Motor Mind B enhanced
Motor Controller Data Sheet

Revision 2
July 19, 2010

SOLUTIONS CUBED, LLC
256 East First Street Chico, CA 95928
phone: 530.891.8045 fax: 530.891.1643
www.solutions-cubed.com

Table of Contents

List of Figures

1.0 Revision Log

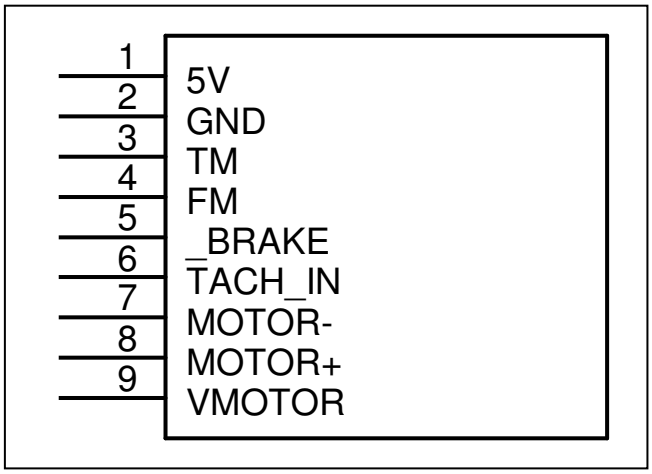
Date	Rev	Description	By
04-05	1	Original Implementation	L. Glazner
07-10	2	WRITE_PI description modified to reflect no EEPROM storage, and added default values	L. Glazner

2.0 Introduction

Motor Mind B enhanced
Serial DC Motor Controller

FEATURES

- Up to 1.75A continuous current (6A peak), 6-36VDC brush motors
- PWM frequency of 242Hz or 15.5KHz
- TTL serial interface, 2.4KBPS or 9.6KBPS
- Enhanced command set from earlier Motor Mind B modules
- Backwards compatible with earlier Motor Mind B modules
- Tiny 1.0" x 1.2" packaging, 9-pin 0.1" SIP
- Under-voltage, over-temperature, over-current protection built in
- 0-65,535Hz tachometer
- Closed loop PI Filter speed control
- Pulse counter mode of operation
- User accessible EEPROM



COMPONENT SIDE

2.1 DESCRIPTION

The Motor Mind B enhanced DC Motor Controller is capable of controlling one brushed DC motor. Control is implemented through a TTL (logic level) serial interface.

The Motor Mind B enhanced (MMBe) is pin and functionally compatible with earlier versions of the Motor Mind B. Care has been taken to make this new revision as close as possible to the previous version of the Motor Mind B (MMB). Significant differences are detailed in section 3.7 of this datasheet. Physically they me be distinguished by the lack of a heat sink on the MMBe.

Beyond the physical changes the MMBe has additional commands and functionality designed into it. A primary improvement is the H-bridge IC used. This H-bridge has greater current handling capacity and better protection from under-voltage, over-temperature, and over-current conditions. Fault and communication LEDs were added to provide visual indications of functionality. The MMBe now accommodates 2.4KBPS and 9.6KBPS baud rates, and can be configured to operate with a pulse-width-modulation frequency of 15.5KHz. The previous MMB operated with software generated PWM frequency of 61Hz, which was derived from the time-base required for tachometer measurements and serial communication. The MMBe defaults to 15.5KHz PWM frequency but can be programmed to operate at the much lower 242Hz to allow for backwards compatibility. New designs should use the more efficient (and less audible 15.5KHz) frequency.

Enhanced tachometer, speed control, and count commands may be used to implement speed control or rudimentary position control. The SPDCON command may now be implemented in the original increment/decrement mode, or a Proportional/Integral (PI) filter may be implemented.

The board measures 1.0" x 1.2" x 0.3", and comes in a 9-pin SIP package (0.1" spacing).

Note: In this datasheet hexadecimal values are represented by a prefix of "0x". For example decimal 10 is represented as 0x0A.

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 electrical characteristics should be used to determine applicable ranges of operation.

Storage Temperature	-50℃ to +150℃
Operating Temperature	-40℃ to +85℃
Motor Voltage (VM)	+6V to +36.0V
Voltage on pins TM,FM,TACH_IN, _BRAKE, 5V	-0.3V to +5.5V
Voltage on VM, M+, M-	40V transient spike
Motor Current Load	6A peak / 1.75A continuous

