DC Integrated Torque Servo YZ-AIM_v2.56

I. Product Features

1. 15-bit absolute encoder, up to 32768 pulses per revolution.
2. Multi-turn absolute value (requires battery). Pulse mode: automatically returns to the position at power-off when powered on again. Communication mode: can record position after power-off.
3. Multi-stage DD motor structure, high torque output.
4. Integrated servo, simplified wiring, ultra-small size.
5. Low noise, low vibration, high-speed positioning, high reliability.
6. FOC field-oriented vector control, supports position/speed closed-loop.
7. Can operate in a zero-lag given pulse state, following with zero lag.
8. 16-bit electronic gear function.
9. Modbus RTU communication (19200, 8, N, 1), baud rate can be set.
10. Position mode, supports pulse + direction signal.
11. Features stall, over-current, and over-voltage protection.

II. Parameter Table

Model Parameters	42AIM15	42AIM10	
Power Supply	Voltage	24VDC±10%	24VDC±10%
	Current	2.2A	1.6A
Motor Parameters	Torque	0.48NM	0.33NM
	Rated Speed	1000RPM	1000RPM
	Max Speed	1500RPM	1500RPM
	Power	50W	35W
Feedback Signal	Single-turn 15-bit magnetic encoder (32768 pulses per turn)		
Cooling Method	Natural cooling		
Weight			
Position Control Mode	Max Input Pulse Frequency	500KHz	

	Pulse Command Mode	Pulse + Direction, A-phase + B-phase
	Electronic Gear Ratio	Setting range 1~65535 to 1~65535
	Position Sampling Frequency	2KHz
Protection Function	Stall alarm	
Communication Interface	RS485 (Modbus RTU 19200,8,N,1)	
Operating Environment	Ambient Temperature	0~40°C
	Max Motor Temperature	85°C
	Humidity	5~95%

II. Driver Interface

1. Power and Control Signal Interface

Terminal No.	Name	Function
1	+24V	DC power positive, +24V. Reversing polarity will directly short the power supply and may damage the driver.
2	GND	DC power ground. Reversing polarity will directly short the power supply and may damage the driver.
3	PU+ (+5V)	Pulse control signal: effective on the rising edge of the pulse; PU- high level 3.3#5V, low level 0#0.5V.
4	PU- (PU)	For reliable response to pulse signals, the pulse width should be greater than 1.2μs. A resistor is required if using +12V or +24V.

5	DIR+ (+5V)	Direction signal: high/low level signal. To ensure reliable motor direction change, the direction signal should be established at least 5μs before the pulse signal.
6	DIR- (DIR)	DIR- high level 3.3#5V, low level 0#0.5V.

Terminal number: Facing the terminal, the left is the first. The AIM series uses a differential interface circuit that can be applied to differential signals, single-ended common cathode and common anode interfaces, built-in high-speed optocouplers, allowing reception of long-line driver, open collector and PNP output circuit signals.

2. Communication and Output Interface

Terminal number: Facing the terminal, the bottom row from left to right is 12345, the top row from left to right is 6 7 8 9 10.

Terminal No.	Name	Function
1	NC	
2	485A	485 communication positive terminal
3	485B	485 communication negative terminal
4	EN+	Enable signal positive terminal. 3.3~5V, if the voltage is 24V, a 2K resistor needs to be connected in series.
5	EN-	Enable signal negative terminal.
6	COM	Output signal and 485 power common ground.
7	WR	Alarm signal output, internal optocoupler NPN output. Normally high impedance state, conducts with COM when alarming.
8	RDY/PF	Servo ready signal/ positioning complete signal.

		Signal is present (conducts) after power-on and automatic operation. Signal is present (conducts) when the following error is less than 0.5°, no signal (high impedance state) when the following error is greater than 0.5°.
9	ZO	Encoder zero point output. Optocoupler NPN output conducts when there is a zero point signal.
10	485_5V	485 communication 5V power supply, requires an external power supply. (This power is supplied through the controller)

3. Status Indication and Alarm

After power-on, the red and green lights both turn on once to check if the LEDs are working properly. Then the green light is on and the red light is off for the normal state. If an alarm state is encountered, the cause can be judged by the red light flashing, or the alarm code can be read through Modbus.

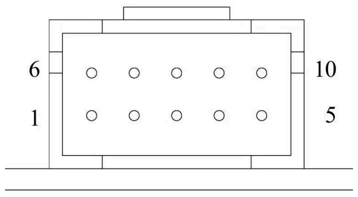
Alarm Code	Red Light Flashing	Alarm Cause	Alarm Handling
0x10	One long flash	Battery power loss alarm	Only a prompt, does not stop the machine.
0x12	One long flash, 2 short flashes	Overcurrent alarm	Stops the machine.
0x14	One long flash, 4 short flashes	Stall alarm	Stops the machine.
0x15	One long flash, 5 short flashes	Overvoltage alarm	Stops the machine. Deceleration of a large inertial load will generate electricity, which may cause an overvoltage alarm. A discharge module needs to be added to the power supply, or a large capacitor needs to be added to

			the power supply for energy storage.

Note: Stall alarm, the stall time can be set, see the register description for details.

