Modbus-RTU

manual

Shenzhen Just Motion Control Electromechanics Co., Ltd.

Preface

Shenzhen Just Motion Control Electromechanics Co., Ltd.

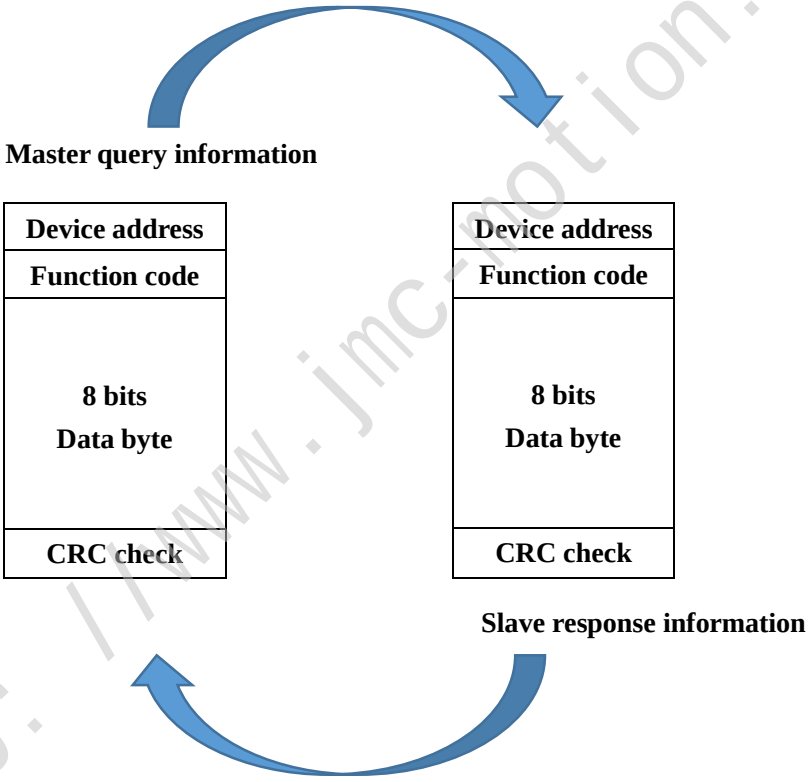
version	writer	approval
V2.2	jevon	R & D department

Catalog

1 Modbus-RTU Protocol

Modbus protocol, designed by MODICON, is a bus protocol that allows the master to share data with one or more slaves. The data consists of 16-bit registers. Master can R/W single register or multiple registers. This chapter describes the use of ModBus serial communication protocol based on Shenzhen Just Motion Control Electromechanics Co., Ltd., the communication protocol is the Modbus-RTU standard protocol.

Serial communication port is to use RS-485 compatible serial interface that defines the connector, the wiring cable, the signal level, the transmission baud rate and parity check. The controller communication uses the master and slave technology, that is, the master can start the data transmission and request. Other devices (slaves) response to the request or process the action requested. Master device should include master-slave processors or PLC. Slaves include servo drives and stepper drives. The master-slave query - feedback mechanism is as follows:



2 Modbus-RTU message format

Modbus-RTU is a master-slave technology, and the CRC check range from the device address bits to the data bits; detailed message format of the function code, see the appendix. Modbus-RTU message frame is as follows:

Address field	**function code**	**Data**	**CRC check code（2 byte）**

3 Modbus-RTU wiring

The Modbus-RTU has a common physical layer with standard RS-485 and can be configured with 1 to 32 slave addresses. The topological structure of the RS-485 network is constructed,Usually in the last slave device parallel 120 ohm termination resistor. JMC's Modbus-RTU bus driver supports only RS-485 two-wire half-duplex wiring.

4 transfer protocol

Message processing:

As a slave, the drive will wait for receiving the data sent by the master station, and when receiving the data, it is judged whether it is the data of own slave and responds.. When the time out 3.5T times did not receive the next frame of data, the slave drive default error data received, the previous data cleared, re-receive new data.

Message frame structure:

The message exchanged by "master-slave" starts with the slave address, followed by the function code , Transmission of data. The structure of the data field depends on the function code used. The CRC check code is transmitted at the end of the message frame.

Slave address	Function	Data	CRC check
1 byte	1 byte	N byte	2 byte

Address	Modbus Slave address 1~32
Function	Modbus function code
Data	Modbus data：Register address, the number of register addresses,the data of register
CRC check	Message frame checksum

Broadcast message：

The master uses slave address 0 to address all slaves on the bus.

Broadcast messages are only allowed with write function code 0x06 and 0x10.

Broadcast messages do not require reply message frames from slave.

Support the following Modbus function code:

1) 0x03：read hold register;
2) 0x06：write a single register;
3) 0x10：write multiple registers.

Data field DATA：

Data field DATA is used to transmit function code and specific data, for example:
Bytes number, start address of register, read and write address number, data value and so on.

CRC check：

The last frame of the message is a CRC 16 checksum of 2 bytes. The checksum is calculated as follows:$X^{16} + X^{15} + X^2 + 1$. The lower byte is transmitted first, followed by the upper byte.

Error response:

When detecting the master request message error, such as the receive register address is invalid, will set the highest bit of the function code to 1.
Subsequent transmission is a one-byte error code describing the error.

The error code and its description:

Error code	Conforms to the definition of ModBus specification	Description
1	Illegal function	Function code is illegal
2	Illegal data address	Slave has an illegal data address
3	Illegal data value	Slaves have illegal data values
4	The associated device has failed	An internal error from the slave
5	Write the read-only register	Can not execute the corresponding write operation
6	Busy, reject the message	The slave is not ready to receive the message
7	Read the write-only register	Can not perform the corresponding read operation
8	Data write error	The number of bytes does not match the number of registers
9	Parity error	Parity error (with parity configuration)
10	CRC check error	An error occurred in the CRC check, requesting retransmission of data
11	R / W register does not exist	The R / W register address does not exist, can not be completed

5 communication protocol

0x1001 error register

The device's internal error will be mapped to this register.

Register	Data type	Access type	Default value
1001h	UNSIGNED16	RO	0

Bit 0: generic error
Bit 1: current error
Bit 2: voltage error
Bit 3: temperature error
Bit 4: communication error
Bit 5: over position error protection (Only stepper servo and servo drive)
Bit 6: Reserve (Default 0)
Bit 7: Motor phase loss (Stepping servo drive)

0x1008 Manufacturer Device name

Device manufacturer and specific model

Register	Data type	Access type	Default value
1008h	Vis-String	RO	JMC XXX

0x1009 Manufacturer hardware version

Device hardware version number

Register	Data type	Access type	Default value
1009h	Vis-String	RO	XXX

0x100A Manufacturer software version

Device internal software version number

Register	Data type	Access type	Default value
100Ah	Vis-String	RO	JMC XXX

0x6000 Format register

32-bit register storage format, when 0x6000=0, 32-bit register the high 16-bit in the previous address, 16-bit lower in the next address; when 0x6000=1, 32-bit register the low 16-bit in the previous address, the high16-bits in the next address.

Register	Data type	Access type	Default value
6000h	UNSIGNED16	RO	0

0x6040 control word

The control word for the drive.Used to enable or disable the drive power and brake output, start and stop the motor in different operating modes, clear the error alarm and so on.

Structure of the control word description:

Register	Data type	Access type	Default value
6040h	UNSIGNED16	WR	0

Bit definition of control word：

Byte	BIT	Bit definition			
		Position mode	Velocity mode	Homing mode	Torque mode
LSB	0	0—>1：Parameter and variable initialization（Brake shut down）.			
	1	0—>1：Drive power supply（Brake open）.			
	2	0—>1：quick stop			
	3	0—>1：Motor enable 1—>0：Motor enable disable			
	4	0—>1：Position sampling	Reserve	Start homing operation	Reserve
	5	0：Complete the current position and then execute the next position; 1：Run directly to the next given position；	Reserve	Reserve	Reserve
	6	0：Absolute positioning； 1：Relative positioning；	Reserve	Reserve	Reserve
	7	0—>1：Reset and clear error			
MSB	8	0—>1：Halt 1—>0：normal operation			
	9	0：Finish the previous position stop and run the next position； 1：The current position and the next position do not stop；	Reserve	Reserve	Reserve
	10	Reserve			
	11	Reserve			
	12	Reserve			
	13	Reserve			
	14	Reserve			
	15	Reserve			

Control word state switching command：

Command	The bit of the Control word				
	Error reset bit8	Enable operation bit3	Quick stop bit2	Enable voltage bit1	Initialization device bit0
Shut down	0	X	1	1	0
Initialization	0	0	1	1	1

device					
Enable device operation	0	1	1	1	1
Quick stop	0	X	0	1	X
Forbid enable	0	0	1	1	1
Error reset	0 **à** 1	X	X	X	X

0x6041 Status word

The status word can only be read, reflecting the current status of the drive. Its structure is defined as follows:

Register	Data type	Access type	Default value
6041h	UNSIGNED16	RO	0

Bit definition of status word：

<table>
<tr><td rowspan="2">Byte</td><td rowspan="2">BIT</td><td colspan="7">Bit definition</td></tr>
<tr><td colspan="2">Position mode</td><td colspan="2">Velocity mode</td><td colspan="2">Homing mode</td><td>Torque mode</td></tr>
<tr><td rowspan="8">LSB</td><td>0</td><td colspan="7">0：Unprepared initialization 1：Ready to initialize</td></tr>
<tr><td>1</td><td colspan="7">0：Initialization drive is not completed 1：Initialization drive completion</td></tr>
<tr><td>2</td><td colspan="7">0：Motor enable disable 1：Motor enable</td></tr>
<tr><td>3</td><td colspan="7">0：Drive normal status 1：Drive error status</td></tr>
<tr><td>4</td><td colspan="7">0：Drive not work 1：Drive work normally</td></tr>
<tr><td>5</td><td colspan="7">0：Normal operation 1：Quick stop</td></tr>
<tr><td>6</td><td colspan="7">0：Normal operation 1：Device enters the initialization state</td></tr>
<tr><td>7</td><td colspan="7">0：Normal operation 1：Warning</td></tr>
<tr><td rowspan="8">MSB</td><td>8</td><td colspan="7">0：Normal operation 1：Motor halt</td></tr>
<tr><td>9</td><td colspan="7">0：motor is stoped 1：Motor is running</td></tr>
<tr><td rowspan="4">10</td><td rowspan="2">0：Not arrived</td><td rowspan="2">0</td><td>Bit8=0:Not reaching the target velocity</td><td rowspan="2">0</td><td>Bit8=0:Not reaching the homing position</td><td rowspan="4">Reserve</td></tr>
<tr><td>Bit8=1:Deceleration</td><td>Bit8=1:Deceleration</td></tr>
<tr><td rowspan="2">1：arrived</td><td rowspan="2">1</td><td>Bit8=0:reaching the target velocity</td><td rowspan="2">1</td><td>Bit8=0:reaching the homing position</td></tr>
<tr><td>Bit8=1:The velocity is 0</td><td>Bit8=1:The speed is 0</td></tr>
<tr><td>11</td><td colspan="7">SW mechanical origin limit</td></tr>
<tr><td rowspan="3">12</td><td>0 ： can’t set new position</td><td colspan="2">0：The velocity is not 0</td><td colspan="2">0：Homing operation not complete</td><td rowspan="3">Reserve</td></tr>
<tr><td>1 ： can set new position</td><td colspan="2">1：The velocity is equal to 0</td><td colspan="2">1 ： Homing operation complete</td></tr>
<tr><td rowspan="2">13</td><td>0：Normal operation</td><td colspan="2">0 ： No maximum acceleration</td><td colspan="2">0：Homing operation not error</td></tr>
<tr><td>1：over position error</td><td colspan="2">1:maximum acceleration</td><td colspan="2">1：Homing operation error</td></tr>
</table>

