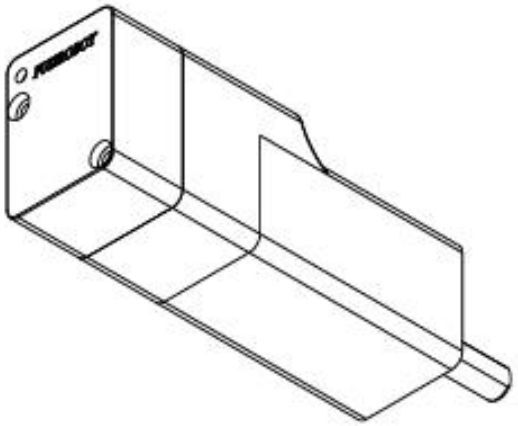
PUSIROBOT

CQPUSI ROBOT CONTROL SYSTEM

User Manual

PMC006B1 Series

Miniature Integrated Stepper Motor Controller



1. Version Control

1） Update Records

Date	Author	Version	Remark
2024-7-1	Liu	V1.0.0	Initial
2024/12/30	Liu	V1.0.1	Update the register address
2025/3/11	Liu	V1.0.2	Add a unit description for the high-speed mode
2026/9/11	Liu	V1.0.3	Add parameters for analog quantity adjustment

Catalog

1 Introduction 6
1.1 Statement of intellectual property right 6
1.2 Disclaimer 6
2 Overview 7
2.1 General Description 7
2.2 Features 7
2.3 Production & Ordering Information 8
3 Connector Description 8
3.1 Terminal port location 8
3.2 Motor connection J1 8
3.3 Modbus network connection 9
3.4 Limit switch connection 9
3.5 Other limit switch connections 10
3.6 Mechanical switch connection 11
3.7 Analog Adjusting Speed/Positioning 11
4 Drive mode 11
4.1 Normal open-loop/closed-loop mode 11
4.2 High-speed torque mode 12
5 MODBUS communication 12
5.1 MODBUS introduction 12
5.2 MODBUS frame structure 13
5.3 MODBUS communication configuration 13
5.3.1 Node ID 13
5.3.2 Baud rate 14
5.3.3 Group ID 14
5.4 System information acquisition 14
5.4.1 Device node name 14
5.4.2 Hardware version 14
5.4.3 Software version 14
5.4.4 System control 15
5.5 Motor control parameters 15
5.5.1 Error status 15
5.5.2 Controller status 16
5.6 Basic Instructions 16
5.6.1 Stop stepping command 16
5.6.2 Micro stepping 16
5.6.3 Maximum phase current(Operating current) 17
5.6.4 Motor position 17
5.6.5 Encoder position (absolute value encoder closed loop) 17
5.6.6 Current reduction 18
5.6.7 Motor enable 18
5.6.8 Operation mode 18
5.6.9 Real-time speed(Closed-loop) 20

5.7 Position mode 20
5.7.1 Position mode rotating direction 20
5.7.2 Position mode operating speed 20
5.7.3 Position mode start speed 21
5.7.4 Position mode stop speed 21
5.7.5 Position mode acceleration coefficient 21
5.7.6 Position mode deceleration coefficient 22
5.7.7 Position mode relative displacement command 22
5.7.8 Position mode absolute displacement command 23
5.8 External emergency stop 23
5.8.1 External emergency stop enable 24
5.8.2 The trigger mode of external emergency stop 24
5.8.3 Sensor type 24
5.8.4 Debouncing delay 25
5.9 General IO port 25
5.9.1 IO port direction 25
5.9.2 General IO port value 26
5.10 Closed-loop control 27
5.10.1 Encoder resolution 27
5.10.2 KP parameter 27
5.10.3 KI parameter 28
5.10.4 KD parameter 28
5.10.5 Pre-filtering parameter 28
5.10.6 Post-filtering parameter 28
5.10.7 Stall length parameter(closed-loop) 29
5.10.8 Torque ring enable(High-speed torque mode switch) 29
5.10.9 Autosave when power is off enable 29
5.11 PP motion mode 29
5.11.1 PP mode parameter 1 30
5.11.1.1 PP mode Acceleration 30
5.11.1.2 PP mode Deceleration 30
5.11.1.3 PP mode Start speed 30
5.11.1.4 PP mode Stop speed 31
5.11.2 PP mode parameter 2 31
5.11.2.1 PP mode Control word 31
5.11.2.2 PP Status word 31
5.11.2.3 PP model Running Speed 32
5.11.2.4 PP model Target Location 32
5.11.3 PP model work timing 32
5.12 PV Mode 35
5.13 Analog parameters 35
5.13.1 Analog enable 35
5.13.2 Analog start_adc 36
5.13.3 Analog end_adc 36

5.13.4 Analog adjustment interval 36
5.13.5 Analog regulating trigger value 36
5.13.6 Analog speed/position 1 37
5.13.7 Analog speed/position 2 37
5.13.8 Analog adjustment lower limit 37
5.13.9 Analog adjustment upper limit 38
5.13.10 Analogue input read 38
5.14 Synchronous position Motion mode 38
5.14.1 SP speed 38
5.14.2 SP position 38
5.14.3 SP commands 39
5.15 Soft limit function 39
5.16 Power loss behavior 40
5.16.1 Power-down behavior control word 40
5.16.2 Power-down off-line voltage 40
5.16.3 Power voltage read 41
5.17 Open and closed loop function switching 41
5.18 Temperature threshold 41
5.19 Motor wiring configuration 41
5.20 Homing configuration 42
5.20.1 Homing speed 42
5.20.2 Homing distance 42
5.21 offline programming 42
5.21.1 The total number of offline program instructions 43
5.21.2 Offline automatic operation enable 43
5.21.3 Offline program pointers 43
5.21.4 Set the content of the offline command 43
5.21.5 Save offline commands 43
5.21.6 Test run instructions 44
6 Electrical Characteristics 44
7 Dimensions 44
8 Appendix 1 instruction table 46
9 Appendix 2 Communications Example 49
9.1 Modbus/RTU supported function codes 49
9.2 Master Station communication parameter setting： 49
9.3 Modbus master station message write operation 49
9.4 Modbus/RTU control examples 49
9.4.1 Relative position control 49
9.4.2 Absolute position control 50
9.4.3 Speed mode control 50
9.4.4 Commonly used fixed parameter settings 51
10 Appendix 3 CRC code 51
11 Appendix 3 Modbus/RTU 16bits CRC check example 52
12 Appendix 2 Error code table 55

1 Introduction

1.1 Statement of intellectual property right

PMC006B1 series controller has been applied for the following national patent:

• Controller scheme and method have been applied for the protection of the invention patent.

• Controller circuit has been applied for the protection of utility model patent.

• Controller appearance has been applied for the protection of appearance patent protection.

PMC006B1 series controller has embedded firmware code, it would be considered as a violation of intellectual property protection act and regulations that any behavior of trying to destroy the function of firmware code protection. If this behavior acquires the software or other achievements of intellectual property protection without authorization of CQPUSI, CQPUSI has the right to stop such behavior by filing a lawsuit according to the act.

1.2 Disclaimer

The using method of the device and other content in the description of this manual is only used to provide convenience for you, and may be update in futu-re version. To ensure the application conforms to the technical specifications is the responsibility of your own. CQPUSI does not make any form of statement

or guarantee to the information, which include but not limited to usage, quality, performance, merchantability or applicability of specific purpose. CQPUSI is not responsible for these information and the consequences result caused by s-uch information. If the CQPUSI device is used for life support and/or life saf-ety applications, all risks are borne by the buyer. The buyer agrees to protect the CQPUSI from legal liability and compensation for any injury, claim, lawsu-it or loss caused by the application.

2 Overview

2.1 General Description

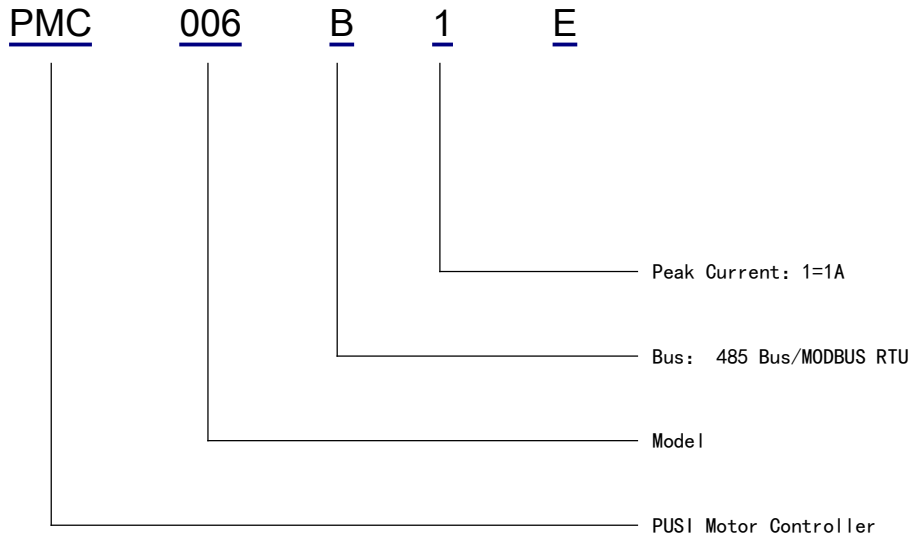
PMC006B1 is a kind of miniature integrated stepper motor microstepping controller, which can be directly installed in the rear of NEMA11 series stepper motor. The PMC006 series controllers are available in a variety of models based on MODBUS RTU bus control and different current levels. With the PMC006B1 stepper motor controller, it is easy to implement an industrial control network system with up to 32 nodes, and encoder-based closed-loop control can be realized according to user requirements. It is suitable for a wide range of high-precision and wide-range industrial applications.

2.2 Features

- ✓ Wide range of 9-36V single voltage supply
- ✓ Output current 0.2A ~ 2A. Adjustable phase current by commands
- ✓ Automatic control of S curve acceleration and deceleration
- ✓ Support Position/Velocity/PP/PV mode etc. motion mode
- ✓ 4000PPR incremental encoder by default; Supports single-turn and multi-turn absolute encoders;
- ✓ Support limit function;
- ✓ Support normal open-loop mode, normal closed-loop mode, closed-loop high-speed torque mode
- ✓ Open-loop and closed-loop applications can be switched
- ✓ Support for enabling the automatic deviation correction function in high-speed torque mode
- ✓ Support 0/2/4/8/16/32/64/128 microstepping resolution
- ✓ Suitable for 4/6/8 lines of 2 phase stepper motor
- ✓ Automatic over-temperature, over-current, under-voltage and over-voltage protection
- ✓ Analog Input Application (Special edition)

2.3 Production & Ordering Information

In order to serve you quicker and better, please provide the product number in following format when ordering PMC006B1：

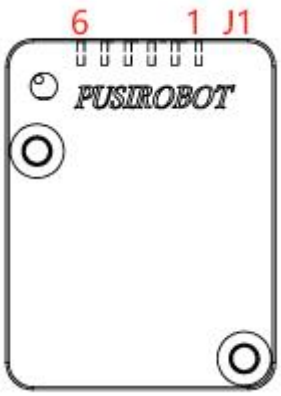


Remark:

E：Closed-loop type.

3 Connector Description

3.1 Terminal port location



3.2 Motor connection J1

Pin no	1	2	3	4	5	6
Designator	VCC	GND	485B	485A	EXT2/AIN	DVDD

Description:

VCC：Supply voltage, 9-36V.

GND：Supply voltage ground.