III. Driver Wiring Diagram and Control Method

1. Typical Driver Wiring Diagram



2. Command Pulse + Direction Position Control Mode

If 3200 pulses per revolution are required

Electronic gear setting 32768 (encoder pulses per revolution) to 3200 (pulses per revolution to be set)

After simplification: 256 to 25

If 8192 pulses per revolution are required (default parameter)

Electronic gear setting 32768 (encoder pulses per revolution) to 8192 (pulses per revolution to be set)

After simplification: 4 to 1

Note: Simplify as much as possible. The numerator of the electronic gear is 32768, which is too large and will affect the following performance.

Command pulse frequency = (required motor speed in RPM / 60) * pulses per revolution

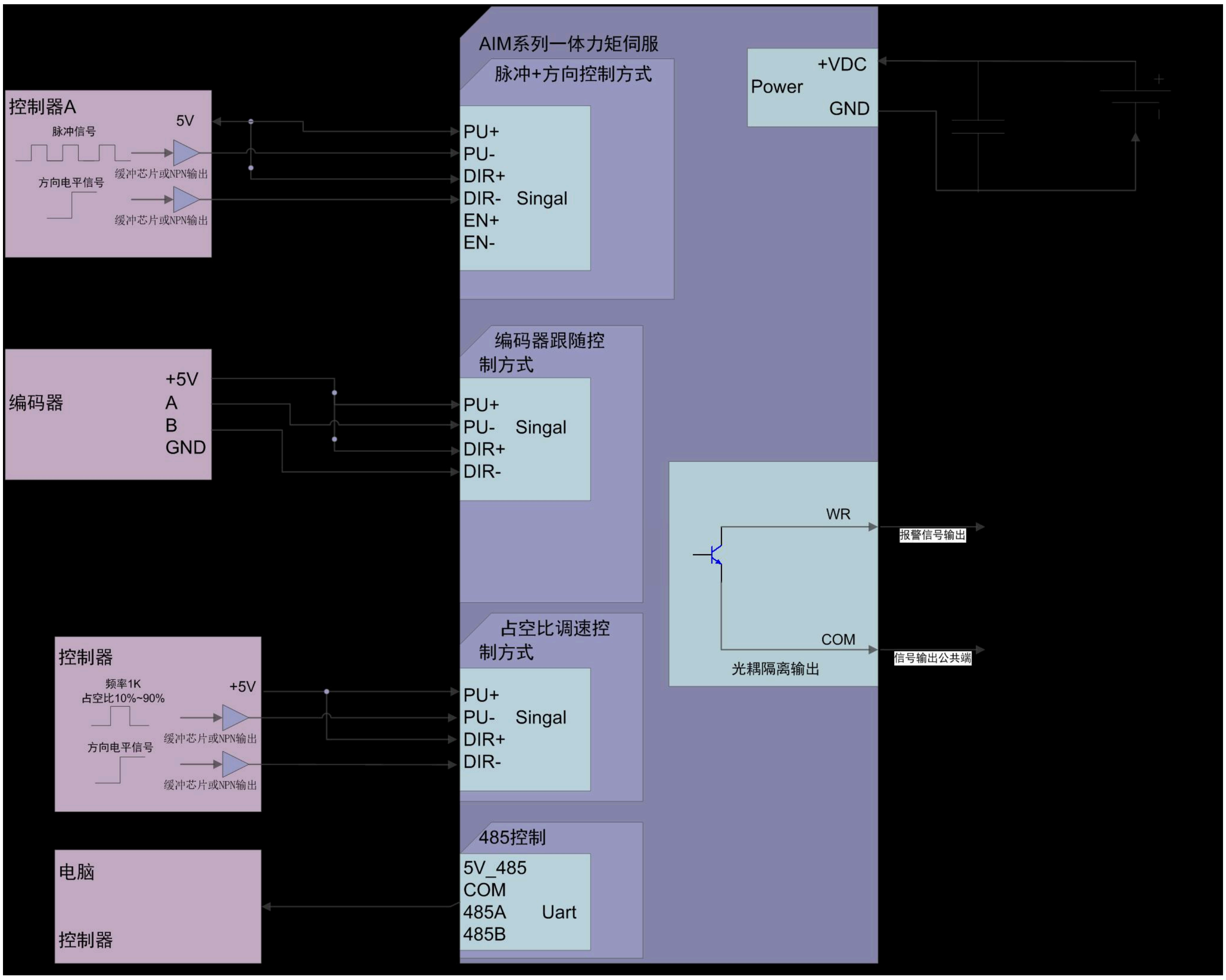
For example: required motor speed 1000 RPM, pulses per revolution 8192