	14	CW clockwise direction limit
	15	CCW counter clockwise direction limit

The status word indicates the state of the device：

Status word value（Binary）	State
XXXX XXXX X0XX 0000	Uninitialized device
XXXX XXXX X01X 0001	Initialized device
XXXX XXXX X01X 0011	Device power on and operation
XXXX XXXX X01X 0111	Device enable
XXXX XXXX X00X 0111	Quick stop active
XXXX XXXX X0XX1111	Device error alarm occurs
XXXX XXXX X0XX1000	The device is in an error state

0x605A Quick stop code

The stop code determines how to stop when the command is quickly stopped

Register	Data type	Access type	Default value
605Ah	INTEGER16	RW	0

Quick stop code	Perform the operation
1	Stop with the current deceleration
2	Stop with fast stop speed
3...32767	Stop immediately

0x605D Halt code

The halt code determines how to halt when the command is suspended.

Register	Data type	Access type	Default value
605Ah	INTEGER16	RW	0

Halt code	Perform the operation
1	halt with the current deceleration
2	halt with fast stop speed
3…32767	halt immediately

0x6060 Modes of operation

Operating mode is used to select the appropriate sport mode

Register	Data type	Access type	Default value
6060	INTEGER16	WO	0
Mode of operation		operation	
1		Position mode	
3		Velocity mode	
4		Torque mode（serve）no use	

6	Homing mode

The device supports three modes: velocity mode, position mode and homing mode.

0x6061 **Mode code response**

The mode code response indicates the current operating mode. The return value is related to the corresponding mode status. (index 6060h)

Register	Data type	Access type	Default value
6061	INTEGER16	RO	0

0x6064 Position actual value

The current position represents the position of the current moment in the position mode.

Register	Data type	Access type	Default value
6064	INTEGER32	RO	0

0x606C Velocity actual value

The current speed indicates the speed of the current time.

Register	Data type	Access type	Default value
606C	INTEGER32	RO	0

Unit rps / min unit.

e.g .: If the value of index 606C is 100, it means the current speed is 10rps.

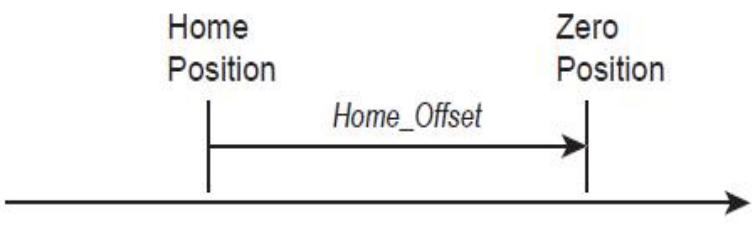
0x607A Target position

The target position is the position that the drive should move in position mode. The related parameters are the target velocity, acceleration and deceleration. The target position is related to different subdivisions and can be treated as a calculation or related parameter according to the bit6 of the control word.

Register	Data type	Access type	Default value
607A	INTEGER32	RW	0

0x607C Home offset

Home offset indicates the offset position between the home position and the zero position. After finding the mechanical origin, all the parameters will be cleared after a short distance from the mechanical far point. This is shown below.



Register	Data type	Access type	Default value
607C	INTEGER32	RW	0

0x6081 Target velocity

The target speed is the speed at which the drive runs at a constant speed

Register	Data type	Access type	Default value
6081	INSIGNED32	RW	0

This object is consistent with the actual speed (index 0x606C) and operating speed (0x60FF) units.The setting value is 10 times the actual value, for example: the value is 100, the actual speed value is 10r / s.

0x6083 Acceleration

Motor running acceleration

Register	Data type	Access type	Default value
6083	UNSIGNED16	RW	0

The unit of this value is rps / s. The drive internally divides this value by 10.

e.g .:Set this value to 10 rps / s, which the drive internally sees as 1 rps / s.

0x6084 Deceleration

Motor running deceleration

Register	Data type	Access type	Default value
6084	UNSIGNED16	RW	0

The unit of this value is rps / s. The drive internally divides this value by 10.

e.g .:Set this value to 10 rps / s, which the drive internally sees as 1 rps / s.

0x6085 Quick stop deceleration

This deceleration is used to quickly stop, according to the quick stop code selection.

Register	Data type	Access type	Default value
6085	UNSIGNED16	RW	0

The unit of this value is rps / s. The drive internally divides this value by 10.

e.g .:Set this value to 10 rps / s, which the drive internally sees as 1 rps / s.

0x6098 Homing method

The value determines the choice of homing mode

Register	Data type	Access type	Default value
6098	INTEGER16	RW	0

0x6099 the speed to find mechanical origin

the speed of finding mechanical origin：

Register	Data type	Access type	Default value
6099	UNSIGNED32	RW	0

Note: The range of speed value: 0 ~ 100000, beyond this range will lead to homing operation failed

0x609A Homing acceleration

Homing acceleration is used for the acceleration（deceleration ）speed in homing mode.

Register	Data type	Access type	Default value
609A	UNSIGNED16	RW	0

The unit of this value is rps / s. The drive internally divides this value by 10.

e.g .:Set this value to 10 rps / s, which the drive internally sees as 1 rps / s.

0x609B the speed to find zero origin

the speed of finding zero origin：

Register	Data type	Access type	Default value
609B	UNSIGNED32	RW	0

TIP:The range of speed value: 0 ~ 100000, beyond this range will lead to homing operation failed

6 Motion control mode

The following intoduce is based on the Modbus-RTU communication protocol for the JMC -RC series products .

6.1 Position mode

Position mode: The point-to-point motion control mode is realized by parameters such as acceleration, deceleration, target speed and target position. When all the parameters are set, the drive will run according to the corresponding parameters. During the movement, the position of the next point can be set during the operation of one point, so as to realize the continuous motion control.

Enable the operation of the drive

The drive is disabled after the drive is powered on or reset. Writes control word 0x000F to the drive control register to put the device into operation enabled state.

Enable position mode

To enable position mode, you first need to write 0x01 to object dictionary 6060h. View Current operation mode from object dictionary 6061h .When a point is running, a new point can be set, and the second position is run directly after the current point bit is completed.

Setting running parameters

Set the target position (607Ah), target speed (6081h), acceleration (6083h), deceleration (6084h) parameters.

Start / stop running

After writing the above operating parameters of the position operation, set bit4 of the control word to start the motor running. During the operation of the motor, if the control word bit8 is set, the motor will stop running.

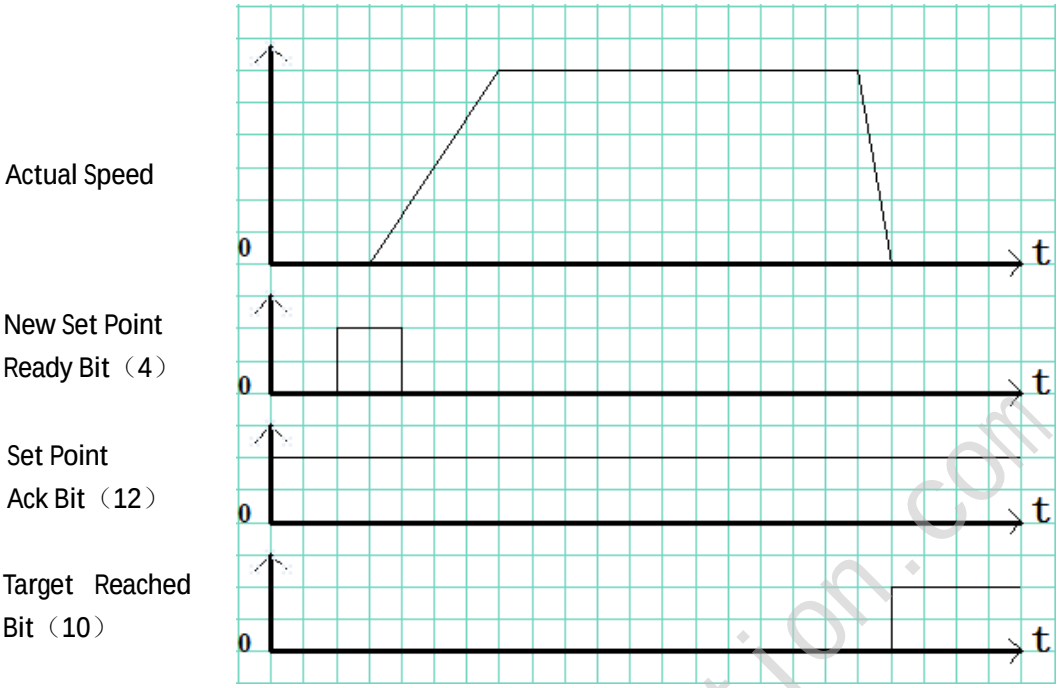
The bit of control word

New target position acquisition: When the status word bit12 is 1, the bit 4 of the control word is changed from 0 to 1, the current position value will be acquired, and if the status word bit12 is 0, the current position value will not be acquired .

