EZmotion

User Guide

MMS7xx-x1 All-In-One Servo Motors
MMP7xx-x1 Driver Modules

Table of Contents

Overview

Introduction

The MMS7xx-x1 all-in-one servo motors are fully integrated solutions for motion control applications. They integrate a position sensor, controller, and driver inside the motor housing to provide a complete smart motor solution.

The motors can operate in speed, position, or torque control modes. The motors are controlled through either an RS-485 serial interface, or the simple input/output (I/O) interface (see Figure 1). Configurable parameters can be set with an easy-to-use, PC-based program, which interfaces to the motor through the USB and RS-485 interface. Once parameters have been optimized, they can be saved in the motor's non-volatile memory (NVM).

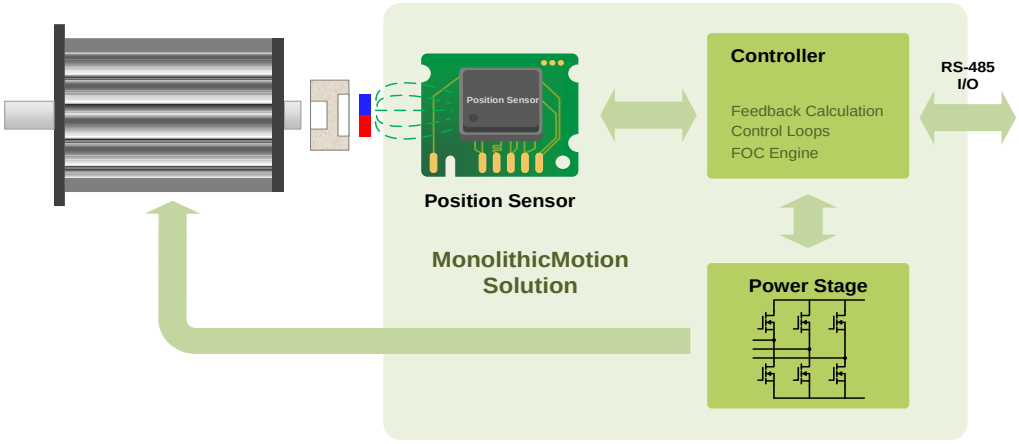


Figure 1: Motor Control Block Diagram

In addition to complete motor assemblies, a driver module is also available. The MMP7xx-x1 driver module offers a control and drive solution that can be integrated with other motors.

The MMP7xx-x1 driver module includes a magnetic position sensor for rotor feedback, a field-oriented-control (FOC) motor controller, and a power stage.

Applicable Products

Table 1 lists the products that are applicable to this user guide.

Table 1: Applicable Products

All-In-One Servo Motors		Driver Module	
MMS7xx-R1	MMS757188-36-R1-1	MMP7xx-R1	MMP757188-70-R1-1
	MMS757141-36-R1-1		MMP757141-70-R1-1
	MMS757094-36-R1-1		MMP757094-70-R1-1
	MMS742038-24-R1-1		MMP742038-36-R1-1
	MMS742052-24-R1-1		MMP742052-36-R1-1
	MMS742077-24-R1-1		MMP742077-36-R1-1
	MMS742105-24-R1-1		MMP742105-36-R1-1
MMS7xx-S1	MMS757188-36-S1-1	MMP7xx-S1	MMP757188-70-S1-1
	MMS757141-36-S1-1		MMP757141-70-S1-1
	MMS757094-36-S1-1		MMP757094-70-S1-1
	MMS742038-24-S1-1		MMP742038-36-S1-1
	MMS742052-24-S1-1		MMP742052-36-S1-1
	MMS742077-24-S1-1		MMP742077-36-S1-1
	MMS742105-24-S1-1		MMP742105-36-S1-1

Safety Warnings

To prevent personal injury or damage to the motor or other equipment, follow the guidelines listed below:

- Always secure the motor before applying power, since the motor may move unexpectedly, jump, or fall when it starts.
- Keep hair and loose clothing away from the motor.
- Avoid the shaft and any attached mechanical parts when operating the motor.
- Do not open or disassemble the motor.
- Bond the motor enclosure to a protective ground when the motor is installed in a system.
- Ensure that the power source connected to the motor is 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, or whether a fault condition has occurred (see Figure 2).

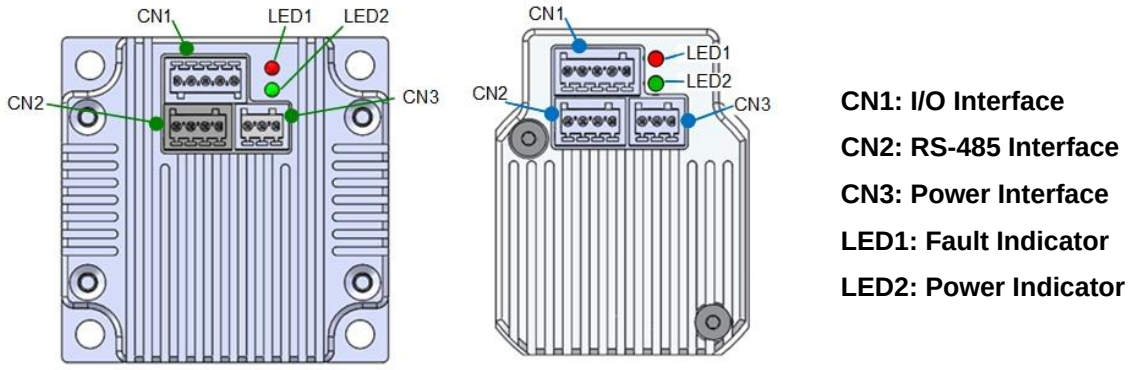


Figure 2: The Rear Surface of the Motor

Table 2 lists matching connectors. Users can find the accessory package on www.EZmotion.co. For example, the MMA03-3001 provides all of the connectors listed below.

Table 2: Matching Connectors

Connector	Pins	Manufacturer	Part Number
IO	5	KAIFENG	KF12EKD-2.5-5P-1G
RS-485	4	KAIFENG	KF12EKD-2.5-4P-1G
Power	3	KAIFENG	KF12EKD-2.5-3P-1G

1.2 Power

DC power is connected to the motor through the 3-pin connector on the rear of the motor housing (see Figure 3). Apply a DC voltage within the model's specification range. The power supply should be fused or current-limited, and it must be capable of supplying current up to the motor's target current limit

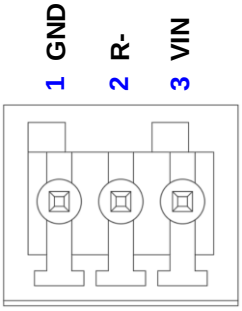


Figure 3: DC Power Connector

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

1.3 Control Interface

The MMS7xx-S1 servo motor and MMP7xx-S1 driver module are controlled by the I/O interface, which means that they only accept control signals from the I/O interface. The RS-485 interface in the MMS7xx-S1 and MMP7xx-S1 is used for debugging and reading/writing registers.

The MMS7xx-R1 servo motor and MMP7xx-R1 driver module may be controlled either using high-level commands through an RS-485 interface, or by using a simple I/O interface.

1.3.1 I/O Interface

The I/O interface is optically isolated (see Figure 4).

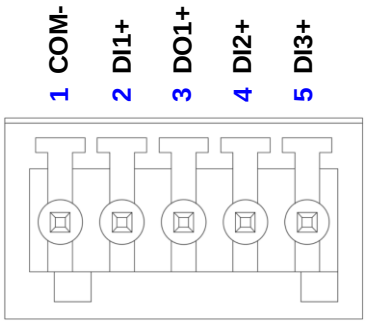


Figure 4: I/O Interface

Table 3 shows the signals used by the I/O interface.