Pulse frequency = 1000/60 * 8192 = 136533HZ

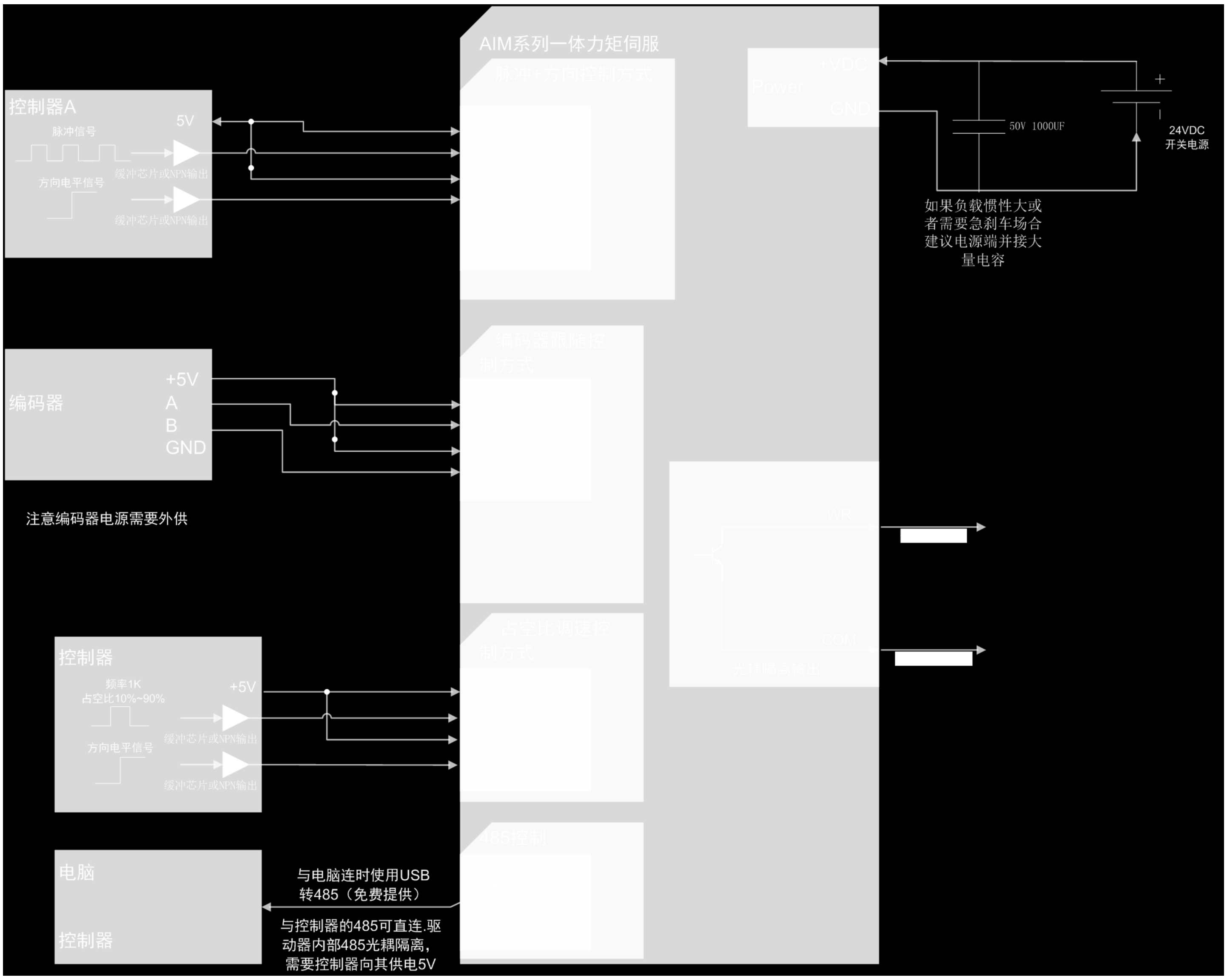
3. Orthogonal Command Pulse Position Control Mode

By setting the special function (address 0x19) to 2 and re-powering, it will be in encoder following mode. This mode can be used for encoder following. For example, if a shaft is connected to an encoder, connect the encoder output to the driver (wiring method as in the typical driver wiring diagram), and the driver can control the servo motor to follow the input encoder signal. The ratio of the control encoder to the motor rotation angle can be set by adjusting the electronic gear.

Forward pulse:



Reverse pulse:



Motor rotation direction: PU rising edge leads DIR rising edge for forward rotation. PU rising edge lags DIR rising edge for reverse rotation.

IV. Parameter Debugging

1. Internal Acceleration/Deceleration Curve

Whether to use the internal acceleration/deceleration curve is selected according to the different output signals of the controller.

Use internal acceleration curve: When the motor acceleration is less than 60000, the driver will enable the internal acceleration/deceleration curve, and the specific acceleration will be the same as the set value.

Use case: Using the internal acceleration curve will cause the phenomenon of lagging pulses. Some occasions that do not require real-time following can use the internal acceleration curve. Some controllers give pulses directly to the corresponding speed frequency without acceleration and deceleration. In this case, using the internal acceleration/deceleration curve can reduce the programming difficulty of the controller.

Disable internal acceleration curve: When the motor acceleration is greater than or equal to 60000, the driver allows acceleration and deceleration according to external pulses, and the internal acceleration is invalid.

Use case: For example, in an engraving machine, the pulses output by the controller have acceleration and deceleration, so the internal acceleration curve of the driver is not needed. If it is used at this time, it will lag behind the actual pulses.

2. Ball Screw Load

First, let's introduce torque. Let's use a 400W motor with 1.3NM. The load is a ball screw with a pitch of 5mm, which means that the load moves 5mm for one revolution of the motor shaft. In this case, Equivalent force arm of the load = 5mm / 3.14 = 1.592 mm

Then the thrust that the motor can provide is Thrust transmitted through the ball screw = 1.3NM / (1.592mm * 0.001) = 816 N

The weight of the load that can be pushed is about 80KG, which is vertical. It can be slightly larger for horizontal pushing.

