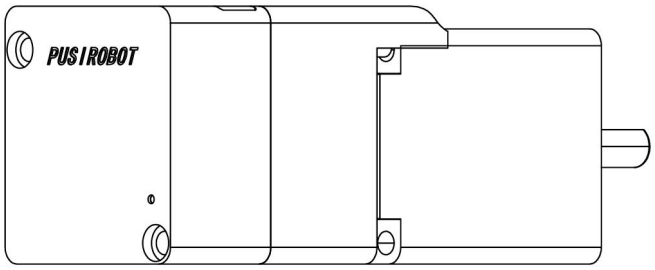
PUSIROBOT

CQPUSI ROBOT CONTROL SYSTEM

User Manual

PMC006C2 Series

Miniature Integrated Stepper Motor Controller



1. Version Control

1） Update Records

Date	Author	Version	Remark
2014-10-19	huangcheng	V0.1.0	Initial
2014-11-25	Liur	V0.1.1	Fix typo
2014-12-08	huangcheng	V0.1.2	Add system control object, signal interface J1 add BOOT and RESET signal description, merge external emergency stop enable and external emergency stop trigger mode object into one object, modify tool screenshot and add related function description.
2015-5-6	huangcheng	V0.1.3	PDO function supplement
2016-6-15	wentao	V0.1.4	Add senseless stall detection
2016-9-22	Liur	V0.1.5	Add function description for close loop
2016-11-7	huangcheng	V0.1.6	Add external stop3
2016-11-25	wentao	V0.1.7	Close loop function supplement
2017-10-8	liur	V0.1.8	Add pvtb mode, update feature & function
2018-6-15	jiawei	V0.1.9	Migrated from pmc007cxsx
2018-7-23	hc	V0.2.0	Modify instruction
2018-09-27	hc	V0.2.1	1、 Add analog input function 2、 Add step notify related objects
2018-11-30	Tanlu	V0.2.2	Supplement description for Ext port
2018-12-3	huangcheng	V0.2.3	Add sensor type parameter description
2019-02-25	huangcheng	V0.2.4	1、Functional description of adding PV/PP mode 2、Adding analog location function description
2019-07-03	Liur	V0.2.6	Fix timing figure
2024-10-19	Tony	V1.0.0	Migrated from PMC007C2
2024-12-22	Liu	V1.0.1	Add Description of High-Speed Torque Mode
2025-3-1	Liu	V1.0.2	Add Image of All-in-One Machine

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 Motor connection J2 9
3.4 CAN network connection 9
3.5 Limit switch connection 9
3.6 Other limit switch connections 10
3.7 Mechanical switch connection 11
3.8 Analog Adjusting Speed 11
4 Drive mode 11
4.1 Normal open-loop/closed-loop mode 11
4.2 High-speed torque mode 12
5 CANopen communication 12
5.1 CANopen introduction 12
5.2 CAN frame structure 13
5.3 CAN communication configuration 13
5.3.1 Node ID 13
5.3.2 Baud rate 13
5.3.3 Group ID 14
5.3.4 Device node name 14
5.3.5 Hardware version 14
5.3.6 Software version 14
5.3.7 System control 15
5.4 Condition monitoring 15
5.4.1 Motor error status 15
5.4.2 Controller status 15
5.5 Micro stepping/Current 16
5.5.1 Micro stepping 16
5.5.2 Maximum phase current(Operating current) 16
5.6 Working mode switching 16
5.7 Position mode 18
5.7.1 Position mode rotating direction 18
5.7.2 Position mode start speed 18
5.7.3 Position mode stop speed 18
5.7.4 Position mode acceleration coefficient 19
5.7.5 Position mode deceleration coefficient 19