485B: Connect the transceiver interface of the 485 module

485A: Connect the transceiver interface of the 485 module

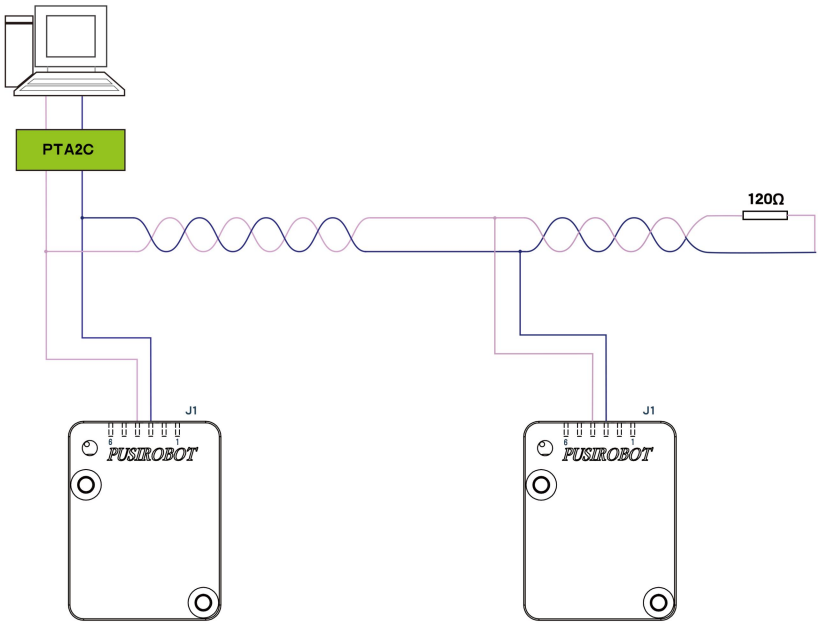
DVDD：Controller voltage output(+5V). Maximum current is 100mA.

EXT2/AIN: Limit interface (or analog input interface), 0-3.3V

WARNING: The voltage of all signal ports must be between -0.3V~ 5V, and exceeding the range may cause permanent damage to the controller.

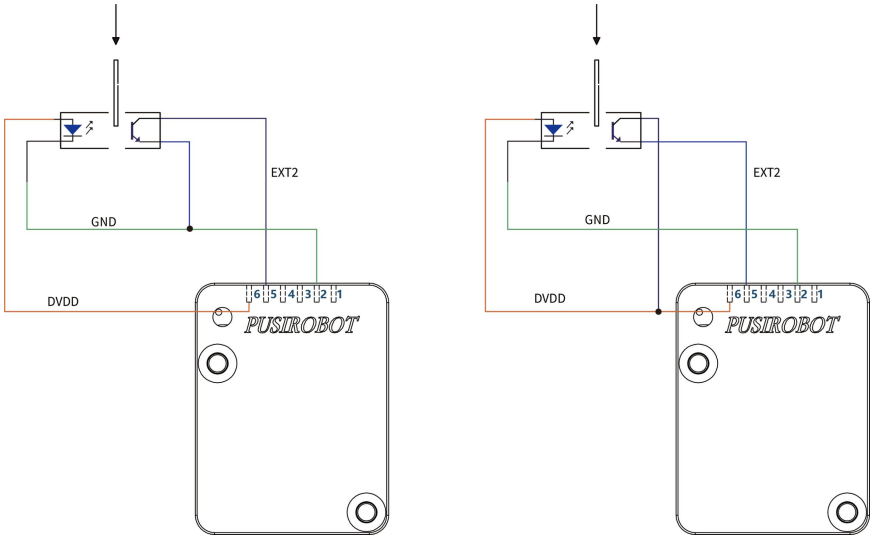
3.3 Modbus network connection

Modbus/RTU has a common physical layer with standard RS-232 or RS-485, and can be configured with 1~32 slave addresses; RS-485 networks are built topology, usually with a 120 ohm termination resistor connected in parallel at the end of the slave device. Modbus/RTU supports half-duplex wiring, and we generally recommend half-duplex wiring to build RTU communication networks.



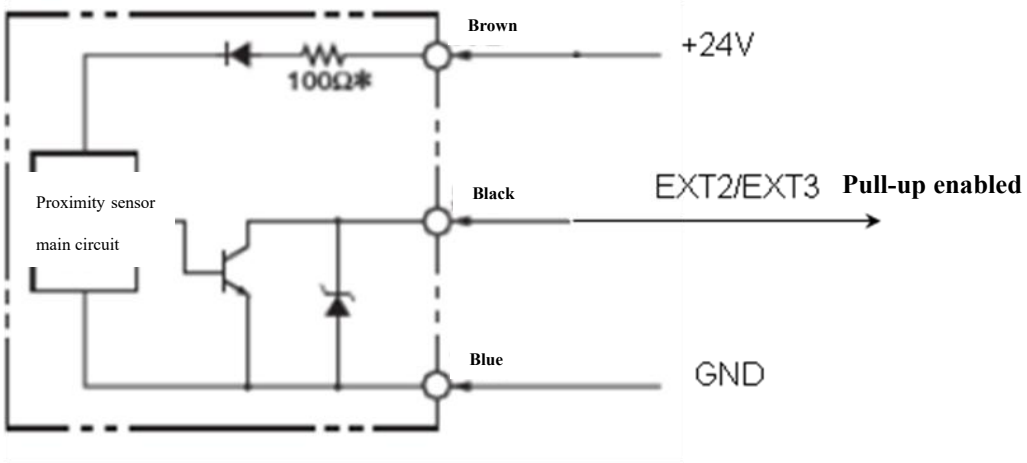
3.4 Limit switch connection

The PMC006B1 EXT2 pin can be used to connect an external limit (home position) switch, and the trigger mode of each pin can be selected in real time by command. GPIO8 is configured as a high-level output, as shown on the left in the figure below, EXT2 is set to pull-up enable, and the rising edge is triggered. As shown on the right in the figure below, EXT2 is set to pull-down enable and falling edge trigger.



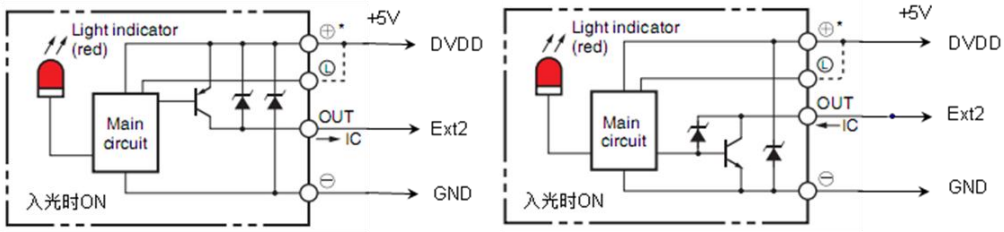
3.5 Other limit switch connections

The EXT2 pin of the PMC006B1 controller can be used to connect to an external limit switch, but its acceptance voltage cannot exceed 5V. Users can freely configure the trigger edge and pull-up mode, the initial level will change with the configuration mode, the default falling edge is set to high, and the rising edge is set to low. When using a 24V HVDC three-wire NPN open-collector output proximity switch (such as Omron E2EC/X☐C☐ or E2E-X☐D1S series), the connection method is shown in the figure below, and the EXT2 needs to be configured with the initial level high and the rising edge triggered. Since the input port can only accept a 5V voltage range, it cannot be connected to a 24V input signal.



There are two types of common external limit switches, one is a four-wire through-beam optocoupler without an internal circuit, and the other is a three-wire or five-wire optocoupler with an internal circuit, which is divided into PNP type and NPN type. And there is a difference between light conduction and shading conduction. Different settings are required for different types of optocouplers. For NPN or PNP type sensors with internal circuitry, it is necessary to select the appropriate operating mode and trigger method. On the left of the figure below is the PNP sensor, the operation mode is light on, and the EXT2 port is configured to trigger the falling edge and enable the pull-down. On the right side of the figure is the NPN sensor, the action mode

is ON when the light is in, and the EXT2 port is configured to be triggered by the rising edge and enabled by pull-up.



3.6 Mechanical switch connection

When using mechanical button switches or relay contacts for limiting, direct connection should be used. When the initial level is high, the signal interface is displayed as low when connected to GND, as shown on the left side of the figure below. When the initial level is low, it is displayed as high when connected to the DVD, as shown on the right in the figure below.



3.7 Analog Adjusting Speed/Positioning

PMC006B1 controller can use the analog speed control function as well as the analog positioning function in offline operation mode, in which case the EXT2 pin is used as an analog input port.

In offline mode, the EXT2 pin is used as an analog input port and can be directly connected to an external input voltage in the range of 0~3.3V. When using PLC or other industrial control equipment to output 0-10V, 0-24V or 4~20mA analog control, special instructions are required to distinguish the version.

Analog speed regulation: Set a speed range, enter the minimum value corresponding to the minimum value of the speed, and enter the maximum value corresponding to the maximum value of the speed. By adjusting the input value, the motor rotates at the corresponding speed.

Analog positioning: Set a position interval, enter the minimum value corresponding to the position minimum, and enter the maximum value corresponding to the position maximum. The current position of the motor is adjusted to change the input value so that the motor stops at the specified position.

Note: If you need to purchase, please contact the sales staff in advance.

4 Drive mode

4.1 Normal open-loop/closed-loop mode

The applicable speed range of normal mode is 10-1200RPM, and the unit of operating parameters such as speed and start-stop speed is PPS. Open-loop and closed-loop switching can be made by modifying register 0x6036-6037.

4.2 High-speed torque mode

The high-speed torque mode is suitable for the speed range of 1000-5000RPM, and the controller can be switched from normal mode to high-speed torque mode by enabling the torque loop (0x605d). The high-speed torque function is as follows:

1. In torque mode, the motor will automatically match the magnetism when powered on to ensure accurate position;

2. The controller microstepping is fixed and cannot be manually modified;

3. The automatic deviation correction function is started when the pre-filter > 20;

4. If disabled, the PID parameter does not take effect. After enabled, the default parameters KP 15, KI 8, and KD 25 are modified according to the operation requirements (it is not recommended to modify them unless there are special requirements), the value is written to the register, and the power-off save instruction is written once before it takes effect. The closed-loop version works.

5. The torque mode will automatically slow down when encountering resistance, and will automatically increase speed when the resistance decreases; The torque is maintained at a constant maximum value during automatic deviation correction, and the deviation correction speed is 400RPM;

6. Units of High Speed Torque Mode:

a. Start speed, stop speed, maximum speed: 0.1lines/ms, for example, set 8000, that is, 800000lines/s, 4000ppr encoder is 3000rpm; Default parameters: startup speed 20 x 0.1lines/ms, stop speed 20 x 0.1lines/ms

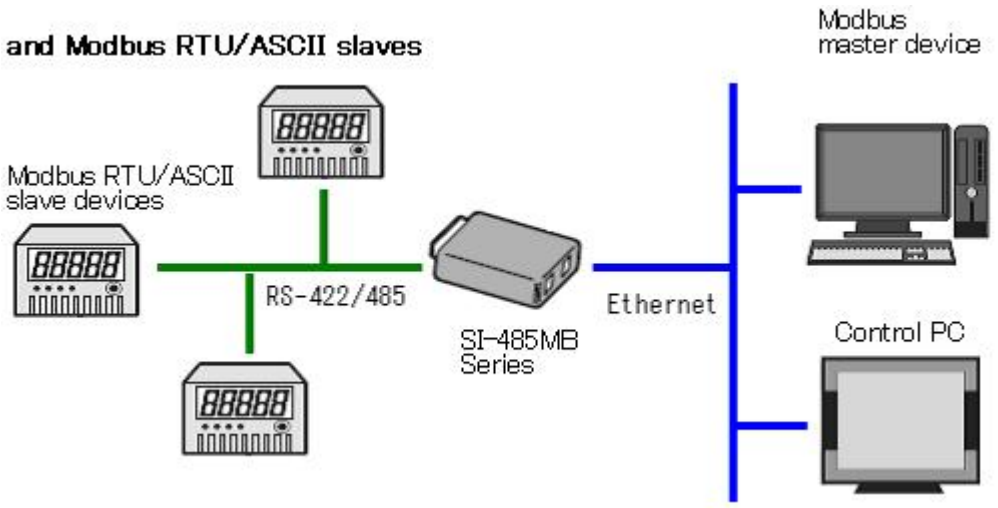
b. Acceleration and deceleration: 0.01 lines/ms^2, for example, 5 represents 50000 lines/s^2, default parameters: acceleration 32 * 0.01 lines/ms^2, deceleration 32 * 0.01 lines/ms^2