Since the distance the load moves for one revolution of the motor with a ball screw load is short, the driver parameters (acceleration can be larger, such as 20000, position loop KP can be larger, such as 3000) can be set. Servo motors are most suitable for this type of load.

3. Belt Pulley Load

Servo motors are actually not very suitable for this type of load. Because the diameter of the belt pulley is generally relatively large, for example, 30mm in diameter. Then for one revolution of the motor, the distance the load moves is 30mm * π = 94.2, which is many times larger than the 5mm of the ball screw mentioned above.

Then the thrust that the motor can provide is Thrust transmitted through the belt = 1.3NM / (30mm * 0.001) = 43.3 N

The weight of the load that can be pushed is about 4.3KG. So servo motors are actually not suitable for connecting to synchronous wheels, because the distance the load moves for one revolution of the synchronous wheel is too long, and the force arm is long. If a servo motor is to be used in this situation, you can choose a synchronous wheel with as small a diameter as possible, or connect a small synchronous wheel to the motor shaft and a large synchronous wheel to the load end. This will reduce the speed by several times and achieve better results. In this case, the driver parameters (acceleration is set to be smaller, such as 5000) are set to reduce acceleration and deceleration because the equivalent inertia of the load is large.

4. Disk Load

This type of load cannot be driven directly by a servo motor, and a reducer is generally required. For example, a disk with a diameter of 200mm and a weight of 10KG. The radius is 100mm, and the equivalent radius of the weight is 50mm. The force arm is very large. If a servo motor is to be connected to this type of load, a reducer must be connected first.

If the disk is not particularly heavy, you can sacrifice some positioning accuracy and rigidity to control it. The specific method is to set the motor acceleration to a relatively small value, for example, around 1000. Set the speed KI to 2000 to cancel the integral action. Change the position KP to 1000. With these parameters, a general disk load can also be used.

5. Automatic Single-Turn Origin Search Function

The automatic origin search function is selected by changing the parameters of register address 0x19 (special function). If you need to automatically find the origin after power-on, the setting method is as follows:

modbus enable send 1

Special function (address 0x19) send 10~32768 (32768 corresponds to 360° of the motor)

Parameter save send 1

It will automatically find the origin after re-powering. Since it is an absolute encoder, it can automatically find any position in one revolution after power-on. (The DIR polarity is 1 or 0 to set the direction of origin search)

6. Automatic Mechanical Origin Search Function

The automatic mechanical origin search function is selected by changing the parameters of register address 0x19 (special function). If you need to automatically find the mechanical origin after power-on, the setting method is as follows:

modbus enable send 1

Special function (address 0x19) send 1 (at this time it will automatically find the mechanical origin immediately)

Parameter save send 1 (if you need to automatically find the mechanical origin after re-powering, you can save this parameter)

After re-powering, it will automatically reverse until the motor stalls, and then the motor will reverse 36° as the origin. (The DIR polarity is 1 or 0 to set the direction of origin search)

7. Automatic Origin Search Function (Origin Switch)

The automatic mechanical origin search function is selected by changing the parameters of register address 0x19 (special function). If you need to automatically find the origin after power-on, the setting method is as follows:

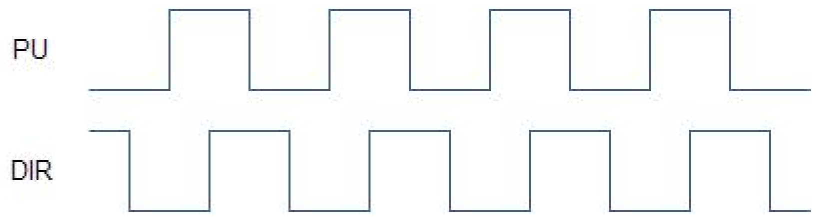
modbus enable send 1

Special function (address 0x19) send 8 (at this time it will automatically find the mechanical origin immediately)

Parameter save send 1 (if you need to automatically find the mechanical origin after re-powering, you can save this parameter)

After re-powering, it will automatically reverse until there is an EN signal, and then the motor will forward rotate to the encoder origin as the origin. (The DIR polarity is 1 or 0 to set the direction of origin search)

The origin switch needs to be connected to the EN signal. The wiring diagram is as follows:



8. Clear Position by Communication Method

Clear absolute position: If you need to clear the absolute position to 0 during operation, first send 0 to the numerator of the electronic gear (the electronic gear is invalid in communication mode and is used for this special function. If it is communication control, you can directly save the numerator of the electronic gear as 0), then send 0 to the absolute position (0x16), and the absolute position will be cleared to 0 directly.