5.7.6 Position mode operating speed 20
5.7.7 Position mode relative displacement command 20
5.7.8 Position mode absolute displacement command 21
5.8 Stop stepper command 21
5.9 Commonly used parameters 21
5.9.1 Motor position 21
5.9.2 Encoder position (absolute value encoder closed loop) 22
5.9.3 Current reduction 22
5.9.4 Motor enable 22
5.9.5 Real-time speed(Closed-loop) 23
5.9.6 Temperature threshold 23
5.9.7 Motor wiring configuration 24
5.9.8 Automatic power-down save enabled 24
5.10 External emergency stop 24
5.10.1 External emergency stop enabled 24
5.10.2 The trigger mode of external emergency stop 25
5.10.3 Sensor type 25
5.10.4 Debouncing delay 25
5.11 General IO port 26
5.11.1 General IO port setting 26
5.11.2 General IO port value 27
5.12 Closed-loop parameter 28
5.12.1 Encoder resolution 28
5.12.2 KP parameter 28
5.12.3 KI parameter 28
5.12.4 KD parameter 28
5.12.5 Pre-filtering parameter 29
5.12.6 Post-filtering parameter 29
5.12.7 Stall length parameter 29
5.13 High-speed torque mode switch 29
5.14 Open loop application switch 30
5.15 PP mode 30
5.15.1 PP mode parameter 1 30
5.15.2 PP mode parameter 2 31
5.15.3 PP model work timing 32
5.16 Synchronous position Motion mode 35
5.16.1 SP speed 36
5.16.2 SP position 36
5.16.3 Synchronous start and stop 36
5.17 Analog parameters 36
5.17.1 Analog enabled 37
5.17.2 Analog start_adc 37
5.17.3 Analog end_adc 37
5.17.4 Analog adjustment lower limit 37

5.17.5 Analog adjustment upper limit 37
5.17.6 Analog Adjustment Interval 38
5.17.7 Analog adjustment trigger value 38
5.17.8 Analog speed/position 1 38
5.17.9 Analog speed/position 2 38
5.18 offline programming 38
5.18.1 Offline programming parameter 1 38
5.18.2 Offline programming parameter 2 39
5.19 Analogue input read 40
5.20 Step notification 40
5.21 Low-voltage protection 41
5.21.1 Low-voltage protection control word 41
5.21.2 Offline voltage threshold 41
6 User-defined programs 41
7 Tool software operation introduction 41
7.1 Graphic programming support 42
7.2 Scripting language support 42
8 Electrical Characteristics 43
9 Dimensions (Unit: mm) 43
10 Appendix 1 PMC006C2 Object dictionary table 44
Analog start_adc 51
11 Appendix 2 CANopen Communication Examples 52
11.1 SDO Read/Write Examples 52
11.1.1 SDO Read 52
11.1.1.1 Data frame format 52
11.1.1.2 SDO Read example 53
11.1.2 SDO Write in 53
11.1.2.1 Data frame format 53
11.1.2.2 SDO Write example 54
12 Appendix 3 PDO configuration example 55
12.1 PDO overview 55
12.1.1 The structure PDO——Mapping parameter 55
12.1.2 The structure PDO——Communication parameter 56
12.1.3 PDO Trigger mode 57
12.2 PDO Configuration example 58
13 Appendix 4 SDO abort code error 59

1 Introduction

1.1 Statement of intellectual property right

PMC006C2 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.

PMC006C2 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 future 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 such information. If the CQPUSI device is used for life support and/or life safety applications, all risks are borne by the buyer. The buyer agrees to protect the CQPUSI from legal liability and compensation for any injury, claim, lawsuit or loss caused by the application.