5 MODBUS communication

5.1 MODBUS introduction

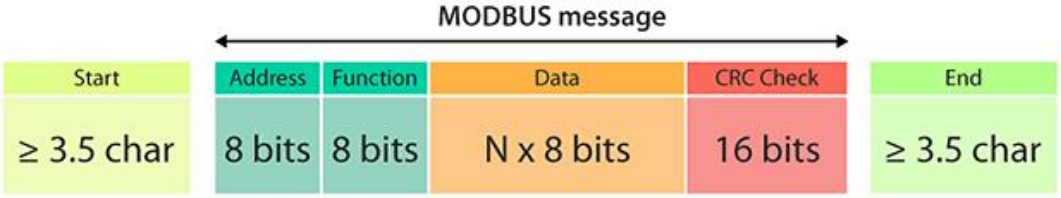
Modbus protocol, which is a bus protocol that allows the master station and one or more slave stations to share data, consists of 16-bit registers. The master can read and write individual or multiple registers.

the standard Modbus port on the controller is using a RS-232 compatible serial interface that defines connectors, wiring cables, signal levels, transmission baud rates, and parity. Controller communication uses master-slave technology, that is, the host can start data transmission, called query. Other devices (slave) return the response to the query or process the actions required by the query. Host equipment shall include master processors, programmers and PLC. The slave includes a programmable controller, a servo drive and a step drive. MODBUS RTU network structure is shown below：



5.2 MODBUS frame structure

Modbus/RTU is a master-slave technique, and the CRC verification range is from the device address bit to the data bit; the detailed message format of each function code, please see appendix. Modbus/RTU message frames are as follows:



as a master-slave communication technology, data can be transmitted between a master station (e.g.: PC) and 32 substations (e.g.: PMC007Bx). The following protocols must be observed between master and slave stations:

1) All information transmitted on the RS485 communication loop will be initialized and controlled by the master station;

2) No communication can begin at a sub-station;

3) All communications in the RS485 loop are transmitted in frame format;

4) If the master or sub-station receives a frame format containing unknown commands, it does not respond;

5.3 MODBUS communication configuration

PMC006 is set to 1 node ID and baud rate of 9600 by default, and users can modify the settings through the supporting MODBUS master debugging tool.

5.3.1 Node ID

Object name	Node ID
Instruct address	0x2028

Object type	U8,rw
Range	1-127
Storage type	ROM
Default value	1

5.3.2 Baud rate

Object name	Baud rate
Instruct address	0x202a/0x202b
Object type	U8,rw
Range	4800/9600/19200/38400/51200/115200
Storage type	ROM
Default value	9600

5.3.3 Group ID

Object name	Group ID
Instruct address	0x200e
Object type	U8,rw
Range	1-127
Storage type	RAM
Default value	1

In a MODBUS RTU network, the function remains.

5.4 System information acquisition

5.4.1 Device node name

Object name	Device node name
Instruct address	0x101a..21
Object type	string,ro
Range	-
Storage type	ROM
Default value	-

5.4.2 Hardware version

Object name	Hardware version
Instruct address	0x1022..23
Object type	string,ro
Range	-
Storage type	ROM
Default value	-

5.4.3 Software version

Object name	Software version
Instruct address	0x1024..25

Object type	string,ro
Range	-
Storage type	ROM
Default value	-

5.4.4 System control

Object name	System control
Instruct address	0x200f
Object type	U8,ro
Range	1，2，3
Storage type	RAM
Default value	-

System control values are defined as follows:

1: Jump to bootloader

2: Save Object Dictionary parameters

3: Reset factory settings

Note: The parameters stored in the object dictionary with the storage type ROM are temporarily stored in memory after being written by the command, if you need to save them permanently, you need to save the object dictionary parameters by powering off. This operation cannot be written to the process to prevent fast erase and write the ROM to accelerate the consumption of service life.

5.5 Motor control parameters

5.5.1 Error status

Object name	Error status
Instruct address	0x6000
Object type	U8,rw
Range	bit
Storage type	RAM
Default value	0

Driver state are defined as follows:

Bit0:OTS, Overheating Warning

Bit1:AOCP, motor overcurrent

Bit2: BOCP, the core position is wrong

Bit3:APDF, the rotation direction is wrong when magnetizing

Bit4:BPDF, the rotation angle is wrong when the magnet is correct

Bit5: UVLO, Low supply voltage warning

Bit6: SERR, encoder packet error

Bit7: EERR, the encoder is in the wrong position

For example, write 0x18 to the corresponding bit, clear the bit3 and bit4 flag bits, that is, clear the wrong rotation direction when pairing magnetism and the wrong state of rotation angle when facing magnetism, and write FF as fully clear.

Note: bit2/bit3/bit4/bit6/bit7 only works for high-speed torque mode.

5.5.2 Controller status

Object name	Controller status
Instruct address	0x6001
Object type	U8,rw
Range	bit
Storage type	RAM
Default value	0

Controller status is defined as follows:

Bit0：External stop 1

Bit1：External stop 2

Bit2：Stall state

Bit3：busy state

Bit4：External stop 3

Bit8：Resets the status bit

In addition to the busy state, you can write 1 to clear the response state, and the usage is the same as 6000h.

5.6 Basic Instructions

5.6.1 Stop stepping command

Object name	Stop stepping command
Instruct address	0x6053
Object type	U8,rw
Range	0
Storage type	RAM
Default value	0

The instruction immediately terminates the motor operation without a deceleration process.

5.6.2 Micro stepping

Object name	Micro stepping
Instruct address	0x600c
Object type	U16,rw
Range	0,2,4,8,16,32,64,128
Storage type	ROM
Default value	0

The motor is 360° per revolution, and the step angle of a conventional 2-phase stepper motor is 1.8°, i.e. 200 pulses per revolution are required at 0 microstepping. In the same way, 400 pulses per revolution are required under 2 microstepping, and 1600 pulses per revolution are

required under 8 microstepping.

In high-speed torque mode, microstepping can only be read and cannot be set.

5.6.3 Maximum phase current(Operating current)

Object name	Maximum phase current(Operating current)
Instruct address	0x600d
Object type	U16,rw
Range	0-2000
Storage type	ROM
Default value	658

The maximum phase current is the supply current to the motor when it is working normally, and is generally set to the rated current of the motor. In some cases, it can be adjusted appropriately, generally with a range of +-20% and not more than 50%. Excessive current will cause the motor to heat up seriously, and long-term operation of the motor may have the risk of demagnetization, affecting the service life of the motor.

5.6.4 Motor position

Object name	Motor position
Instruct address	0x600e..f
Object type	S32,rw
Range	-2^31 ~ (2^31-1)
Storage type	RAM
Default value	0

The current motor position is represented by the number of pulses per turn. Incremental is a quadruple frequency of PPR, for example, 4000 wires are 16,000 pulse motors rotated once. The absolute accuracy is 2^14, 16384 pulses per turn.

In the open-loop mode, the motor position is not saved, and the position information is automatically cleared after the controller is powered off.

In incremental closed-loop mode, the motor position is not saved in conventional applications. There is a switch inside the controller that records the position of the motor before the power is lost. It is generally used in conjunction with brakes to prevent the motor from moving when the power is lost. Otherwise, the saved location is not informative.

In the closed-loop mode of the absolute value of a single turn, the controller can record the change of the motor position within one turn, and the value of the extra turn will be automatically discarded after the power is lost. For value storage greater than one circle, you can refer to the same application method for incremental expressions.

Multi-turn absolute closed-loop mode, which can record the change of motor position in real time, and can still work after power failure. The position of the limit switch can be dispensed with.

5.6.5 Encoder position (absolute value encoder closed loop)

Object Name	Encoder position

Instruct address	0x6089..8a
Instruct address	S32,rw
Object type	-2^31 ~ (2^31-1)
Range	RAM
Storage type	0

For now only single-turn and multi-turn value reading is open, and the encoder feedback position is read in real time, and the value cannot be changed by command.

5.6.6 Current reduction

Object name	Current reduction coefficient
Instruct address	0x6010
Object type	U8,rw
Range	0-3
Storage type	ROM
Default value	2

The value is defined as follows:

0：Decay 0%;

1：Decay 50%

2：Decay 75%

3：Decay 87.5%

0-3 corresponds to 4 gears of 100%, 50%, 25% and 12.5% idle current supply, the ratio is based on the percentage of the maximum phase current of 600Bh. The default is 25% for 2nd gear, and when the motor is idle, it will switch to idle current power supply.

5.6.7 Motor enable

Object name	Motor enable
Instruct address	0x6011
Object type	U8,rw
Range	0,1
Storage type	RAM
Default value	1

The value of the motor is defined as follows:

0: Offline

1: Motor enable

Release control of the motor immediately after setting offline, the motor loses power supply and loses holding torque.

The controller is enabled by default, and no settings need to be manually written for power-on.

5.6.8 Operation mode

Object name	Operation mode
Instruct	0x6007

address	
Object type	U8,rw
Range	0,1,4,5
Storage type	RAM
Default value	0

The value of the motor operation mode is defined as follows:

0：Position mode

1：Speed mode(including analog speed regulation)

4：PP (Profile Position) mode (including analog positioning)

5：PV(Profile Velocity) mode

Location Mode:

The setting of the position mode is simple, easy to operate, you can use the relative or absolute running motion mode, after setting the running speed and direction of movement, give the corresponding running instructions, the motor will move to the corresponding position, because the acceleration and deceleration is the gear setting, the acceleration and deceleration can not be accurately adjusted, the equipment or parameters need to be debugged according to the load situation, the stepping command setting needs to be sent when the controller is idle, so to change the speed, change the stepping position, you need to wait for the motor to stop and then send the corresponding instructions, before it can run, Otherwise, the command will not be received.

Speed Mode:

After entering the speed mode, set the running speed and the motor will rotate all the time. Unless the limit or stop stepping command is triggered or the speed mode is exited, the motor will rotate according to the running speed, and the running speed can be changed (positive positive rotation, negative negative reversal) and speed adjustment can be adjusted by setting the running speed during operation, without stopping the movement. The speed mode and the position mode share the same parameter register, and the acceleration and deceleration and start-stop speed cannot be modified during operation. The speed mode is easy to set and stable, which is convenient for long-distance variable speed movement. If it is at high speed, changing direction directly may cause stalling. The speed setting needs to be adjusted according to the actual operation.

PP Mode:

PP mode uses trapezoidal acceleration and deceleration movement, each run can change the acceleration and deceleration, start and stop speed, running speed, running direction and target position, through the unified execution of the control word, you can switch the running task or preset the next task during the movement. PP mode can also use absolute or relative operation modes, and the acceleration and deceleration value settings are open to better control the smoothness of the operation and perform more complex operation planning.

PV Mode:

PV mode is called Profile Velocity Mode, which also uses trapezoidal acceleration and deceleration mode, and shares parameters such as start-stop speed, acceleration and deceleration, and running speed with PP mode. After entering the PV mode, you need to set the running speed for the first run, use the control word to start the run, and then change the speed only need to modify the value of the running speed.