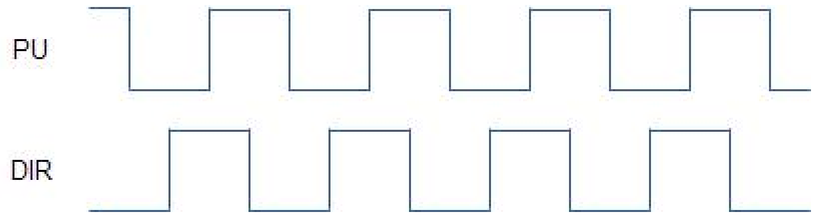
Emergency stop: In communication mode, if there are many remaining pulses to go and an emergency stop is required. First send 0 to the numerator of the electronic gear (the electronic gear is invalid in communication mode and is used for this special function. If it is communication control, you can directly save the numerator of the electronic gear as 0, and then send 0 to the target position (0x0C) to perform an emergency stop. There is also a small deceleration distance for an emergency stop, and the length of the deceleration distance is controlled by the position loop KP.

9. Default to Communication Control on Power-up

As long as the numerator of the electronic gear is set to 0 and saved, after re-powering, the modbus enable is 1 by default.

V. Modbus Control Method

1. Hardware Connection



The internal 485 of the driver is isolated by optocouplers, which solves the problem of a host being easily interfered with and damaged when connected to multiple slaves.

2. Register Description

The driver can be controlled by Modbus (RTU mode). The host can set driver parameters and control operation by reading and writing registers of Modbus. The function codes supported by the driver are 0x3 (read register), 0x6 (write register), 0x78 (write target position), 0x7a (modify device address).

The register list is as follows:

Address	Parameter Name	Read-only/Read-write	Parameter Range	Parameter Description
0x00	Modbus Enable	Read-write	0~1	0: Modbus disabled 1: Modbus enabled
0x01	Driver Output Enable	Read-write	0~1	0: Driver output disabled 1: Driver output enabled
0x02	Motor Target Speed	Read-write	0~3000 r/min	In speed mode, target speed. In position mode, maximum speed
0x03	Motor Acceleration	Read-write	0~60098 (r/min)/s	When the parameter is less than 60000, the driver generates an acceleration/deceleration curve internally. When the parameter is equal to 60000, there is no acceleration process, and the deceleration is determined by the position KP. When it is greater than 60000 to 60098, the units and tens digits 0~98 correspond to position feedforward 0%~98%. The larger the position feedforward, the smaller the pulse following lag.

0x04	Field Weakening Angle	Read-write	0~306 r/min	Internal parameters do not need to be set separately.
0x05	Speed Loop Proportional Coefficient	Read-write	0~10000	Represents 0.0~10.0. The larger the value, the stronger the rigidity. If the units digit is even: the pulse input polarity is effective at the moment of disconnection. If the units digit is odd: the pulse input polarity is effective at the moment of conduction.
0x06	Speed Loop Integral Time	Read-write	2~2000 ms	Integral time 2~2000ms. The smaller the value, the stronger the rigidity.
0x07	Position Loop Proportional Coefficient	Read-write	60~30000	Position KP, the larger the value, the stronger the rigidity. If the units digit is even: the alarm output is normally open (normally open, normally closed on alarm). If the units digit is odd: the alarm output is normally closed (normally closed, normally open on alarm).

0x08	Speed Feedforward	Read-write	0~12.0V/KRPM	327 represents 1V/KRPM, no need to set it yourself.
0x09	DIR Polarity	Read-write	0~1	0: External DIR is not conducting, clockwise rotation. 1: External DIR is conducting, clockwise rotation.
0x0A	Electronic Gear Numerator	Read-write	0~65535	16-bit electronic gear numerator. If the electronic gear numerator is 0, special functions can be realized. See the previous description for details.
0x0B	Electronic Gear Denominator	Read-write	1~65535	16-bit electronic gear denominator.
0x0C	Target Position Low 16 bits	Read-only		High 16 bits of the number of steps to go.
0x0D	Target Position High 16 bits	Read-only		Low 16 bits of the number of steps to go.
0x0E	Alarm Code	Read-only		
0x0F	System Current	Read-only	0~32767	Actual current is x/2000(A).
0x10	Motor Current Speed	Read-only	-30000~30000 r/min	Actual motor speed = Motor current speed / 10.

0x11	System Voltage	Read-only	0~32767	Actual voltage is x/327(V).
0x12	System Temperature	Read-only	0~100	Degrees Celsius.
0x13	System Output PWM	Read-only	-32768~32767	Represents -100%~100%.
0x14	Parameter Save Flag	Read-write	0~1	0: Parameters not saved. 1: Saving parameters. 2: Saving complete.
0x15	Device Address	Read-only	0~255	Device address.
0x16	Absolute Position Low 16 bits	Read-write		High 16 bits of the number of steps taken.
0x17	Absolute Position High 16 bits	Read-write		Low 16 bits of the number of steps taken.
0x18	Maximum allowable output at rest	Read-write	0~609	0~609 corresponds to the maximum allowable output of 0~60.9%. The units digit 1~9 corresponds to the stall alarm time. Units digit 0 means no stall alarm, and the output is automatically reduced to 1/2 of the original after 3S of stalling.
0x19	Special Function	Read-write	0~100	0: Pulse + direction mode. 1: Automatically find the mechanical origin and rotate forward 36°

				(automatically reverse to the mechanical zero point on power-up, and then move forward 36° and stop). 2: Encoder following mode. 3: Speed mode, duty cycle speed regulation (10%~90% corresponds to 0~1000RPM). 10~32768: The angle to which it automatically turns on power-up. The algorithm is: X*360°/32768. 4: Automatically find the mechanical origin and rotate forward to the encoder zero point (automatically reverse to the mechanical zero point on power-up, and then move forward to the encoder zero point and stop). 5: Z_OUT analog origin signal mode: On power-up, give the motor a pulse to control the motor to slowly reverse until it stalls. Zout outputs a conduction

				signal, and then the controller gives the motor a pulse to rotate forward until the ZOUT signal disappears, which is the system origin (Zout disappears after the encoder zero point appears once, which can ensure the accuracy of the origin). 8: Automatically find the origin (origin is connected to the EN signal). On power-up, it automatically reverses until the EN has a signal, and then rotates forward until the EN has no signal and stops.

