3**	**Checksum**	
response	H'01'	H'20'	H'4E'	H'00'	H'00'	H'6F'	

5.10.2 ASCII Protocol

This command is not implemented in ASCII mode.

5.11 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.

5.11.1 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'		

5.11.2 ASCII Protocol

Command: X01<CR><LF>
Response: OK<CR><LF>
How it appears on screen:
X01
OK

5.12 RESET Command:

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

5.12.1 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'		

5.12.2 ASCII Protocol

Command: Y01<CR><LF>
Response: OK<CR><LF>
How it appears on screen:
Y01
OK

5.13 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 3.11). 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.

Figure18: 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	1152
11	RCMIN	R/W	2	0 to +65,535	576
12	RCBAND	R/W	2	0 to +65,535	15
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	2	0 to +1023	15
27	DESIREDPOSITION	R	4	-2,147,483,648 to +2,147,483,647	N/A
28	AMPSLIMIT	R/W	2	0 to +1023	512
29	AMPS	R	2	0 to +1023	N/A
30	FUNCTION2	R/W	2	0 to +65,535	0
31	THERMISTOR	R	2	0-1023	N/A

6.2 Register Descriptions:

Figure19: 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 (PWMOUT). 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 1736ns. 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 1736ns. 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 1736ns 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 1736ns.

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 3rd channel on many quadrature encoders, and occurs with each encoder revolution.
23	VNLIMIT	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	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	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	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	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.
28	AMPSLIMIT	0-1023 in roughly 20mA increments. This value sets the active current limit threshold. Setting this to 0 disables the active current limit function. Example, 512 sets the current limit at ~ 10.24A
29	AMPS	0-1023 in roughly 20mA increments. This is the current measurement the Motion Mind is reading. Example, 512 would be approximately 10.24A
30	FUNCTION2	The FUNCTION2 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.
31	THERMISTOR	10-bit AD temperature measurement ˚C = (THERMISTOR register – 1104) x -0.165

6.3 FUNCTION Register and FUNCTION2 Register Bits:

The FUNCTION and FUNCTION2 registers contain a number of bits that are used to enable or disable certain functions. Both registers are 16 bits (2 bytes) wide. These registers are written to or read like and other register (WRITE and READ commands). Their contents may be stored in nonvolatile memory using the STORE command.

For example, if you wanted to set the SATPROT, ACTIVESTOP, and ENABLEDB bits you would write binary '00001000 10010000' to the FUNCTION register (hexadecimal 0x0890 which also equals decimal 2192). Most calculators can do these conversions.

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0

Figure20: 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/WRITE 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/WRITE 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/WRITE 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	ACTIVESTOP	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.
9	ADSTEP	When set: Forces modes 0,1, and 2 to use digital value for motor speed change limit (instead of analog). 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	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).

Figure20 (continued): FUNCTION Register Bits

Bit	Name	Description
11	ENABLEDB	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.
12	RCPOS-ENCFDBCK	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	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
14	DISABLEPID	When set: PID output in closed loop modes is disabled When clear: PID output is normal in closed loop modes
15	DISABLEBLINK	When set: The D6 mode indicator LED blink on power is skipped When clear: The D6 mode indicator LED blink on power up occurs as normal

If you wanted to set the FREQX2 and ADX8 bits you would write binary ‘0000000000010010’ to the FUNCTION2 register (hexadecimal 0x0012 which also equals decimal 18). Most calculators can do these conversions.

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Figure21: FUNCTION2 Register Bits

Bit	Name	Description
0	FREQ/2	When set: PWM frequency is divided by 2 (9KHz). This setting reduces PWM resolution from 10 bits to 9 bits, although this change does not impact user interfacing with the device. Bits 0 and 1 are sequentially tested so the highest bit set has its function implemented. When clear: No effect
1	FREQX2	When set: PWM frequency is multiplied by 2 (36KHz). This setting reduces PWM resolution from 10 bits to 9 bits, although this change does not impact user interfacing with the device. Bits 0 and 1 are sequentially tested so the highest bit set has its function implemented. When clear: No effect
2	ADX2	Only used in MODE5 with FUNCTION.RCPOS-ENCFDBCK (bit 12) set. This bit multiplies the analog control signal used to determine the DESIRED POSITION (taken from J4 P4). Using this can expand the range of motion of a motor using encoder feedback. For example, an analog signal of 2.5V that converts to 512 bits would command the motor to position the encoder at 1024 counts (512 x 2). Bits 2, 3,4,5,6 and 7 are sequentially tested so the highest bit set has its function implemented. When set: Multiplies control voltage used to determine DESIRED POSITION by 2. When clear: No effect.
3	ADX4	As above but.. When set: Multiplies control voltage used to determine DESIRED POSITION by 4. When clear: No effect.