During PV mode operation, the motor will rotate according to the set running speed, and the stop can be achieved by exiting or switching the mode/setting the running speed to 0/step stop command. During the operation, the running speed can be set to change the running direction (positive positive rotation, negative negative reversal) and adjust the speed magnitude, without pausing the movement to set the parameters. Because the acceleration and deceleration are controllable and the acceleration and deceleration are uniform, it is easier to calculate the running time, and changing the running direction at high speed will not cause stalling.

5.6.9 Real-time speed(Closed-loop)

Object name	Real-time speed(pps)
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x607f/6080
Object type	S32, ro
Range	-300000~+300000
Storage type	RAM
Default value	0

For parameters that are only available under the closed-loop function, an error will be reported when the open-loop reading is performed.

Real-time velocity is a signed variable with a direction of 1 when positive and a direction of 0 when negative. At high speed torque, the reading value is 100 times the actual operating speed.

5.7 Position mode

5.7.1 Position mode rotating direction

Object name	Rotating direction
Instruct address	0x6002
Object type	U8,rw
Range	0,1
Storage type	RAM
Default value	1

The value of the rotation direction is defined as follows:

0: forward

1: backward

5.7.2 Position mode operating speed

Object name	Location mode operating speed（pps）
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x6003..4
Object type	S32,rw

Range	-200000 ~ +200000
Storage type	RAM
Default value	0

Note: the speed is a signed variable. Positive represents that the direction is 1, and negative represents that the direction is 0. So in the displacement mode it is recommended to set the speed firstly , and then set the direction.

5.7.3 Position mode start speed

Object name	Start speed(Unit: pps)
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x6008
Object type	U16,rw
Range	0-0xFFFF
Storage type	ROM
Default value	Normal mode 600 High-speed mode 32

5.7.4 Position mode stop speed

Object name	Stop speed(Unit: pps)
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x6009
Object type	U16,rw
Range	0-0xFFFF
Storage type	ROM
Default value	Normal mode 600 High-speed mode 32

If the start speed and stop speed jump directly, for example, jump from 0 to the maximum speed of starting speed and then accelerate or decelerate from maximum speed to stop speed and then jump to 0, so the start and stop speed cannot be set to 0.

5.7.5 Position mode acceleration coefficient

Object name	Acceleration coefficient
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x600a
Object type	U16,rw
Range	0-8
Storage type	ROM
Default value	8

5.7.6 Position mode deceleration coefficient

Object name	Deceleration coefficient
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x600b
Object type	U8,rw
Range	0-8
Storage type	ROM
Default value	8

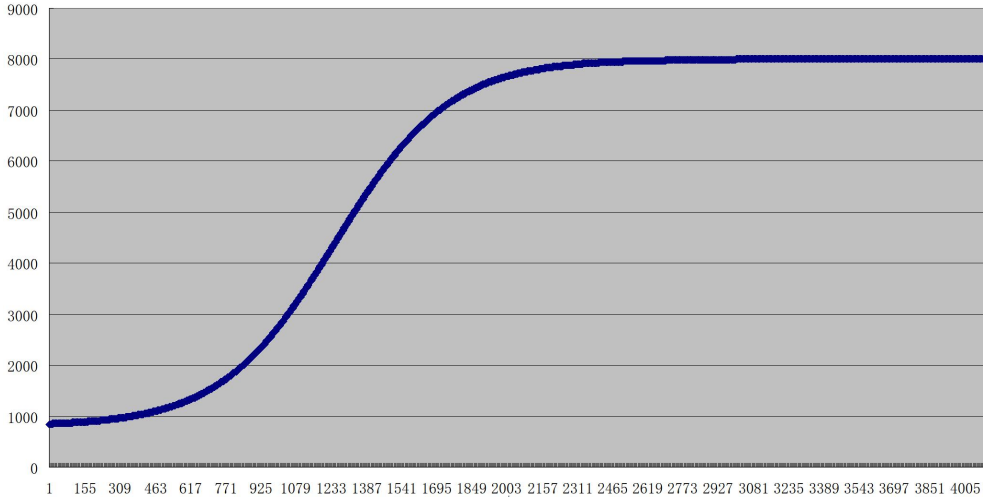


Figure 5-1

The PMC006B1 controller uses S curve acceleration and deceleration. As shown in Figure 5-1, the start-up speed, stop speed, acceleration and deceleration can be configured separately, and the acceleration and deceleration support 1~8 for a total of 8 gears, and the corresponding acceleration values for each gear are as follows.PMC006B1 The acceleration value of the 1st gear is the largest, and the acceleration value of the 8th gear is the smallest.

Gear	Acceleration and deceleration value（PPS2）
0	Acceleration and deceleration cannot be enable
1	77440
2	48410
3	27170
4	21510
5	14080
6	10460
7	6915
8	5210

5.7.7 Position mode relative displacement command

Object name	Relative displacement command

Instruct address	0x6005..6
Object type	U32,rw
Range	0x0-0xFFFFFFFF
Storage type	RAM
Default value	0

Relative to the current motor position, let the motor run for a set number of steps, and an error will be reported if the displacement is 0. Write step number, then the controller will control the motor rotate a given number of steps which is calculated based on the current microstepping settings at the setting direction, speed and acceleration.

In open-loop mode, the number of steps is calculated based on the current microstepping setting.

In the incremental encoder closed-loop mode, the input unit is 4 times the resolution of the encoder, for example, CPR=4000, then the motor rotates once when 16000 is input.

In the closed-loop mode of the absolute encoder, the input unit is the same as the encoder counting unit, for example, if the accuracy is 14 bits, then the motor rotates once when 16384 is entered.

5.7.8 Position mode absolute displacement command

Object name	Absolute displacement command
Instruct address	0x6044..45
Object name	S32,rw
Range	-2^31 ~ (2^31-1)
Storage type	RAM
Default value	0

Run directly to the specified position, and an error will be reported if the displacement is 0. The absolute displacement command gives the target position, and the controller will automatically calculate the direction and the required number of steps, and control the stepper motor to rotate the specified position according to the set speed and acceleration.

Note: The above operating parameters are only for normal mode

5.8 External emergency stop

The PMC006B1 controller provides a dedicated limit switch input port EXT2, which can be used as an emergency stop or zero search function.

When the emergency stop function is enabled, if the corresponding input pin detects a valid trigger edge, the controller will lock the motor and stop responding to the stepping command, and the user can read the controller status to see which input pin triggered the emergency stop. Only when the user clears the corresponding status bit does the controller continue to respond to new step commands.

5.8.1 External emergency stop enable

Object name	External emergency stop
Instruct address	0x6013
Object type	U8,rw
Range	Bit
Storage type	ROM
Default value	0

This represented by 1bit that each external emergency stop enable. 0 indicates the prohibition, and 1 indicates enabling. Its definition is as follows:

bit1: External emergency stop 2 enable settings

Set to 0 to turn off the limit enable, set to 2, turn on the limit enable.

5.8.2 The trigger mode of external emergency stop

Object name	The trigger mode of external emergency stop
Instruct address	0x6014
Object type	U8,rw
Range	Bit
Storage type	ROM
Default value	0

Trigger mode of each external emergency stop is represented by 1bit. 0 indicates falling edge trigger, and 1 indicates rising edge trigger. Its definition is as follows:

bit1： The trigger mode of external emergency stop 2

The same goes for setup and use

5.8.3 Sensor type

Object name	Sensor type
Instruct address	0x6015
Object type	U8,rw
Range	0-1
Storage type	ROM
Default value	0

You can modify the initial level value of EXT2 under different trigger modes, the default value is 0, the initial high level of the falling edge, and the initial low level of the rising edge
Set to 1, the falling edge is initially low, and the rising edge is initially high. 0000000000
Its definition is as follows:

0: when the trigger mode is configured as the rising edge, the controller is configured as the internal pull-down resistance,When configured as a falling edge, the controller is configured with an internal pull-up resistor; typically used for an NPN type of sensor;

1: When the trigger mode is configured as the rising edge, the controller is configured as the internal pull-up resistance, when the trigger mode is configured as the falling edge, the

controller is configured with the internal pull-down resistance, typically used for an PNP type of sensor;

5.8.4 Debouncing delay

The trigger delay of external emergency stop can be modified by 0x6042 object. The controller delays the setting time after the detected edge signal, and then detects whether its level state is correct or not, then triggers the motor to stop urgently, otherwise the motor will continue to rotate.

Object name	EXT1/EXT2/EXT3 stabilize delay（ms）
Instruct address	0x6042
Object type	U8,rw
Range	0~200
Storage type	ROM
Default value	30

The debouncing delay parameter needs to be stabilized in MS, and it can still meet the requirements of limit level detection, which can be analogous to the debounce delay application of keyboard keys.

5.9 General IO port

The PMC006B1 controller provides 2 general IO (GPIO) ports, 1 external emergency stop input (EXT2) ports

5.9.1 IO port direction

Object name	IO port direction
Instruct address	0x602e
Object type	U16,rw
Range	0, 1
Storage type	ROM
Default value	0

The direction of each IO port is represented by 1bit. 0 represents input, and 1 represents output. The meaning of each bit is as follow:

Bit0：GPIO1
Bit1：GPIO2
Bit2：GPIO3
Bit3：GPIO4
Bit4：GPIO5
Bit5：GPIO6
Bit6：GPIO7
Bit7：GPIO_EXT1
Bit8：GPIO_EXT2
Bit9：GPIO_ENC1

Bit10：GPIO_ENC2

Bit11：GPIO8

Bit12: FSET

Bit13: GPIO9

Among them, the direction of the emergency stop input port and the encoder input port is fixed as the input port and cannot be configured.

2、IO port configuration

Object name	IO port configuration
Instruct address	0x602f..30
Object type	U32,rw
Range	0-0x3fffff
Storage type	ROM
Default value	0

Each port is configured by 2 bits. If the IO port is configured as a input port, the meaning of the value is as follows:

0：FLOATING

1：IPU

2：IPD

3：AIN

If the IO port is configured as a output port, the meaning of the value is as follows:

0：OD

1：PP

The definition of the IO port configuration is defined as follows:

Bit1-0：GPIO1

Bit3-2：GPIO2

Bit5-4：GPIO3

Bit7-6：GPIO4

Bit9-8：GPIO5

Bit11-10：GPIO6

Bit13-12：GPIO7

Bit15-14：GPIO_EXT1

Bit17-16：GPIO_EXT2

Bit19-18：GPIO_ENC1

Bit21-20：GPIO_ENC2

Bit23-22：GPIO8

Bit25-24: FSET

Bit27-26: GPIO9

5.9.2 General IO port value

Object name	General IO port value
Instruct address	0x6031
Object type	U16,rw

Range	bit
Storage type	RAM
Default value	0

The value of each IO port is represented by 1bit, 0 indicates a high level, 1 indicates a low level, and writing value to the port is not valid for the input port. The meaning of each bit is as follows:

Bit0:GPIO1,
Bit1:GPIO2,
Bit2:GPIO3,
Bit3:GPIO4,
Bit4:GPIO5,
Bit5:GPIO6,
Bit6:GPIO7,
Bit7:GPIO_EXT1,
Bit8:GPIO_EXT2,
Bit9:GPIO_ENC1,
Bit10:GPIO_ENC2,
Bit11:GPIO8,
Bit12:FSET,
Bit13:GPIO9

5.10 Closed-loop control

PMC006B1 supports 200-4000CPR incremental photoelectric encoder and uses PID to realize closed loop control. The following is a detailed description of the closed-loop parameters.

5.10.1 Encoder resolution

Object name	Encoder resolution
Instruct address	0x6054
Object type	U16,rw
Range	Incremental encoder closed loop: 200,400,500,600,800,1000,1200,1600,2000,4000 Absolute encoder closed loop:14
Storage type	ROM
Default value	4000,14

Note: After changing the encoder resolution, the power of controller must be re-energize.

PPR: Encoder Accuracy/Number of Lines/Resolution

CPR: Encoder count

CPR = PPR * 4 CPR is the number of steps per lap.