3. Modbus Communication Format

a. modbus host read data and slave response format (function code 03)

Host read data Format

Device address Function code First register high byte address First register low byte address Number of registers high byte Number of registers low byte CRC high byte CRC low byte

0x01 0x03 0x00 0x00 0x00 0x01 0x84 0x0a

Slave response

Device address Function code Data length First data high byte First data low byte CRC high byte CRC low byte

0x01 0x03 0x02 0x00 0x01 0x79 0x84

The data received by the serial port are all unsigned numbers. If the register is a signed number, the binary complement format is sent. The algorithm for converting to a signed number is as follows (VB code):

```
If modbus.data(11) > 32767 Then disp_modbus_data.PU = (modbus.data(11) - 32768) *
65536 + modbus.data(10) disp_modbus_data.PU = -((&H7FFFFFFF - disp_modbus_data.PU) +
1) Else disp_modbus_data.PU = dmodbus.data(11) * 65536 + modbus.data(10) End If
```

Note: modbus.data(11) is the high 16 bits of the target position modbus.data(10) is the low 16 bits of the target position

b. modbus host write data and slave response format (function code 06)

Host write data Format

Device address Function code First register high byte address First register low byte address Data high byte Data low byte CRC high byte CRC low byte

0x01 0x06 0x00 0x00 0x00 0x01 0x48 0x0a

Slave response Format

Device address Function code First register high byte address First register low byte address Data high byte Data low byte CRC high byte CRC low byte

0x01 0x06 0x00 0x00 0x00 0x01 0x48 0x0a

c. modbus host write pulse number (function code 0x10)

Host writes double-byte data (write PU pulse number)

Device address Function code First register high byte address First register low byte address Number of registers high byte Number of registers low byte Data length

0x01 0x10 0x00 0x0c 0x00 0x02 0x04

PU:8~15 bits PU:0~7 bits PU:24~31 bits PU:16~23 bits CRC high byte CRC low byte

0x27 0x10 0x00 0x00 0xf8 0x8b

The number of pulses is a signed number. The algorithm for converting a negative number (assuming this number is X) into a 32-bit hexadecimal number is as follows (vb code):

```
If X < 0 Then X = &H7FFFFFFF + (X + 1) PU24_31 = Fix(X / (256 * 65536)) + &H80 Else
PU24_31 = Fix(X / (256 * 65536)) End If PU16_23 = Fix(X / 65536) mod 256 PU8_15 =
Fix(X / 256) mod 256 PU0_7 = X mod 256
```

Note: fix() is the rounding function

Slave response Format

Device address Function code First register high byte address First register low byte address Number of registers high byte Number of registers low byte CRC high byte CRC low byte

0x01 0x10 0x00 0x0c 0x00 0x02 0x81 0xcb

d. modbus host write incremental pulse number (special function code 0x78)

Host special function code 0x78 format (write PU pulse number)

Device address Function code PU:24~31 bits PU:16~23 bits PU:8~15 bits PU:0~7 bits CRC high byte CRC low byte

0x01 0x78 0x00 0x00 0x27 0x10 0xbb 0xfc

Slave response Format

Device address Function code PU:8~15 bits PU:0~7 bits PU:24~31 bits PU:16~23 bits CRC high byte CRC low byte

0x01 0x78 0x27 0x0e 0x00 0x00 0xca 0xb7

e. modbus host write absolute position (special function code 0x7b)

Host special function code 0x78 Format (write PU pulse number)

Device address Function code PU:24~31 bit PU:16~23 bit PU:8~15 bits PU:0~7 bits CRC high byte CRC low byte

0x01 0x7b 0x00 0x00 0x27 0x10 0xff 0xfc

Slave response Format

Device address Function code PU:8~15 bits PU:0~7 bits PU:24~31 bit PU:16~23 bit CRC high byte CRC low byte

0x01 0x7b 0x27 0x10 0x00 0x00 0xee 0xb1

4. Modbus Synchronous Control of Multiple Motors

Synchronously control up to 100 servo motors via modbus. The communication format is as follows:

Modbus-rtu write command

Host (computer or other host) writes:

Command format

1BYTE 1 BYTE 2BYTE 2BYTE 1BYTE Number of registers*8 BYTE 2BYRE

Device address Function code Register address Number of registers Data length Data content CRC

0x00 0x10 0x00 0x16 0x00 0x10 0xXX See table below HI LO

Device address: The default is 0, so all address motors can receive it.

Function code: 0x10 is compatible with standard function codes and supports as many standard devices as possible.

