User Guide

MSM-Series Motors

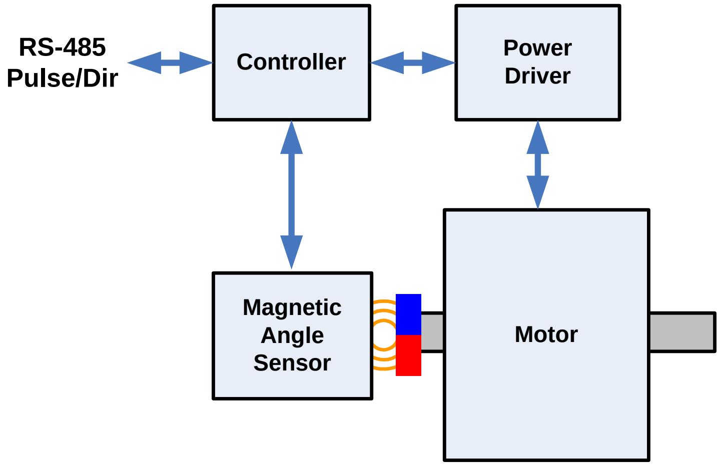
MMP-Series Motor Controllers

Table of Contents

Overview

Introduction

The MSM-series motors are fully integrated solutions for servo motor applications. They integrate a position sensor, controller, and driver inside the motor housing to provide a complete "smart motor" solution.



The motors may be operated in speed, position, or torque control modes. The motors are controlled through either an RS-485 serial interface, or with simple PULSE/DIR signals. Programmable parameters can be set with an easy-to-use PC-based program, which interfaces to the motor through USB and the RS-485 interface. Once parameters have been optimized, they can be saved in non-volatile memory in the motor.

In addition to complete motor assemblies, a controller/driver module is also available. The MMP series offers a control and driver solution that can be integrated with other motors.

The MMP module includes a magnetic angle sensor for rotor feedback, a field-oriented-control (FOC) motor controller, and a motor driver.

Evaluation Kit Contents

The MSM-series motors and MMP-series controllers are provided with an interface to be able to program and control the motor from a PC computer. The kits contain the following:

- MSM-series motor, or MMP-series controller
- Mating connectors
- USB to RS-485 interface box
- USB cable

In addition, the GUI PC software and USB driver software need to be downloaded from the MPS website.

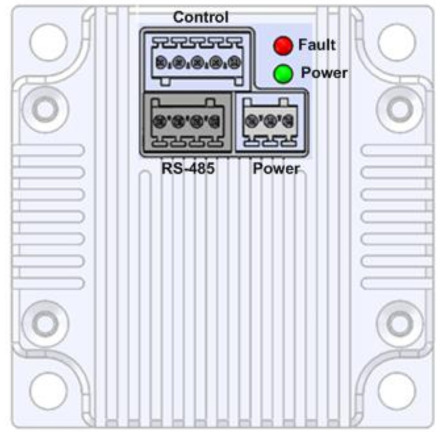
Safety Warnings

To prevent personal injury or damage to the motor or other equipment, follow these precautions:

- *Always secure the motor before applying power. The motor may move unexpectedly, and can jump or fall when it starts.*
- *Keep hair and loose clothing away from the motor.*
- *Keep away from the shaft and any attached mechanical parts when operating the motor.*
- *Do not open or disassemble the motor.*
- *When installed in a system, the motor enclosure should be bonded to a protective ground.*
- *The power source connected to the motor should be fused or otherwise current-limited.*

Section 1. Hardware

1.1 Connections



Power and control connections are made to the rear surface of the motor. LEDs indicate the presence of power, and that a fault condition has occurred.

Mating connectors are listed below:

Connector	Pins	Manufacturer	Part Number
Control	5	Wurth	691382000005
RS-485	4	Wurth	691382000004
Power	3	Wurth	691382000003

1.2 Power

DC power is connected to the motor through the 3-pin connector on the rear of the motor housing. Apply a DC voltage within the range specified on the datasheet for the particular model used. The power supply should be fused or current-limited, and be capable of supplying current up to the desired current limit of the motor.

The R- pin may be connected to an external power resistor and to VIN. This resistor can be used to limit the bus voltage by dissipating energy when energy is returned from the motor. This can happen during regenerative braking or other conditions.

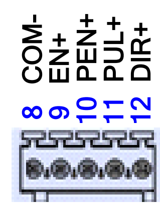
1.3 Control Interface

The motor may be controlled either using high-level commands through an RS-485 interface, or by using a simple pulse and direction interface.

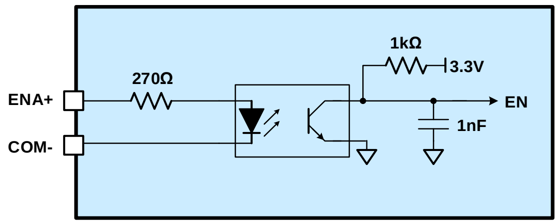
1.3.1 Pulse Control Interface

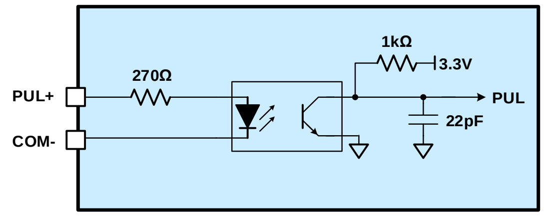
The pulse control interface is optically isolated. The interface uses the following signals:

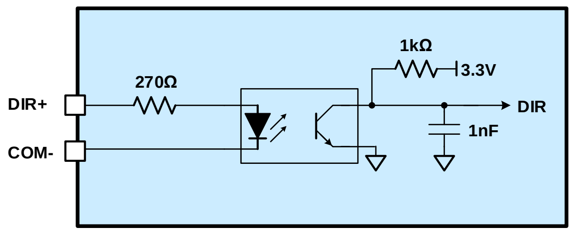
Signal	Direction	Pin	Description
ENA+	To motor	9	When driven high, enables the motor.
DIR+	To motor	12	When driven high, sets CW rotation direction.
PUL+	To motor	11	Pulse or PWM input to control position or speed.
PEND+	From motor	10	Driven active when position is reached.

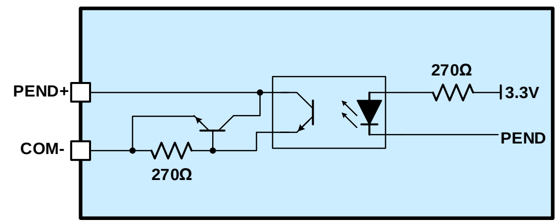


The internal circuitry connected to these signals is shown below:









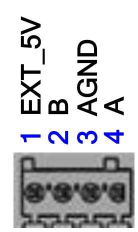
In pulse control mode, the motor moves a pre-programmed amount each time a pulse is inputted. The rotation direction is set by the state of the DIR input signal, and motion is enabled or disabled by the ENA signal.

When the motion is complete, the PEND output signal goes active to indicate that the motor has completed the motion.