Table 3: I/O Interface Signals

Signal	Direction	Pin	Description
COM-	Bidirectional	1	Common return.
DI1+	To motor	2	Default: ENA. Enables the motor when driven high. Alternate: Home switch, negative switch, or positive switch.
DO1+	From motor	3	Default: PEND. Driven active when the position is reached. Alternate: ALARM.
DI2+	To motor	4	Default: STEP, or PWM input to control position or speed. Alternate: Home switch, negative switch, or positive switch.
DI3+	To motor	5	Default: DIR. Sets CW rotation direction when driven high. Alternate: Home switch, negative switch, or positive switch.

Figure 5 shows the internal circuitries connected to these signals.

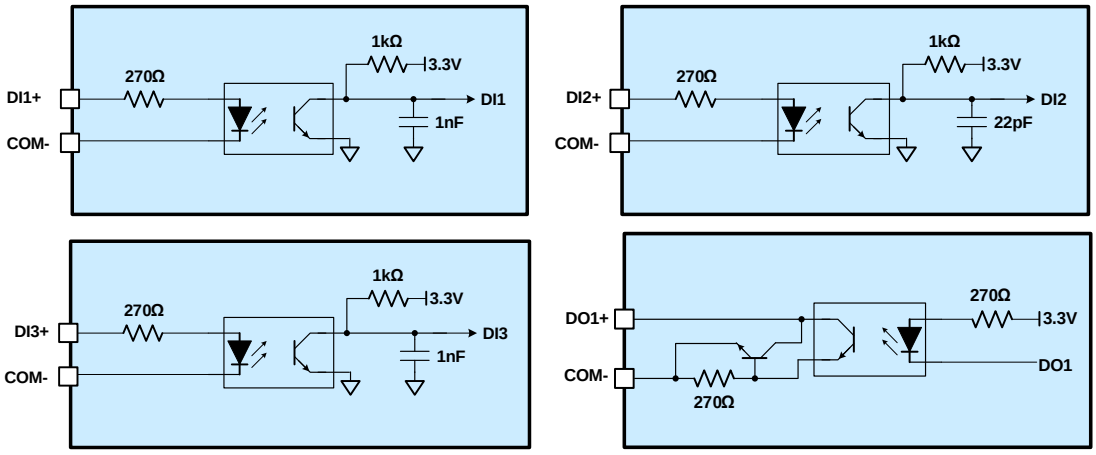


Figure 5: Internal Circuitries of the I/O Interface

The I/O interface supports STEP/DIR control for positon mode, and PWM/DIR control for speed and torque mode. The STEP or PWM signal is input via the DI2+ pin. The rotation direction is set by the state of the DI3 input signal, and motion is enabled or disabled by the DI signal.

When the motion is complete, the DO1 signal goes active to indicate that the motor has completed the motion.

1.3.2 RS-485 Interface

Figure 6 on page 7 shows the RS-485 control interface.

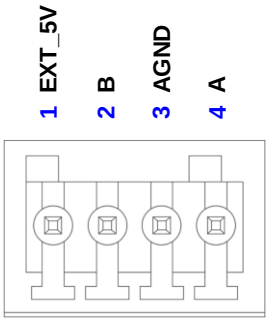


Figure 6: RS-485 Control Interface

Table 4 lists the RS-485 control interface signals.

Table 2: RS-485 Control Interface Signals

Signal	Direction	Pin	Description
EXT_5V	To motor	1	External 5V power for firmware configurations
B	Bidirectional	2	RS-485 data B (inverting)
AGND	N/A	3	Ground for EXT_5V
A	Bidirectional	4	RS-485 data A (non-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 this value can be changed by writing to the BAUDRATE register.

The module supports multi-slave communication. The slave address can be configured by writing to the RS485_ADDR register, which is available from 0x00 to 0x7F. Address 0x00 is the broadcast address, meaning that all slaves respond when the master uses 0x00 as the slave address.

The registers can be read and written to via the MotionLAB software tool (see Section 4 on page 17 for more details).

The master address is the address that the communication kit (or other master) uses. For successful communication, the master request address and slave address should be the same.

To communicate with the module without applying the main VIN power (e.g. to configure the NVM), supply 5V DC to the EXT-5V pin. This pin is not connected during normal operation.

Figure 7 shows the connection between a master controller and the MMS7xx-R1 series motor (or MMP7xx-R1 series module).

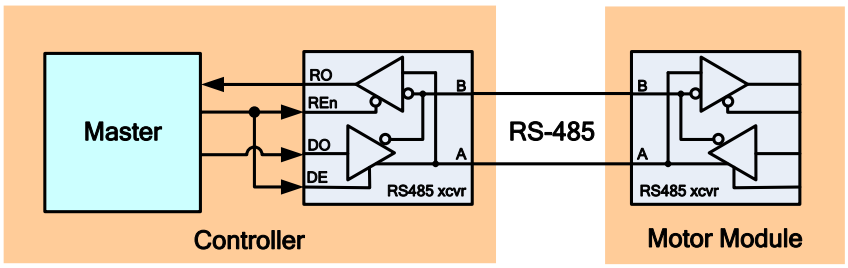


Figure 7: The Internal Circuitry of the RS-485 Control Interface

1.4 Mechanical Mounting

The MMS7xx-x1 series motors are mechanically mounted from a front flange, such as NEMA frame motors. For mounting dimensions, refer to the specifications for each motor.

The MMP7xx-x1 series modules are designed to mount to the back of standard NEMA motors. Refer to the specification for each module for more details.

Section 2. Communication

The motor/driver module uses the MODBUS RTU protocol to exchange messages between the master and slave nodes. A master-slave system has one node (the master node) that issues explicit commands to one of the slave nodes, and does not communicate with other slaves. The slave node never transmits data unless it has received a request from the master node. The slave nodes never communicate with each other.

In the module, control parameters are stored as discontinuous holding registers. They can be accessed using the MODBUS function codes 0x03 (read holding registers), 0x06 (write single register), and 0x10 (write multiple registers).

2.1 Communication Protocol

The data frame format contains an address field, a function code, data, and a CRC (see Figure 8). For each byte in the data frame, the data format is 8 data bits, 1 stop bit, and 1 odd polarity check bit.

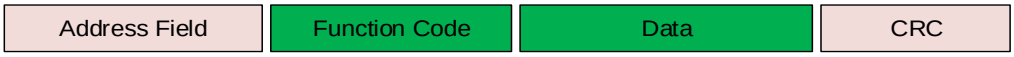


Figure 8: MODBUS RTU Data Frame

MODBUS allows for 256 different addresses. Address 0 is the broadcast address; all slave nodes recognize the broadcast address. Addresses 1–247 are valid individual addresses, and addresses 248–255 are reserved.

An exception response is sent when a communication error is detected. Figure 9 shows the exception data frame format.

Figure 9: MODBUS Exception Data Frame

For the supported function codes, the data field is defined in the following sections.

2.1.1 Read Holding Registers (0x03)

This function code is used to read the contents of holding registers in a slave device.

All the registers in the module are 16 bits in length, which occupies one holding register. The data format for the request, response, and error frames are below.

Request

Slave Address	Function Code	Register Address		Quantity of Registers		CRC
1 Byte	0x03	High Byte	Low Byte	High Byte	Low Byte	2 Bytes

Response

Slave Address	Function Code	Number of Bytes Returned	Register Value 1		……	CRC
1 Byte	0x03	1 Byte	High Byte	Low Byte	……	2 Bytes

Error

Slave Address	Error Code	Exception Code	CRC
1 Byte	0x80 + 0x03	0x01 or 0x02 or 0x03 or 0x04	2 Bytes

Table 5 on page 9 shows an example of a read register.