Register address: The default is 0x16, which is the address of the absolute position.

Number of registers: If controlling 1 motor, it is 0x4, if 2 motors, it is 0x8, and so on.

Data length: Any number can be sent.

Data content

4byte 2byte 2byte 4byte 2byte 2byte ……

1-axis position 1-axis speed 1-axis acceleration 2-axis position 2-axis speed 2-axis acceleration ……

32-bit number 16-bit number 16-bit number 32-bit number 16-bit number 16-bit number ……

[15~8] [7~0] [31~24] [23~16] [15~8] [7~0] [15~8] [7~0]

Data content: Each motor will intercept its corresponding data according to its own address. If it is a motor with address 1, it is the first 8 bytes, and the content is as shown in the table above.

1-axis position: The absolute position of motor with address 1.

1-axis speed: The target speed of motor with address 1.

1-axis acceleration: The acceleration of motor with address 1.

5. CRC Check Example Code

```
unsigned short CRC16(unsigned char *puchMsg, unsigned short usDataLen) { unsigned
char uchCRCHi = 0xFF ; unsigned char uchCRCLo = 0xFF ; unsigned uIndex ;
while (usDataLen--) { uIndex = uchCRCHi ^ *puchMsgg++ ; uchCRCHi = uchCRCLo ^
auchCRCHi[uIndex] ; uchCRCLo = auchCRCLo[uIndex] ; } return (uchCRCHi << 8 |
uchCRCLo) ; }
```

6. Modbus Host Control Process

a: Position mode

Set the DIP switch SW1 to OFF and then power on to enter position mode.

First, you can use the host computer software we provide to set the following parameters:

1. Modbus enable Send 1 (you can only change other parameters when modbus enable is 1, and the external pulse signal is invalid.)

HEX source command: 01 06 00 00 00 01 48 0A

2. Motor acceleration Send 5000 (set the acceleration according to actual needs, if not set, the default parameter 20000 will be used)

HEX source command: 01 06 00 03 13 88 74 9C

3. Target speed Send 1500 (set the running speed according to the actual running needs, if not set, the default parameter 2800 will be used)

HEX source command: 01 06 00 02 05 DC 2A C3

4. Electronic gear numerator Send 0 (after the electronic gear numerator is saved as 0, the mdobus enable is 1 by default next time it is powered on)

HEX source command: 01 06 00 0A 00 00 A9 C8

5. Parameter save flag Send 1 (after sending this parameter, the previously set parameters are saved internally)

HEX source command: 01 06 00 14 00 01 08 0E

6. Re-power on and see if the parameters have been saved correctly. The above settings can be done with the provided host computer software, and the HEX source code does not need to be sent through the serial port by yourself.

After the parameters are set, you can send position commands through PLC or single-chip microcomputer, or your own designed host computer software. To send a position command, you only need to send the position to be moved through the 0x10 command.

1. Send incremental position (the meaning of incremental position is that the sent data is the position that the motor needs to move forward or backward)

For example, if you need to move forward one circle (assuming the motor encoder is a 1000-line encoder, the number of pulses per circle is 4000)

HEX source command: 01 10 00 0C 00 02 04 0F A0 00 00 F0 CC

For example, if you need to move forward and backward one circle (assuming the motor encoder is a 1000-line encoder, the number of pulses per circle is -4000) -4000 The binary calculation method is as follows: The binary of 4000 is 00 00 0F A0.

(Note: 0 = FF FF FF FF +1)

-4000 is 0 - 00 00 0F A0 =FF FF FF FF - 00 00 0F A0 +1=FF FF F0 5F +1 = FF FF F0 60

HEX source command: 01 10 00 0C 00 02 04 F0 60 FF FF C1 54

2. Send absolute position (the meaning of absolute position is that the position is defined as 0 when it is just powered on or the absolute position is cleared or the origin is automatically found. The absolute position is to go to the newly sent position. For example, the first time you send 4000, it is one circle. The second time you send it, it has already reached the position of 4000. If you send the same command again, the motor will not move)

For example, if you need the motor to go to the position of 2 circles (assuming the motor encoder is a 1000-line encoder, the number of pulses for 2 circles is 8000)

HEX source command: 01 10 00 16 00 02 04 1F 40 00 00 74 89

For example, if you need the motor to return to the origin (when the numerator of the electronic gear is 0, sending 0 is to clear the current position, so send 1 to return to the origin. At this time, a pulse will not affect the accuracy)

HEX source command: 01 10 00 16 00 02 04 00 01 00 00 23 49

Note: To control the motor, you only need to send the required position first (try to use the absolute position command, because it can be sent repeatedly, and it will still go to the same position). Then you can judge whether to execute the next command by reading the absolute position and comparing whether it has reached the set position (note that you need to allow an error of +-2 when judging). Or you can connect the PF signal. After reaching the position, the driver will give an optocoupler output switch signal.