1.3.2 RS-485 Control Interface

The RS-485 serial interface uses the following signals:

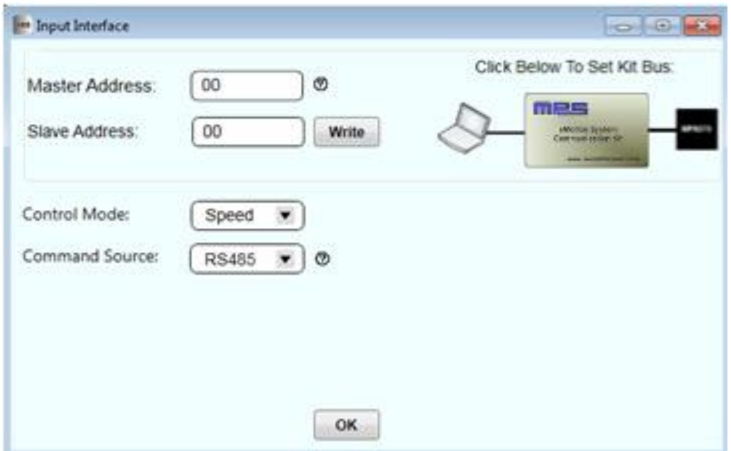
Signal	Direction	Pin	Description
EXT_5V	To motor	1	External 5V power for firmware programming
AGND	N/A	3	Ground for EXT_5V
A	Bidirectional	4	RS-485 data A (non-inverting)
B	Bidirectional	2	RS-485 data B (inverting)



The RS-485 has an asynchronous serial interface, and uses standard RS-485 transceivers. The baud rate is set to 115200bps by default, but can be changed by writing to the BAUDRATE register, or by using the eMotion System^{TM} Virtual Bench program.

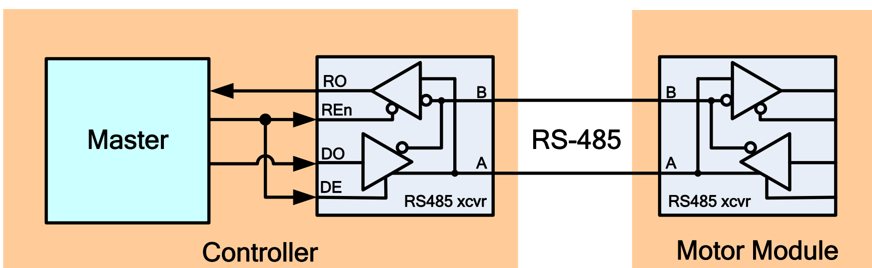
The module supports multi-slaver communication. To change the slaver address, enter the slave address and click write. Address 0x00 is the broadcast address; all slavers will respond when the master using 0x00 as the slave address.

The master address is the address that the communication kit (or other master) is using. The master address and slave address should be same to achieve successful communication.

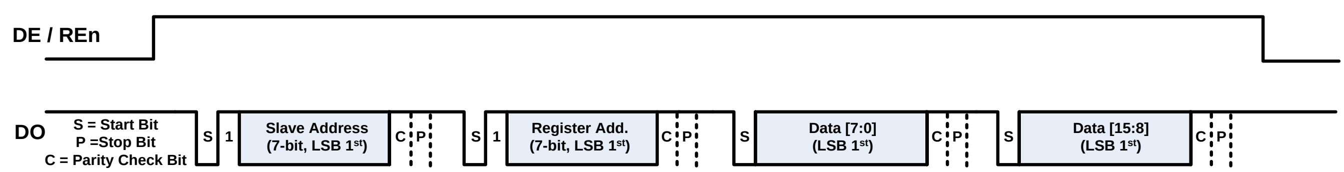


To communicate with the module without applying the main VIN power (e.g. to program the nonvolatile memory), supply 5V DC to the EXT-5V pin. In normal operation, this pin is not connected.

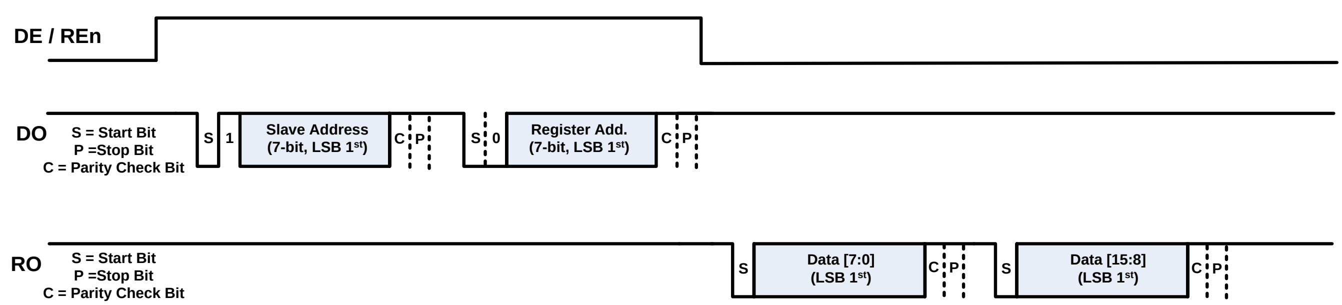
The connection between a master controller and the MSM-series motor (or MMP-series controller) is shown below:



To write data to the motor, a data packet is sent that contains a slave address, register address, and data. The 1 sent at the beginning of the register address byte indicates a write operation. Use odd parity check.



To read data from the motor, the slave address and register address are sent, and then the motor returns the data. The 0 sent at the beginning of the register address byte indicates a read operation.



1.4 Mechanical Mounting

The MSM-series motors are mechanically mounted from a front flange, like other NEMA frame motors. See the datasheets for each motor for mounting dimensions.

MMP-series controllers are designed to mount to the back of standard NEMA motors. See the datasheets for each controller for details.

Section 2. Operational Modes

2.1 Position Control Mode

2.1.1 RS-485 Control Mode

Using RS-485 control, the position mode is programmable in absolute command mode or incremental command mode. The mode is set through the POS_CMD_TYPE register. Absolute mode commands the motor to an absolute position; incremental commands move the motor relative to its present position.

In absolute command mode, the maximum command limit is ±32,768 mechanical revolutions. In incremental command mode, the minimum command resolution should be greater than 0.2° due to the resolution limitations of the internal position sensor.

The command is set through the POS_CMD register (0x4A & 0x4B), as shown below:

Forward:

POS_CMD[31:0] = (Revs + Theta / 360) * 2^{16}

0x4A[15:0] = POS_CMD[31:16]

0x4B[15:0] = POS_CMD[15:0]

Reverse:

POS_ CMD[31:0] = 2^{32} – (Revs + Theta / 360) * 2^{16}

0x4A[15:0] = POS_CMD[31:16]

0x4B[15:0] = POS_CMD[15:0]

Revs: Target position command full revolutions.

Theta: Target position command angle.

Below are examples to set the position to 2 revolutions plus 45°:

Forward:

0x4A[15:0] = 2 = 0002h

0x4B[15:0] = 45/360 * 2^{16} = 2000h

Reverse:

0x4A [15:0] = 2^{16} – 2 = FFFDh

0x4B[15:0] = 2^{16} – 45/360 * 2^{16} = E000h

The maximum speed limit to approach position command is programmable through the SPEED_LIMIT register. The default value is 3000rpm. Write 0x0000 to register 0x76 to update the position command.

2.1.2 PUL/DIR Control Mode

In PUL/DIR command control mode, the position works in incremental mode. Each rising edge on the PUL input will move the motor by a programmable increment. The number of pulses per revolution is programmable to be 512, 1024, 2048, or 4096 pulses per mechanical revolution. This is set through the NSTEP register. The default value is 4096.