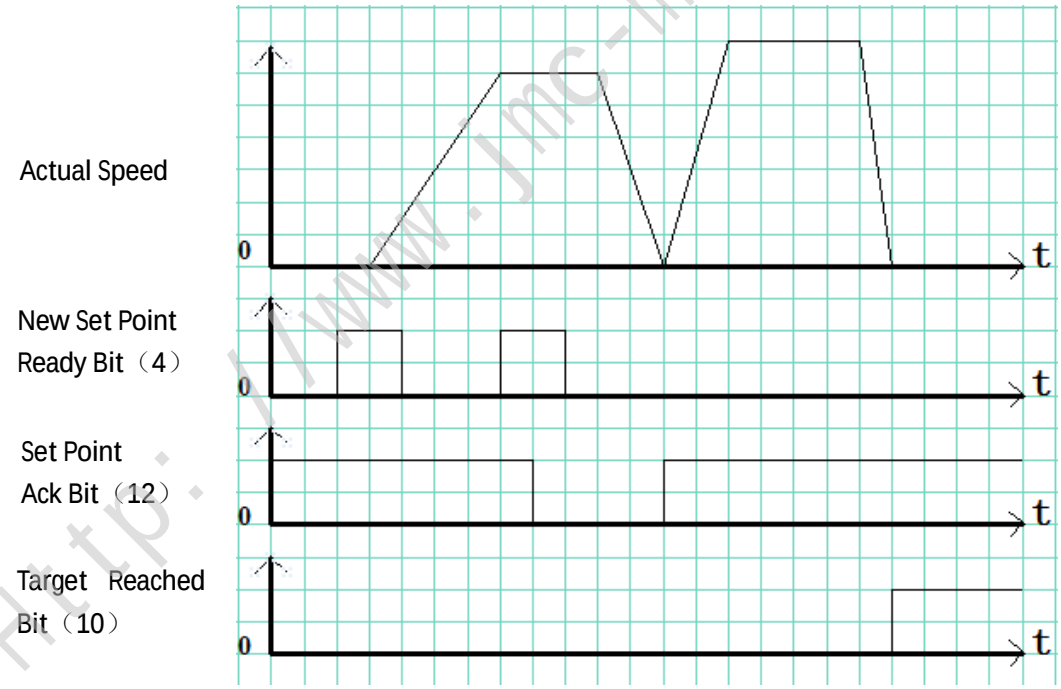
If the control word bit 9 is 0, the drive will run to the last set point and stop. If bit 9 is 1, the drive will complete the movement of the previous point according to the current speed and then change to the next point of the running speed to run to the next point.

Set bit 5 to 1, the newly set position will take effect immediately, and the drive will run immediately at the new position and at the new speed.

Set bit 6 to 1, the final position of the operation is the sum of two positions before and after, that is, the relative positioning. When bit 6 is 0, the final position of operation is the latest position, that is, absolute positioning.

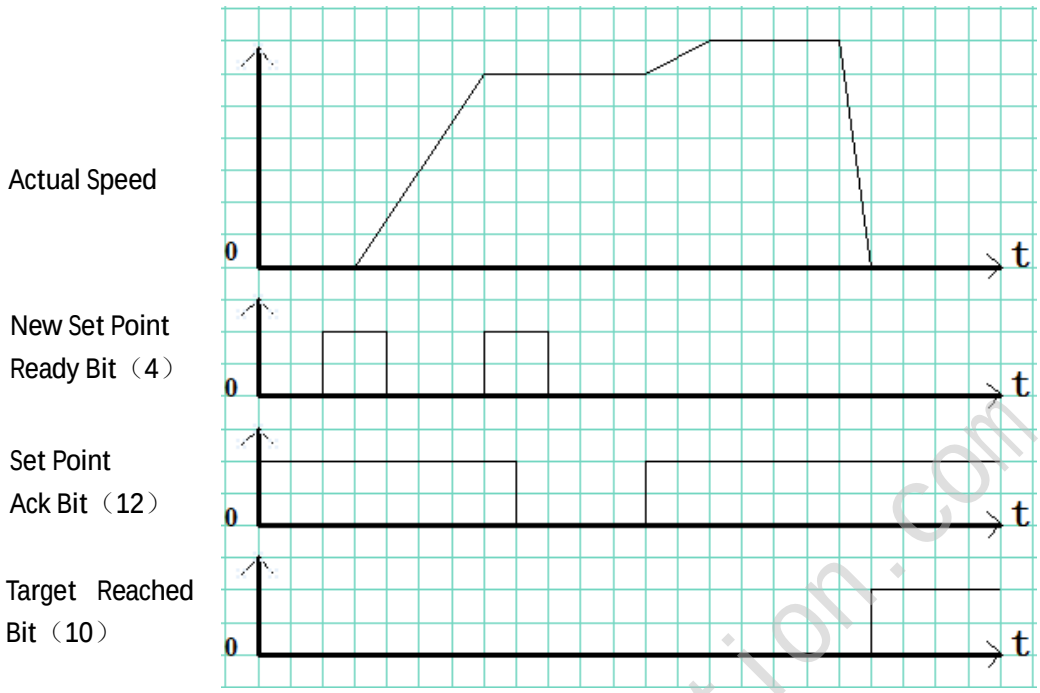


Single point motion



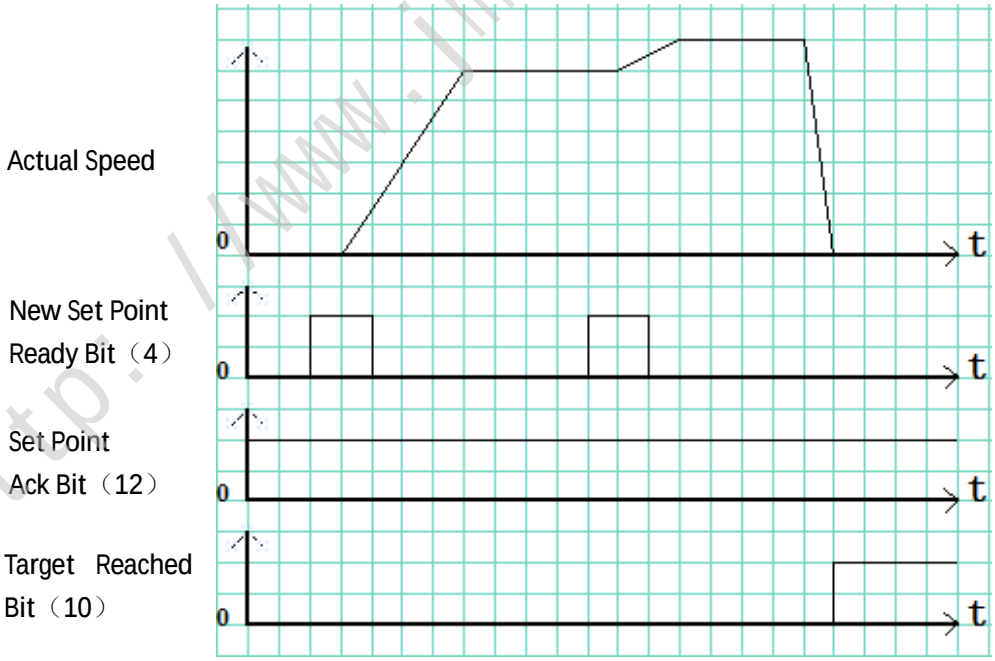
Multi-point motion , the motor will stop running between two positions

In this way, the control word bit 9 and bit 5 are all 0. motors will stop during the two operation.



Multi-point movement, the motor will do not stop the continuous movement

In this mode, the bit9 of the control word is 1, the 5th bit is 0, and The motor runs at a constant speed according to the speed set by the first point before reaching the first point. After reaching the first point, the motor will run at the speed set by the second point, and the motor will not stop.



Multi-point motion, the speed of switching to second points directly after setting second points

In this way, the ninth bit of the control word and fifth bit are placed 1, the motor will switch directly to the second point's velocity without completing the first point's motion. The motor is running at a continuous speed.

Programming description

JMC's ModBus-RTU the value of drive register ：

Register address	Value (the actual value)	Data type	Unit	Description
607Ah	200000（200000）	INT32		Target position
6081h	50（5）	INT32	rps	Target speed
6083h	100（10）	UNINT16	rps/s	Target accelerate
6084h	100（10）	UNINT16	rps/s	Target deceleration
6085h	100（10）	UNINT16	rps/s	Quick stop deceleration

Note: The number of addresses in the 16-bit register is 1 and the number of addresses in the 32-bit register is 2. 32-bit registers can only be written in function code 0x10 .

Assuming the slave1 want to write control word (0x6040 = 0x000F), operation mode (0x6060 = 0x0001), the target speed (0x6081 = 0x00000064), the target acceleration (0x6083 = 0X0064), the target deceleration (0x6084 = 0x0064), quick stop deceleration (0x6085 = 0x0064), the target position (0x607A = 0x00030D40).It is possible to write in continuous (0x10 function code) mode or write (0x06 function code) mode in single register.

Continuous write mode is as follows:

Write the message：01 10 60 40 00 09 12 00 0F 00 01 00 00 00 64 00 64 00 64 00 64 00 03 0D 40 40 1C

Feedback message：01 10 60 40 00 09 1F DB

Write the message explained as follows:

Message ：	01	10	60 40	00 09	12	00 0F	00 01
Introduction：	Slave address	Function code	Start register address	The total number of write addresses	Write the total number of bytes	Write 0x6040 data	Write 0x6060 data

00 00	00 64	00 64	00 64	00 64	00 03	0D 40	40 1C
Write 0x6081high-16bit data	Write 0x6081low-16bit data	Write 0x6083 data	Write 0x6084 data	Write 0x6085 data	Write 0x607Ahigh-16bit data	Write 0x607Alow-16bit data	CRC check code

Feedback message explained as follows:

Message ：	01	10	00 09	1F DB
Introduction：	Slave address	Function code	The total number of write addresses	CRC check code

the mode of writing Single register is as follows:

Want to write target acceleration (0x6083) = 100 (0x0064).

Write the message 1：01 06 60 83 00 64 67 C9

Want to write target deceleration (0x6084) = 100 (0x0064).