3.2 Electrical Characteristics

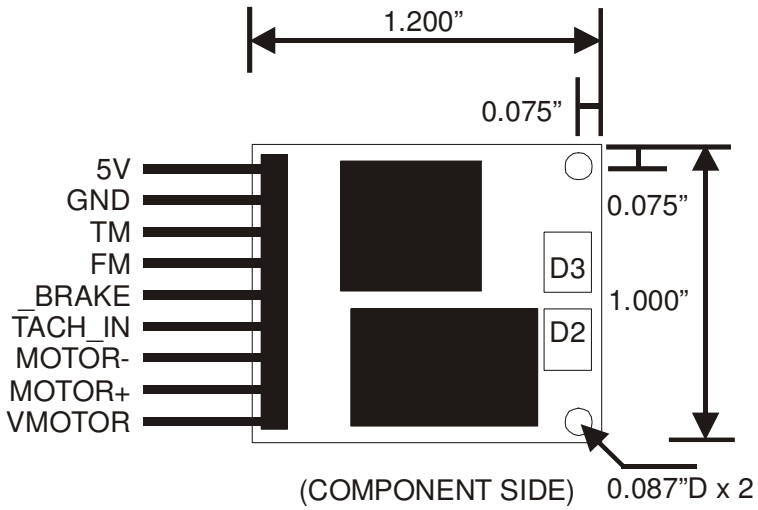
Characteristic	Min	Typ	Max	Unit	Notes
Supply Voltage	6		36	V	VMOTOR voltage
Supply Current 5V pin		2		mA	Indicator LEDs off
5V Pin Voltage	4.5		5.5	V	
Short circuit current limit			15	A	Extended short-circuit could damage PCB traces, other components
Over-current trip point	6	8	10	A	The higher the temperature the lower the trip point
Peak load current			6	A	<100ms pulsed at a period of 1S
Max continuous motor current	1.5	1.75	2.0	A	Continuous current rating at room temp. 95% duty-cycle, resistive load.
Under-voltage shutdown VM pin	3.5	4.7	5.6	V	VM voltage that causes under-voltage fault
Under-voltage turn-on VM pin		5.3	6.0	V	VM voltage that allows H-bridge to turn on
Over-temperature shutdown	150	175	200	℃	Hysteresis is 20℃ (if shutoff occurs at 175℃ then fault clears at 150℃)
Low Level Input FM, _BRAKE, TACH pins			0.5	V	pin pulled to +5V with 12.5kΩ resistor
High Level Input FM, _BRAKE, TACH pin	2.0			V	pin pulled to +5V with 12.5kΩ resistor
Low Level Output TM pin			0.6	V	TM pin has a 270Ω resistor in series on the PCB
High Level Output TM pin	3.8		4.8	V	TM pin has a 270Ω resistor in series on the PCB
PWM Frequency	0.242		15.5	KHz	User selects frequency through serial interface
Tachometer Frequency	0		65,535	Hz	
Tachometer Resolution		8		Hz	
TACH_IN pin voltage	0		20	V	This pin has a 1.5KΩ series resistor and a 5.6V zener diode for protection

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 mounting holes may be used with 2-56 machine screws and nuts

Figure 1: Mechanical Dimensions



Mechanical Landmark Descriptions

Landmark	Type	Description
D2	LED – green	COMM- Indicates serial communication was received correctly
D3	LED – red	FAULT- Indicates an over-temperature, over-current, or under-voltage condition exists.
J1	1x9 0.1” right angle header	Main input and motor connector

3.4 Connectivity Overview

J1 Main Input Pin Descriptions

Pin	Name	Type	Description
1	*+5V*	POWER	+5V supply provided to MMBe controller
2	*GND*	POWER	Motor ground return and +5V logic return
3	*TM*	OUTPUT	To Master - TTL data out to Master; 8N1 TTL levels 2.4KBPS or 9.6KBPS, has a series 270Ω resistor to prevent shorting to ground
4	*FM*	INPUT	From Master - TTL data in from Master; 8N1 TTL levels 2.4KBPS or 9.6KBPS
5	*_BRAKE*	INPUT	Logic low shuts down motor
6	*TACH_IN*	INPUT	TTL tachometer or encoder input has a series 1.5KΩ resistor to a 5.6V zener diode.
7	*MOTOR-*	POWER	Negative motor lead connection
8	*MOTOR+*	POWER	Positive motor lead connection
9	*VMOTOR*	POWER	Motor supply voltage 6-36VDC

3.5 Indicator LEDs

D2 (green) will light briefly when serial communication is received, and will also blink once on power-up. D3 (red) will light when the motor voltage is less than ~5.6V, the current exceeds 6A, or the temperature on the PCB exceeds 175°C. The red LED will turn off when the fault condition disappears, but a FAULT bit will remain set in the STATUS register until that register is read.

3.6 Differences Between the MMBe and earlier MMB

Area of difference	MMBe	MMB
Hardware Changes		
Physical	no heat sink, has indicator LEDS	silver heat sink
Protection	under-voltage, over-current, over-temperature	over-temperature
Peak Current	6A	3.5A
Continuous Current no cooling	1.75A	1.3A
Motor Voltage	6-36VDC	7-30VDC
TM Pin	Has a series 270Ω resistor to prevent shorting to ground. Any external pull-up resistors used on this pin should be 10KΩ or greater	Has pull-up resistor on PCB and is only actively drive low (pseudo open-collector).
Firmware Changes		
1-wire communication (sharing the TM-FM pins with a single i/o on the master unit)	1-wire communication is possible but care should be taken to ensure that the FM-TM connection at the master unit is primarily configured as an input. The TM pin on the MMBe is always an output and unless responding to commands is driven to +5V. A series 270Ω resistor protects this pin from shorting which would otherwise occur if FM were driven low while connected to TM.	Pseudo-open collector TM pin allows-1 wire communication
PWM Frequency	242Hz or 15.5Khz (can be toggled with CHANGE_FREQ command)	61Hz
How to exit SPDCON mode	Any command other than SPDCON or SPDCON_FREQ will cause SPDCON mode to be exited	FM line must be held low for GATETIME x 2, then returned to +5V
SPDCON control	Includes both the increment/decrement method and a proportional-integral (PI) control loop. Mode of SPDCON can be toggled with SPDCON_MODE command.	Motor speed is incremented or decremented based on desired speed
How to exit COUNT command before count is completed	Send EXIT_COUNT command.	A hard power reset is required
Serial response time to TACH command	5ms	Varied based on GATETIME value
Tachometer range	0-65,535Hz	0-65,535Hz
Tachometer resolution	8Hz, options for sending GATETIME have been removed. GATETIME value is still accepted but no longer effects tachometer measurement time or resolution.	Varied based on GATETIME value
Baud Rate	2.4KBPS or 9.6KBPS (can be toggled with CHANGE_BAUD command).	2.4KBPS
Command response times	SPDCON-TACH commands may take as long as 5ms to respond if optional GATETIME is not sent. Other commands respond in 500-1000us.	Commands are responded to within 500-1000us
STATUS register	The reset flag bit was removed and additional bits have been defined (PIMODE, FAULT, BRAKEON, FREQLOW, BAUD9600).	Direction and reset bits are available

3.7 Two's Compliment Number System

The SETDC_DIR command accepts negative numbers. 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. For 8-bit number you can also subtract the absolute value of the negative number from 256. Example: 256 - | -65 | => 256 – 65 = 191 = 0xBF.