The command to read the absolute position is as follows: 01 03 00 16 00 02 25 CF

7. Modify Baud Rate

The baud rate can be modified by sending it through the host computer software we provide. You must follow the steps below:

A. Modbus enable (address 0) Send 1

B. Motor acceleration (address 3) Send 803 (Note 803:115200 802:38400 801:19200 800:9600)

C. Field weakening angle (address 4) Send 129

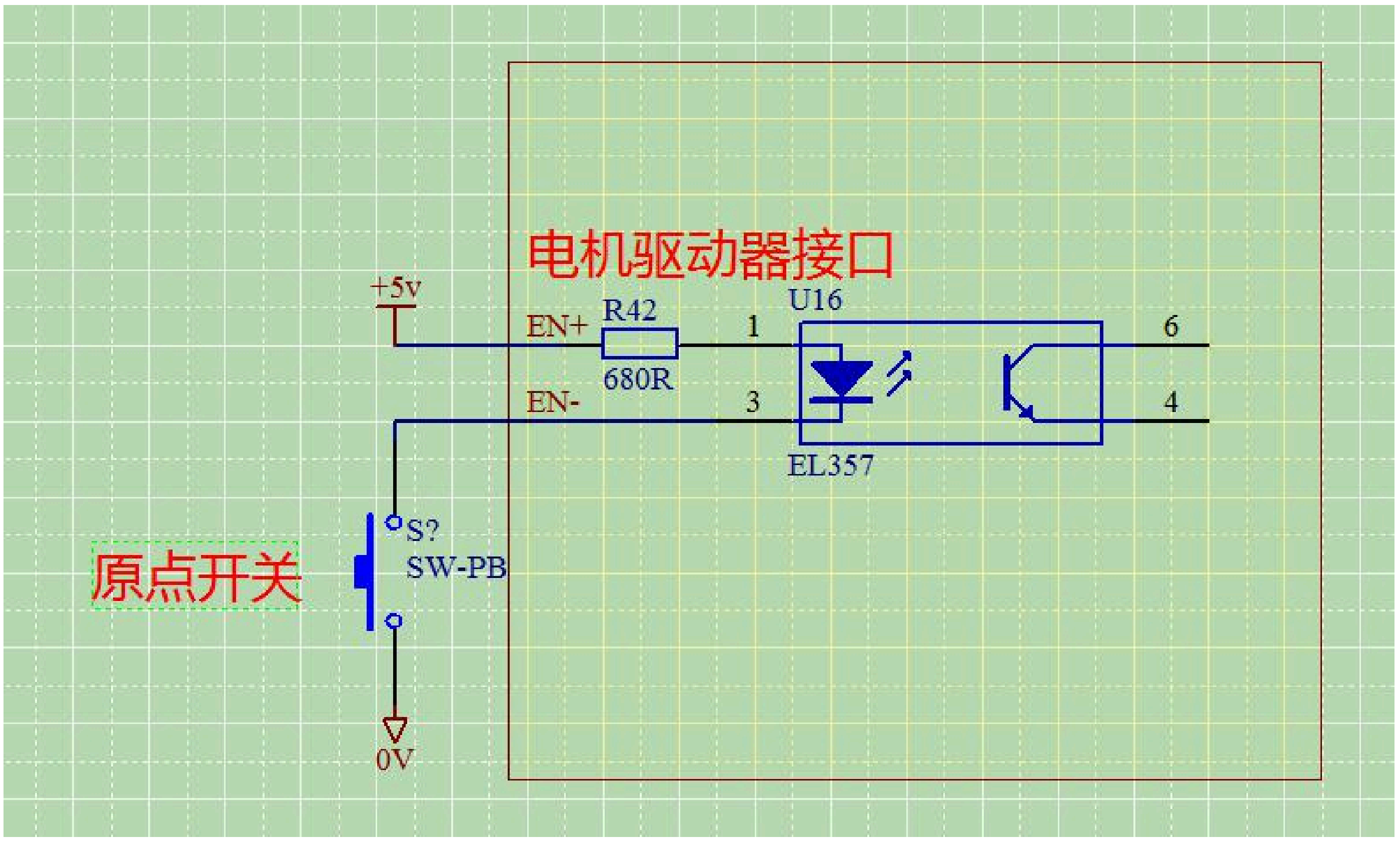
D. Modbus enable (address 0) Send 506

It will take effect after re-powering.

Note: You do not need to send a parameter save, because this is to modify internal parameters. Just follow the steps above strictly.

VI. Host Computer Software User Manual

This driver provides a host computer software for monitoring and testing the driver. You can view and set the internal parameters of the driver through the software.



As shown in the figure above, the software is divided into several parts such as waveform display and motor running parameters. The functions and roles of each part are introduced below.

Waveform display: There are 4 channels in total, represented by 4 colors. The colors are the same as the font colors in the motor running parameters. That is: blue represents current, green represents the output pulse width, red represents the current speed, and black represents voltage.

Motor running parameters: displays the real-time data of the motor's operation.

Driver setting parameters: displays the DIP switch of the driver and the direction enable setting. If it is in modbus mode, this column is invalid.

Driver running status: This column will display the alarm status of the driver. If there is no alarm, it will display that it is running normally.

Modbus control parameters: The parameters in this column are the internal parameters of the driver. If you want to modify these parameters, you must first write 1 to the modbus enable. For the meaning of specific parameters, please refer to the register description.

Modbus read: This column can set the address of the driver, the period of reading driver data, and whether to read.

Modbus send: This column is used to modify the driver parameters. First select the parameter type, then set the parameter data, and then click send.