Table 5: Read Register IQ_LATCH (0x63) using Slave Address 0x01

Request		Response	
Field Name	**Data (Hex)**	**Field Name**	**Data (Hex)**
Address	0x01	Address	0x01
Function Code	0x03	Function Code	0x03
Starting Address (High)	0x00	Byte Count	0x02
Starting Address (Low)	0x63	Register Value (High)	0x00
Quantity of Registers (High)	0x00	Register Value (Low)	0x00
Quantity of Registers (Low)	0x01	CRC (Low)	0xB8
CRC Low	0x74	CRC (High)	0x44
CRC High	0x14	-	-

2.1.2 Write to a Single Register (0x06)

This function code is used to write a single holding register in a slave device. The data format for the request, response, and error frames are below.

Request

Slave Address	Function Code	Register Address		Value to Write		CRC
1 Byte	0x06	High Byte	Low Byte	High Byte	Low Byte	2 Bytes

Response

Slave Address	Function Code	Register Address		Written Value		CRC
1 Byte	0x06	High Byte	Low Byte	High Byte	Low Byte	2 Bytes

Error

Slave Address	Error Code	Exception Code	CRC
1 Byte	0x80 + 0x06	0x01 or 0x02 or 0x03 or 0x04	2 Bytes

Table 6 lists an example of a write register.

Table 6: Write Register SPD_KP (0x22) using Slave Address 0x01

Request		Response	
Field Name	**Data (Hex)**	**Field Name**	**Data (Hex)**
Address	0x01	Address	0x01
Function Code	0x06	Function Code	0x06
Starting Address (High)	0x00	Starting Address (High)	0x00
Starting Address (Low)	0x22	Starting Address (Low)	0x22
Quantity of Registers (High)	0xFD	Register Value (High)	0xFD
Quantity of Registers (Low)	0xFE	Register Value (Low)	0xFE
CRC (Low)	0xE8	CRC (Low)	0xE8
CRC (High)	0xD0	CRC (High)	0xD0

2.1.3 Write to Multiple Registers (0x10)

This function code writes to a block of continuous registers in a slave device. This function can be used to write 32-bit control parameters with register length of 2.

Request

Slave Address	Function Code	Start Register Address	Register Number	Byte Count	Register Value 1	……	CRC
1 Byte	0x10	2 Bytes	2 Bytes	1 Byte	2 Bytes	……	2 Bytes

Response

Slave Address	Function Code	Start Register Address		Register Number		CRC
1 Byte	0x10	High Byte	Low Byte	High Byte	Low Byte	2 Bytes

Error

Slave Address	Error Code	Exception Code	CRC
1 Byte	0x80 + 0x10	0x01 or 0x02 or 0x03 or 0x04	2 Bytes

Table 6 lists an example of a write to multiple registers.

Table 6: Write POS_CMD (0x4A and 0x4B) to Slave 0x01

Request		Response	
Field Name	**Data (Hex)**	**Field Name**	**Data (Hex)**
Address	0x01	Address	0x01
Function code	0x10	Function code	0x10
Starting Address (High)	0x00	Starting Address (High)	0x00
Starting Address (Low)	0x4A	Starting Address (Low)	0x4A
Register Number (High)	0x00	Register Number (High)	0x00
Register Number (Low)	0x02	Register Number (Low)	0x02
Bytes to Write	0x04	CRC (Low)	0x60
Value to Write (High)	0x00	CRC (High)	0x1E
Value to Write (Low)	0x01	-	-
Value to Write (High)	0x00	-	-
Value to Write (Low)	0x00	-	-
CRC (Low)	0x26	-	-
CRC (High)	0x20	-	-

Section 3. Operational Modes

3.1 Position Control Mode

3.1.1 RS-485 Control Mode

The RS-485 control mode is for the MMS7xx-R1 and MMP7xx-R1 only. Using RS-485 control, the position mode can be configured 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, while the incremental command moves the motor relative to its current position.

In absolute command mode, the maximum command limit is ±32,768 mechanical revolutions. In incremental command mode, the minimum command resolution should exceed 0.2° due to the internal position sensor's resolution limitations.

The command is set through the POS_CMD register (0x4A and 0x4B) (see Table 7). In this table, Revs is the target position for command full revolutions, and Theta is the target position command's angle.

Table 7: POS_CMD Register

Forward Direction	Reverse Direction
POS_CMD, bits[31:0] = (Revs + Theta / 360) x 2^{16} 0x4A, bits[15:0] = POS_CMD, bits[31:16] 0x4B, bits[15:0] = POS_CMD, bits[15:0]	POS_CMD, bits[31:0] = 2^{32} – (Revs + Theta / 360) x 2^{16} 0x4A, bits[15:0] = POS_CMD, bits[31:16] 0x4B, bits[15:0] = POS_CMD, bits[15:0]

Table 8 shows an example when the position is set to 2 revolutions plus 45°.

Table 8: POS_CMD Register Example

Forward Direction	Reverse Direction
0x4A, bits[15:0] = 2 = 0002h 0x4B, bits[15:0] = 45/360 x 2^{16} = 2000h	0x4A, bits[15:0] = 2^{16} – 2 = FFFDh 0x4B, bits[15:0] = 2^{16} – 45/360 x 2^{16} = E000h

After a new position command is given in position mode, the reference generation block inside the device calculates the position curve every 100µs to make the motion smooth (see Figure 10).

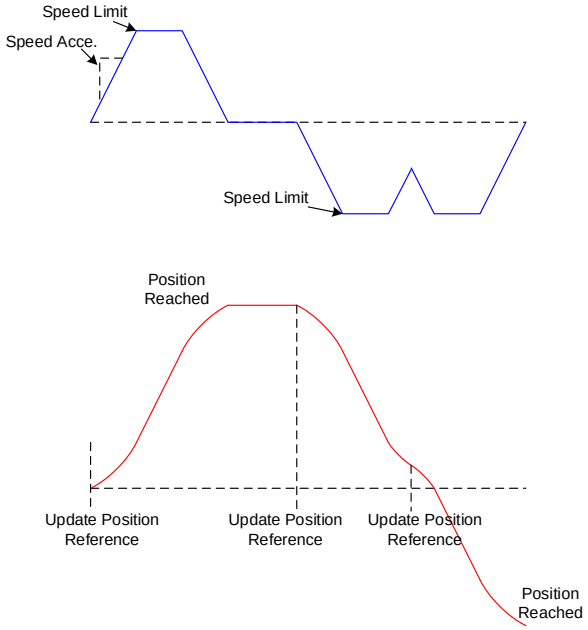


Figure 1: Position Reference Profile

If the speed limit has not been reached, and there is enough angle left for the motor to decrease the speed and stop, the motor either increases its speed or maintains its speed. When the speed reaches the speed limit, the motor limits the speed at the set value. If the reference generation block detects that the remaining angle is not enough for the motor to ramp down the speed using the set speed slope, the generation block slows the motor so that the speed is zero once the position is reached.

If a new position command is given when the motor speed is ramping down, the reference generation block increases the motor speed again to ensure that the motor reaches the target position as quickly as possible.

The position mode slope is set via the PROFILE_ACCE (0x4F) and SPD_CMD (0x4D and 0x4E) registers. PROFILE_ACCE sets the speed ramp slope in position mode. SPD_CMD sets the position ramp slope, which is the same as the speed limit (the default value is 3000rpm).

Write 0x0000 to register 0x76 to update the position command.

3.1.2 STEP/DIR Control Mode

The STEP/DIR control mode is for all applicable products listed in Table 1 on page 3.

In STEP/DIR command control mode, the position works in incremental mode. Each rising edge on the STEP input moves the motor by a configurable increment. The number of steps per revolution can be configured to 512, 1024, 2048, or 4096 steps per mechanical revolution. This is set through the NSTEP register. The default value is 4096.