Write the message 2：01 06 60 84 00 64 D6 08

Write the message explained as follows:

Message1：	01	06	60 83	00 64	67 C9
Introduction:	Slave address	Function code	Register address	Data input	CRC check code

Message 2：	01	06	60 7A	00 03	D6 08
Introduction:	Slave address	Function code	Register address	Data input	CRC check code

Preload target location (607A) = 200000 (0x00030D40). The data type of this register is INT32. Data can only be written in continuous write mode (0x10 function code). Note that the mode of the 0X6000 definition is different, and its 32 - bit data store the high and low differences in the register, and the message is as follows:

If 0x6000=0：

Write the message：01 10 60 7A 00 02 04 00 03 0D 40 29 96

Feedback message：01 10 60 7A 00 02 7E 11

If 0x6000=1：

Write the message：01 10 60 7A 00 02 04 0D 40 00 03 9F 8F

Feedback message：01 10 60 7A 00 02 7E 11

Write the message explained as follows:

Message：	01	10	60 7A	00 02	04	00 03	0D 40	29 96
Introduction:	Slave address	Function code	Start register address	write The total number of addresses	Write the total number of bytes	Write 0x607A high-16bit data	Write 0x607A low-16bit data	CRC check code

Feedback message explained as follows:

Message：	01	10	60 7A	00 02	7E 11
Introduction:	Slave address	Function code	Start register address	Write the total number of bytes	CRC check code

Want to write target position register(0x607A),and the message is as follows:

If 0x6000=0：

Read the message：01 03 60 7A 00 02 FB D2

Feedback message：01 03 04 00 03 0D 40 0F 53

If 0x6000=1：

Read the message：01 03 60 7A 00 02 FB D2

Feedback message：01 03 04 0D 40 00 03 B9 4A

Read the message explained as follows:

Message ：	01	03	60 7A	00 02	FB D2
Introduction:	Slave address	Function code	Start register address	The total number of write addresses	CRC check code

Feedback message explained as follows:

Message ：	01	03	04	00 03	0D 40	B9 4A
Introduction:	Slave address	Function code	Write the total number of bytes	Read 0x607A high-16bit data	Read 0x607A low-16bit data	CRC check code

Position mode code routine(0x6000=0)

*****Power the motor*****

01 06 60 40 00 01 57 DE 'Drive initialization'

01 06 60 40 00 03 D6 1F 'The drive normal, but the motor disable, loosens the brake'

01 06 60 40 00 0F D6 1A 'power the Motor ,operation Enabled'

*****Set position mode*****

01 06 60 60 00 01 56 14

*****Set point parameters*****

01 10 60 81 00 02 04 00 00 00 0A 12 06 'Set target speed 1rps'

01 06 60 83 00 64 67 C9 'Set acceleration 10rps/s'

01 06 60 84 00 64 D6 08 'Set deceleration10rps/s'

the instruction of setting position parameters can be written in function code 0x10 mode at once :

01 10 60 81 00 04 08 00 00 00 0A 00 64 00 64 10 52

*****Single point motion*****

01 10 60 7A 00 02 04 00 03 0D 40 29 96 'Set target position 200000 steps'

01 06 60 40 00 1F D7 D6 'Sampling new position and running'

01 06 60 40 00 0F D6 1A 'the control word bit 4 can be set to 0 for the next fast sampling'

*****Multi-point motion, need to stay between two points,absolute position*****

01 10 60 81 00 02 04 00 00 00 32 13 D4 'Set target speed 5rps'
01 10 60 7A 00 02 04 00 03 0D 40 29 96 'Set target position 200000 steps'
01 06 60 40 00 1F D7 D6 'Sampling new position'
01 06 60 40 00 0F D6 1A 'the control word bit 4 can be set to 0 for the next fast sampling'
01 10 60 81 00 02 04 00 00 00 64 93 EA 'The speed of setting second segments is 10 rps'
01 10 60 7A 00 02 04 00 09 27 C0 17 54 'Set target position 600000steps'
01 06 60 40 00 1F D7 D6 'Sampling new position'

*****Multi-point motion, no need to stay between two points*****

01 10 60 81 00 02 04 00 00 00 32 13 D4 'Set target speed 5rps'
01 10 60 7A 00 02 04 00 03 0D 40 29 96 'Set target position 200000steps'
01 06 60 40 02 5F D7 46 'Sampling new position'
01 06 60 40 02 4F D6 8A 'the control word bit 4 can be set to 0 for the next fast sampling'
01 10 60 81 00 02 04 00 00 00 64 93 EA 'The speed of setting second segments is 10 rps'
01 10 60 7A 00 02 04 00 09 27 C0 17 54 'Set target position 600000steps'
01 06 60 40 02 5F D7 46 'Sampling new position'

*****Multi-point motion, which directly changes the speed of operation*****

01 10 60 81 00 02 04 00 00 00 32 13 D4 'Set target speed 5rps'
01 10 60 7A 00 02 04 00 03 0D 40 29 96 'Set target position 200000steps'
01 06 60 40 00 7F D7 FE 'Sampling new position'
01 06 60 40 00 6F D6 32 'the control word bit 4 can be set to 0 for the next fast sampling'
01 10 60 81 00 02 04 00 00 00 64 93 EA 'The speed of setting second segments is 10 rps'
01 10 60 7A 00 02 04 00 09 27 C0 17 54 'Set target position 600000 steps'
01 06 60 40 02 7F D6 9E 'Sampling new position'

6.2 Velocity mode

Velocity mode introduction

velocity mode: The velocity mode includes the target speed (0x6081), acceleration (0x683), deceleration (0x6084) parameters. During running can be suspended by the control word to pause the movement.

Enable the operation of the drive

The drive is disabled after the drive is powered on or reset. Writes 0x000F to the drive control register to put the device into operation enabled state. Before writing the speed parameter, the control word bit 8 can be set to 1 to suspend the motor running,that is ,write 0x010F to the control word (0x6040). After writing the parameters, the operation can be enabled.

Enable velocity mode

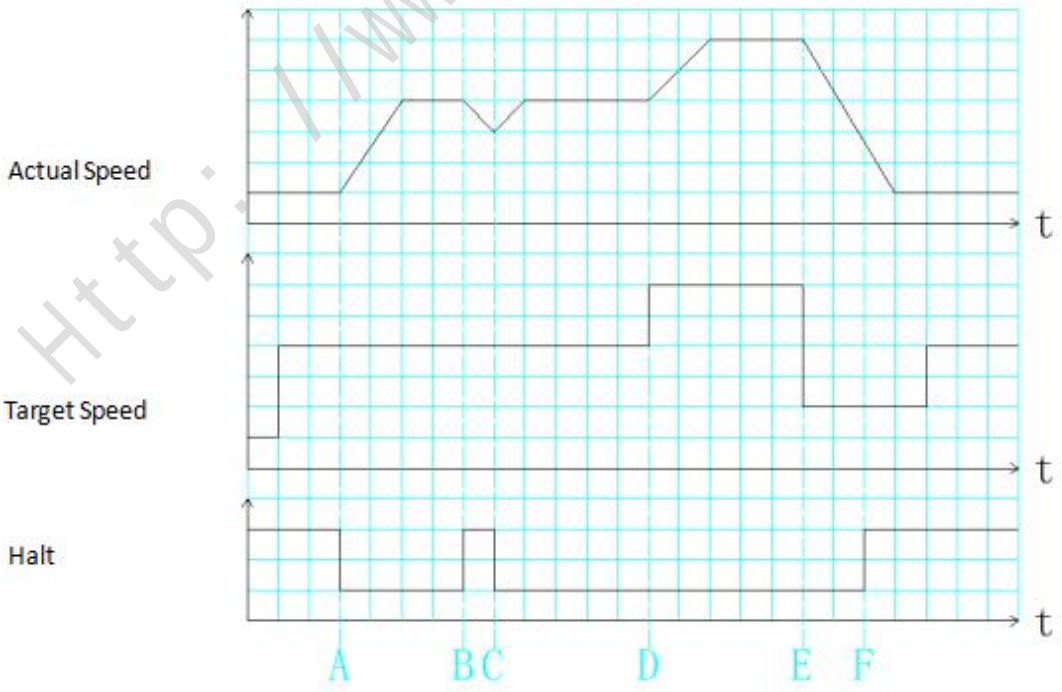
Write 0x0003 to the object dictionary 0x6060h, enabling velocity mode.

Setting running parameters

Set speed (0x6081h), acceleration (0x6083h), deceleration (0x6084h) parameters.

start /stop

Write 0x000F to the control word to start the motor. Write 0x010F to the control word to stop the motor.



Velocity mode code routine(0x6000=0)

******Power the motor******

01 06 60 40 00 01 57 DE 'Drive initialization'

01 06 60 40 00 03 D6 1F 'The drive normal, but the motor disable, loosens the brake'

01 06 60 40 00 0F D6 1A 'power the Motor ,operation Enabled'

******Set velocity mode******

01 06 60 60 00 03 D7 D5 'Set velocity mode'

01 06 60 40 01 0F D7 8A 'Pause motor operation'

******Set running parameters******

01 10 60 81 00 02 04 00 00 00 0A 12 06 'Set target speed 1rps'

01 06 60 83 00 64 67 C9 'Set acceleration 10rps/s'

01 06 60 84 00 64 D6 08 'Set deceleration 10rps/s'

the instruction of setting velocity parameters can be written in function code 0x10 mode at once :

01 10 60 81 00 04 08 00 00 00 0A 00 64 00 64 10 52

******Start/Pause running ******

01 06 60 40 00 0F D6 1A 'start running'

01 10 60 81 00 02 04 00 00 00 64 93 EA 'Set speed 10rps'

01 06 60 40 01 0F D7 8A 'Pause running'

6.3 Homing mode

In homing mode, we will go back to homing operation through three limit switches, namely CW limit, CCW limit, and mechanical origin limit (optional).

Enable the operation of the drive

The drive is disabled after the drive is powered on or reset. Writes 0x000F to the drive control register to put the device into operation enabled state.

Enable homing mode

Writes 0x06 to object dictionary 6060h, enabling home mode.

Setting running parameters

Need to set parameters of homing method (0x6098h), the speed to find mechanical origin (0x6099h), homing acceleration (0x609Ah), home offset (0x607C) and the speed to find zero origin(0x609B).

Start homing operation

In the object dictionary 0x6098h choose to homing method. The fourth bits in the control word are turned over and start to homing operation . You can check the status of the status word.

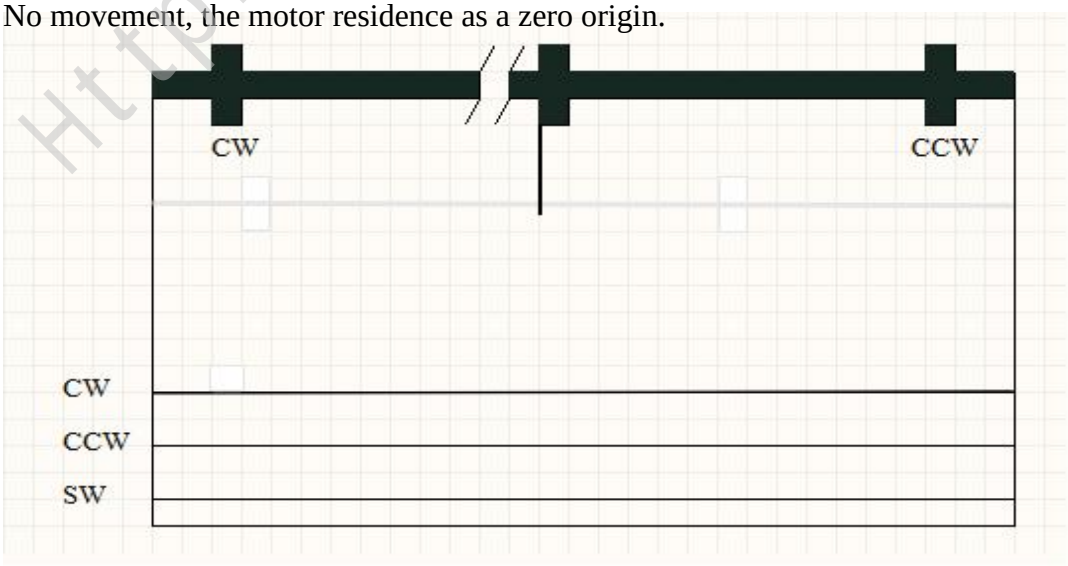
Home offset

Home offset is the offset distance between the homing position and the zero position we set, and the direction of the offset can be on the left or right of the mechanical origin.