To smoothly ramp the speed of the motor, the velocity up/down slope is programmable through the POS_CMD_SLOPE register.

The loop parameters can be optimized through the KP_POS and KP_GAIN_POS registers, according to the system requirements with the real mechanical load.

The maximum torque limit is programmable with the MAX_LIMIT_IQ register.

2.2 Speed Control Mode

2.2.1 RS-485 Control Mode

In speed control mode, the speed command (in rpm) is set through the RS-485 interface. The command is set through the SPD_CMD register (0x4D & 0x4E), as shown below:

Forward:

SPD_ CMD[31:0] = (Speed / 60) * 2^{32} / 10,000
0x4D = SPD_ CMD[31:16]
0x4E = SPD_ CMD[15:0]

Reverse:

SPD_ CMD[31:0] = 2^{32} – (Speed / 60) * 2^{32} / 10,000
0x4D = SPD_ CMD[31:16]
0x4E = SPD_ CMD[15:0]

Speed: Target speed in rpm.

Below is an example to set the speed to 3,000rpm:

Forward:

SPD_ CMD[31:0] = (3,000 / 60) * 2^{32} / 10,000 = 0147AE14h
0x4D = SPD_ CMD[31:16] = 0147h
0x4E = SPD_ CMD[15:0] = AE14h

Reverse:

SPD_ CMD[31:0] = 2^{32} – ((3,000 / 60) * 2^{32} / 10,000) = FFFCB965h
0x4D = SPD_ CMD[31:16] = FFFCh
0x4E = SPD_ CMD[15:0] = B965h

In speed mode, the velocity ramp-up/ramp-down slope is programmable with the SPD_CMD_SLOPE register.

The loop parameters can be optimized through registers KP_SPD, KP_GAIN_SPD, KI_SPD, KI_GAIN_SPD, and KC_SPD, based on the real mechanical load.

The maximum torque limit is programmable with the MAX_LIMIT_IQ register. See details in the Torque Control Mode section on page 11. Write 0x0000 to register 0x76 to update the speed command.

2.2.2 PWM/DIR Control Mode

In PWM/DIR command control mode, the motor speed is controlled by the duty cycle of the PWM input. The real motor speed is SPEED_CMD x duty_cycle, and the SPEED_CMD is programmable (register 0x4D, register 0x4E). The PWM signal frequency should be between 100Hz and 10kHz.

The DIR pin can control the direction. When DIR is at a high level, the motor moves forward.

2.3 Torque Control Mode

2.3.1 RS-485 Control Mode

In torque control mode, the torque command (which corresponds to phase current) is set directly through the RS-485 interface. The command is set through the IQ_CMD register, as shown below:

If iq is positive:

$$IQ\_CMD[11:0] = iq * 1.5 * 0.01 * KAD * 1024 / 1.6$$

If iq is negative:

$$IQ\_CMD[11:0] = 2^{12} – (iq * 1.5 * 0.01 * KAD * 1024 / 1.6)$$

iq: Torque current in A.

KAD: Amplifier gain coefficient of current sensing. Default value is 2.

Below is an example to set 5A torque current:

When iq is positive:

$$IQ\_CMD[11:0] = 5 * 1.5 * 0.01 * 2 * 1024 / 1.6 = 60h$$

When iq is negative:

$$IQ\_CMD[11:0] = 2^{12} – (5 * 1.5 * 0.01 * 2 * 1024 / 1.6) = FA0h$$

iq: Torque current in A.

2.3.2 PWM/DIR Mode

In PWM/DIR command control mode, the motor torque is controlled by the duty cycle of the PWM input. The real motor torque is IQ_REF x duty_cycle, and the IQ_REF is programmable (register 0x2F). The PWM signal frequency should be between 100Hz and 10kHz. The DIR pin can control the torque direction. When DIR is at a high level, the torque is positive, which means the motor tries to go forward.

In torque mode, the maximum speed clamp is programmable through the T_MAX_SPD and ANTI_TORQUE_GAIN registers. The loop parameters can be optimized through the CURRENT_KP and CURRENT_KI registers.

2.4 Fault Indication

The control module has robust protection to avoid unexpected failure modes and external component damage. The fault type can be determined either from the register value 0x53 or from the red LED flash times.

Fault Type	Register 0x53 Flag Bit	LED Flash Times
OCP	Bit 2	3
Rotor lock	Bit 3	4
Overload	Bit 4	5
Over-temperature	Bit 5	6

Section 3. PC-Based Software Control

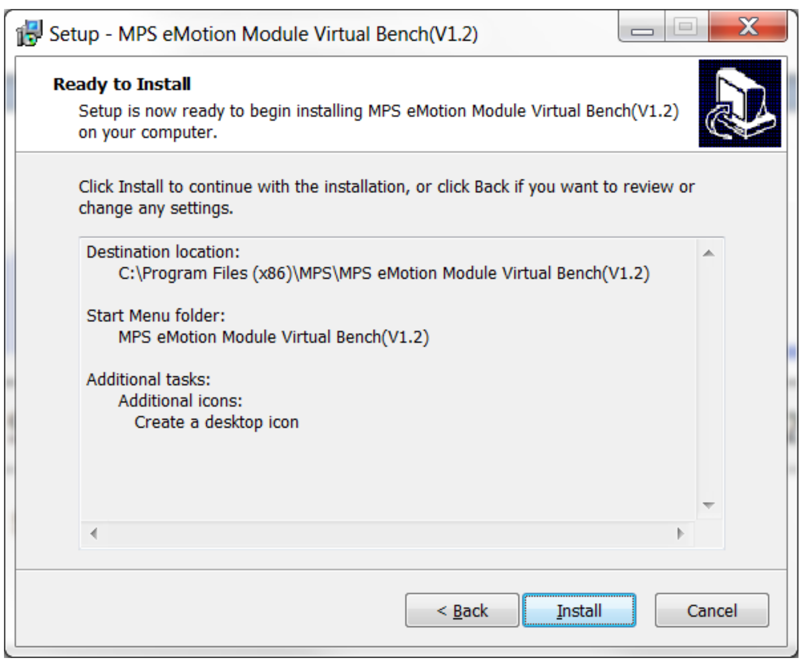
3.1 Quick Start Guide

The MSM-series motors are pre-programmed with initial parameters, so the motor can simply be connected to power and to a PC (via the USB-RS-485 interface), which will spin the motor. Before connecting the motor, install the software and connect as described below.

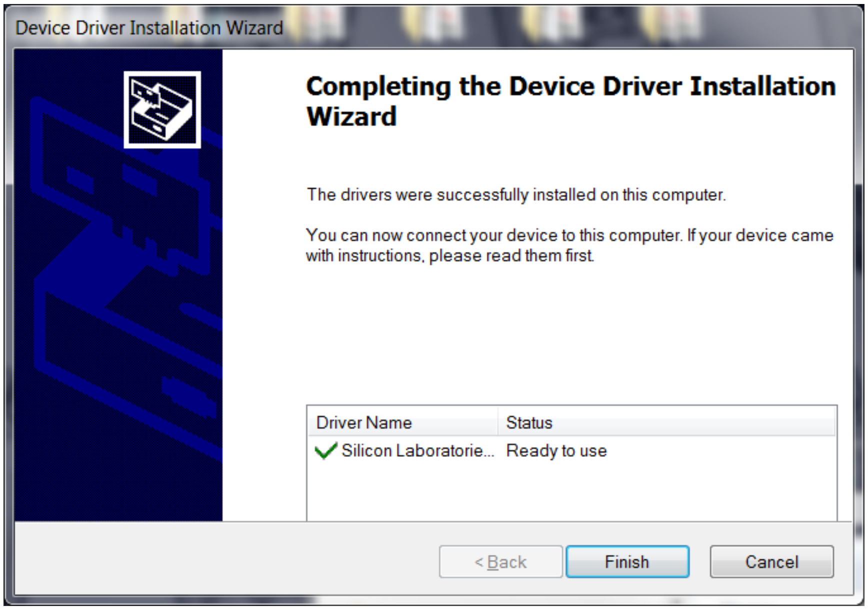
3.2 Software Installation Procedure

To use a PC to program and run the motor, first download and install the driver and GUI software.