The number of steps per revolution can also be set by the on-board switch (SW1). NSTEP_TYPE (register 0x34, bit[7]) determines whether the steps per revolution is set via the register value or the switch status. Table 9 lists the relationship between the SW1 status and the number of steps per revolution. Note that pin 1 of SW1 should be off all the time.

Table 9: Relationship Between SW1 Status and Steps per Revolution

Pin 2 Status	Pin 3 Status	Pin 4 Status	Steps per Revolution
On	On	On	4096 steps per revolution
On	On	Off	2048 steps per revolution
On	Off	On	1024 steps per revolution
On	Off	Off	512 steps per revolution
Other combinations			512 steps per revolution

To smoothly ramp the speed of the motor, the velocity up/down slope can be configured through the PROFILE_ACCE 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 can be configured via the MAX_LIMIT_IQ register.

3.2 Speed Control Mode

3.2.1 RS-485 Control Mode

RS-485 control mode is for the MMS7xx-R1 and MMP7xx-R1 only. 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 and 0x4E) (see Table 10 on page 13). In Table 10, Speed is the target speed (in rpm).

Table 10: SPD_CMD Register

Forward Direction	Reverse Direction
SPD_CMD, bits[31:0] = (Speed / 60) x 2^{32} / 10,000 0x4D = SPD_CMD, bits[31:16] 0x4E = SPD_CMD, bits[15:0]	SPD_CMD, bits[31:0] = 2^{32} - (Speed / 60) x 2^{32} / 10,000 0x4D = SPD_CMD, bits[31:16] 0x4E = SPD_CMD, bits[15:0]

Table 11 lists how to set the speed to 3,000rpm.

Table 11: SPD_CMD Register Example

Forward Direction	Reverse Direction
SPD_CMD, bits[31:0] = (3,000 / 60) x 2^{32} / 10,000 = 0147AE14h 0x4D = SPD_CMD, bits[31:16] = 0147h 0x4E = SPD_CMD, bits[15:0] = AE14h	SPD_CMD, bits[31:0] = 2^{32} - ((3,000 / 60) x 2^{32} / 10,000) = FFFCB965h 0x4D = SPD_CMD, bits[31:16] = FFFCh 0x4E = SPD_CMD, bits[15:0] = B965h

In speed mode, the velocity ramp up/down slope can be configured via the PROFILE_ACCE register. After updating the speed reference, the motor speed ramps to the target speed using the set slope rate. If the slope rate setting changes during ramping, the actual slope rate changes immediately, and the motor changes speed according to the newly set ramp slope.

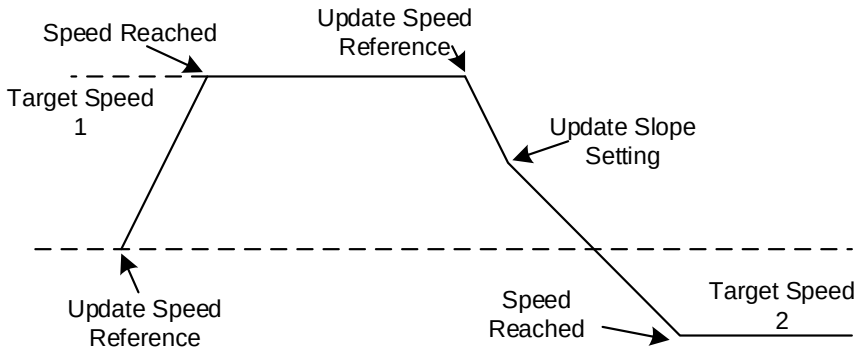


Figure 11: Speed Reference Profile

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

The maximum torque limit can be configured via MAX_LIMIT_IQ. For more details, see the Torque Control Mode section on page 14.

Write 0x0000 to register 0x76 to update the speed command.

3.2.2 PWM/DIR Control Mode

PWM/DIR control mode is for all applicable products listed in Table 1 on page 3.

In PWM/DIR command control mode, the motor speed is controlled by the duty cycle of the PWM input. The real motor speed is (SPD_CMD x Duty Cycle), and the SPD_CMD can be configured (registers 0x4D and 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 clockwise.

3.3 Torque Control Mode

3.3.1 RS-485 Control Mode

RS-485 control mode is for the MMS7xx-R1 and MMP7xx-R1 only. In torque control mode, the torque command (which corresponds to the phase current) is set directly through the RS-485 interface; otherwise, it is set through the IQ_CMD register.

If the torque current (I_Q) is positive, IQ_CMD, bits[11:0] can be calculated with Equation (1):

$$\text{IQ\_CMD, bits[11:0]} = I_Q \times 1.5 \times 0.01 \times K_{AD} \times 1024 / 1.6 \quad (1)$$

Where K_{AD} is the amplifier gain coefficient for current-sensing. If I_Q is negative, IQ_CMD, bits[11:0] can be calculated with Equation (2):

$$\text{IQ\_CMD, bits[11:0]} = 2^{12} - (I_Q \times 1.5 \times 0.01 \times K_{AD} \times 1024 / 1.6) \quad (2)$$

Check the corresponding specification for the actual amplifier gain coefficient.

Below is an example to set a 5A torque current when the amplifier gain is 2.

If I_Q is positive, IQ_CMD, bits[11:0] can be calculated with Equation (3):

$$\text{IQ\_CMD, bits[11:0]} = 5 \times 1.5 \times 0.01 \times 2 \times 1024 / 1.6 = 60h \quad (3)$$

If I_Q is negative, IQ_CMD, bits[11:0] can be calculated with Equation (4):

$$\text{IQ\_CMD, bits[11:0]} = 2^{12} - (5 \times 1.5 \times 0.01 \times 2 \times 1024 / 1.6) = FA0h \quad (4)$$

3.3.2 PWM/DIR Mode

PWM/DIR control mode is for all applicable products listed in Table 1 on page 3.

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_CMD x Duty Cycle), where IQ_CMD can be configured via 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 rotate clockwise.

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

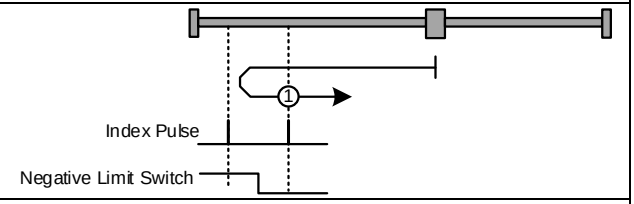
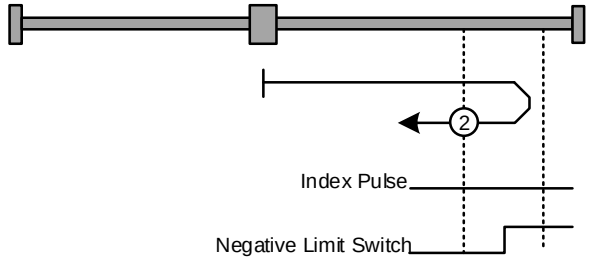
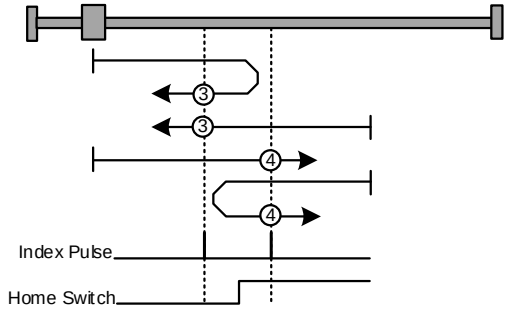
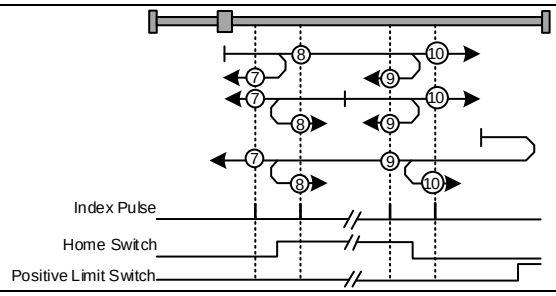
3.4 Homing Mode