Diagram of homing operation

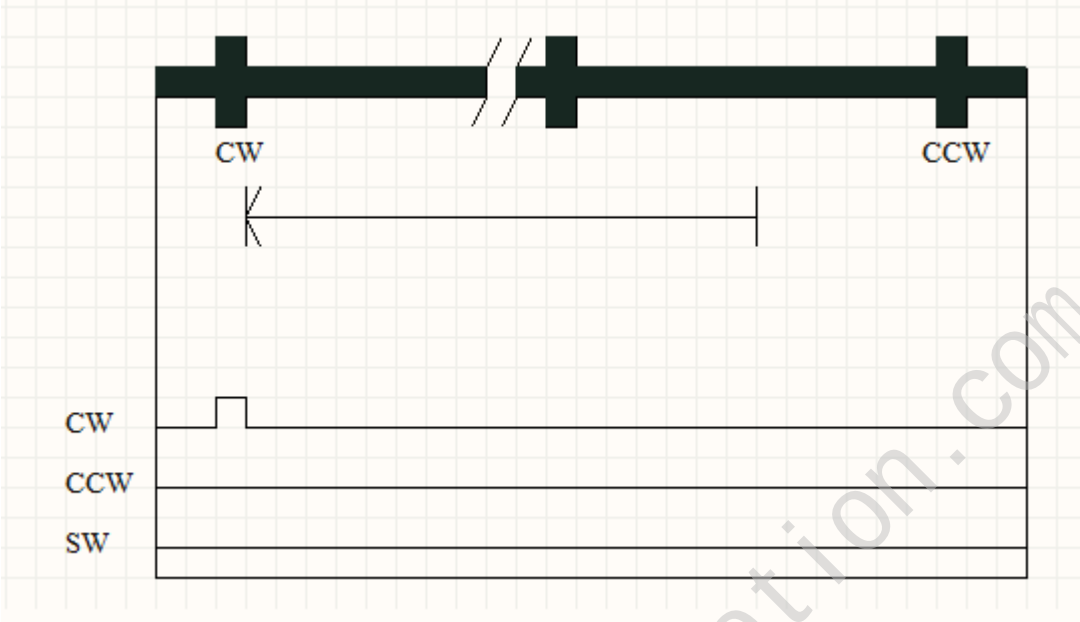
Homing method 0

No movement, the motor residence as a zero origin.



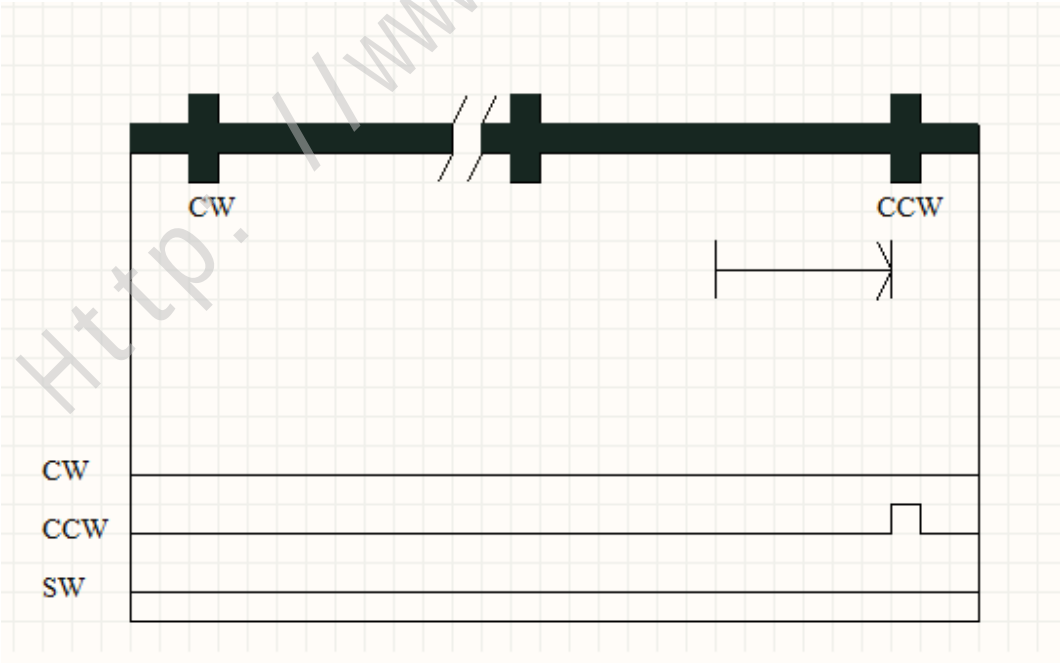
Homing method 1

Move left and stop when moving back to CW limit. (maximum homing speed of 300r / min)



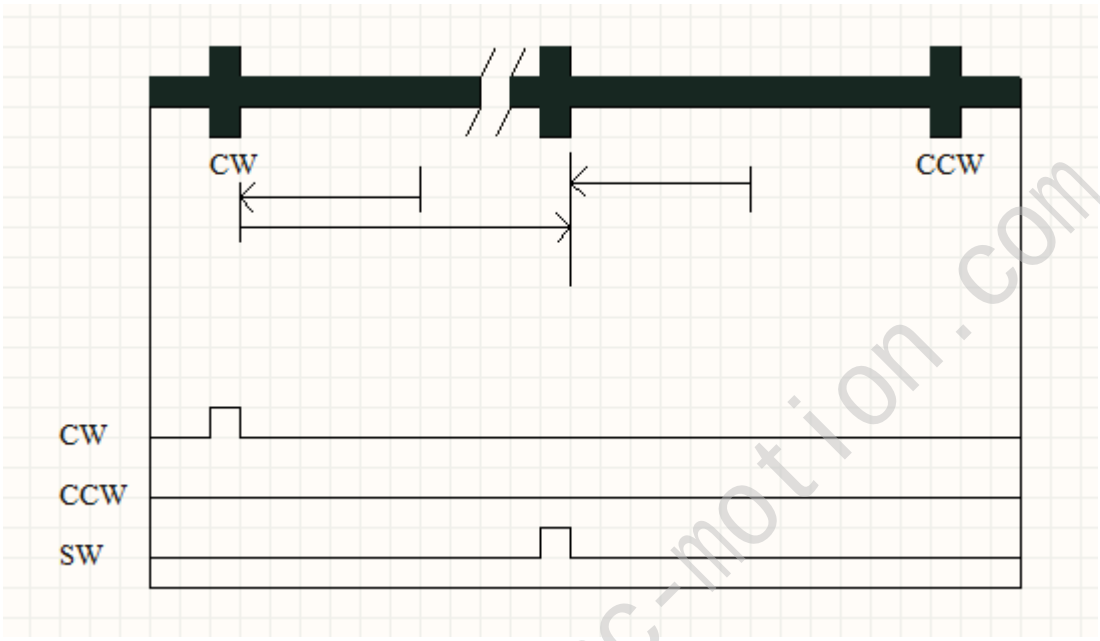
Homing method 2

Move right and stop when moving back to CCW limit. (maximum homing speed of 300r / min)



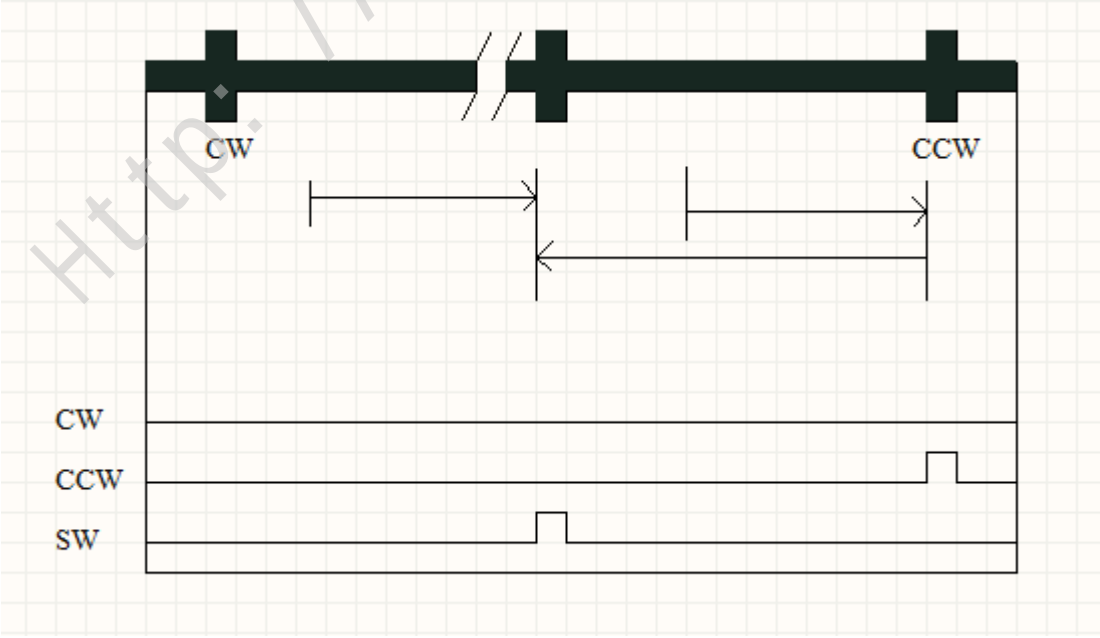
Homing method 3

Move to left, (continue to the left without touching the SW limit) if the CW limit is reached, move to right and stop at the mechanical origin limit. (maximum homing speed of 300r / min)