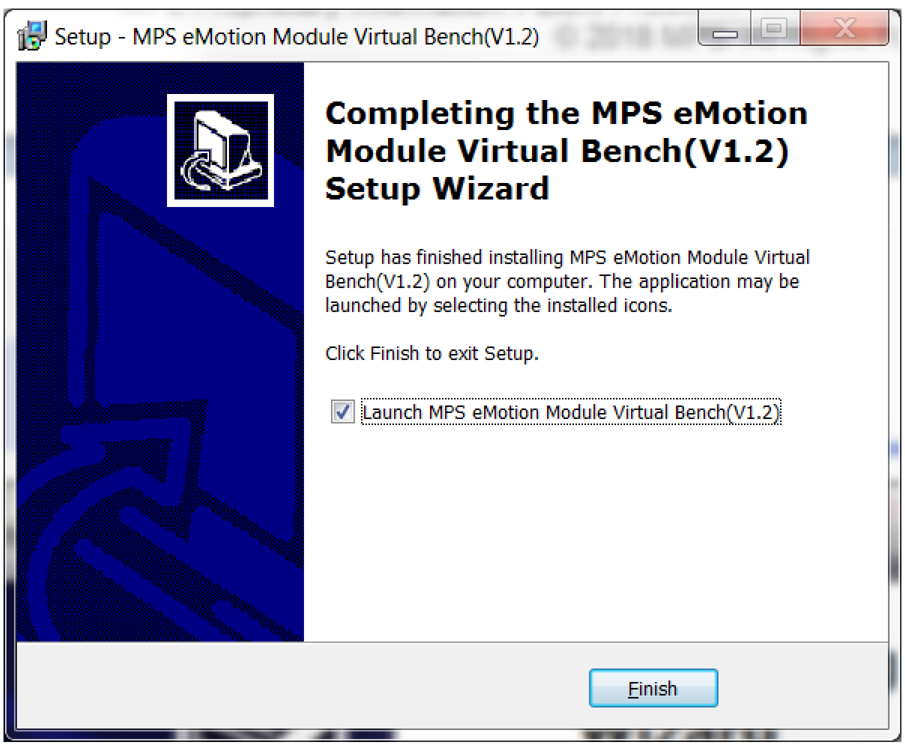
1. Download the GUI and driver installer from www.monolithicpower.com
2. Navigate to the folder containing the installer.
3. Double-click on the installer to launch it.
4. Follow the prompts to:
    - Select the language
    - Accept the license agreement
    - Specify the installation location (or use the default)
    - Specify the shortcut location (or use the default)
    - Choose to create desktop or quick launch icons (or not)
    - Select "Install"



- The device driver installation will then begin.

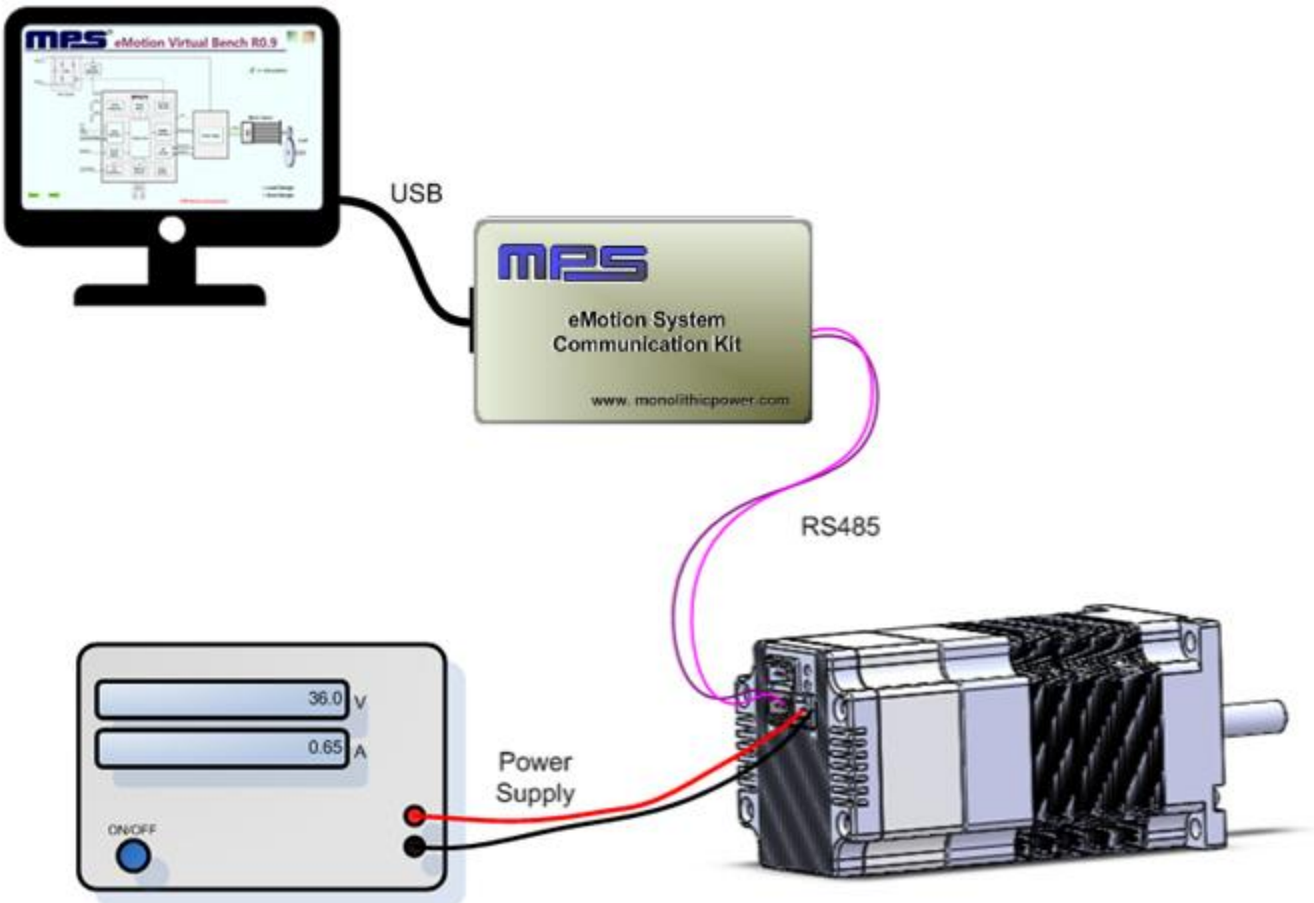


- Confirm installation is complete.