Two's Compliment Examples

Number	Absolute Value	Hexadecimal	Compliment	Two's Compliment
-127	127	0x7F	0x80	0x81
-65	65	0x41	0xBE	0xBF

3.8 The PI Filter

Speed control mode can be used with an increment or decrement control scheme, or a Proportional Integral (PI) filter may be applied to the error signal. In the standard increment/decrement mode the Master unit issues a desired frequency with the SPDCON command, and the motor speed is incremented (if the tachometer is lower than the desired value) or decremented (if the tachometer is higher than the desired value). When configured for PI mode the error signal (desired frequency – actual frequency) is multiplied by a Proportional term, then a sum of errors-over-time is multiplied by an Integral term, and added to the previous result. The resulting 32-bit two's compliment number is scaled down to a value between +255 to -255. This value is applied to the motor as the duty cycle or speed control. It is recommended that PI mode be used with the 15.5KHz frequency setting.

If tuned correctly the PI filter will provide responsive control of the motor speed and account for changes in the motor load. Very low frequencies are difficult to control, so the PI filter is best used with frequencies in the hundreds of Hertz or higher range. The components of the PI filter are modified with the WRITE_PI command. The STATUS.PIMODE bit must be set before the PI filter will be used in SPDCON mode. This bit may be toggled with the SPDCON_MODE command. SPDCON motor drive updates occur 8 times per second.

Component of PID	Description
PTERM	The Proportional term is multiplied by the error signal (desired frequency – actual frequency). This term has the greatest effect on changing the motor speed, and will typically be much larger than the Integral term. This is an 8-bit value. Default value is 100.
ITERM	While the PI mode is in operation the error signals are summed in a register. The Integral term is then multiplied by the sum of errors and the result is added to the proportional term and error signal product. The integral term allows for small corrections to motor speed to occur over time. This value will be much smaller than the Proportional term. This is an 8-bit value. Default value is 5.
PISCALAR	The result of the previous multiplications will be much greater than the +/-255 used for setting the motor speed duty cycle. Therefore the result must be scaled down. The resulting output is divided by $2^{PISCALAR}$. This is an 8-bit value. Default value is 14.
STATUS.PIMODE	Set this bit to enable PI mode for use with SPDCON operation.

3.9 Tuning the PI Filter

Tuning the PI requires a bit of trial and error. For starters you can change the ITERM to zero, and then modify the PTERM until you get roughly the correct speed. Then increment the ITERM to fine tune control. The PISCALAR is the "number of divide-by-twos" applied to the PI output to get the motor speed update ($PWM = PI/2^{PISCALAR}$). So making the PISCALAR larger reduces the PWM (divides it by 2), decreasing it increases the PWM (doubles it). Therefore, the PISCALAR will have a large effect on the way PI mode drives the motor. If your motor is ramping (zooms up to top speed, then back to zero) increase the PISCALAR. If changes in the PTERM do not create motor movement, then decrease the PISCALAR. You may find that one value of the PISCALAR causes cycling, and the next higher value doesn't drive the motor at all. That's okay, this would be the point where adjusting the PTERM would take place.

3.10 External Capacitance on 5V / VMOTOR Pins

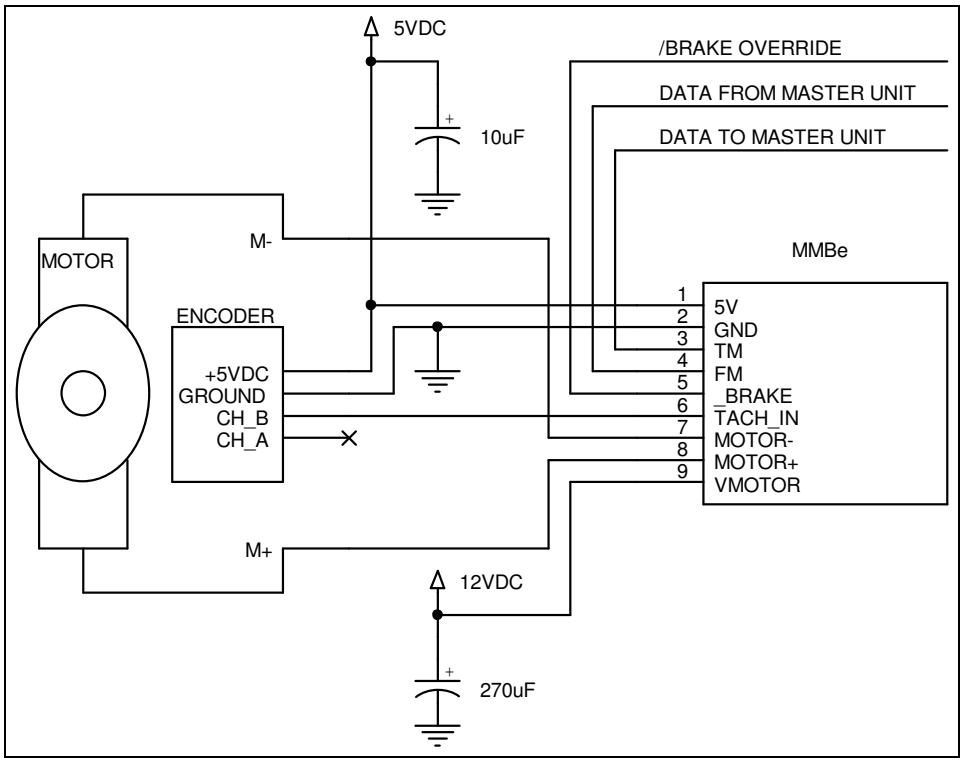
Electrolytic capacitors may be important components in your circuit. Some capacitance should be located near the MMBe to provide short bursts of current and to smooth power supply spikes. If connections to your power supply or battery exceed 6" you should consider adding capacitance to the 5V and VMOTOR pins. The VMOTOR capacitance should be capable of providing a ripple-current of ½ of your peak current. Low ESR capacitors are good candidates for use at the VMOTOR pin. See Figure 2 for an example of capacitor placement