5.10.2 KP parameter

Object name	KP parameter

Instruct address	0x6057
Object type	U8,rw
Range	1-255
Storage type	ROM
Default value	32

This parameter affects the transient response characteristic of the system.

5.10.3 KI parameter

Object name	KI parameter
Instruct address	0x6058
Object type	U8,rw
Range	1-255
Storage type	ROM
Default value	3

This parameter affects the cumulative error characteristics of the system.

5.10.4 KD parameter

Object name	KD parameter
Instruct address	0x6059
Object type	U8,rw
Range	1-255
Storage type	ROM
Default value	16

This parameter affects the transient response characteristic of the system.

5.10.5 Pre-filtering parameter

Object name	Pre-filtering parameter
Instruct address	0x605a
Object type	U8,rw
Range	1-128
Storage type	ROM
Default value	40

This parameter affects the speed characteristics of the system. When speed or microstepping is high, it is recommended to use larger parameter values.

5.10.6 Post-filtering parameter

Object name	Post-filtering parameter
Instruct address	0x605b
Object type	U32, rw
Range	1-65535
Storage type	ROM
Default value	1130

This parameter is reserved for the time being.

5.10.7 Stall length parameter(closed-loop)

Object name	Stall length parameter
Instruct address	0x605c
Object type	U16,rw
Range	1-32768
Storage type	ROM
Default value	640

The threshold for stalling judgment: the higher the value, the lower the sensitivity or the harder it is to detect a stall.

5.10.8 Torque ring enable(High-speed torque mode switch)

Object name	Torque ring enable
Instruct address	0x605d
Object type	U8,rw
Range	0-1
Storage type	ROM
Default value	0

Toggle the switch in high-speed torque mode, write 1 into high-speed torque mode, and write 0 into normal mode.

5.10.9 Autosave when power is off enable

Object name	Autosave when power is off enable
Instruct address	0x605e
Object type	U8,rw
Range	0-1
Storage type	ROM
Default value	0

The closed-loop takes effect, and the controller automatically detects the power-off of the system after it is enabled, and writes the current motor position into the EEPROM. At the same time, for singleturn and incremental encoder motors, it is necessary to ensure that the motor does not rotate after the power failure, which can be used with the solenoid valve. If the motor rotates in the event of a power failure, this saved value is meaningless.

5.11 PP motion mode

Set the working mode to 4 to PP mode(Profile Position Mode), which adopts trapezoidal acceleration and deceleration, and can set the start speed, stop speed, acceleration, deceleration, running speed and target position independently. In the process of PP mode operation, the host computer can be received to write a new set of parameters, and finally by writing the control word to let the controller run smoothly from the previous motion parameters to the new

parameters, or after the old parameters are completed, and then run with the new parameters.

5.11.1 PP mode parameter 1

5.11.1.1 PP mode Acceleration

Object name	PP acceleration， pps/s
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x6067..68
Object type	U32,rw
Range	Normal mode >150 High-speed mode >0
Storage type	ROM
Default value	Normal mode 32000 High-speed mode 50

5.11.1.2 PP mode Deceleration

Object name	PP deceleration， pps/s
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x6069..6a
Object type	U32,rw
Range	Normal mode >150 High-speed mode >0
Storage type	ROM
Default value	Normal mode 32000 High-speed mode 50

5.11.1.3 PP mode Start speed

Object name	PP start speed， pps/s
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x606b..6c
Object type	U32,rw
Range	Normal mode >150 High-speed mode >20
Storage type	ROM
Default value	Normal mode 600 High-speed mode 100

5.11.1.4 PP mode Stop speed

Object name	PP stop speed
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x606d..6e
Object type	U32,rw
Range	Normal mode >150 High-speed mode >20
Storage type	ROM
Default value	Normal mode 600 High-speed mode 100

5.11.2 PP mode parameter 2

5.11.2.1 PP mode Control word

Object name	PP Control word
Instruct address	0x6070
Object type	U16,rw
Range	ROM
Storage type	0-0xFFFF
Default value	0

The function description for Control word object 6070：

• Bit 4：Start task. Start the task when bit value switch from 0 to 1.

• Bit 5：The task trigged by Bit4 will be immediately executed if this bit is set 1.The task will be executed after last task completed if this bit is set 0.

• Bit 6：The target position (6074..75) is relative position when this bit is set 0, the target position is absolute position if this bit is set 1.

• Bit 8 (Halt): This bit is for PV motion mode, motor will be accelerated to target speed in preferred slope if this bit change from 1 to 0. Motor will be decelerated to zero if this bit change from 0 to 1.

5.11.2.2 PP Status word

Object name	Pp status word
Instruct address	0x6071
Object type	U16,rw
Range	ROM
Storage type	0-0xFFFF
Default value	0

The function of status word is following:

• bit0: When this bit is set, the speed will change upon reaching the first target position. In other words, braking will not occur before reaching the first target, as the motor should not stop at this position.

• bit1: This bit will be set to "1" when the final target has been reached.

• bit2: This bit confirms that a valid new target point has been received. This bit is set and reset synchronously with the "New Target Point" bit in the control word.

Exception: When a motion task is not yet completed, and the next motion task should only execute after the current task is finished, a new motion task is started. In this case, this bit will only be reset when the command has been accepted and the controller is ready to execute the new motion task.

When a motion task is enabled and another motion task has already been set, all other motion tasks will be ignored; to indicate this situation, this bit is set.

5.11.2.3 PP model Running Speed

Subindex 0x03: running speed, the symbol represents the direction of rotation, the positive sign rotates positively, and the negative sign reverses

Object name	Pp running speed
Unit	Normal mode pps High-speed mode 0.1 lines/ms
Instruct address	0x6072..73
Object type	S32,rw
Range	ROM
Storage type	-300000- -150,150-300000
Default value	32000

5.11.2.4 PP model Target Location

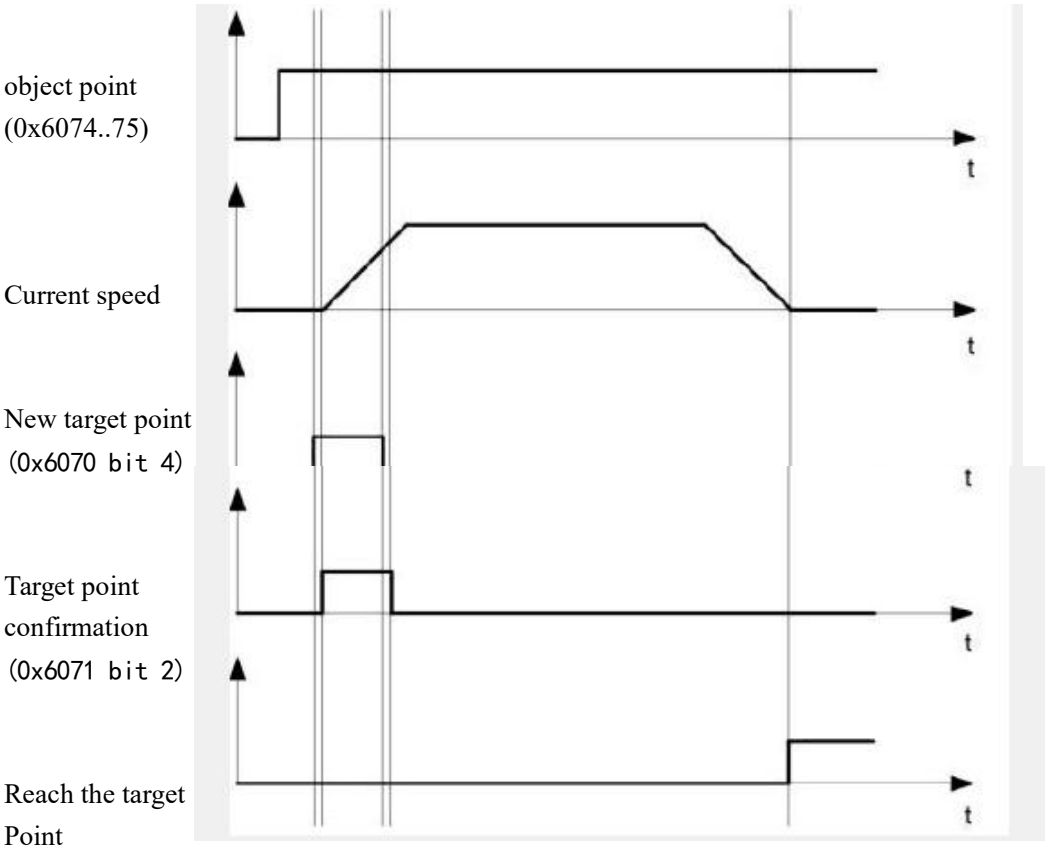
Subindex 0x04: PP Model Target Location

Object name	Pp Target location
Instruct address	0x6074..75
Object type	S32,rw
Range	ROM
Storage type	-2^31~2^31
Default value	0

5.11.3 PP model work timing

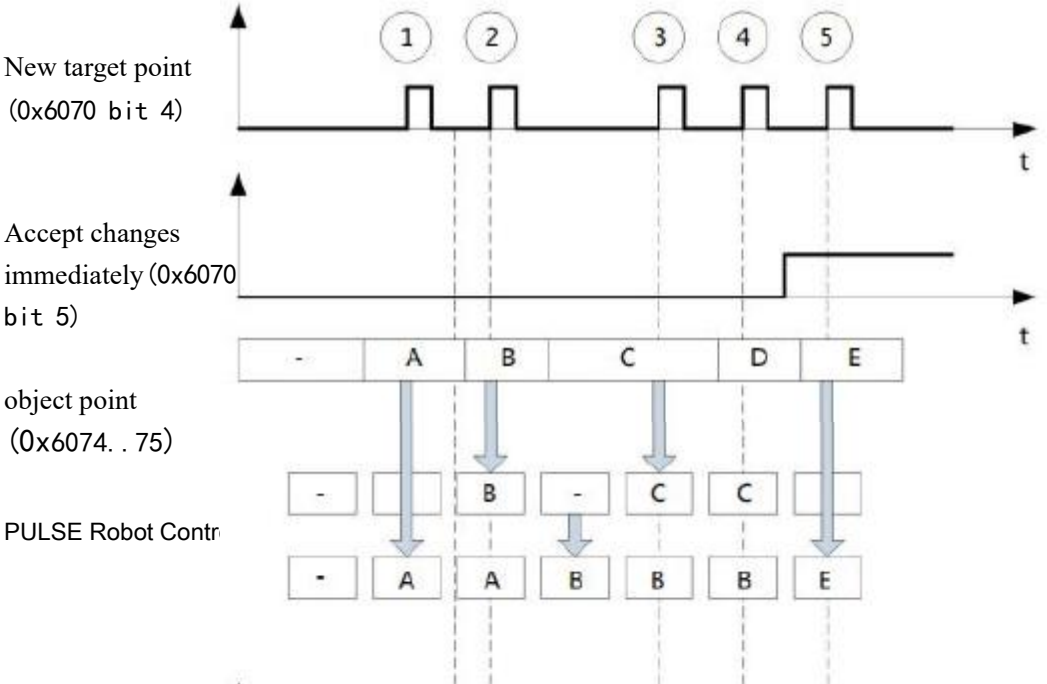
A new target position is set in the target position object (6074..75). Next, the bit 4 in the control word object (6070) is set for trigger the operation command. If the target location is valid, the controller will reply through the bit 12 in the object status word to locate the start of the

operation. When the location is reached, the bit 10 in the status word will immediately set to a "1".



Other running commands can be stored in the cache (see point in time 1 in the figure below), and bit 12 in the status word object (6071 Set the target point response) will be set to "0". During the motion to the target position, a second target position can be sent to the controller to prepare for it. At this point, you can reset all parameters, such as velocity, acceleration, deceleration, etc. (point in time 2). If the cache is idle again, the next point in time can enter the queue (point in time 3).