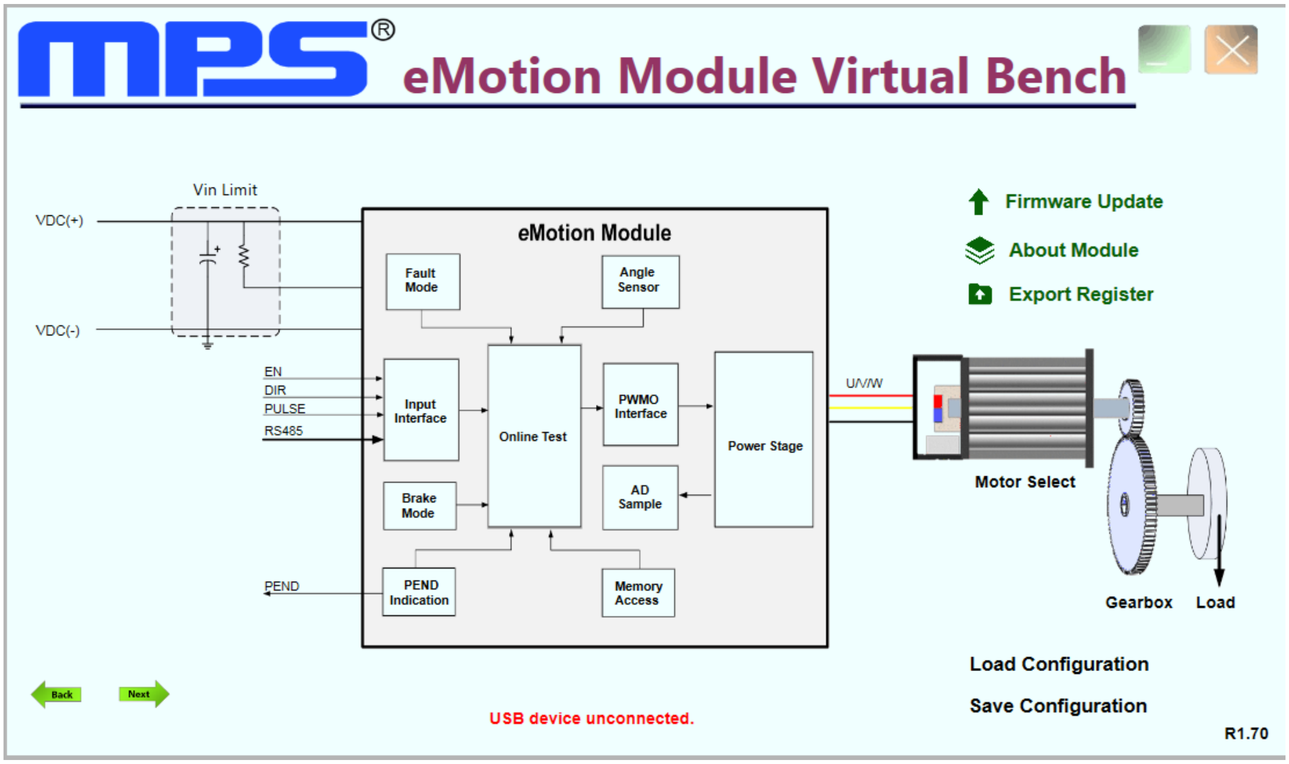


3.3 Connecting the Interface

To communicate with the motor, connect the MPS USB to RS-485 interface to the motor RS-485 connector and to the host PC via the supplied USB cable.



Section 4. eMotion System^{TM} Virtual Bench



The eMotion System^{TM} Virtual Bench program allows control and programming of the motor. Once programmed, the motor may be controlled via the RS-485 interface or using the PUL/DIR interface.

The following sections describe the different functions in the eMotion System^{TM} Virtual Bench.

4.1 Basic Operation

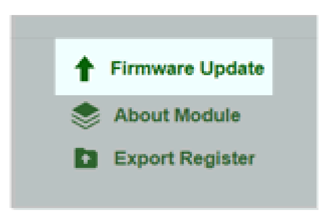
4.1.1 Communication

The eMotion System^{TM} Virtual Bench program communicates with the motor through a USB to RS-485 interface.

When properly connected, the program will report "**Communication success**" at the bottom of the window. If there is an issue with the USB cable, or if the USB driver has not been loaded, "**USB device unconnected**" will be displayed. If the USB communication is successful but there is no response from the motor (as would be the case if the motor was not powered), then "**Communication failed**" will be displayed.

4.1.2 Firmware Updates

The internal firmware may be updated using eMotion System^{TM} Virtual Bench. Click on "Firmware Update," and select the firmware update file.



4.1.3 Loading and Saving Configuration Files

Load Design

Save Design

The design parameters for a motor may be saved to a file and loaded into the eMotion System™ Virtual Bench. For MSM-series motors, a configuration file is provided that is optimized for that motor.

To load a configuration, click "Load Design" and select the file to be loaded. To save a design, click "Save Design" and enter a location and file name.

4.1.4 Aligning and Testing the Angle Sensor

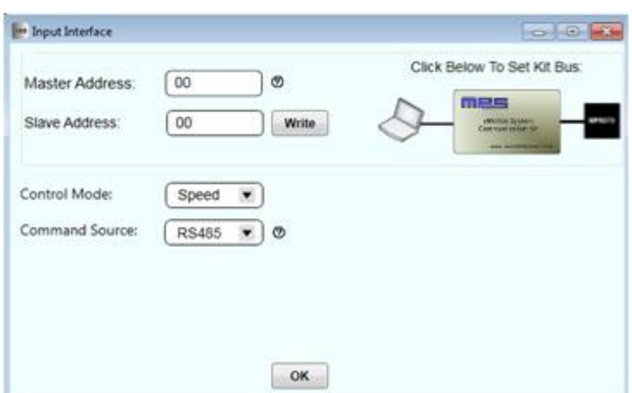
The orientation of the magnet on the motor shaft may be arbitrarily aligned with respect to the rotor magnets. An alignment step drives the rotor to a known position and establishes the orientation of the magnet.

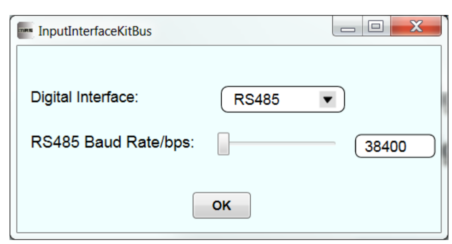
If the "Aligning Done" indicator is red, alignment needs to be performed. To perform alignment, click the "Start" button under "Aligning Test." The "Aligning Done" indicator should turn green.

To check sensor mounting, click the "Start" button under "Checking Sensor Mounting." After the check, the indicator should turn green.

4.1.5 Setting the Control Mode

Select the control mode (RS-485 or pulse) as well as position control or speed control modes by clicking the "Input Interface" button. Click "Set Kit Bus" to change the baud rate of the RS485 interface.