3.11 Tachometer Measurements

Tachometer measurements used by the TACH, SPDCON, and SPDCON_FREQ commands may have an additional GATETIME appended to some commands. The GATETIME is no longer used in the MMBe, but it will still accept commands with this value attached. Tachometer measurements now occur every 125ms (resolution = 8Hz) and are ongoing in the background of the MMBe operating system. Therefore tachometer measurements are available immediately upon receipt of the TACH, or SPDCON_FREQ command. Reading the tachometer more often than 125ms is possible, but the frequency measurement provides new data only every 125ms.

The commands READ_FREQCAL and WRITE_FREQCAL may be used to calibrate the frequency measurements to account for differences in the internal clock used by the MMBe. The FREQCAL register is calibrated to within +/-0.5%(50Hz) of 10KHz at the factory.

Figure 2: Standard Tachometer or Encoder Connection Schematic



3.12 Count Measurements

The COUNT command can use the same schematic as that shown above. But you should expect motor inertia to drive motor past the exact desired count. The READ_COUNTER can provide you with the actual number of pulses counted including the overshoot due to inertia. When a COUNT command is executed the background monitoring of the tachometer is stopped. Sending additional COUNT or READ_COUNTER commands does not restart the tachometer measurements, but all other commands will cause the tachometer measurements to begin.

The internal counter used by the COUNT command will rollover at 65,536 counts. Since the motor will not stop moving immediately after the correct number of pulses is counted, it is recommended that counts greater than 65,000 should not be used.

4.0 Communication Protocol

4.1 Overview

The MMBe is controlled via a serial data interface. The protocol requires data to be received at TTL levels (0V = logic 0, 5V = logic 1), 8 data bits, no parity bit, 1 stop bit (8N1). Data is arranged least significant bit first (LSB). The default baud rate is 2.4KBPS, but using the CHANGE_BAUD command allows the user to toggle the baud rate between 2.4KBPS and 9.6KBPS. Each byte of data should be sent with no more than 500us between bytes. A special note regarding the TM (data to master) pin is required. This pin has a 270Ω series resistor to protect this output from shorting ground. If pull-up or pull-down resistors are used external to the MMBe they should be 10KΩ or larger (FYI: pull-up/down resistors are unnecessary on the TM pin).

Serial commands are sent starting with the sync byte (decimal 85, hexadecimal 0x55), the command byte, and any additional data required by the command. No checksum is required.

A typical application would have some master controller (a popular one is the Parallax BASIC Stamp, www.parallax.com). But you may also control it via a PC serial COMM port. With a PC the data voltages must be converted from RS232 (+12V = logic 0, -12V = logic 1) to TTL levels. Figures 3 and 4 are examples of the connections required for these applications.