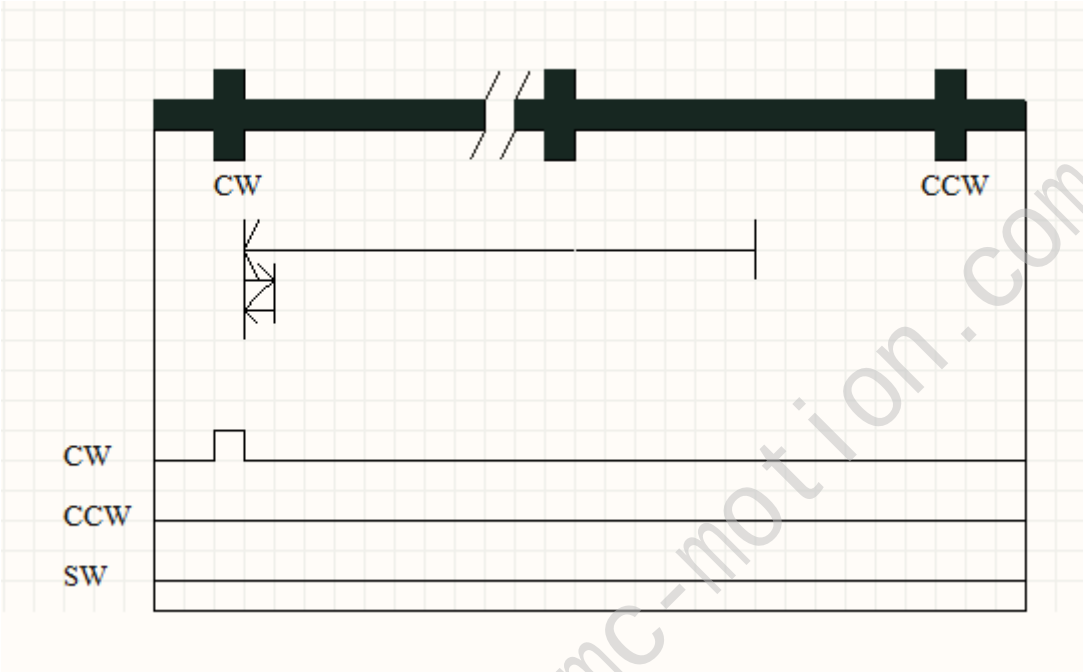
Homing method 4

Move to right, (continue to the right without touching the SW limit) if the CCW limit is reached, move to left and stop at the mechanical origin limit. (maximum homing speed of 300r / min)



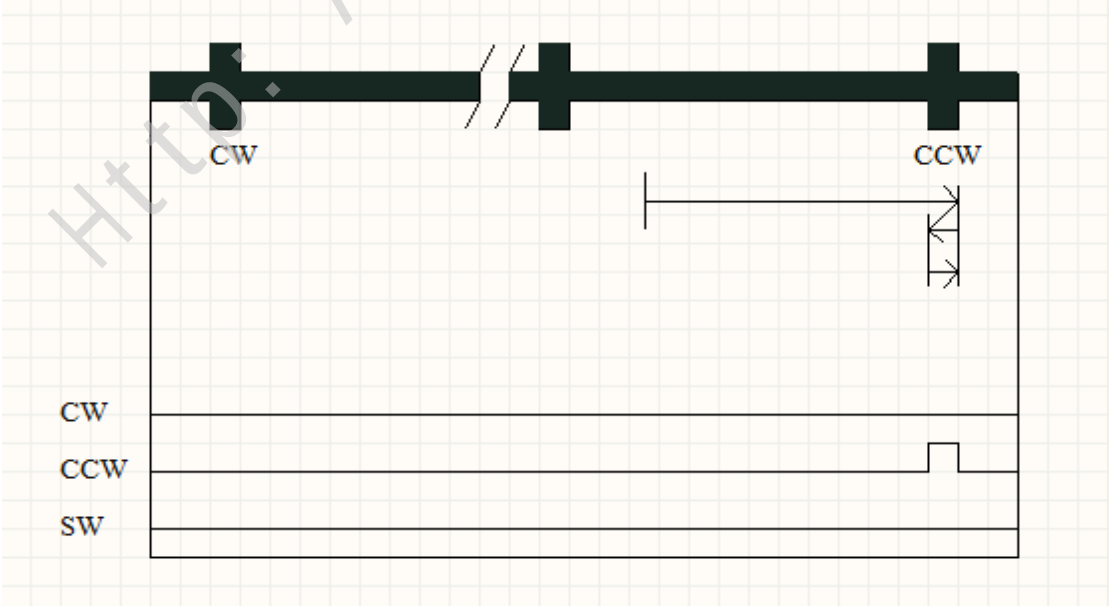
Homing method 5

High speed move to the left after the limit switch to quickly reduce, and then move to the right at low speed, After crossing the limit switch, low speed to the left and stop when moving back to CW limit. (maximum homing speed of 1200r / min)



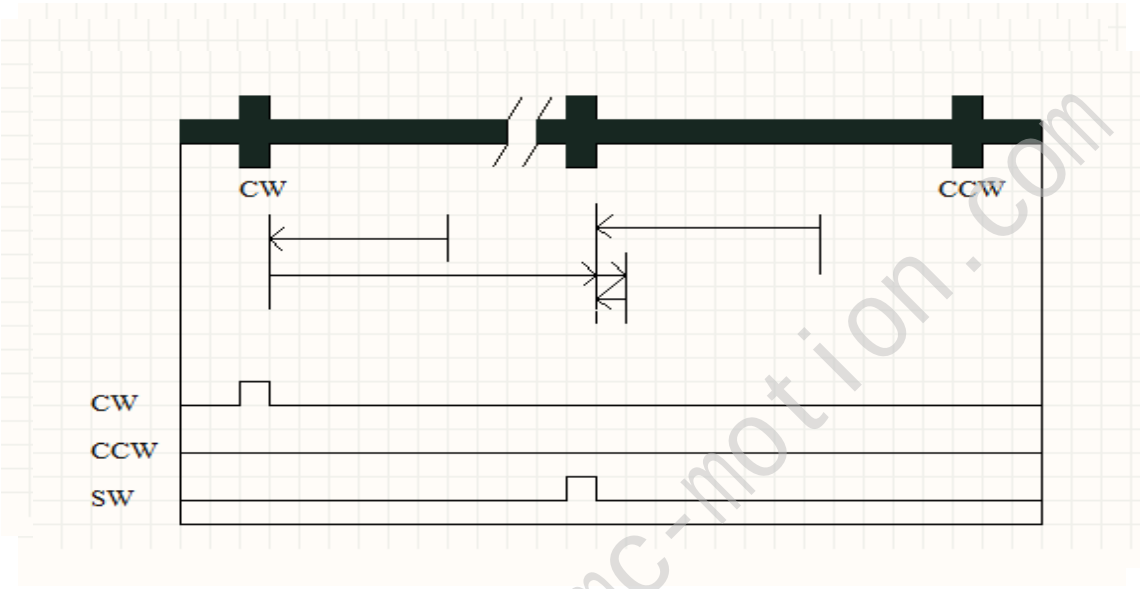
Homing method 6

High speed move to the right after the limit switch to quickly reduce, and then move to the left at low speed, After crossing the limit switch, low speed to the right and stop when moving back to CCW limit. (maximum homing speed of 1200r / min)



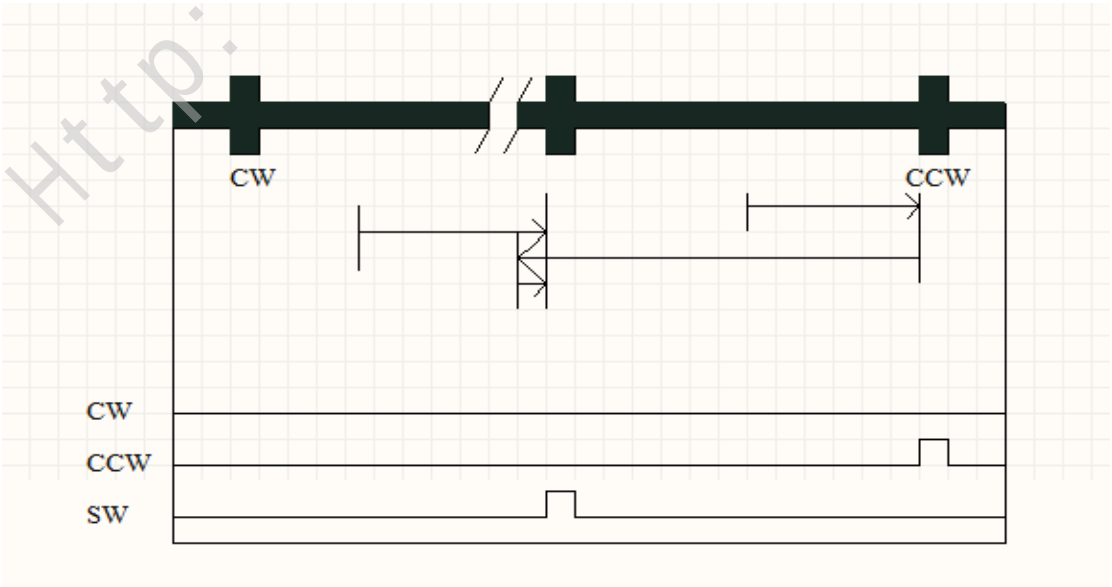
Homing method 7

High speed move to left, if it does not get the mechanical origin, it will continue to move to left.If the CW limit switch is getted, it moves to right by high speed. When it get the machine origin limit switch, it rapidly decreases and then moves to the right by low speed ,After crossing the limit switch, low speed to the left and stop at mechanical origin switch . (maximum homing speed of 1200r / min)



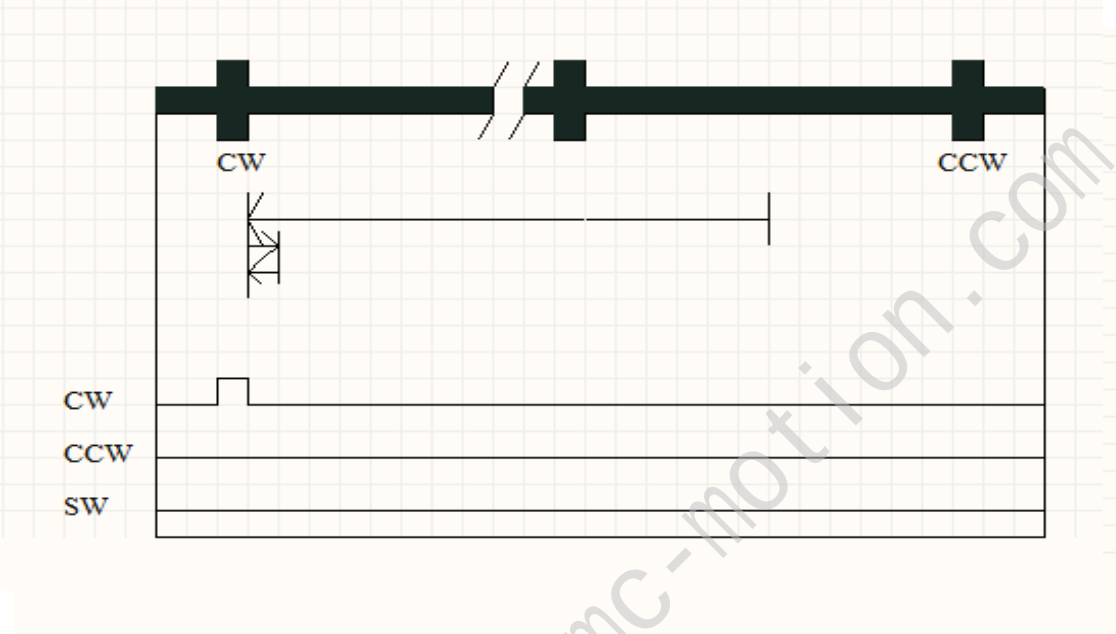
Homing method 8

High speed move to right, if it does not get the mechanical origin switch, it will continue to move to the right. If the CCW limit switch is getted, it moves to left by high speed. When it gets the machine origin limit switch, it rapidly decreases and then moves to the left by low speed ,After crossing the limit switch, low speed to the right and stop at the limit switch . (maximum homing speed of 1200r / min)