Figure21 (continued): FUNCTION2 Register Bits

Bit	Name	Description
4	ADX8	As above but.. When set: Multiplies control voltage used to determine DESIRED POSITION by 8. When clear: No effect.
5	ADX16	As above but.. When set: Multiplies control voltage used to determine DESIRED POSITION by 16. When clear: No effect.
6	ADX32	As above but.. When set: Multiplies control voltage used to determine DESIRED POSITION by 32. When clear: No effect.
7	ADX64	As above but.. When set: Multiplies control voltage used to determine DESIRED POSITION by 64. When clear: No effect.
8	OVERTEMP	When set: The over-temperature protection is disabled When clear: Temperature over 90˚C will cause the H-bridge to turn off. The H-bridge will be re-enabled automatically when the temperature drops below 80˚C.
9	PULSEFB	When set: In mode 4 a 0-4.096ms pulse at J6 INDEX pin may be used as the feedback source. When clear: The feedback source is not from a pulse at J6 INDEX pin
10	BAUD38400	When set: The controller serial communication operates at 38.4KBPS when this bit is set at power up. In order to use this communication rate the FUNCTION2.BAUD38400 bit should be set and stored in EEPROM and then the device should have its power cycled. When clear: The serial communication rate is determined by the SW1 hardware setting.
11	NEWVELMODE	When set: If operating in Velocity mode the position register contents (Desired and Actual) are replaced with velocity values. This forces the PID to work directly on the velocity values. PID outputs are then added to the previous PID output. This means a PID error of 0 results in no change in the motor drive signal, versus the normal mode of operation where a PID error of 0 results in a stopped motor. When Clear: The velocity operating mode uses position data to lead the drive signal in order to create the desired velocity.
12	BAUD115200	When set: The controller serial communication operates at 115.2KBPS when this bit is set at power up. In order to use this communication rate the FUNCTION2.BAUD115200 bit should be set and stored in EEPROM and then the device should have its power cycled. When clear: The serial communication rate is determined by the SW1 hardware setting.
13	FASTPID	When set: PID loop time is reduced from 5mS to 500uS. Included in Rev A firmware. Should not be used for when serial control or velocity control is implemented. When clear: PID loop defaults to 5mS
14	Unused	

6.4 STATUS Register Bits:

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

Figure22: STATUS Register Bits

Bit	Name	Description
0	NEGLIMIT	Set when the input *NEG_LIM* (J4 P16) is at 0V
1	POSLIMIT	Set when the input *POS_LIM* (J4 P17) is at 0V
2	BRAKE	Set when the input *_BRAKE* (J4 P12) is at 0V
3	INDEX	Set when the input *IND/RC* (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	CURRENTLIMIT	Set when the AMPS register has exceeded the AMPS LIMIT setting and the PWM signal is being regulated down
8	PWMLIMIT	Set when the PWMOUT register has exceeded the PWM LIMIT setting and the PWM signal is being regulated down
9	INPOSITION	Set when the in closed loop mode and the DESIRED POSITION equals the ACTUAL POSITION register. If the dead band is enabled this bit is set when the DESIRED POSITION equals the ACTUAL POSITION and remains set as long as the ACTUAL POSITION is DESIRED POSITION +/- DEADBAND
10	TEMPFAULT	Set when the thermistor reads higher than ˚90C, cleared when the temperature drops below ˚80C
11	Unused	
12	Unused	
13	Unused	
14	Unused	
15	Unused	

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. Exceptions to this warrantee include any sales outside of the continental United States where shipping cost precludes product exchanges.

8.0 Errata Sheet

The ERRATA sheet tracks known issues 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.

Analog Measurement Jitter: Firmware Revisions - 1: The analog measurements exhibited 1-2 bits of jitter. The measurements were converted to an average of 8 measurements to eliminate the jitter. This increases the effective update rate of analog measurements to 40ms.

ASCII READ Response Times: Firmware Revisions 1, 2: As register contents increase in size the time it takes to convert register values to ASCII values increases. These times were exceeding the 5ms PID update rate and causing errors in timing for position and velocity control. Revision 3 addresses most of these issues with the exception of the READ 97, 98, and 99 commands. More efficient serial communication can be achieved by using the binary serial interface.