If the cache is full, the new target point will be ignored (point in time 4). If bit 5 in the control word object (602e, 1 bit: "change the target point now") is set, the controller will not use cache when working, and the new running command will be executed directly (point in time 5).



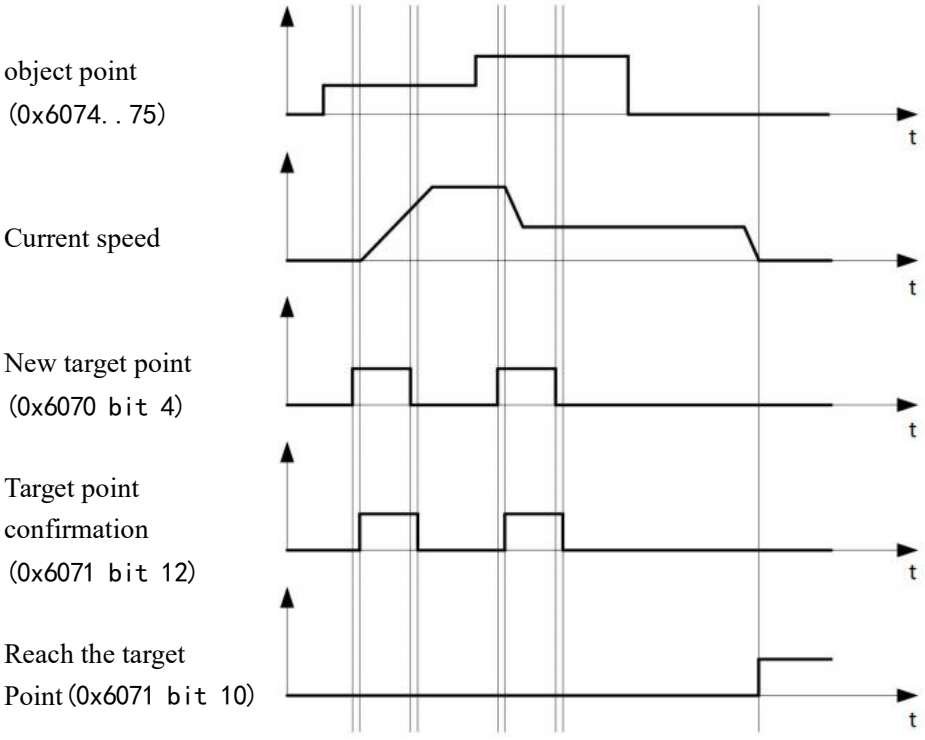
Saved target point
Target point to be
processed

Target point
confirmation
(0x6071 bit 12)

Reach the target
Point (0x6071 bit 10)

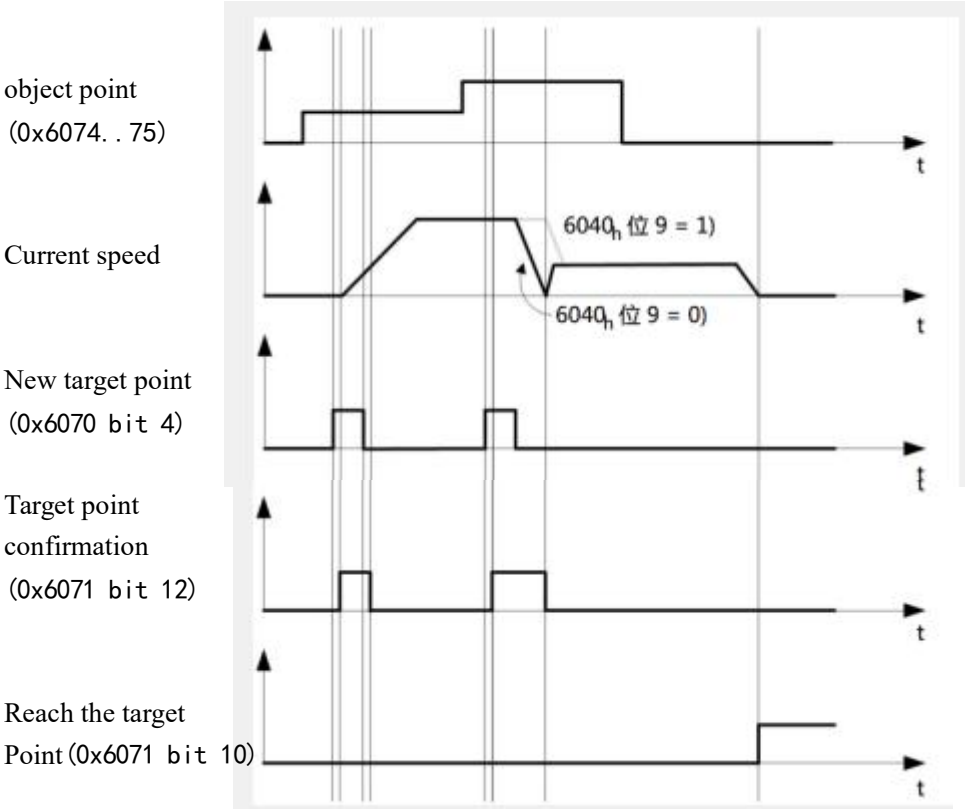
The conversion process of the second target position:

The following figure shows the conversion process of the second target position when moving to the first target position. In this figure, the bit 5 of the control word object (602e, 1) is set to "1" and the new target value will be accepted immediately.



The method of moving to the target position:

If the bit 9 in the control word object (6070) is a "0", it will first fully travel to the current target position. In this example, the final speed of the first target position is equal to zero. If bit 9 is set to "1", the final speed will be maintained until the target position is reached, and then the newly set motion parameters will take effect.



5.12 PV Mode

The working mode is set to 5 into (Profile Velocity Mode) PP mode, which adopts ladder acceleration and deceleration, and shares the starting speed, stop speed, acceleration, deceleration and running speed parameters with PP mode.

When the value of bit 8 (Halt) of the control word changes from "1" to "0", the motor will accelerate to the target speed with a preset starting speed in slope. When the value of the bit changes from "0" to "1", the motor slows down and stops moving. In the process of motion, a new running speed can be sent out, and the controller will smooth over to the newly set speed.

5.13 Analog parameters

The PMC006B1 has an analog signal input port and uses a 12-bit ADC internally. It can be configured into analog mode through software. First, set the analog-related parameters, and enable the analog function. Below is a detailed description of the analog-related features.

5.13.1 Analog enable

Analog enable,1 open, 0 closed

Object name	Analog enable
Instruct address	0x6077

Object type	U8,rw
Range	ROM
Storage type	0、1
Default value	0

5.13.2 Analog start_adc

Analog start_adc, Adjust when the analog curve shifts.

Object name	Analog start_adc
Instruct address	0x6078
Object type	U16,rw
Range	ROM
Storage type	0-4095
Default value	0

5.13.3 Analog end_adc

Analog end_adc, Adjust when the analog curve shifts.

Object name	Analog end_adc
Instruct address	0x6076
Object type	U16,rw
Range	ROM
Storage type	0-4095
Default value	4095

5.13.4 Analog adjustment interval

Analog adjustment interval, Unit ms

The controller checks the analog input value at this time, and if the difference between the AD input value and the last input value is greater than the threshold value, the position will be adjusted once.

Object name	Analog adjustment interval
Instruct address	0x6079
Object type	U16,rw
Range	ROM
Storage type	0-65535
Default value	100

5.13.5 Analog regulating trigger value

The analog quantity adjusts the trigger value, and when the difference between the

acquired AD code and the last acquired AD code is converted to a position greater than this value, the controller will adjust the position once.

Object name	Analog regulating trigger value
Instruct address	0x607a
Object type	U16,rw
Range	ROM
Storage type	0-65535
Default value	30

5.13.6 Analog speed/position 1

Analog speed/position 1: Speed/position corresponding to the starting AD code of the analog quantity

Object name	Analog speed/position 1
Instruct address	0x607b..7c
Object type	S32,rw
Range	ROM
Storage type	-2^31~2^31
Default value	0

5.13.7 Analog speed/position 2

Analog speed/position 2: Speed/position corresponding to AD code 4095

Object name	Analog speed/position 2
Instruct address	0x607d..7e
Object type	S32,rw
Range	ROM
Storage type	-2^31~2^31
Default value	0

5.13.8 Analog adjustment lower limit

Analog adjustment lower limit: It only starts adjusting when the analog value is above this threshold; when it's below, it keeps the minimum speed or position.

Object name	Analog adjustment lower limit
Instruct address	0x6066
Object type	U16, rw
Range	ROM

Storage type	0-4095
Default value	0

5.13.9 Analog adjustment upper limit

Analog adjustment upper limit: Stop adjusting when the analog value exceeds this value, and keep the maximum speed or maximum position when it's above this value

Object name	Analog adjustment upper limit
Instruct address	0x6081
Object type	U16, rw
Range	ROM
Storage type	0-4095
Default value	0

5.13.10 Analogue input read

The default voltage input is 0-3.3V, and special versions can support 4-20mA or 0-24V analog input, 12-bit ADC.

Object name	Analogue input
SDO ID	0x609c
Object type	U16,rw
Range	0~4095
Storage type	RAM
Default value	0

5.14 Synchronous position Motion mode

The synchronous positioning motion mode can first set the absolute position and speed of the specified node to run, and then make multiple axes move at the same time through the synchronous start command.

5.14.1 SP speed

Object name	SP speed
Instruct address	0x6047..48
Object type	S32,rw
Range	-2147483648-2147483647
Storage type	RAM
Default value	2

5.14.2 SP position

Object name	SP position
Instruct address	0x6049..4a

Object type	S32,rw
Range	-2147483648-2147483647
Storage type	RAM
Default value	0

5.14.3 SP commands

Object name	SP commands
Instruct address	0000
Object type	U16, rw
Range	0, a,
Storage type	RAM
Default value	0

a: Synchronous operation command, absolute displacement

For controllers that need to be synchronized, set to different station numbers, the same group (i.e., the same group ID), and set the motion mode to position mode. The synchronous operation is operated in the absolute positioning mode of position mode, and the speed and position are set to 6047, 6048/6049, 604a respectively. After setting the values for the controllers at different stations, the instructions are sent to the groups to perform the movement by broadcasting.

The broadcast instruction is fixed at 06h to write a single function code, the address is 0, the register number is 0, the fourth byte is the group ID value, the fifth byte is the motion mode, and the synchronous execution value is A

06H instruction (Write a single hold register)

485 address	Register address	Register value
0	0000	010a

00 06 00 00 01 0A 09 8C

09 8C is the crc check code, and there is no response packet after the command is sent.

5.15 Soft limit function

PMC006B1 supports soft limit control to limit motor operation. When the switch is enabled, you can set the upper and lower limit positions to take effect, and when it runs to the upper limit and greater than the upper limit, the motor will stop the current forward rotation action and stop immediately, and the forward rotation command cannot be executed, and it can only reverse the operation to the upper limit range. The same goes for the lower limit.

Object name	Soft limit enable
Instruct address	0x6061
Object type	U16,rw
Range	0~3
Storage type	RAM

Default value	0

bit0 and bit1 control switches. bit1 is the lower limit and bit0 is the upper limit.

Object name	Analogue input
Instruct address	0x6062/6063
Object type	S32,rw
Range	0-0xffffffff
Storage type	RAM
Default value	0

The upper limit bit position, the hexadecimal number with signs, and the value greater than 0x7fffffff is negative.

Object name	Analogue input
Instruct address	0x6064/65
Object type	S32,rw
Range	0-0xffffffff
Storage type	RAM
Default value	0

The lower limit position, a signed hexadecimal number, a value greater than 0x7fffffff is a negative number.