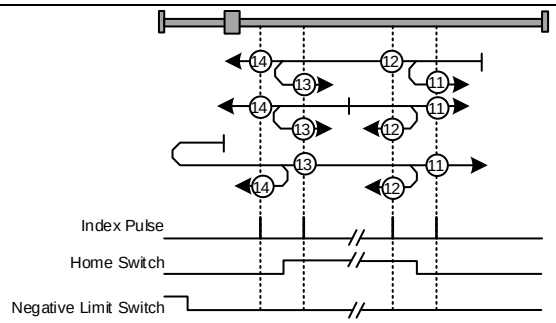
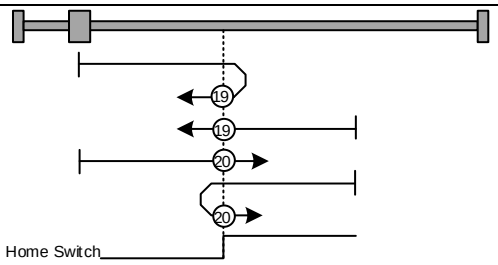
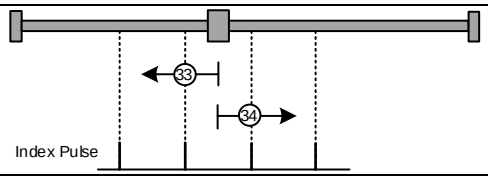
Homing mode is used for seeking the home position (also called the datum, reference point, or zero point). There are various methods to achieve this function using a limit switch at the ends of travel, or a home switch (zero-point switch) at the middle of travel. Most of the methods also use the index (zero) pulse from an angle sensor. Homing with limited torque methods are also supported. These methods are not required to limit switch signals, which can simplify the system and lower the cost.

The user should specify the speed, acceleration, and the method of homing. There are two homing speeds; typically, the faster speed is used to find the home switch (homing speed switch), while the slower speed is used to find the index pulse (homing speed zero).

The controller supports methods 1 to 14, 17 to 30, 33, and 34, as defined in CiA DSP 402 standard. In addition, the controller supports homing with torque limit methods. Table 12 on page 15 shows the detailed descriptions and illustrations of these methods.

Table 12: Homing Methods

Method	Description	Diagram
-3	Homing clockwise with a limited torque.	These two methods allow for homing without a limit switch or home switch. The motor goes in one direction until it reaches the mechanical range limit. The motor output torque is limited by the homing torque.
-2	Homing counterclockwise with a limited torque.	
1	Homing on a negative limit switch and index pulse.	
2	Homing on a positive limit switch and index pulse.	
3, 4	Homing on a positive home switch and index pulse.	
7, 8, 9, 10	Homing on a home switch and index pulse (positive initial motion).	

11, 12, 13, 14	Homing on a home switch and index pulse (negative initial motion).	
17–30	These methods are similar to methods 1–14, except that the home position is not dependent on the index pulse. It is only dependent on the relevant home or limit switch transitions. For example, methods 19 and 20 are similar to methods 3 and 4.	
33, 34	Homing on the index pulse.	

3.5 Fault Indication

The control module has robust protections 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 (see Table 13).

Table 13: Fault Indication

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 4. MotionLAB

The MMS7xx-x1 series motors are pre-configured with initial parameters, so the motor can simply be connected to power and to a PC (via the USB to RS-485 interface) to spin the motor.

EZmotion provides a user-friendly, PC-based virtual bench GUI software, MotionLAB, which offers an easy way to configure and monitors the performance of the motor and module (see Figure 12).

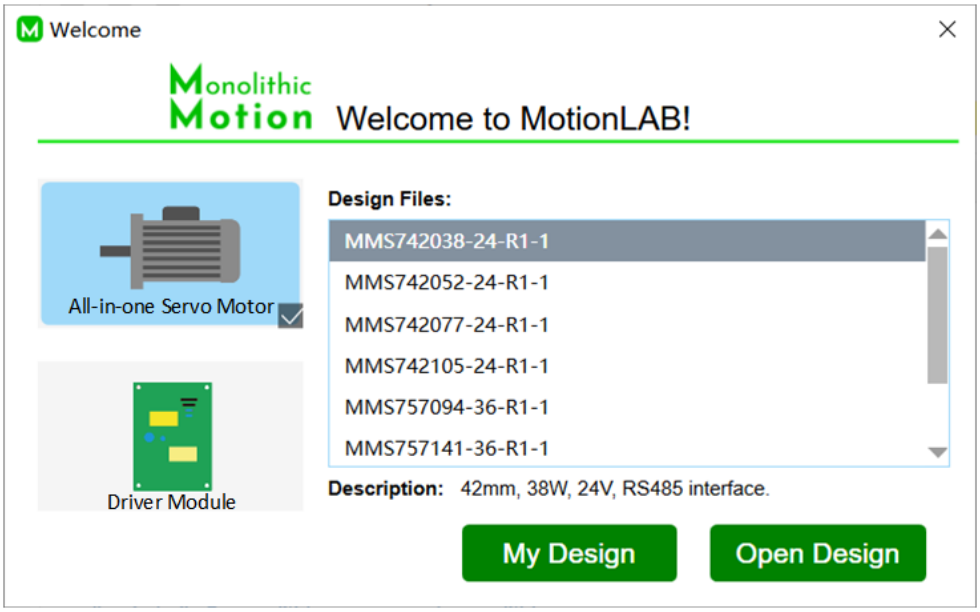


Figure 12: MotionLAB

Download the MotionLAB software and its driver installer from the EZmotion website. The user guide for MotionLAB is also available on the website.

Install MotionLAB and its driver. Read its user guide carefully before using MotionLAB software to run the motor.

Section 5. Register Map

Addr	D[15:0]
Position Command	
1Eh	POS_THRESHOLD, bits[15:0]
4Ah	POS_CMD, bits[31:16]
4Bh	POS_CMD, bits[15:0]
50h	NSTEP, bits[15:0]
Position Loop Settings	
1Ah	KP_POS, bits[15:0]
1Bh	KP_GAIN_POS, bits[15:0]
1Ch	SPEED_LIMIT, bits[15:0]
Speed Commands	
29h	SPD_THRESHOLD, bits[15:0]
4Dh	SPD_CMD, bits[31:16]
4Eh	SPD_CMD, bits[15:0]
4Fh	PROFILE_ACCE, bits[15:0]
Speed Loop Settings	
22h	KP_SPD, bits[15:0]
23h	KP_GAIN_SPD[15:0]
24h	KI_SPD, bits [15:0]
25h	KI_GAIN_SPD, bits[15:0]
26h	KC_SPD, bits[15:0]
27h	MAX_LIMIT_IQ, bits[11:0]
Torque Command	
2Fh	IQ_CMD, bits[11:0]
Torque Loop Settings	
2Dh	T_MAX_SPD, bits[15:0]
2Eh	ANTI_TORQUE_GAIN, bits[15:0]
43h	CURRENT_KP, bits[15:0]
44h	CURRENT_KI, bits[15:0]
Homing Mode Parameters	
30h	DO1_FUNCTION, bits[15:0]
31h	DI2_FUNCTION, bits[15:0]
32h	DI1_FUNCTION, bits[15:0]
33h	DI3_FUNCTION, bits[15:0]
35h	IO_POLARITY, bits[15:0]
39h	HOMING_METHOD, bits[15:0]
3Ah	HOMING_SPEED_SWITCH, bits[15:0]
3Bh	HOMING_SPEED_ZERO, bits[15:0]
3Ch	HOMING_TORQUE, bits[15:0]
3Dh	HOMING_STALL_TIME, bits[15:0]
45h	HOMING_ACCE, bits[15:0]
48h	HOMING_OFFSET_ROUND, bits[15:0]
49h	HOMING_OFFSET_ANGLE, bits[15:0]
Operating Mode	
34h	CONFIG, bits[15:0]