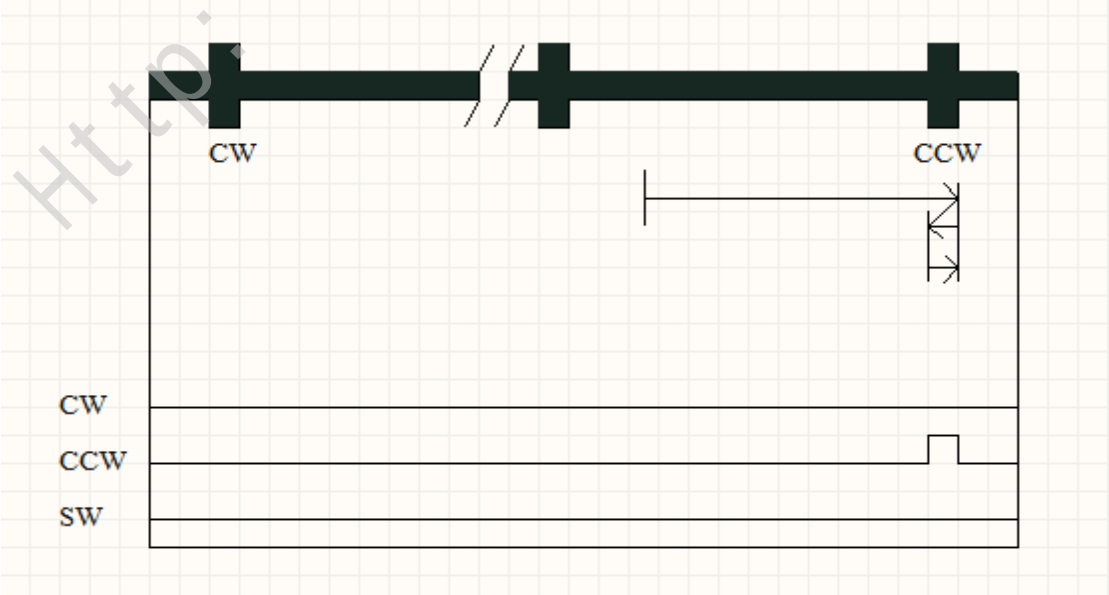
Homing method 9

High speed move to the left after the limit switch to quickly reduce, and then move to right by low speed, After crossing the limit switch, low speed move to the left and stop at the CW limit. (maximum homing speed of 1200r / min)(Note: This mode can set positive and negative homing offset)



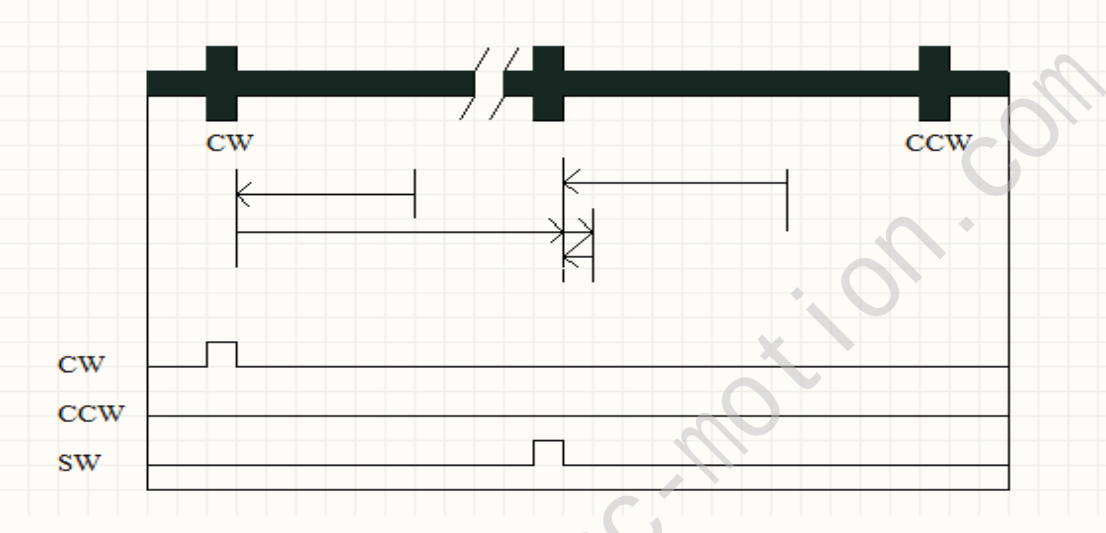
Homing method 10

High speed move to right after the limit switch to quickly reduce, and then move to left by low speed, After crossing the limit switch, low speed move to the right and stop at CCW limit. (maximum homing speed of 1200r / min)(Note: This mode can set positive and negative homing offset)



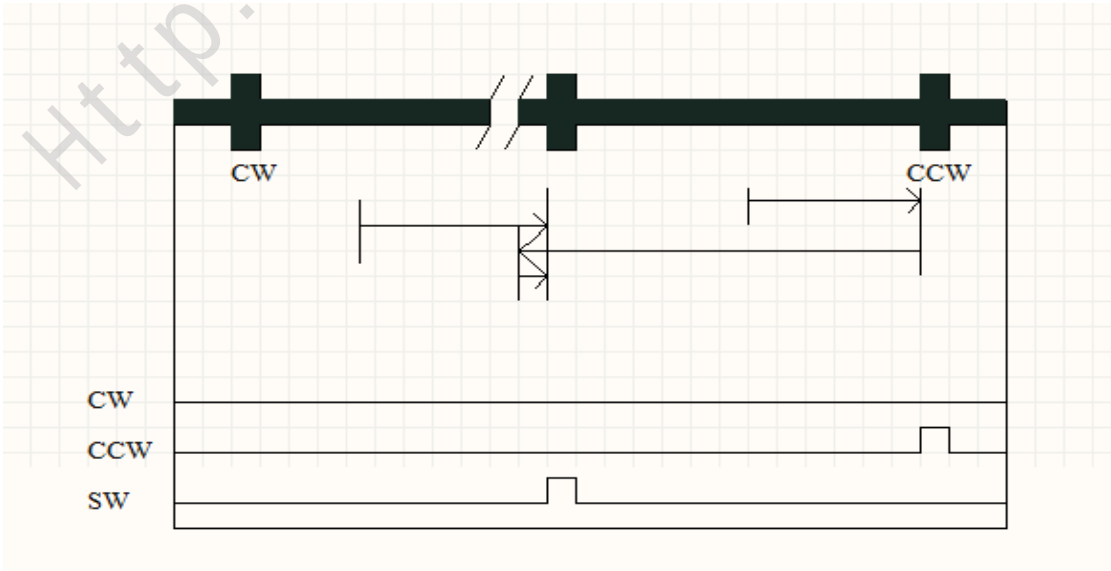
Homing method 11

The high speed is left to run, if it does not meet the mechanical origin, it will continue to run to the left.If the CW limit switch is operated, it runs to the right at a high speed. When it encounters the machine origin limit switch, it rapidly decreases and then moves to the right and low speed ,After crossing the limit switch, low speed to the left and stop when moving back to limit switch . (maximum homing speed of 1200r / min)(Note: This mode can set positive and negative home offset)



Homing method 12

High speed move to right, if it does not arrive the mechanical origin switch, it will continue to move to right. If the CCW limit switch is arrived, it moves to left by high speed. When it reached the machine origin limit switch, it rapidly decreases and then moves to left by low speed ,After crossing the limit switch, low speed to the right and stop at limit switch . (maximum homing speed of 1200r / min)(Note: This mode can set positive and negative home offset)



Homing mode code routine（0x6000=0）

*****Power the motor*****
01 06 60 40 00 01 57 DE 'Drive initialization'
01 06 60 40 00 03 D6 1F 'The drive normal, but the motor disable, loosens the brake'
01 06 60 40 00 0F D6 1A 'power on Motor ,operation Enabled'

*****Set homing mode*****
01 06 60 60 00 06 17 D6 'Set homing mode'

*****Set homing method*****
01 06 60 98 00 01 D7 E5 'Select homing method 1'
*****Set homing operation parameter*****
01 06 60 9A 03 E8 B7 5B 'Set homing acceleration100rps/s '
01 10 60 99 00 02 04 00 00 00 64 93 40 'Set homing speed 10rps '
01 10 60 9B 00 02 04 00 00 00 64 12 99 'Set the speed of finding zero origin 10rps '
01 10 60 7C 00 02 04 00 00 03 E8 5C 62 'Set homing offset 1000 steps'
The homing method and the instruction of setting homing parameters can be written in function code 0x10 mode at once :
01 10 60 7C 00 08 10 00 00 03 E8 00 01 00 00 00 64 03 E8 00 00 00 64 09 42

**** start/halt running ****
01 06 60 40 00 1F D7 D6 'start running'
01 06 60 40 01 1F D6 46 'halt running'

7 Function code supported by the device

7.1 Introduction

JMC's ModBus-RTU agreement supports the following function code:

Function	Code	Description	Mark
read hold register	0x03	Read multiple words	The maximum word length：63 word
write a single register	0x06	Write a single word	-
write multiple registers	0x10	Write multiple words	-

7.2 read register

request

Salve address	1Byte	0xXX
Function code	1Byte	0x03
Start address	2Byte	0xXXXX
The total number of addresses	2Byte	1~126
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

respond

Salve address	1Byte	0xXX
Function code	1Byte	0x03
The total number of bytes	1Byte	-
Return data	N bytes of data	-
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

Example

request

Salve address	Function code	Start address	The total number of addresses	CRC check
XX	0x03	Hi Lo	Hi Lo	Hi Lo
1Byte	1Byte	2Byte	2Byte	2Byte

Note: Hi = high byte, Li = low byte.

respond

Salve address	Function code	The total number of bytes	The data of the first address	.....	The data of the last address	CRC check
XX	0x03	XX	Hi LO		Hi Lo	Hi Lo
1Byte	1Byte	1Byte	2Byte		2Byte	2Byte

Note: Hi = high byte, Li = low byte.