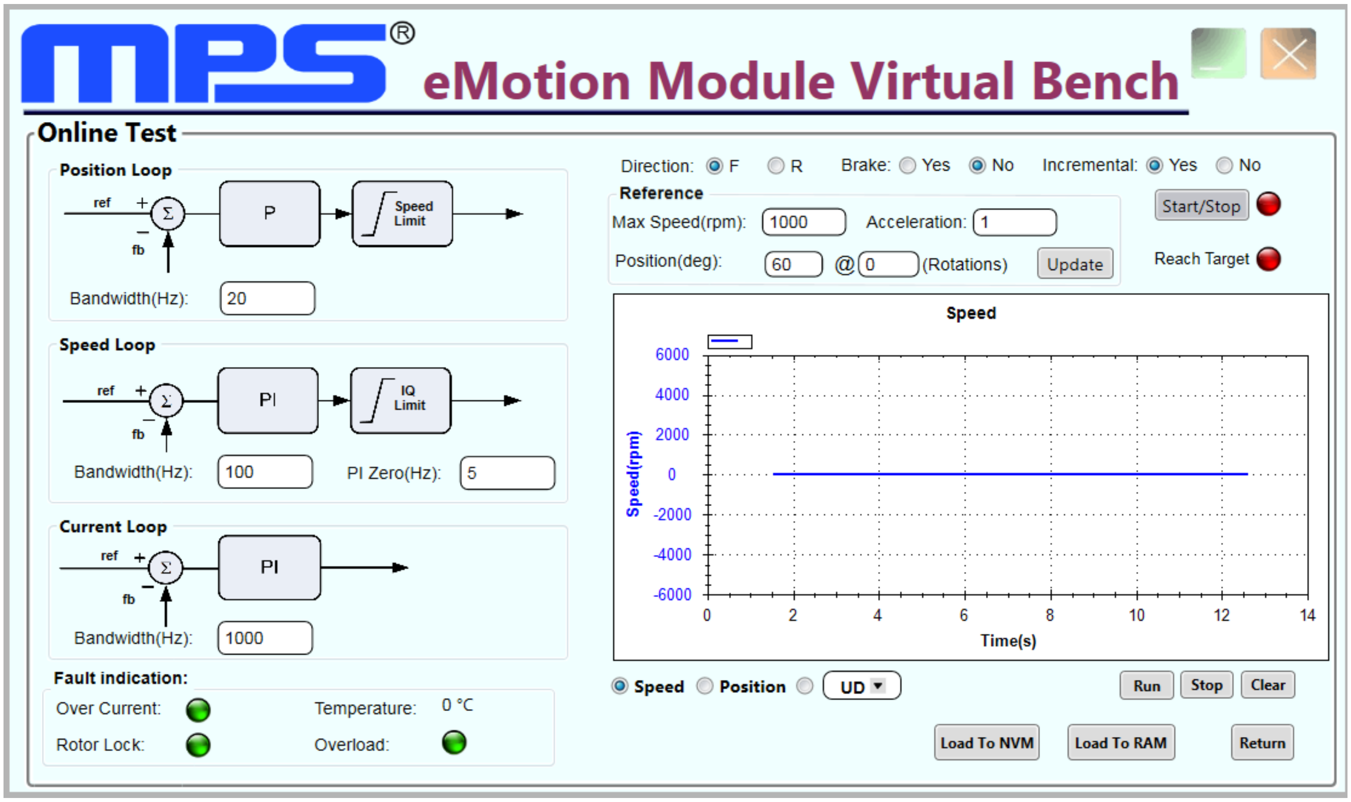


4.1.6 Running the Motor

Clicking the "Online Test" button opens up a new screen that allows the motor to be run and control loop parameters to be adjusted to optimize motor performance in application.

If a configuration file has been loaded, many of the parameters will already be populated correctly.

In position control mode, the screen looks like this:

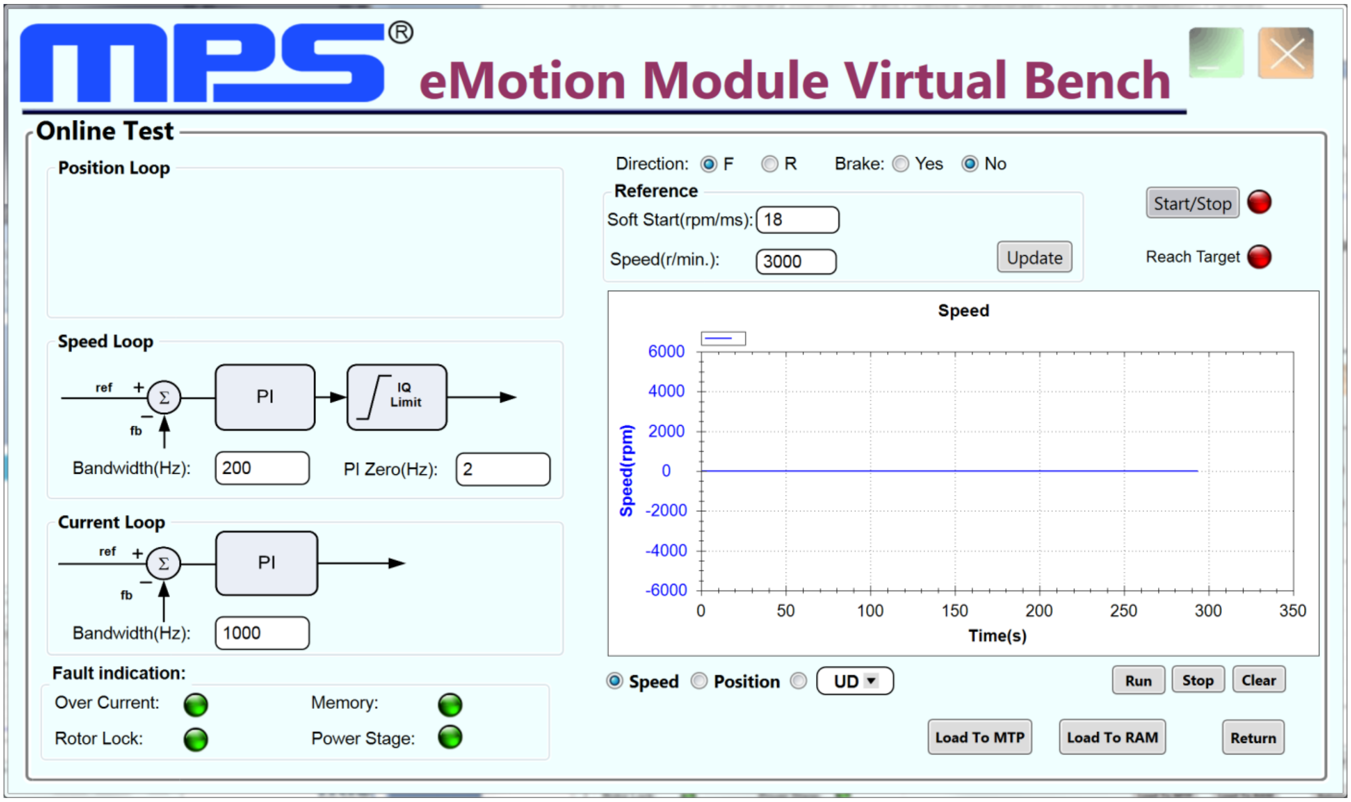


The "Reference" section allows selection of the rotation direction, incremental or absolute positioning mode, and the desired target position.

Loop parameters, including gain, bandwidth, and limits may be set for the three control loops (torque, speed, and position).

The motor speed and/or position are reported in the graph, which can be run, stopped, or cleared.

In speed control mode, the reference section is different:



Here, the target speed can be programmed (as well as the rotation direction).