Figure 3: PC Serial Port Connection Schematic

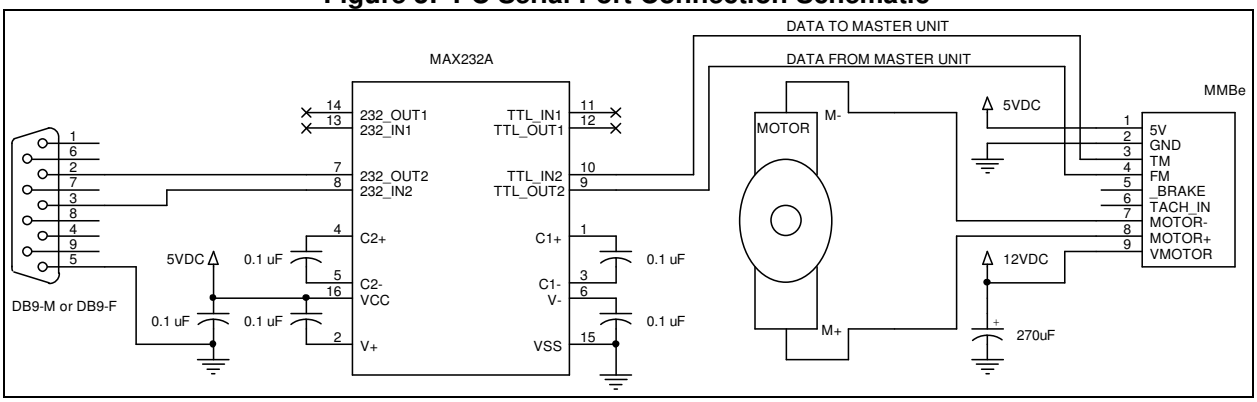
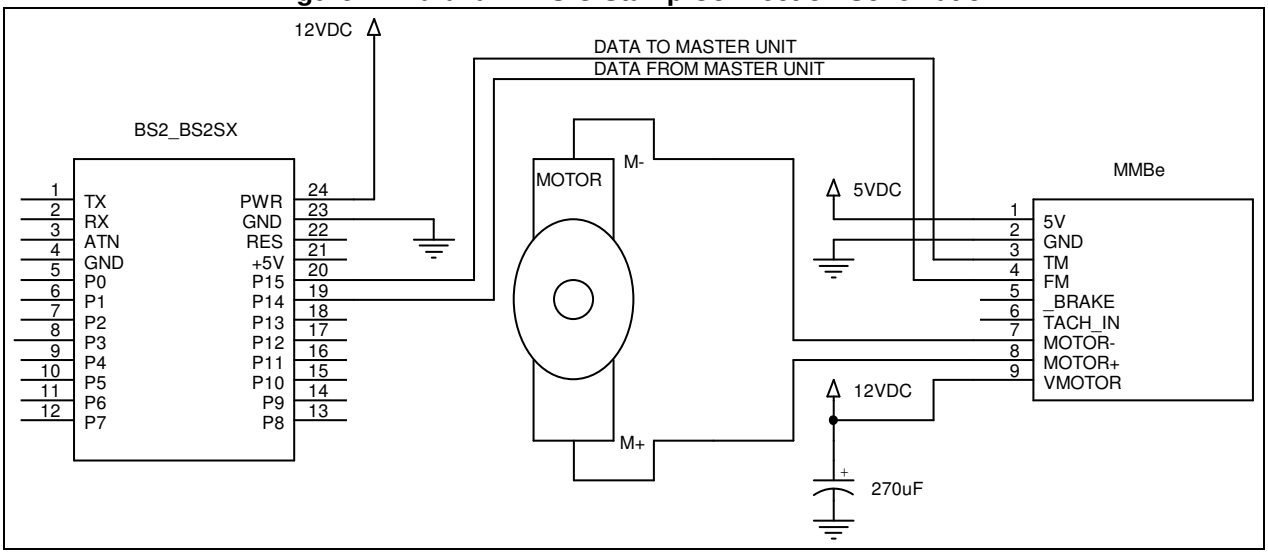


Figure 4: Parallax BASIC Stamp Connection Schematic



4.2 Response to Commands

Many of the commands available in the MMBe do not automatically elicit responses. But in motor control applications it is often desirable to have some feedback to ensure that the commands have been received. For this reason a confirmation response has been built into the MMBe (and earlier versions). Setting the highest bit of any command byte will cause the MMBe to respond first with a confirmation, and then with any response built into the command itself. For example the command byte for SETDC is decimal 3 (0x03), sending a command byte of 131 (0x83) will cause the confirmation byte to be returned. A confirmation byte is a decimal 170 (0xAA). Confirmation bytes are not shown in the following examples. Responses to commands will typically occur within 500us-1000us of sending the data. There are three exceptions to this rule. First, commands that include the option of a GATETIME value (TACH and SPDCON) may take as long as 5ms to respond. Second, commands that write to EEPROM (WRITE_EEPROM for example) may take 10ms-40ms to respond, although the actual time required should be much less than that. Finally, the COUNT command will respond with the STATUS and SPEED values after the count is complete.

4.3 Command Summary

The following table summarizes the commands available in the MMBe. Please note that the confirmation byte (CONF) and GATETIME are optional. Regarding the confirmation byte, it will only be returned if the highest bit of the command byte is set. The GATETIME may be sent with the TACH and SPDCON commands but no longer has an effect on tachometer measurements.

Command	Command Byte decimal (hex)	Additional Byte(s) (* denotes optional)	Response (* denotes optional)	Writes to EEPROM
STOP	0 (0x00)		CONF*	NO
REV	1 (0x01)		CONF*	NO
TACH	2 (0x02)	GATETIME*	CONF*	NO
SETDC	3 (0x03)	SPEED	CONF*	NO
SPDCON	4 (0x04)	FREQHI, FREQLO, GATETIME*	CONF*	NO
STATUS	5 (0x05)		CONF*, STATUS, SPEED	NO
COUNT	6 (0x06)	COUNTHI, COUNTLO	CONF*, STATUS, SPEED	NO
EXIT_COUNT**	7 (0x07)		CONF*	NO
SETDC_DIR**	8 (0x08)	SPEED (2's compliment)	CONF*	NO
WRITE_EEPROM**	9 (0x09)	ADDRESS, DATA	CONF*	YES
READ_EEPROM**	10 (0x0A)	ADDRESS	CONF*, DATA	NO
READ_COUNTER**	11 (0x0B)		CONF*, COUNTHI, COUNTLO	NO
CHANGE_BAUD**	12 (0x0C)		CONF*	YES
CHANGE_FREQ**	13 (0x0D)		CONF*	YES
READ_FIRMWARE**	14 (0x0E)		CONF*, FIRMWARE REV	NO
READ_PI**	15 (0x0F)		CONF*, PTERM, ITERM, PISCALAR	NO
WRITE_PI**	16 (0x10)	PTERM, ITERM, PISCALAR	CONF*	NO
SPDCON_MODE**	17 (0x11)		CONF*	YES
SPDCON_FREQ**	18 (0x12)		CONF*	NO
READ_FREQCAL**	19 (0x13)		CONF*, FREQCAL	NO
WRITE_FREQCAL**	20 (0x14)	FREQCAL	CONF*	YES

*Optional GATETIME data or optional CONF response
**New command not available on earlier MMB versions

4.4 STOP Command

The STOP command will stop the motor. Stopping the motor may also be implemented by sending a SETDC or SETDC_DIR command with a SPEED of 0.