Protection Parameters	
53h	ERROR_STATUS, bits[15:0]
54h	RETRY, bits[15:0]
55h	PROTECTION, bits[15:0]
56h	VDC_LIMIT, bits[15:0]
58h	DEADTIME[15:0]
59h	ADGAIN, bits[15:0]
5Ah	IOCP, bits[15:0]
5Bh	V_UVLO, bits[15:0]
6Bh	OVERTEMP, bits[15:0]
Switching Frequency	
57h	FSW, bits[15:0]
RS-485 Communication	
5Ch	RS485_ADDR, bits[15:0]
5Dh	RS485_BAUDRATE, bits[15:0]
Servo Information	
5Eh	TEMPERATURE, bits[15:0]
5Fh	POSITION, bits[31:16]
60h	POSITION, bits[15:0]
61h	SPEED, bits[15:0]
62h	SPEED, bits[31:16]
63h	IQ_LATCH, bits[15:0]
64h	ID_LATCH, bits[15:0]
65h	UQ_LATCH, bits[15:0]
66h	UD_LATCH, bits[15:0]
6Ah	STATUS, bits[15:0]
6Ch	VIN_SENSE, bits[15:0]
Operating Command	
70h	RUN, bits[15:0]
71h	BRAKE, bits[15:0]
73h	LOAD_TO_FLASH, bits[15:0]
74h	LOAD_FROM_FLASH, bits[15:0]
76h	UPDATE_COMMAND, bits[15:0]

Section 6. Register Details

Position Commands

0x1E: POS_THRESHOLD			
Bits	**Name**	**Default**	**Description**
15:0	POS_ THRESHOLD	0024h	Sets the position to reach the target threshold (LSB). 1 LSB = 360 / 65536°. The target is reached when the target position and the real feedback position error are below this value.

0x4A: POS_CMD, Bits[31:16]			
Bits	**Name**	**Default**	**Description**
15:0	POS_CMD	0000h	Position command (higher 16 bits), calculated with the following equation: POS_CMD, bits[31:0] = (Revolutions + Theta / 360) x 2^{16} 0x4A, bits[15:0] = POS_CMD, bits[31:16]

0x4B: POS_CMD, Bits[15:0]			
Bits	**Name**	**Default**	**Description**
15:0	POS_CMD	0000h	Position command (lower 16 bits), calculated with the following equation: POS_CMD, bits[31:0] = (Revolutions + Theta / 360) x 2^{16} 0x4B, bits[15:0] = POS_CMD, bits[15:0]

0x50: NSTEP			
Bits	**Name**	**Default**	**Description**
15:0	NSTEP	0004h	Sets the stepping resolution per step input. 4: 4096 steps per revolution 5: 2048 steps per revolution 6: 1024 steps per revolution 7: 512 steps per revolution

Position Loop Settings

0x1A: KP_POS			
Bits	**Name**	**Default**	**Description**
15:0	KP_POS	00FAh	Sets the position loop proportional gain.

0x1B: KP_GAIN_POS			
Bits	**Name**	**Default**	**Description**
15:0	KP_GAIN_POS	8008h	Sets the gain ratio of the position loop proportional gain.

0x1C: SPEED_LIMIT			
Bits	**Name**	**Default**	**Description**
15:0	SPEED_LIMIT	02FFh	Sets the maximum speed in position mode. The unit is LSB / 100µs, with 2^{16} LSBs per revolution. In the calculation below, the speed limit is set to 3,000rpm: SPEED_LIMIT = 3000(rpm) x 2^{16} / 60 / 10,000

Speed Commands

0x29: SPD_THRESHOLD			
Bits	**Name**	**Default**	**Description**
15:0	SPD_ THRESHOLD	0006h	Sets the speed to reach the target threshold (LSB/100µs). 1 LSB / 100µs = 9.155rpm. The target is reached when the target speed and real feedback speed error are below this value.

0x4D: SPD_CMD, Bits[31:16]			
Bits	**Name**	**Default**	**Description**
15:0	SPD_CMD	000Ah	Speed command (higher 16 bits). The speed in forward direction can be calculated with the following equation: SPD_CMD, bits[31:0] = Speed / 60 x 2^{32} / 10,000 The speed in reverse direction can be calculated with the following equation: SPD_CMD bits[31:0] = 2^{32} - (Speed / 60 x 2^{32} / 10,000) 0x4D = SPD_CMD, bits[31:16].

0x4E: SPD_CMD, Bits[15:0]			
Bits	**Name**	**Default**	**Description**
15:0	SPD_CMD	0000h	Speed command (lower 16 bits). The speed in forward direction can be calculated with the following equation: SPD_CMD, bits[31:0] = Speed(rpm) / 60 x 2^{32} / 10,000 The speed in reverse direction can be calculated with the following equation: SPD_CMD, bits[31:0] = 2^{32} - (Speed(rpm) / 60 x 2^{32} / 10,000) 0x4E = SPD_CMD, bits[15:0].

0x4F: PROFILE_ACCE			
Bits	**Name**	**Default**	**Description**
15:0	PROFILE_ACCE	1300h	Sets the profile acceleration for the position and speed trajectory generator (in LSB / 100µs / 100µs, with 2^{16} LSBs per revolution). Calculate the slope with N (in rpm/ms) using the following equation: PROFILE_ACCE = N(rpm/ms) x 2^{16} x 2^{16} / 6,000,000

Speed Loop Settings

0x22: KP_SPD			
Bits	**Name**	**Default**	**Description**
15:0	KP_SPD	FDE8h	Sets the gain ratio for the speed loop proportional gain. It is recommended to use MotionLAB software to set the loop parameters.

0x23: KP_GAIN_SPD			
Bits	**Name**	**Default**	**Description**
15:0	KP_GAIN_SPD	8009h	Sets the gain ratio for the speed loop proportional gain. It is recommended to use MotionLAB software to set the loop parameters.

0x24: KI_SPD			
Bits	**Name**	**Default**	**Description**
15:0	KI_SPD	0309h	Sets the speed loop integral gain. It is recommended to use MotionLAB software to set the loop parameters.

0x25: KI_GAIN_SPD			
Bits	**Name**	**Default**	**Description**
15:0	KI_GAIN_SPD	8013h	Sets the gain ratio for the speed loop proportion gain. It is recommended to use MotionLAB software to set the loop parameters.

0x26: KC_SPD			
Bits	**Name**	**Default**	**Description**
15:0	KC_SPD	0032h	Sets the speed loop anti-integral gain. It is recommended to use MotionLAB software to set the loop parameters.

0x27: MAX_LIMIT_IQ			
Bits	**Name**	**Default**	**Description**
11:0	MAX_LIMIT_IQ	03FFh	Sets the maximum torque current limit for the speed loop output. If I_Q is positive, the limit can be calculated with the following equation: MAX_LIMIT_IQ, bits[11:0] = I_Q x 1.5 x 0.01 x K_{AD} x 1024 / 1.6 If I_Q is negative, the limit can be calculated with the following equation: MAX_LIMIT_IQ, bits[11:0] = 2^{12} - (I_Q x 1.5 x 0.01 x K_{AD} x 1024 / 1.6) Where I_Q is the torque current (in A), and K_{AD} is the amplifier gain's coefficient for current-sensing. The default value is 2.