To run the motor, follow the steps below:

1. Set the reference information (target speed or position).
2. Set the loop parameters. For MSM-series motors, these should have already been provided from the configuration file.
3. Click “Load to RAM” to transfer the data to the controller.
4. Click “Start/Stop” to run the motor.
5. If any changes are made to the parameters, click “Load to MTP” to save the new parameters into nonvolatile memory in the motor. (Or click “Save Design” to save to a file after exiting this screen).

Return to the main GUI screen by clicking the “Return” button.

Section 5. Register Map

<table>
<tr><th>Addr</th><th colspan="9">D[15:0]</th></tr>
<tr><td colspan="10">Position Command</td></tr>
<tr><td>4AH</td><td colspan="9">POS_CMD[31:16]</td></tr>
<tr><td>4BH</td><td colspan="9">POS_CMD [15:0]</td></tr>
<tr><td>4CH</td><td colspan="9">POS_CMD_SLOPE[15:0]</td></tr>
<tr><td>1DH</td><td colspan="6"></td><td colspan="3">ACCE[4:0]</td></tr>
<tr><td>50H</td><td colspan="9">NSTEP[15:0]</td></tr>
<tr><td colspan="10">Position Loop Settings</td></tr>
<tr><td>1AH</td><td colspan="9">KP_POS[15:0]</td></tr>
<tr><td>1BH</td><td colspan="9">KP_GAIN _POS[15:0]</td></tr>
<tr><td>1CH</td><td colspan="9">SPEED_LIMIT[15:0]</td></tr>
<tr><td colspan="10">Speed Command</td></tr>
<tr><td>4DH</td><td colspan="9">SPEED_CMD[15:0]</td></tr>
<tr><td>4EH</td><td colspan="9">SPEED_CMD[31:16]</td></tr>
<tr><td>4FH</td><td colspan="9">SPEED_CMD_SLOPE[15:0]</td></tr>
<tr><td colspan="10">Speed Loop Settings</td></tr>
<tr><td>22H</td><td colspan="9">KP_SPD[15:0]</td></tr>
<tr><td>23H</td><td colspan="9">KP_GAIN _SPD [15:0]</td></tr>
<tr><td>24H</td><td colspan="9">KI_SPD[15:0]</td></tr>
<tr><td>25H</td><td colspan="9">KI_GAIN_SPD[15:0]</td></tr>
<tr><td>26H</td><td colspan="9">KC_SPD[15:0]</td></tr>
<tr><td>27H</td><td colspan="9">MAX_LIMIT_IQ[11:0]</td></tr>
<tr><td colspan="10">Torque Command</td></tr>
<tr><td>2FH</td><td colspan="9">IQ_CMD[11:0]</td></tr>
<tr><td colspan="10">Torque Loop Settings</td></tr>
<tr><td>2DH</td><td colspan="9">T_MAX_SPD[15:0]</td></tr>
<tr><td>2EH</td><td colspan="9">ANTI_TORQUE_GAIN[15:0]</td></tr>
<tr><td>43H</td><td colspan="9">CURRENT_KP[15:0]</td></tr>
<tr><td>44H</td><td colspan="9">CURRENT_KI[15:0]</td></tr>
<tr><td colspan="10">Operating Mode</td></tr>
<tr><td>34H</td><td colspan="4">Reserved</td><td colspan="2">POS_CMD_TYPE</td><td colspan="2">CMD_MODE</td><td>MODE[1:0]</td></tr>
<tr><td colspan="10">Protection Parameters</td></tr>
<tr><td>53H</td><td></td><td>OTP[5]</td><td>OVER_LOAD[4]</td><td colspan="2">LOCK[3]</td><td>OCP[2]</td><td>PSFT[1]</td><td colspan="2"></td></tr>
<tr><td>56H</td><td colspan="6">VDC_LIMIT[15:1]</td><td colspan="3">VDC_LIMIT_EN[0]</td></tr>
<tr><td>59H</td><td colspan="9">AD_GAIN[15:0]</td></tr>
<tr><td>5AH</td><td colspan="9">IOCP[15:0]</td></tr>
<tr><td>58H</td><td colspan="9">DEAD_TIME[15:0]</td></tr>
<tr><td>6BH</td><td colspan="9">OVER_TEMPERATURE[15:0]</td></tr>
<tr><td colspan="10">Switching Frequency</td></tr>
<tr><td>57H</td><td colspan="9">FSW[15:0]</td></tr>
<tr><td colspan="10">RS485 Communication & Operating Command</td></tr>
<tr><td>5CH</td><td colspan="9">RS485_ADDR[15:0]</td></tr>
<tr><td>5DH</td><td colspan="9">RS485_BAUDRATE[15:0]</td></tr>
<tr><td>70H</td><td colspan="9">RUN[15:0]</td></tr>
<tr><td>71H</td><td colspan="9">BRAKE[15:0]</td></tr>
<tr><td>73H</td><td colspan="9">LOAD_TO_FLASH[15:0]</td></tr>
<tr><td>74H</td><td colspan="9">LOAD_FROM_FLASH[15:0]</td></tr>
<tr><td>76H</td><td colspan="9">UPDATE_COMMAND</td></tr>
</table>

Section 6. Register Detail

Position Command

0x4A: POS_CMD[31:16]			
Bit[15:0]	Name	Default	Description
[15:0]	POS_CMD	0000H	Position command high 16 bits. POS_CMD[31:0] = (Revolutions + Theta / 360) * 2^{16} 0x4A[15:0] = POS_CMD[31:16]
0x4B: POS_CMD[15:0]			
Bit[15:0]	**Name**	**Default**	**Description**
[15:0]	POS_CMD	0000H	Position command low 16 bits. POS_CMD[31:0] = (Revolutions + Theta / 360) * 2^{16} 0x4B[15:0] = POS_CMD[15:0]
0x4C: POS_CMD_SLOPE[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	POS_CMD_ SLOPE	0064H	Position command ramping slope, unit LSB/100μs, 2^{16} LSBs per revolution. *Example below is to set 100rpm ramping slope:* POS_CMD_SLOPE = 100(rpm) * 2^{16} / 60 / 10,000
0x1D: ACCE[4:0]			
Bit[15:0]	Name	Default	Description
[4:0]	ACCE	0x01	Acceleration rate in position mode. The real acceleration rate is ACCE * 5760rpm/s.
0x50: NSTEP[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	NSTEP	0000H	Stepping resolution per pulse input: NSTEP = 4: 4096 pulses per turn NSTEP = 5: 2048 pulses per turn NSTEP = 6: 1024 pulses per turn NSTEP = 7: 512 pulses per turn

Position Loop Settings

0x1A: KP_POS [15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KP_POS	00FAH	Position loop proportional gain.
0x1B: KP_GAIN _POS [15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KP_GAIN_PO S	8008H	Gain ratio of the position loop proportional gain.
0x1C: SPEED_LIMIT[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	SPEED_LIMIT	02FFH	Maximum speed in position mode, unit is LSB/50μs, 2^{16} LSBs per revolution. *Example below is to set 3,000rpm speed limit:* SPEED_LIMIT = 3000(rpm) * 2^{16} / 60 / 10,000

Speed Command

0x4D: SPEED_ CMD[31:16]			
Bit[15:0]	Name	Default	Description
[15:0]	SPEED_CMD	000AH	Speed command high 16 bits. *For forward direction:* SPD_ CMD[31:0] = Speed / 60 * 2^{32} / 10,000 *For reserve direction:* SPD_ CMD[31:0] = 2^{32} – (Speed / 60 * 2^{32} / 10,000)