Example of STOP Command

Send	Sync	Command
Decimal	85	0
Hex	0x55	0x00

4.5 REV Command

The REV command will flip the direction the motor is turning. The MMBe will always power up running forward (MOTOR+ = VMOTOR, MOTOR- = GND during the positive duty cycle). REV toggles the polarity of the voltage applied to the motor. In other words if the motor is running in reverse and the REV command is sent it will start running forward. See also the new command SETDC_DIR where the motor speed and direction are combined into a single command.

Example of REV Command

Send	Sync	Command
Decimal	85	1
Hex	0x55	0x01

4.6 TACH Command

The MMBe monitors pulses at the TACH_IN pin on an ongoing basis. This measurement is accessed with the TACH command. The optional GATETIME value may be appended to the command, or left off. Tachometer measurements are updated every 125ms, and have a resolution of 8Hz. While you may read the tachometer value more often than every 125ms, the actual value is not updated any faster than that. Frequency measurements are factory calibrated to be within +/-0.5%(50Hz) or 10KHz, additional calibration could be attempted using the READ_FREQCAL and WRITE_FREQCAL commands. After the TACH command is received the frequency of the signal at the TACH_IN pin is returned as 2 bytes, with the high byte sent first. To actual frequency is FREQHI x 256 + FREQLO.

Example of TACH Command with frequency of 10,000Hz (0x2710)

Send	Sync	Command	GateTime (optional)
Decimal	85	2	64
Hex	0x55	0x02	0x40

Receive	FreqHi	FreqLo
Decimal	39	16
Hex	0x27	0x10

4.7 SETDC Command

SETDC is used to set the positive duty-cycle of the motor. This controls the motor speed. Duty-cycle is expressed as a percentage of time ON (positive duty-cycle). The SPEED value is an 8-bit value, with 255 (0xFF) equaling 100% ON (full speed), and 0 (0x00) equaling 0% ON (not running). The duty-cycle also relates to the average voltage across your motor. For example, in a 12V system a 50% duty-cycle would equal and average voltage across the motor of 6V (50% at 12V + 50% at 0V). To send a 25% duty-cycle you would send 25% of 255 (0xFF), or 64 (0x40).

When using the higher frequency mode (15.5KHz) you should expect less resolution in low speed control. The turn on/off time for the H-bridge on the MMBe is too slow and swamps out the lower duty cycle values. Therefore the H-bridge will not turn on unless the SPEED value is greater than 40 (15%).

Example of SETDC Command with speed of 75% (191, or 0xBF)

Send	Sync	Command	Speed
Decimal	85	3	191
Hex	0x55	0x03	0xBF

4.8 SPDCON Command

SPDCON command consists of the sync byte, command byte, and the desired frequency (0 to 65,535) sent as two bytes, after the desired frequency a GATETIME value may be appended to the command (but the GATETIME is not used by the MMBe). This command compares the desired frequency with the actual frequency of the signal at the TACH_IN pin. It then adjusts the motor speed based on the difference between the desired and actual frequencies (frequency error). The MMBe applies motor speed changes in an effort to maintain the desired frequency. There are two modes of operation. The user may also change the operating mode by sending the SPDCON_MODE command.

Increment/Decrement Mode: This is the default mode, and is compatible with earlier versions of the MMB. In this mode the motor speed is incremented or decremented based on the frequency error. This mode is slow to respond to changes in the motor load, but also unlikely to overcompensate for small load changes. Motor SPEED adjustments occur 8 times per second.

PI Mode: More detailed descriptions of the Proportional Integral (PI) Filter occur earlier in this datasheet. But this mode of speed control will provide a much more responsive speed control. The PTERM, ITERM, and PISCALAR, associated with PI operation may be modified with the WRITE_PI command. It is recommended that the 15.5KHz PWM setting be used with PI mode.

The frequency at the TACH_IN pin may be read with the SPDCON_FREQ command. SPDCON mode is exited by sending any command other than SPDCON or SPDCON_FREQ.

Example of SPDCON Command with frequency of 10,000Hz (0x2710)

Send	Sync	Command	FreqHi	FreqLo	GateTime (optional)
Decimal	85	4	39	16	64
Hex	0x55	0x04	0x27	0x10	0x40

4.9 STATUS Command

The STATUS command is used to read the STATUS byte and SPEED byte from the MMBe. Reading the STATUS byte allows you to determine if fault conditions have occurred, find out which direction the motor is turning, and a number of other things. Reading the SPEED register can be useful upon exiting SPDCON mode to see what the current drive state of the motor is.

Example of STATUS Command with FAULTON set and SPEED = 128(0x80)

Send	Sync	Command
Decimal	85	5
Hex	0x55	0x05

Receive	Status	Speed
Decimal	2	128
Hex	0x02	0x80

The contents of the STATUS register are broken down into individual bits that may be read. All bits in the STATUS register are factory programmed to 0. A bit that is set is equal to “1”, cleared is equal to “0”. The CHANGE_FREQ, CHANGE_BAUD, and SPDCON_MODE commands modify bits in the STATUS register. When these commands are used the new contents of the STATUS register are written to EEPROM and the new settings are loaded on future power-ups. For example, if you use the CHANGE_FREQ command to set the STATUS.FREQLOW bit, you do not need use it each time the device powers up. Sending the command a second time would clear the STATUS.FREQLOW bit.