Torque Command

0x2F: IQ_CMD			
Bits	**Name**	**Default**	**Description**
15:12	RESERVED	0h	Unused.
11:0	IQ_CMD	000Ah	Sets the torque current . If I_Q is positive, the value can be calculated with the following equation: IQ_CMD, bits[11:0] = I_Q x 1.5 x 0.01 x K_{AD} x 1024 / 1.6 If I_Q is negative, the value can be calculated with the following equation: IQ_CMD, bits[11:0] = 2^{12} - (I_Q x 1.5 x 0.01 x K_{AD} x 1024 / 1.6) Where I_Q is the torque current (in A), and K_{AD} is the amplifier gain's coefficient for current-sensing. The default value is 2.

Torque Loop Settings

0x2D: T_MAX_SPD			
Bits	**Name**	**Default**	**Description**
15:0	T_MAX_SPD	0064h	Sets the maximum speed in torque operating mode (in LSB / 100µs, with 2^{16} LSBs per revolution). In the calculation below, the speed limit is set to 3,000rpm: T_MAX_SPD = 3,000(rpm) x 2^{16} / 60 / 10,000

0x2E: ANTI_TORQUE_GAIN			
Bits	**Name**	**Default**	**Description**
15:0	ANTI_TORQUE_GAIN	000Ah	Sets the anti-saturation gain for the speed limit in torque mode. It is recommended to use MotionLAB software to set the loop parameters.

0x43: CURRENT_KP			
Bits	**Name**	**Default**	**Description**
15:0	CURRENT_KP	3BB5h	Sets the proportional gain setting for the torque loop. It is recommended to use MotionLAB software to set the loop parameters.

0x44: CURRENT_KI			
Bits	**Name**	**Default**	**Description**
15:0	CURRENT_KI	2BA6h	Sets the integral gain for the torque loop. It is recommended to use MotionLAB software to set the loop parameters.

Homing Mode

0x30: DO1_FUNCTION			
Bits	**Name**	**Default**	**Description**
15:0	DO1_FUNCTION	0	0: PEND 1: ALARM

0x31: DI2_FUNCTION			
Bits	**Name**	**Default**	**Description**
15:0	DI2_FUNCTION	0	0: STEP 1: Home switch 2: Negative switch 3: Positive switch

0x32: DI1_FUNCTION			
Bits	**Name**	**Default**	**Description**
15:0	DI1_FUNCTION	0	0: DIR 1: Home switch 2: Negative switch 3: Positive switch

0x33: DI3_FUNCTION			
Bits	**Name**	**Default**	**Description**
15:0	DI3_FUNCTION	0	0: ENA 1: Home switch 2: Negative switch 3: Positive switch

0x35: IO_POLARITY			
Bits	**Name**	**Default**	**Description**
15:4	Reserved	0	Unused.
3	DI3_POLARITY	1	0: None inverted 1: Inverted
2	DI1_POLARITY	1	0: None inverted 1: Inverted
1	DI2_POLARITY	1	0: None inverted 1: Inverted
0	DO1_POLARITY	1	0: None inverted 1: Inverted

0x39: HOMING_METHOD			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ METHOD	0	-3: Homing with the set torque (clockwise) -2: Homing with the set torque (counterclockwise) -1: Reserved 0: No homing operation required 1–35: Methods 1–35 (see Table 12 on page 15 for more details)

0x3A: HOMING_SPEED_SWITCH			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ SPEED_ SWITCH	0	Sets the speed during the search for a switch. The unit is in LSB / 100µs, where LSB / 100µs = rpm x 2^{16} / 60 / 10,000.

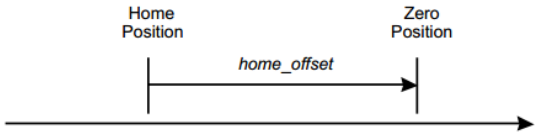
0x3B: HOMING_SPEED_ZERO			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ SPEED_ZERO	0	Sets the speed during the search for zero. The unit is in LSB / 100µs, where LSB / 100µs = rpm x 2^{16} / 60 / 10,000.

0x3C: HOMING_TORQUE			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ TORQUE	0	Sets the torque limit during homing, calculated with the following equation: HOMING_TORQUE = I_Q x 1.5 x 0.01 x K_{AD} x 1024 / 1.6 Where I_Q is the torque current (in A), and K_{AD} is the amplifier gain's coefficient for current-sensing. The default value is 2.

0x3D: HOMING_STALL_TIME			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ STALL_TIME	0	The unit for this time is ms.

0x45: HOMING_ACCE			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ACCE	0	Sets the acceleration and deceleration during homing (the unit is in LSB / 100μs / 100μs, with 2^{16} LSBs per revolution). Calculate the deceleration with N (in rpm/ms) using the following equation: Homing_ACCE = N(rpm/ms) x 2^{16} x 2^{16} / 6,000,000

0x48: HOMING_OFFSET_ROUND			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ OFFSET_ ROUND	0	Sets the homing offset (higher 16 bits) (see 0x49: HOMING_OFFSET_ANGLE below for more details).

0x49: HOMING_OFFSET_ANGLE			
Bits	**Name**	**Default**	**Description**
15:0	HOMING_ OFFSET_ ANGLE	0	Sets the homing offset (lower 16 bits), which is the difference between the zero position for the application and the machine home position (found during homing). Once the homing is completed, the zero position is offset from the homing position by adding the home offset to the home position. All the subsequent absolute moves shall be taken relative to this new zero position. 

Operating Mode

0x34: CONFIG			
Bits	**Name**	**Default**	**Description**
15:12	MODE	0h	Selects the operation mode. 0: Speed mode 1: Position mode 2: Torque mode 3: Homing Others: Reserved
11:10	RESERVED	0h	Reserved
9	FEED_FORWARD	0h	Enables the feed-forward function. 0: Disabled 1: Enabled
8	SCURVE	0h	Enables the S-curve function. 0: Disabled. The position reference profile is the ramp profile 1: Enabled
7	NSTEP_TYPE	0h	0: NSTEP is determined by the register value 1: NSTEP is determined by the on-board switch SW1

6	PWM_MODE	0h	Defines the three-phase bridge PWM driver output mode. 0: Outputs six separates PWM signals to drive the three-phase bridge MOSFETs 1: EN and PWM combination mode
5	STANDBY	1h	0: Standby mode disabled. The motor starts to run when powered up 1: Standby mode enabled. A start command is required for start-up
4	POS_CMD_TYPE	1h	0: Absolute position mode 1: Incremental position mode
3:2	CMD_MODE	0h	00: The command source is the RS-485 interface 01: Reserved 10: The command source is the STEP/DIR input 10: The command source is the PWM/DIR input
1:0	RESERVED	0h	Reserved.

Protection Parameters

0x53: ERROR_STATUS			
Bits	**Name**	**Default**	**Description**
15:7	RESERVED	0h	Unused.
6	UVLO	0h	Indicates the under-voltage (UV) status.
5	OTP	0h	Flag that indicates whether over-temperature protection (OTP) has been triggered.
4	OVERLOAD	0h	Flag that indicates whether overload protection has been triggered.
3	LOCK	0h	Flag that indicates whether lock protection has been triggered.
2	OCP	0h	Flag that indicates whether over-current protection (OCP) has been triggered.
1	PSFT	0h	Flag that indicates whether a power stage fault has been triggered.
0	MEMORY	0h	Indicates the memory fault status.

0x54: RETRY			
Bits	**Name**	**Default**	**Description**
15:5	RESERVED	0h	Unused.
4	RETRY_EN	0h	Enables the fault retry function. 0: Disabled. Fault protections are in latch-off mode 1: Enabled. Fault protections are in retry mode
3:0	RETRY_TIME	0h	Sets the fault retry time, calculated with the following equation: Retry time = (RETRY_TIME + 1) x 0.5s.