			0x4D = SPD_ CMD[31:16]
0x4E: SPEED_REF[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	SPEED_ CMD	0000H	Speed command low 16 bits. *For forward direction:* SPD_ CMD[31:0] = Speed(rpm) / 60 * 2^{32} / 10,000 *For reserve direction:* SPD_ CMD[31:0] = 2^{32} – (Speed(rpm) / 60 * 2^{32} / 10,000) 0x4E = SPD_ CMD[15:0]
0x4F: SPEED_REF_SLOPE[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	SPEED_ CMD_SLOPE	1300H	Speed command slope, LSB / 100µs / 100µs, 2^{16} LSBs per revolution. *Example below is to set slope with N RPM/ms:* SPEED_ CMD_SLOPE = N(rpm) * 2^{16} * 2^{16} / 6,000,000

Speed Loop Settings

0x22: KP_ SPEED [15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KP_ SPEED	FDE8H	The gain ratio of the speed loop proportional gain. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*
0x23: KP_GAIN _SPEED [15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KP_GAIN _SPEED	8009H	The gain ratio of the speed loop proportional gain. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*
0x24: KI_SPEED[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KI_SPEED	0309H	The speed loop integral gain. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*

0x25: KI_GAIN_SPEED[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KI_GAIN_ SPEED	8013H	The gain ratio of speed loop proportion gain. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*
0x26: KC_SPEED[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	KC_SPEED	0032H	The speed loop anti-integral gain. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*
0x27: MAX_LIMIT_IQ[11:0]			
Bit[15:0]	Name	Default	Description
[11:0]	MAX_LIMIT_IQ	03FFH	Maximum torque current limit of speed loop output. *If iq is positive:* MAX_LIMIT_IQ[11:0] = iq * 1.5 * 0.01 * K_{AD} * 1024 / 1.6 *If iq is negative:* MAX_LIMIT_IQ[11:0] = 2^{12} – (iq * 1.5 * 0.01 * K_{AD} * 1024 / 1.6) iq: Torque current in A. K_{AD}: Amplifier gain coefficient of current sensing. Default value is 2.

Torque Command

0x2F: IQ_CMD[11:0]			
Bit[11:0]	Name	Default	Description
[11:0]	IQ_REF	000AH	Torque current command. *If iq is positive:* IQ_CMD[11:0] = iq * 1.5 * 0.01 * K_{AD} * 1024 / 1.6 *If iq is negative:* IQ_CMD[11:0] = 2^{12} – (iq * 1.5 * 0.01 * K_{AD} * 1024 / 1.6) iq: Torque current in A. K_{AD}: Amplifier gain coefficient of current sensing. Default value is 2.

Torque Loop Settings

0x2D: T_MAX_SPEED[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	T_MAX_SPEED	0064H	Maximum speed in torque operating mode, unit LSB/50µs, 2^{16} LSBs per revolution. *Example below is to set 3,000rpm speed limit:* T_MAX_SPEED = 3,000(rpm) * 2^{16} / 60 / 10,000
0x2E: ANTI_TORQUE_GAIN[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	ANTI_TORQUE_GAIN	000AH	Anti-saturation gain of speed limit in torque mode. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*

0x43: CURRENT_KP[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	CURRENT_KP	3BB5H	Proportional gain setting of torque loop. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*
0x44: CURRENT_KI[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	CURRENT_KI	2BA6H	Integral gain setting of torque loop. *It is recommended to use eMotion System™ Virtual Bench to set loop parameters.*

Operating Mode

0x34H			
			POS_CMD_TYPE[4] | **CMD_MODE[3:2]** | **MODE[1:0]**
Bit[15:0]	Name	Default	Description
[1:0]	MODE	0H	Control Mode: 00: Speed mode 01: Position mode 10: Torque mode
[3:2]	CMD_MODE	0H	00: Command source is from RS485 interface. 10: Command source is from PULSE/DIR input.
[4]	POS_CMD_TYPE	1H	0: Absolute position mode 1: Incremental position mode
[5]	STANDBY	1H	0: Standby mode disabled. The motor starts to run when powered up. 1: Standby mode enabled. Need to send start command.
[6:15]	Reserved		

Protection Parameters

0x56			
VDC_LIMIT[15:1]		**VDC_LIMIT_EN[0]**	
Bit[15:0]	Name	Default	Description
[0]	VDC_LIMIT_EN	0000H	VDC limit function enable bit: 0: VDC limit function disabled 1: VDC limit function enabled
[15:1]	VDC_LIMIT	00C7H	VDC limit voltage setting: VDC_LIMIT[15:1] = V_{IN_limit} * 3.887 V_{IN_limit}: Bus voltage protection limit in V.
0x58: DEAD_TIME[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	DEAD_TIME	0008H	Dead time of the half-bridge switching signal. The dead time is set by (DEAD_TIME * 25) + 12.5ns.

0x59: AD_GAIN			
Bit[15:0]	Name	Default	Description
[2:0]	AD_GAIN	0002H	The amplifier gain of the current sensing voltage: ADGAIN = 000: K_{AD} = 12X ADGAIN = 001: K_{AD} = 8X ADGAIN = 010: K_{AD} = 7X ADGAIN = 011: K_{AD} = 6X ADGAIN = 100: K_{AD} = 5X ADGAIN = 101: K_{AD} = 4X ADGAIN = 110: K_{AD} = 3X ADGAIN = 111: K_{AD} = 2X
[3]	AD_MODE	0H	Should be fixed to 0.
0x5A: IOCP[9:0]			
Bit[15:0]	Name	Default	Description
[9:0]	IOCP	03FFH	Phase Current OCP protection threshold: IOCP = I_{limit} * 0.01 * K_{AD} * 1024 / 1.6 I_{limit}: Real phase current protection limit in A.
0x6B: OVER_TEMPERATURE			
Bit[15:0]	Name	Default	Description
[15:0]	OVER_TEPERA TURE	0x0055	Set the over-temperature indication threshold. Unit is °C. When the board reaches this threshold, the red LED will flash 6 times, pause, and repeat again.

Switching Frequency

0x57: FSW[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	FSW	0014H	Switching frequency in kHz.

RS485 Communication & Operating Commands

0x5C: RS485_ADDR[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	RS485_ADDR	0000H	RS485 slave address: Available from 0x00 to 0x7F. 0x00 is alarming address.
0x5D: RS485_BAUDRATE[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	BAUDRATE	0000H	RS485 Baud rate: BAURRATE = f_{bps} / 32 f_{bps}: Target baud rate.

0x70: RUN[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	RUN	0000H	Run/stop the motor: 1: Motor run 0: Motor stop
0x71: BRAKE[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	BRAKE	0000H	Brake/run the motor: 1: Motor brake 0: Motor run

0x73: LOAD_TO_FLASH[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	LOAD_TO_ FLASH	0000H	Write 0x0002 to load register values to flash.
0x74: LOAD_FROM_FLASH[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	LOAD_FROM_ FLASH	0000H	Load register value from flash.

0x76: UPDATE_COMMAND[15:0]			
Bit[15:0]	Name	Default	Description
[15:0]	UPDATE_COM MAND	0000H	Write 0x0000 to update speed/position command.