Contents of STATUS register

Bit	Name	Description
0	MOTDIR	Set when motor is reversed (MOTOR+ = GND; MOTOR- = VMOTOR when on), cleared when motor is running forward (MOTOR+ = VMOTOR; MOTOR- = GND when on). Can be used to determine current direction of motor rotation.
1	FAULTON	Set after an under-voltage (VMOTOR < ~5.6V), over-temperature (H-bridge IC exceed 150°C -175°C, or over-current (6A-10A) condition occurs. Will not be cleared until read by STATUS command or end of COUNT command. When the fault happens the red fault indicator LED will light. If the condition is not persistent the LED will turn off but this bit will remain set until read.
2	BRAKEON	Set when the external _BRAKE pin is asserted (0V), or clear when it is not (5V or unconnected).
3	BAUD9600	Set when operating at 9600BPS, cleared when operating at 2400BPS. Sending CHANGE_BAUD command will toggle the state of this bit.
4	FREQLOW	Set when PWM motor drive signal is operating at 242Hz, cleared when operating at 15.5KHz. Sending CHANGE_FREQ command will toggle the state of this bit.
5	PIMODE	Set when operating SPDCON with a proportional integral (PI) filter, cleared when using the increment/decrement mode of SPDCON. The SPDCON_MODE command will toggle the state of this bit.
6	Unused	
7	Unused	

4.10 COUNT Command

The COUNT command is used to run the MMBe at a desired SPEED until it counts the specified number of pulses defined in the command string. It will then stop the motor and implement a STATUS response (it will send the same data to the master unit as a STATUS command would). The COUNT command may be used as part of a rudimentary position control system. Keep in mind that motor inertia will probably drive the motor past the desired number of counts. The MMBe will not attempt to back the motor up when this occurs. A master unit may implement a READ_COUNTER command that will return the exact number of pulses read by the MMBe, and use this information to make position corrections.

The command below instructs the MMBe to move at 50% speed until 10,000 pulses are counted and then stop. The direction of movement is determined by the last commanded direction change.

Example of COUNT to 10,000 with SPEED = 128 (0x80)

Send	Sync	Command	CountHi	CountLo	Speed
Decimal	85	6	39	16	128
Hex	0x55	0x06	0x27	0x10	0x80

Response when COUNT is completed

Receive	Status	Speed
Decimal	2	128
Hex	0x02	0x80

If for some reason the motor becomes stalled during a COUNT command the master unit will need to send the EXIT_COUNT command. This command is the only way to exit a count while it is in progress. Sending the COUNT command stops the tachometer from measuring frequencies. Sending a command other than COUNT or READ_COUNTER will restart the tachometer frequency measurements.

4.11 EXIT_COUNT Command

This command will cancel a COUNT if it is in progress. This might be used in conjunction with a timer to exit a COUNT command in the event of a motor being stalled, and the requested number of counts is not able to occur.

EXIT_COUNT Command

Send	Sync	Command
Decimal	85	7
Hex	0x55	0x07

4.12 SETDC_DIR Command

The SETDC_DIR command is used to send both the desired speed and the desired motor direction in a single command. Sending a negative 2's compliment number as the SPEED will cause the motor to run in reverse, a positive number runs the motor forward. See the previous section in this datasheet on 2's compliment numbers.

Essentially, the highest bit is used as a sign bit in 2's compliment number therefore the motor speed is carried in the lower seven bits of the SPEED byte. This means that you're sending a value of +/-127 as the SPEED. This value is adjusted within the MMBe to reflect values from +254 to – 254. So using the SETDC_DIR command reduces the resolution you have in duty-cycle by half. Still, most applications won't need the full 256 steps of speed control in both directions, and the benefit of not having to track the STATUS.MOTDIR bit to determine current motor direction is highly useful.

SETDC_DIR Command moving in reverse at 50% SPEED

Send	Sync	Command	Speed
Decimal	85	8	-64
Hex	0x55	0x08	0xC0

4.13 WRITE_EEPROM Command

The MMBe has EEPROM incorporated into the microcontroller. During design we didn't need it all, so we built this command for you to use it. Mighty gracious of us! EEPROM is a form of non-volatile memory. The data you store in EEPROM will remain intact after power is removed then restored. EEPROM has a useable life of about 1,000,000 write-cycles. For that reason you shouldn't write to EEPROM too often (once a second is too often, once every 5 minutes is not).

This command allows you to write to EEPROM address values from 0 to 99, and store 1 byte of data at that address.

Example of WRITE_EEPROM value of 255 to address 10

Send	Sync	Command	Address	Data
Decimal	85	9	10	255
Hex	0x55	0x09	0x0A	0xFF

4.14 READ_EEPROM Command

Since we built in the WRITE_EEPROM command it only made sense to allow you to read it. Provide the MMBe with the address (0 to 99) of the EEPROM you want to read from and the data at that address will be returned.

READ_EEPROM data at address 10, data is 255

Send	Sync	Command	Address
Decimal	85	10	10
Hex	0x55	0x0A	0x0A

Receive	Data
Decimal	255

Hex	0xFF

4.15 READ_COUNTER Command

This command allows you to read the raw counter value used with the COUNT command. It may be useful in determining the exact number of pulses received at the TACH_IN pin during a COUNT. Keep in mind that the COUNT command will stop driving a motor after it has received a set number of pulses, but the motor's inertia will continue to spin it for a short period. This overshoot can be read using the READ_COUNTER command.

Example READ_COUNTER returning a count of 10,000

Send	Sync	Command
Decimal	85	11
Hex	0x55	0x0B

Receive	CountHi	CountLO
Decimal	39	16
Hex	0x27	0x10

4.16 CHANGE_BAUD Command

The MMBe is factory programmed for 2.4KBPS serial communication. But you can increase the rate of communication to 9.6KBPS. The CHANGE_BAUD command toggles between these two settings, and stores the most recent setting in EEPROM. You only need to make this change one time. Whatever baud rate you set the device for will become the default baud rate. The change in baud rate takes place upon reception of this command.