5.16 Power loss behavior

PMC006B1 can detect the power loss of the system and set the corresponding power loss behavior. The following is a detailed description of the power loss behavior settings related objects.

5.16.1 Power-down behavior control word

Object name	Power-down behavior control switch
Instruct address	0x6082
Object type	U16,rw
Range	ROM
Storage type	bit
Default value	0

Bit0: detects power loss and goes offline

The switch is turned on when the corresponding value is 1 and the switch is turned off for 0.

5.16.2 Power-down off-line voltage

Object name	Offline threshold voltage, unit mv
Instruct address	0x6083
Object type	U16,rw

Range	ROM
Storage type	0-65535
Default value	0

When the offline switch is turned on, the motor is offline when the power supply voltage is detected to be below this voltage.

5.16.3 Power voltage read

Object name	Power voltage(mV)
Instruct address	0x605f
Object type	U16,rw
Range	-
Storage type	ROM
Default value	-

5.17 Open and closed loop function switching

Object name	Open and closed loop function switching
SDO ID	0x6036-37
Object type	U32,rw
Range	0-1
Storage type	ROM
Default value	0

Write the value to the register, and write a power-off save instruction before it takes effect, after taking effect, the encoder reading will be shielded, and the open-loop method is used to control the motor operation. It is often used in emergency situations. The closed-loop version works.

5.18 Temperature threshold

Object name	Temperature threshold
SDO ID	0x6035
Object type	U8,rw
Range	100-135
Storage type	ROM
Default value	0x64

When the detection value is higher than the set threshold, the motor state (6000h) will be set to overheat, and the motor enable (600b) will be set to 0 to protect the controller.

5.19 Motor wiring configuration

Object name	Motor wiring configuration
Instruct address	0x6086
Object type	U8, rw
Storage type	ROM

Range	0-3
Default value	3

Using registers for direction control allows changing the motor and encoder reading direction without altering the physical wiring order, where bit0 is the motor wiring order, bit1 is the encoder wiring order, and the default is 03.

5.20 Homing configuration

5.20.1 Homing speed

Object name	Homing speed
Instruct address	0x6098..99
Object type	U32, rw
Storage type	RAM
Range	0-0xFFFFFFFF
Default value	0

5.20.2 Homing distance

Object name	Homing distance
Instruct address	0x609a..9b
Object type	U32, rw
Storage type	RAM
Range	0-0xFFFFFFFF
Default value	0

After configuring the homing speed and homing distance, the motor will run forward until EXT2 is triggered, and the current position will be recorded as the zero point. When homing at the limit, the motor will first run in reverse for a quarter of the homing distance, then move forward to find the zero point.

5.21 offline programming

The controller sets the corresponding parameters according to the process. From top to bottom, run sequentially. At the same time, according to the judgment conditions, jump to the program with the specified serial number and continue to run down in turn. Writing an offline program with instructions requires writing the overall number of instructions first, and then writing each instruction based on the pointer value, starting with clause 0, each writes a pointer first, and then writes the specific value and instruction.

You can refer to the instructions in the configuration file 'PMC007BX_offline_zh_CN.ini' and select the corresponding instructions, as well as the specific functions of the subsets. If there is no delay in each instruction, the execution speed from top to bottom is counted in microseconds. So if the internal process doesn't write the wait condition correctly or compare the judgment condition. Then there may be cases where the process runs into other loops before it is finished. Generally, after sending the operation instruction, it will be followed by an instruction to wait for the completion of the step, or a command to wait for the time, so that the motor can run the set

number of steps. If there is no waiting instruction, the program will continue to execute. At this time, the motor is running, then subsequent commands will not take effect or replace the current running task.

5.21.1 The total number of offline program instructions

Object name	The total number of offline program instructions
SDO ID	0x603b
Object type	Record
Storage type	ROM
Default value	0

The total number of offline instructions that were read and written.

5.21.2 Offline automatic operation enable

Object name	Offline automatic operation enable
SDO ID	0x603c
Object type	Record
Storage type	ROM
Default value	0

Write 1 to start the execution process

5.21.3 Offline program pointers

Object name	Offline program pointers
SDO ID	0x603d
Object type	Record
Storage type	ROM
Default value	0

Indicates the number of serial numbers that are currently being executed

5.21.4 Set the content of the offline command

Object name	Set the content of the offline command
SDO ID	0x603e…f
Object type	Record
Storage type	ROM
Default value	0

603e is the high value, which corresponds to the specific value set by the command.

603f is the low value, corresponding to the command options and functions, the options are in the high position, and the function is in the low position

5.21.5 Save offline commands

Object name	Save offline commands
SDO ID	0x6040

Object type	Record
Storage type	ROM
Default value	0

Save the command and save it to the controller, and write 1 takes effect. It needs to be used with the power-off save command.

5.21.6 Test run instructions

Object name	Test run instructions
SDO ID	0x6041
Object type	Record
Storage type	ROM
Default value	0

Whether to run the command corresponding to the current pointer, write 1 to take effect. After execution, the serial number corresponding to the pointer is sent to perform the test. It is often used to test the function of the command, such as setting the current, speed, and verifying whether it is valid

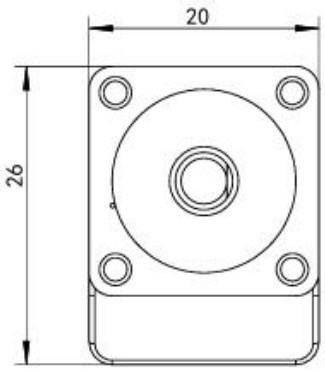
Example of an offline command:

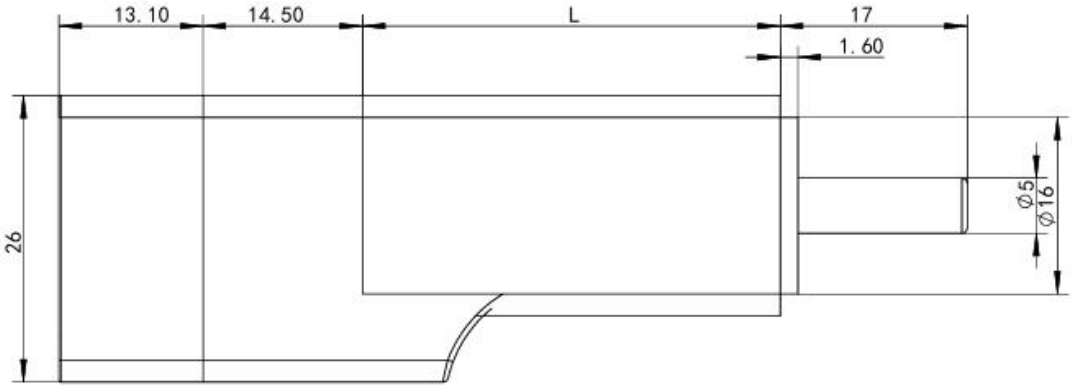
Set the format of the offline command: Take 01 10 60 3E 00 02 04 03 E8 07 77 1B 53 as an example: his data part is 03 E8 07 77. 03E8 is the parameter value of the offline command, which is 1000; 07 77 corresponds to option 7 of the w instruction, i.e., waiting for Nms in the waiting instruction; The full meaning is that the offline command corresponding to the current pointer is to wait for 1000 ms. The specific instructions need to be combined with the files that come with the test software.

6 Electrical Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Supply Power Voltage	Normal 25℃	9	24	36	V
Operation Temperature	24V DC	-40		85	℃
IO maximum current	source/sink current	0	10	20	mA
Output current	Normal 25℃	0.4	4	6	A
IO low Voltage	24V DC	-0.5		1.0	V
IO High Voltage	24V DC	3.0		5.5	V

7 Dimensions


20
26


13.10
14.50
L
17
1.60
26
Ø5
Ø16

8 Appendix 1 instruction table

Modbus address	Name	Data Type	Attr.
2028	Slave address	UINT8	RW
202a..b	Baud rate	UINT32	RW
6000	Error state	UINT8	RW
6001	Controller status	UINT8	RW
6002	Rotation direction	UINT8	RW
6003..4	Max speed	INT32	RW
6005..6	Step command	INT32	RW
6007	Operation mode	UINT8	RW
6008	Start speed	UINT16	RW
6009	Stop speed	UINT16	RW
600a	Acceleration coefficient	UINT8	RW
600b	Deceleration coefficient	UINT8	RW
600c	Microstepping	UINT16	RW
600d	Max phase current	UINT16	RW
600e..f	Motor position	INT32	RW
6010	Current attenuation	UINT8	RW
6011	Motor enable	UINT8	RW
6013	External emergency stop enable	UINT8	RW
6014	Trigger mode of external emergency stop	UINT8	RW
6015	Type of sensor	UINT8	RW
602e	GPIO diriction	UINT16	RW
602f..30	GPIO configuration	UINT32	RW
6031	GPIO value	UINT16	RW
603b	The total number of offline program instructions	UINT16	RW
603c	Offline automatic operation enable	UINT16	RW
603d	Offline program pointer	UINT16	RW
603e…f	Set the content of the offline command	UINT16	RW
6040	Save offline commands	UINT16	RW

6041	Test run instructions	UINT16	RW
6042	Jitter delay of external emergency stop	UINT32	RW
6043	Locked-Rotor configuration	UINT8	RW
6044. . 5	Absolute position step	INT32	RW
6053	Termination step	UINT8	RW
6054	encoder CPR	UINT16	RW
6055. . 6	Position saving value power-down	INT32	RO
6057	Closed-loop parameter KP	UINT8	RW
6058	Closed-loop parameter KI	UINT8	RW
6059	Closed-loop parameter KD	UINT8	RW
605a	Closed-loop Pre-filter parameter	INT8	RW
605b	Closed-loop post-filter parameter	INT16	RW
605c	Closed-loop stall length	INT16	RW
605d	Enable closed-loop torque loop	UINT8	RW
605e	Enable saving automatically when power is off	UINT8	RW
605f	Read power supply voltage	UINT16	RW
6061	Step notification status	UINT8	RW
6062. . 3	Step notification position 1	INT32	RW
6064. . 5	Step notification position 2	INT32	RW
6067. . 8	PP/PV mode Accelerated speed	UINT32	RW
6069. . a	PP/PV mode Dccelerated speed	UINT32	RW
606b. . c	PP/PV mode Initial speed	UINT32	RW
606d. . e	PP/PV mode Stop speed	UINT32	RW
6070	PP/PV mode control word	UINT16	RW

6071	PP/PV mode status word	UINT16	RW
6072. . 3	PP/PV mode running speed	INT32	RW
6074. . 5	PP/PV mode target location	INT32	RW
6077	Enable analog positioning	UINT8	RW
6078	Analog initial AD code	UINT16	RW
6079	Analog adjustment interval time	UINT16	RW
607a	Analog regulating trigger value	UINT16	RW
607b. . c	Minimum value of analog position	INT32	RW
607d. . e	Maximum value of analog position	INT32	RW
607f. . 80	real-time speed	INT32	RW
6082	Power-down behavior control word	UINT16	RW
6083	Power off motor enabling threshold	UINT16	RW
6086	Motor line sequence configuration	UINT16	RW
6089. . 608a	Encoder position	INT32	RW
6098. . 6099	Homing speed	UINT32	RW
609a. . 609b	Homing distance	UINT32	RW
609c	Read analog input	INT16	RW

9 Appendix 2 Communications Example

9.1 Modbus/RTU supported function codes

PMC006B1xP currently supports the following function codes：

1、0x03：read holding register;

2、0x06：write single register;

3、0x10：write multiple registers;

9.2 Master Station communication parameter setting：

1）Baud rate：Same as slave station ;

2） Data bit：8;

3） Stop bit：1;

4） Parity bit：None;

9.3 Modbus master station message write operation

1) Function code 03:

Read the data in user registers 600C and 600D, and the message is as follows:

Read packet: 01 (site) 03 (instruction) 60 0C (register bit) 00 02 (number of reads) 1A 08 (CRC code)

Feedback packet: 01 (site) 03 (command) 04 (number of returned data) 00 00 04 0A (data) 78 F4 (CRC check dignified)

2) Function code 06:

The data value written to user register 600c is 32(0x20): and the packet is as follows:

Write message: 01 (site) 06 (instruction) 60 0C (register bit) 00 20 (write a single data) 56 11 (CRC code)

Feedback message: 01 06 60 0C 00 20 56 11

3) Function code 10:

For example, data 2 (0x0002), 588 (0x292), 0 (0x0000), and 3 (0x0003) are to be written to 600C, 600D, 600E, and 600D, respectively, and the packets are as follows:

Write packets: 01 (site) 10 (instruction) 60 0c (start register bit) 00 04 (write 4 data) 08 (8 bytes) 00 02 02 92 00 00 00 03 (write data) 9f f5 (CRC checksum)

Feedback packet: 01 10 60 0c 00 04 1f c9

9.4 Modbus/RTU control examples

9.4.1 Relative position control

Controller address	Function code	Register address	Data(0x)	Description	Remarks
01	06	600c	0020	Set microstep to 32	
01	06	600d	03e8	Set max phrase current to 1000	
01	06	6003	0000	Set the high 16 bits	

				of speed to 0	
01	06	6004	7d00	Set the low 16 bits of speed to 32000(300RPM）	
01	06	6002	0000	Set the rotation direction to be positive	
01	06	6005	0000	Set the high 16 bits of relative position to 0	
01	06	6006	7d00	Set the low 16 bits of relative position to 32000	
01	03	6001	0000	Read controller statu,bit3 is busy status bit	The value of bit3: 1:busy 0:motor stop

9.4.2 Absolute position control

Controller address	Function code	Register address	Data(0x)	Description	Remarks
01	06	600c	0020	Set microstep to 32	
01	06	600d	03e8	Set max phrase current to 1000	
01	06	6003	0000	Set the high 16 bits of speed to 0	
01	06	6004	7d00	Set the low 16 bits of speed to 32000(300RPM）	
01	06	6044	0000	Set the rotation direction to be positive	
01	06	6045	7d00	Set the high 16 bits of relative position to 0	
01	03	6001	0000	Read controller status, busy status bit of bit3	The value of bit3: 1:busy 0:motor stop

9.4.3 Speed mode control

Controller address	Function code	Register address	Data(0x)	Description	Remarks
01	06	600c	0020	Set microstep to 32	

01	06	600d	03e8	Set max phrase current to 1000	
01	06	6007	0001	Set work mode to speed mode	
01	06	6003	0000	Set the high 16 bits of speed to 0	
01	06	6004	7d00	Set the low 16 bits of speed to 32000(300RPM）	
01	03	6001	0000	Read controller statu,bit3 is busy status bit	The value of bit3: 1:busy 0:motor stop

9.4.4 Commonly used fixed parameter settings

Controller address	Function code	Register address	Data(0x)	Description	Remarks
01	06	600c	0020	Set microstep to 32	
01	06	600d	03e8	Set max phrase current to 1000	
01	06	6008	0258	Set start speed to 600	
01	06	6009	0258	Set stop speed to600	
01	06	600a	0008	Set Acceleration coefficient to 8	
01	06	600b	0008	Set deceleration coefficient to 8	
01	06	6013	0003	Enable external stop	Both of EXT1/EXT2 enabled
01	06	6014	0000	Trigger mode of external emergency stop	Both of EXT1/EXT2 are drop edge .
01	06	200f	0002	Power-down saves all parameters	

10 Appendix 3 CRC code

Cyclic redundancy check CRC area is 2 bytes, containing a 16-bit binary data. the CRC value is calculated by the sending device, and the calculated value is attached to the information. when the receiving device receives the information, the CRC value is recalculated, and the calculated value is compared with the actual value received in the CRC area. if the two are not the same, an error is generated.

CRC start by setting all the 16 bits of the register as "1", and then putting the data of the adjacent 2 8-bit bytes into the current register, only the 8-bit data of each character is used as the starting bit CRC, the generation, and the stop bit and parity bit are not added to the CRC.

The result of every 8 bits of data is shifted one bit to the right (in the LSB direction) during

the generation of CRC, and the MSB, detection is filled with "0". If the LSB is "1", it is different from the preset fixed value, and if the LSB is "0", it is not different or operated.

Repeat the above procedure until the shift is 8 times. After the 8th shift is completed, the next 8-bit data is different from the current value of the register, and after all the information is processed, the final value in the register is CRC value.

CRC generation process:

1. Set the 16-bit CRC register FFFFH.
2. A first 8-bit data performs an XOR operation with a CRC register 8 bits lower, putting the result into the CRC register.
3. Move one bit CRC register to the right, MSB fill zero, check LSB..
4. (if LSB is 1): CRC register performs an XOR operation with the A001H.
   (If LSB is 0): Repeat 3 and move one bit to the right.
5. Repeat 3 and 4 until 8 shifts are completed and 8 bytes are processed.
6. Repeat steps 2 to 5 to process the next 8-bit data until all bytes are processed.
7. the final value of the CRC register is the CRC value.
8. high 8 bits and low 8 bits should be placed separately when putting the CRC value into the information. Add CRC values to the information,send the 16 bit CRC value in the message, first send low 8 bit, then send high 8 bit.

11 Appendix 3 Modbus/RTU 16bits CRC check example

```
// CRC High byte value table
static const uint8_t s_CRCHi[] = {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```

```
24          0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
25          0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
26          0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
27          0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
28          0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
29    } ;
30    // CRC Low byte value table
31    const uint8_t s_CRCLo[] = {
32          0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
33          0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
34          0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
35          0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
36          0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
37          0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
38          0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
39          0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
40          0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
41          0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
42          0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
43          0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
44          0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
45          0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
46          0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
47          0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
48          0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
49          0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
50          0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
51          0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
52          0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
53          0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
54          0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
55          0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
56          0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
57          0x43, 0x83, 0x41, 0x81, 0x80, 0x40
58    };
59    /*
60    *********************************************************************************************************
61    *     function name: CRC16_Modbus
62    *     description: CRC    _Modbus
63    *     Parameters: _pBuf : Data for validation
64    *                 _usLen :Data length
65    *     Return value: 16bit integer value。    For Modbus, this result, high bytes are transmitted first, low bytes are transmitted later..
```

```
66  ********************************************************************************
    ***************************
67  */
68  uint16_t CRC16_Modbus(uint8_t *_pBuf, uint16_t _usLen)
69  {
70      uint8_t ucCRCHi = 0xFF; /* High CRC byte initial */
71      uint8_t ucCRCLo = 0xFF; /* Low CRC byte initial */
72      uint16_t usIndex;    /* CRCindex in the loop */
73
74    while (_usLen--)
75    {
76    usIndex = ucCRCHi ^ *_pBuf++; /* calculate CRC */
77    ucCRCHi = ucCRCLo ^ s_CRCHi[usIndex];
78    ucCRCLo = s_CRCLo[usIndex];
79    }
80    return ((uint16_t)ucCRCHi << 8 | ucCRCLo);
81  }
```

Here's the calling method 1:

```
1   typedef struct
2   {
3       uint8_t RxBuf[H_RX_BUF_SIZE];
4       uint8_t RxCount;
5       uint8_t RxStatus;
6       uint8_t RxNewFlag;
7
8       uint8_t RspCode;
9
10      uint8_t TxBuf[H_TX_BUF_SIZE];
11      uint8_t TxCount;
12
13      uint16_t Reg01H;        /* Save the register header address sent by the host */
14      uint16_t Reg02H;
15      uint16_t Reg03H;
16      uint16_t Reg04H;
17
18      uint8_t RegNum;             /* register number */
19
20      uint8_t fAck01H;        /*  Response command flag 0 indicates execution failure 1
    indicates execution success */
21      uint8_t fAck02H;
22      uint8_t fAck03H;
23      uint8_t fAck04H;
```

```
24          uint8_t fAck05H;
25          uint8_t fAck06H;
26          uint8_t fAck10H;
27
28      }MODH_T;
29      MODH_T g_tModH;
30      uint16_t crc;
31
32      g_tModH.TxBuf[0] = 0x31;
33      g_tModH.TxBuf[1] = 0x32;
34      g_tModH.TxCount = 2;
35      crc = CRC16_Modbus(g_tModH.TxBuf, g_tModH.TxCount);
```

Here's the calling method 2:

```
1       uint8_t _Data[10]; = { 0x31, 0x32};
2       uint8_t  usLen = 2;
3       crc = CRC16_Modbus(_Data, usLen);
4
```

12 Appendix 2 Error code table

MODBUS ERROR CODE		
Code	name	description
01	Illegal function	For the server (or slave), the function code received in the inquiry is an unallowed operation. This may be because the function code is only applicable to new devices and is not achievable in the selected unit. At the same time, it is also pointed out that the server (or slave) handles this kind of request in an error state, for example: because it is unconfigured, and it is required to return the register value.
02	Illegal data address	For the server (or slave), the data address received in the inquiry is an unallowable address. In particular, the combination of reference number and transmission length is invalid. For a controller with 100 registers, the request with offset 96 and length 4 will succeed, and the request with offset 96 and length 5 will generate exception code 02.
03	Illegal data value	For the server (or slave), the value included in the query is not allowed. This value indicates a failure in the remaining structure of the combined request, for example: the implied length is incorrect. It does not mean that because the MODBUS protocol does not know the significance of any special value of any special register, the data item submitted for storage in the register has a value that is not expected by the application.
04	Slave equipment fail	When the server (or slave) is trying to perform the requested

	ure	operation, an unrecoverable error occurs.
05	Ensure	Used with programming commands. The server (or slave) has accepted the request and is processing the request, but it takes a long time to perform these operations. Returning this response prevents a timeout error in the client (or master). The client (or master station) can continue to send the polling procedure completion message to determine whether the processing is completed.
06	Slave equipment bu sy	Used with programming commands. The server (or slave) is processing long-duration program commands. When the Zhang server (or slave station) is idle, the user (or master station) should retransmit the message later.
08	Storage parity error	Used with function codes 20 and 21 and reference type 6 to indicate that the extended file area cannot pass the consistency check. The server (or slave) managed to read the log file, but found a parity error in the memory. The client (or master) can resend the request, but can request service on the server (or slave) device.
0A	Unavailable gateway path	Used with a gateway to indicate that the gateway cannot allocate an internal communication path from input port to output port for processing requests. Usually means that the gateway is misconfigured or overloaded.
0B	Gateway target device failed to respond	Used with the gateway to indicate that no response was obtained from the target device. Usually means that the device is not on the network.

Examples of client requests and server exception responses:

Request		Response	
domain name	Hex	domain name	Hex
Function code	01	Function code	81
Start address(Hi）	04	Error code	02
Start address(Lo）	A1		
Output number(Hi）	00		
Output number(Lo）	01		

In this example, the client addresses the server device's request. Function code (01) is used to read the output status. It will request the output status of address 1245 (hex 04A1). It is worth noting that, as explained in the output field (0001) number, only one output is available. If there is no output address in the server device, the server will return an exception response with an exception code (02). This shows the illegal data address of the slave.

Remarks: The abnormal response message has two fields different from the normal response:

Function code field: In a normal response, the server uses the response function

code field to respond to the originally requested function code. The most significant bit (MSB) of all function codes is 0 (their values are all lower than 80 hex). In the abnormal response, the server sets the MSB of the function code to 1. This makes the function code value in the abnormal response higher than the function code value in the normal response by 80 hexadecimal.

Data field: In a normal response, the server can return the data or statistics table in the data field (any message requested in the request). In the exception response, the server returns the exception code in the data field. This defines the server state that caused the exception.