Example ： read the value of the 32-bit register (0x607Ah)

If 0x6000 = 0：

Read the message：01 03 60 7A 00 02 FB D2

Feedback message：01 03 04 00 03 0D 40 0F 53

If 0x6000 = 1：

Read the message：01 03 60 7A 00 02 FB D2

Feedback message：01 03 04 0D 40 00 03 B9 4A

Read the message explained as follows:

Message ：	01	03	60 7A	00 02	FB D2
Introduction：	Salve address	Function code	Start register address	The total number of addresses	CRC check code

Feedback message explained as follows:

Message ：	01	03	04	00 03	0D 40	0F 53
Introduction：	Salve address	Function code	The total number of bytes	The data of the first address	The data of the second address	CRC check code

7.3 write a single register

request

Salve address	1Byte	0xXX
Function code	1Byte	0x06
register address	2Byte	0xXXXX
The vale of register addresses	2Byte	0xXXXX
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

respond

Salve address	1Byte	0xXX
Function code	1Byte	0x06
register address	2Byte	0xXXXX
The vale of register addresses	2Byte	0xXXXX
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

Note: 32-bit register can not be written with 06 function code, need to use 10 function code to write

Example: Write 0x0003 to slave2's 16-bit register (0x6060h).

Request and Response Data:

Message ：	02	06	60 60	00 03	XX XX
Introduction：	Salve address	Function code	register address	The vale of register addresses	CRC check code

7.4 write multiple registers

request

Salve address	1Byte	0xXX
Function code	1Byte	0x10
Start register address	2Byte	0xXXXX
The total number of write addresses	2Byte	0xXXXX
Write the total number of bytes	1Byte	0xXX
Write n data values	N*2 Byte	0xXX
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

respond

Salve address	1Byte	0xXX
Function code	1Byte	0x10
Start register address	2Byte	0xXXXX
The total number of write addresses	2Byte	0xXXXX
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

Example:

0x0064 is written to the 16-bit register target acceleration (6083h) and the target deceleration (6084h).

If 0x6000 = 0：

Write the message：01 10 60 83 00 02 04 00 64 00 64 53 EC

Feedback message：01 10 60 83 00 04 AE 20

If 0x6000 =1：

Write the message：01 10 60 83 00 02 04 00 64 00 64 53 EC

Feedback message：01 10 60 83 00 04 AE 20

Write the message explained as follows:

Message：	01	10	60 83	00 02	04	00 64	00 64	53 EC
Introduction：	Salve address	Function code	Start register address	The total number of write addresses	Write the total number of bytes	Write 0x6083 data	Write 0x6084 data	CRC Check code

Feedback message explained as follows:

Message ：	01	10	60 83	00 02	53 EC

Introduction：	Salve address	Function code	Start register address	The total number of write addresses	CRC Check code

7.5 Abnormal function code

respond

Salve number	1Byte	0xXX
Function code	1Byte	Request function code|0x80
Abnormal function code	1Byte	0~11
CRC check code 'low byte'	1Byte	0xXX
CRC check code 'high byte'	1Byte	0xXX

Example：

Write 0x0003 to salave1's 16-bit register (0x6090h). An exception occurred, return no this register.

Write the message：01 06 60 90 00 03 D7 E6

Feedback message：01 86 0B 03 A7

Write the message explained as follows:

Message ：	01	06	60 90	00 03	D7 E6
Introduction ：	Salve address	Function code	register address	Written data	CRC check code

Feedback message explained as follows:

Message ：	01	86	0B	03 A7
Introduction ：	Salve address	Abnormal Function code	Write register does not exist	CRC check code

8 Register list

Register	Access type	Data type	Introduction
0x 1001	RO	UNSIGNED16	The device's internal error will be mapped to this register.
0x 1008	RO	Via-String	Manufacturer Device name
0x 1009	RO	Via-String	Manufacturer hardware version
0x 100A	RO	Via-String	Manufacturer software version
0x 6000	RW	UNSIGNED16	the register format of 32-bit data
0x 605A	RW	INTEGER16	The way to quick stop
0x 605D	RW	INTEGER16	The way to quick halt
0x 6040	WO	UNSIGNED16	Drive status and motion’s control word
0x 6060	WO	INTEGER16	Modes of operation
0x 6081	RW	INTEGER32	Target velocity
0x 6083	RW	INTEGER16	Motor running acceleration
0x 6084	RW	INTEGER16	Motor running deceleration
0x 6085	RW	UNSIGNED16	Quick stop deceleration
0x 607A	RW	INTEGER32	The position the drive is moving on
0x 607C	RW	INTEGER32	the offset position between the home position and the zero position.
0x 6098	RW	INTEGER16	Homing method
0x 6099	RW	UNSIGNED32	the speed to find mechanical origin
0x 609A	RW	UNSIGNED16	Homing acceleration/deceleration
0x 609B	RW	UNSIGNED32	the speed to find zero origin
0x 6041	RO	UNSIGNED16	the current status of the drive
0x 6061	RO	INTEGER16	the current mode of the drive
0x 6064	RO	INTEGER32	the position of the current moment in the position mode
0x 606C	RO	INTEGER32	the speed of the current time

TIP: If you want to use function code 0x10 mode to write to the register in succession, the register written must be consecutive in the above table, for example: 0x6041, 0x6061, 0x6064 The three registers are continuous in the above table, so you can use 0x10 mode one time Write parameters to these three registers.

Appendix 1 CRC check

Cyclic Redundancy Check CRC area is 2 bytes and contains a 16-bit binary data. The CRC value is calculated by the sending device and the calculated value is attached to the message. The receiving device recalculates the CRC value when receiving the message and compares the calculated value with the actual value received in the CRC area. If the two are different, It produces an error.

At the beginning of CRC, 16 bits of registers are all placed into'1',then the next two 8-bit data are put into the current register, and only the 8-bit data of each character is used to generate the CRC, start bits, stop bits and parity bits are not added to CRC.
During CRC generation, every 8 bits of data and register's value are used to XOR operation, and the result is shifted to one side to the right direction (to LSB direction), and'0'is used to fill in MSB to detect LSB. If the LSB is '1', it will be XOR with the preset fixed value,and if the LSB is '0' then not XOR.

Repeat the above process, until the shift 8 times, after the completion of the 8th shift, the next 8-bit data and the current value of the register are used to XOR operation, after all the information is processed, the final value of the register for the CRC value.

The process of generating CRC:
1. Set 16-bit CRC register to FFFFH.
2. The first 8-bit data and the CRC register low 8-bit exclusive-OR operation, the results into the CRC register.
3. CRC register moves to one bit to the right, MSB fills in the zero, and checks LSB.
4. (if LSB is 0): repeat 3, and then move right one.(If LSB is 1): CRC register is XOR'd with A001H.
5. Repeat steps 3 and 4 until 8 shifts are completed, completing the 8-bit byte processing.
6. Repeat steps 2 to 5 to process the next 8-bit data until all bytes have been processed.
7. The final CRC register value is the CRC value.
8. The CRC value into the message, the high 8 and low 8 should be placed separately. When sending a 16-bit CRC value in a message, it sends the lower 8 bits first and the higher 8 bits back.

If the CRC value is 1241 (0001 0010 0100 0001):

Slave	Function code	Data Number	Data1	Data2	Data3	Data4	Data5	CRC low	CRC high
								41	12

example:
The various possible CRC values are loaded in two columns, one in the upper 8 bits of the 16-bit CRC, high 8 bits in CRC (0-256), and the other in the lower 8 bits as the lower CRC value.
The CRC obtained in this way is faster than the new CRC value calculated for each new character in the buffer.
Note: This function internally exchanges the high / low byte in the CRC, and in the CRC

value returned, its bytes have been exchanged.

Therefore, the CRC value returned by the function code can be directly sent in the message.

Routine：

The function's return value CRC is of type 'unsigned short'.

```
Generate CRC function:
unsigned char *PMSG;    //To generate a CRC value, point the pointer to a buffer containing binary
                        //data
unsigned short DataLen; //The number of bytes in the buffer.
unsigned short CRC16(PMSG, DataLen)
{
    unsigned char uchCRCHi = 0xFF ; /* Initialize high byte*/
    unsigned char uchCRCLo = 0xFF ; /* Initialize low byte*/
    unsigned uIndex ;
    while (DataLen—)
    {
        uIndex = uchCRCHi ^ *PMSG++ ; /*counter CRC*/
        uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
        uchCRCLo = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Appendix 2 Modbus/RTU 16-bit CRC check routine

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Modbus
{
    public static class Utility
    {
        private static readonly ushort[] m_CrcTable =
        {
            0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
            0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
            0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
            0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
            0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
            0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
            0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
            0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
            0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
            0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
            0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
            0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
            0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
            0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
            0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
            0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
            0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
            0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
            0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
            0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
            0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
            0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
            0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
            0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
            0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
            0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
            0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
            0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
            0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
            0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
            0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
            0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
```

```
        };
        //Calculate vertical redundancy check
        //Parameters: 'data' for LRC operations
        //Return value: LRC calculation result
        public static byte CalculateLrc(byte[] data)
        {
            if (data == null)
            {
                throw new ArgumentNullException('data');
            }
            byte lrc = 0;
            foreach (byte b in data)
            {
                lrc += b;
            }
            lrc = (byte)((lrc ^ 0xFF) + 1);
            return lrc;
        }
        // Calculate the period redundancy check
        // parameter: 'data' is used for CRC calculation
        // Return value: CRC calculation result
        public static byte CalculateLrc(byte[] data)
        {
            if (data == null)
            {
                throw new ArgumentNullException('data');
            }
            ushort crc = ushort.MaxValue;
            foreach (byte b in data)
            {
                byte tableIndex = (byte)(crc ^ b);
                crc >>= 8;
                crc ^= m_CrcTable[tableIndex];
            }
        return BitConverter.GetBytes(crc);
        }
    }
}
```

The following is the calling method:

```
byte[] _Data = new byte[] { 0x18, 0x18};
byte[] _Crc = Modbus.Utility.CalculateCrc(_Data);
```

contact us

Address：Floor2,Building A,Hongwei Industrial Zone No.6, Liuxian 3rd Road, Bao'an District, Shenzhen.China

Telephone：0755-26509689 26502268

Fax：0755-26509289

E-mail:info@jmc-motion.com

Http: //www.jmc-motion.com