CHANGE_BAUD Command

Send	Sync	Command
Decimal	85	12

4.17 CHANGE_FREQ Command

The old MMB had a software generated PWM frequency of 61Hz. Yep, we know this is slow, but it was designed for driving hobby robot platforms, and 61Hz noise was much more palatable than the whine of a 1KHz PWM signal. Since increasing the PWM frequency changes the efficiency of the motor drive circuit, we tried hard to provide a PWM frequency that was similarly slow as the default frequency for the MMBe. This way customers switching from the older products would be less likely to see noticeable changes in motor speed between the two versions of the product.

But, we also recognized that backwards compatibility should not prevent us from using a higher PWM frequency above the audible level. So the default PWM frequency is 15.5KHz, but the frequency may be changed to a lower 242Hz with the CHANGE_FREQ command. This command toggles between the two available PWM frequencies and stores the new setting in EEPROM (as the power up default setting). You can read the STATUS.FREQLOW bit to see which frequency you are using.

Oddly enough the H-bridge used on the MMBe is designed to operate in the audible range (1-2KHz). So operating at 15.5Khz will provide less resolution at the lower end of duty cycles. The H-bridge will not turn on unless the duty cycle is greater than 40 (15%) due to the turn-on/off times of the H-bridge. For a lot of motors this is not a serious issue. But if you need that low-end resolution you can always return to the "old-school" frequency (242Hz).

CHANGE_FREQ Command

Send	Sync	Command
Decimal	85	13
Hex	0x55	0x0D

4.18 READ_FIRMWARE Command

The controller portion of the MMBe is electrically reprogrammable. If Solutions Cubed, LLC becomes aware of bugs in the operating system, and can fix the bugs, we will. Any upgrades to the MMBe are tracked through the firmware revision. When an upgrade occurs it is added to the "errata" sheet (error – data sheet) along with the problem fixed. If problems are known, but cannot be fixed they are also added to the errata sheet. If no errata sheet exists then no problems exist that we are aware of.

Example READ_FIRMWARE command

Send	Sync	Command
Decimal	85	14
Hex	0x55	0x0E

Receive	Revision
Decimal	1
Hex	0x01

4.19 READ_PI Command

The MMBe now has two modes of SPDCON. The default mode increments or decrements the motor SPEED to make the actual frequency equal to the desired frequency. The new PI mode runs the error in motor speed through a PI filter. When tuned correctly the PI filter should be more responsive to motor load changes. This command reads the user adjustable PI variables from the MMBe.

Example of READ_PI Command

Send	Sync	Command
Decimal	85	15
Hex	0x55	0x0F

Receive	PTerm	ITerm	PIScalar
Decimal	64	1	14
Hex	0x40	0x01	0x0E

4.20 WRITE_PI Command

The MMBe now has two modes of SPDCON. The default mode increments or decrements the motor SPEED to make the actual frequency equal to the desired frequency. The new PI mode runs the error in motor speed through a PI filter. When tuned correctly the PI filter should be more responsive to motor load changes. This command writes the user adjustable PI variables to the MMBe. This data is **NOT** stored in EEPROM and should be re-written at each power up.

Example of READ_PI Command

Send	Sync	Command	PTerm	ITerm	PIScalar
Decimal	85	16	128	1	14
Hex	0x55	0x10	0x80	0x01	0x0E

4.21 SPDCON_MODE Command

This command toggles between the two modes of SPDCON. The STATUS.PIMODE bit may be tested to determine which mode you're currently in. When this command is received the new setting is stored in EEPROM and becomes the default mode. It is recommended that the 15.5Khz PWM setting be used with PI mode.

SPDCON_MODE Command

Send	Sync	Command
Decimal	85	17
Hex	0x55	0x11

4.22 SPDCON_FREQ Command

When in SPDCON mode you can read the current frequency measurement with the SPDCON_FREQ command. This command differs from the TACH command in two ways. First, implementing this command does not cause SPDCON to be exited. Second, response to this command sends back the last valid tachometer measurement (which are ongoing in SPDCON mode), and so the information is returned immediately.

Example READ_FREQ returning a frequency of 10,000

Send	Sync	Command
Decimal	85	18
Hex	0x55	0x12

Receive	FreqHi	FreqLo
Decimal	39	16
Hex	0x27	0x10

4.23 READ_FREQCAL Command

The frequency measurements used by the TACH, SPDCON, and SPDCON_FREQ command are based on an internal oscillator in the MMBe. This oscillator's tolerance allows for some variability in the accuracy and timing of frequency measurements. The READ_FREQCAL command reads the current frequency calibration setting in the MMBe. The frequency measurements are calibrated to within 0.5%(50Hz) of a 10KHz test signal at the factory.

Example READ_FREQCAL returning a value of 250

Send	Sync	Command
Decimal	85	19
Hex	0x55	0x13

Receive	FreqCal
Decimal	250
Hex	0xFA

4.24 WRITE_FREQCAL Command

The frequency measurements used by the TACH, SPDCON, and SPDCON_FREQ command are based on an internal oscillator in the MMBe. This oscillator's tolerance allows for some variability in the accuracy and timing of frequency measurements. The WRITE_FREQCAL command is used to change the frequency calibration setting in the MMBe, and store the value in EEPROM. The frequency measurements are calibrated to within 0.5%(50Hz) of a 10KHz test signal at the factory. Reducing the FREQCAL value will reduce the frequency read by the tachometer. Increasing the value increases the frequency read by the tachometer.

Example WRITE_FREQCAL writing a value of 250

Send	Sync	Command	FreqCal
Decimal	85	20	250
Hex	0x55	0x14	0xFA

5.0 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 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.