0x55: PROTECTION			
Bits	**Name**	**Default**	**Description**
15:13	RESERVED	0h	Unused.
12	OVERLOAD_EN	0h	Enables overload protection.
11:8	OVERLOAD_TIME	0h	Sets the overload detection time, calculated with the following equation: Detection time = (OVERLOAD_TIME + 1) x 1s

7:5	RESERVED	0h	Unused.
4	LOCK_EN	0h	Enables the lock protection function. 0: Disabled 1: Enabled
3:0	LOCK_TIME	0h	Sets the lock detection time, calculated with the following equation: Detection time = (LOCK_TIME + 1) x 0.5s

0x56: VDC_LIMIT			
Bits	**Name**	**Default**	**Description**
15:1	VDC_LIMIT	00C7h	Sets the V_{DC} limit, calculated with the following equation VDC_LIMIT, bits[15:1] = V_{IN_LIMIT} x 3.887 Where V_{IN_LIMIT} is the bus voltage protection limit (in V).
0	VDC_LIMIT_EN	0000h	Enables the V_{DC} limit function. 0: Disabled 1: Enabled

0x58: DEADTIME			
Bits	**Name**	**Default**	**Description**
15:0	DEADTIME	0008h	Sets the dead time for the half-bridge switching signal. The dead time is set by (DEADTIME x 25) + 12.5ns.

0x59: ADGAIN			
Bits	**Name**	**Default**	**Description**
15:4	RESERVED	0h	Unused.
3	ADMODE	0h	Fixed to 0.
2:0	ADGAIN	0002h	Sets the amplifier gain for the current-sensing voltage. 000: K_{AD} = 12x 001: K_{AD} = 8x 010: K_{AD} = 7x 011: K_{AD} = 6x 100: K_{AD} = 5x 101: K_{AD} = 4x 110: K_{AD} = 3x 111: K_{AD} = 2x

0x5A: IOCP			
Bits	**Name**	**Default**	**Description**
15:10	RESERVED	0h	Unused.
9:0	IOCP	03FFh	Sets the phase current OCP threshold, calculated with the following equation: $I_{OCP} = I_{LIMIT}$ x 0.01 x K_{AD} x 1024 / 1.6 Where I_{LIMIT} is the real phase current protection limit (in A).

0x5B: V_UVLO			
Bits	**Name**	**Default**	**Description**
15:0	V_UVLO	025Dh	Sets the under-voltage lockout (UVLO) threshold, calculated with the following equation: V_UVLO, bits[15:0] = V_{UVLO} / 3.3 x (10 / 412) x 4096 Where V_{UVLO} is the target UVLO threshold (in V).

0x6B: OVERTEMP			
Bits	**Name**	**Default**	**Description**
15:0	OVERTEMP	0x0055	Set the over-temperature indication threshold (in °C). When the board reaches this threshold, the red LED flashes 6 times, pauses, and repeats the process.

Switching Frequency

0x57: FSW			
Bits	**Name**	**Default**	**Description**
15:0	FSW	0014h	Sets the switching frequency (in kHz).

RS485 Communication

0x5C: RS485_ADDR			
Bits	**Name**	**Default**	**Description**
15:0	RS485_ADDR	0000h	Sets the RS-485 slave address (from 0x00 to 0x7F). 0x00 is the broadcast address.

0x5D: RS485_BAUDRATE			
Bits	**Name**	**Default**	**Description**
15:0	RS485_ BAUDRATE	0000h	Sets the RS-485 baud rate, calculated with following equation: RS485_BAURRATE = f_{BPS} / 32 Where f_{BPS} is the target baud rate.

Servo Information

0x5E: TEMPERATURE			
Bits	**Name**	**Default**	**Description**
15:0	TEMPERATURE	0000h	Sets the PCB temperature (in °C).

0x5F: POSITION, Bits[31:16]			
Bits	**Name**	**Default**	**Description**
15:0	POSITION	0000h	Sets the higher 16 bits of the motor shaft position, which is also the motor shaft rounds.

0x60: POSITION, Bits[15:0]			
Bits	**Name**	**Default**	**Description**
15:0	POSITION	0000h	Sets the lower 16 bits of the motor shaft position, which is also the motor shaft angle. The unit is in LSB, where 1 LSB = 360 / 65536°.

0x61: SPEED			
Bits	**Name**	**Default**	**Description**
15:0	SPEED	0000h	Speed feedback (lower 16 bits). The negative value is represented by a complement code. The speed feedback can be calculated with the following equation: SPEED, bits[31:0] = (rpm / 60) x 2^{32} / 10,000

0x63: IQ_LATCH			
Bits	**Name**	**Default**	**Description**
15:12	RESERVED	0h	Unused.
11:0	IQ_LATCH	000h	Quadrature axis current I_Q feedback. The negative value is represented by a complement code. The feedback can be calculated with the following equation: IQ_LATCH, bits[11:0] = I_Q x 1.5 x K_{AD} x 1024 / 1.6

0x64: ID_LATCH			
Bits	**Name**	**Default**	**Description**
15:12	RESERVED	0h	Unused.
11:0	ID_LATCH	000h	Direct axis current I_D feedback. The negative value is represented by a complement code. The feedback can be calculated with the following equation: ID_LATCH, bits[11:0] = I_D x 1.5 x K_{AD} x 1024 / 1.6

0x65: UQ_LATCH			
Bits	**Name**	**Default**	**Description**
15:12	RESERVED	0h	Unused.
11:0	UQ_LATCH	000h	Sets the quadrature axis voltage.

0x66: UD_LATCH			
Bits	**Name**	**Default**	**Description**
15:12	RESERVED	0h	Unused.
11:0	UD_LATCH	000h	Sets the direct axis voltage.

0x6A: STATUS			
Bits	**Name**	**Default**	**Description**
15:3	RESERVED	0h	Unused.
3	HOMING_END	0h	Flag that indicates whether homing is complete.
2	SPD_REACH	0h	Sets the speed reach target flag. See register 0x29 on page 21 for more details.
1	POS_REACH	0h	Sets the position reach target flag. See register 0x1E on page 20 for more details.
0	RUNNING	0h	Motor running flag

0x6C: VIN_SENSE			
Bits	**Name**	**Default**	**Description**
15:12	RESERVED	0h	Unused.
11:0	VIN_SENSE	000h	Sets the DC voltage ADC sample value. The V_{IN} sense can be calculated with following equation: VIN_SENSE, bits[11:0] = V_{DC} / 3.3 x (10 / 412) x 4096. Where V_{DC} is the DC link voltage.

Operating Commands

0x70: RUN			
Bits	**Name**	**Default**	**Description**
15:0	RUN	0000h	Forces the motor to stop or run. 1: Motor run 0: Motor stop

0x71: BRAKE			
Bits	**Name**	**Default**	**Description**
15:0	BRAKE	0000h	Forces the motor to brake or run. 1: Motor brake 0: Motor run

0x73: LOAD_TO_FLASH			
Bits	**Name**	**Default**	**Description**
15:0	LOAD_TO_ FLASH	0000h	Write 0x0002 to the load register values to the flash memory.

0x74: LOAD_FROM_FLASH			
Bits	**Name**	**Default**	**Description**
15:0	LOAD_FROM_ FLASH	0000h	Loads the register value from the flash memory.

0x76: UPDATE_COMMAND			
Bits	**Name**	**Default**	**Description**
15:0	UPDATE_ COMMAND	0000h	Write 0x0000 to update the speed/position command.

REVISION HISTORY

Revision #	Revision Date	Description	Pages Updated
1.0	10/5/2022	Initial Release	-