2 Overview

2.1 General Description

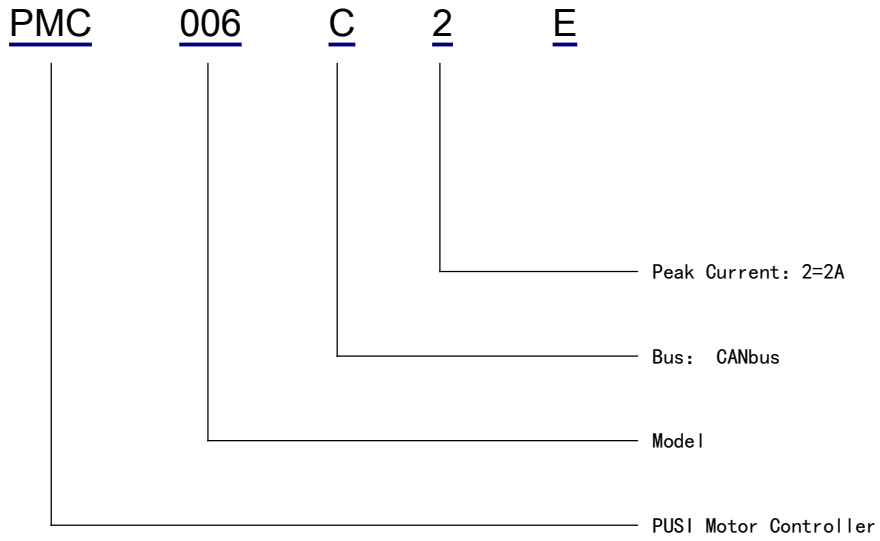
PMC006C2 is a micro all-in-one stepper motor microstepping controller, which can be directly installed on the back cover of the 28 series stepper motor, the controller provides a simple communication network interface, which can easily realize the industrial control network system of up to 120 nodes, and can realize encoder-based closed-loop control according to user requirements. PMC006C2 adopts the industry standard CANOPEN DS301 control protocol, which is suitable for a variety of high-precision, 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 overvoltage protection
- ✓ Analog Input Application (Special edition)
- ✓ Miniature size 28mmx28mmx16mm

2.3 Production & Ordering Information

In order to serve you quicker and better, please provide the product number in following format when ordering PMC006C2：

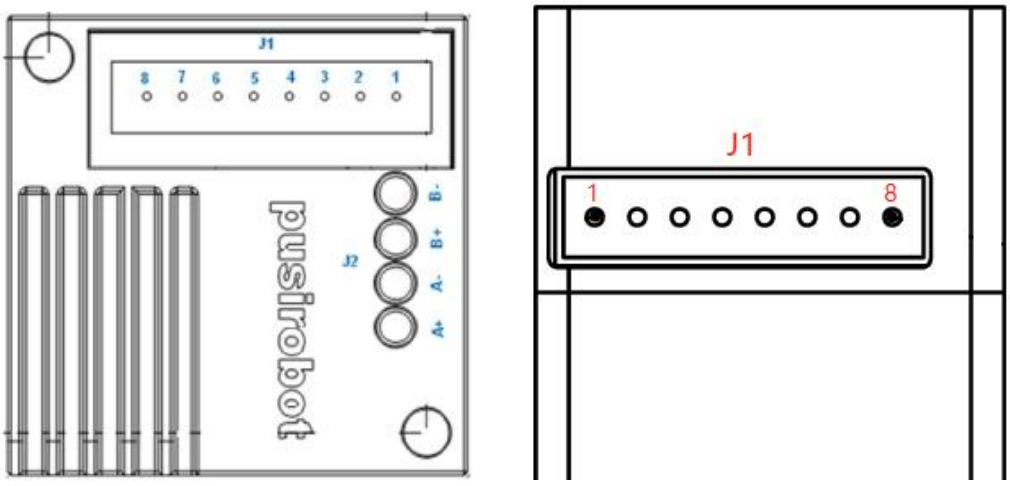


Remark:

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	7	8
Designator	VCC	GND	CANH	CANL	DVDD	EXT2/AIN	GPIO6	GPIO7

Description:

VCC：Supply voltage, 9-36V.

GND：Supply voltage ground.

CANH: Connect the transceiver interface of the CAN module

CANL: Connect the transceiver interface of the CAN module

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

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

GPIO6/ENC1: Digital input/output port, 0-3.3V, open-loop IO corresponds to GPIO6, closed-loop IO corresponds to ENC1;

GPIO7/ENC2: Digital input/output port, 0-3.3V, open-loop IO corresponds to GPIO7, closed-loop IO corresponds to ENC2;

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 Motor connection J2

Pin no:	1	2	3	4
Designator	A+	A−	B+	B−

Description:

A+, A-: Stepper motor phase A

B+, B-: Stepper motor phase B

WARNING: The wrong phase wire of the power supply or motor can permanently damage the controller

3.4 CAN network connection

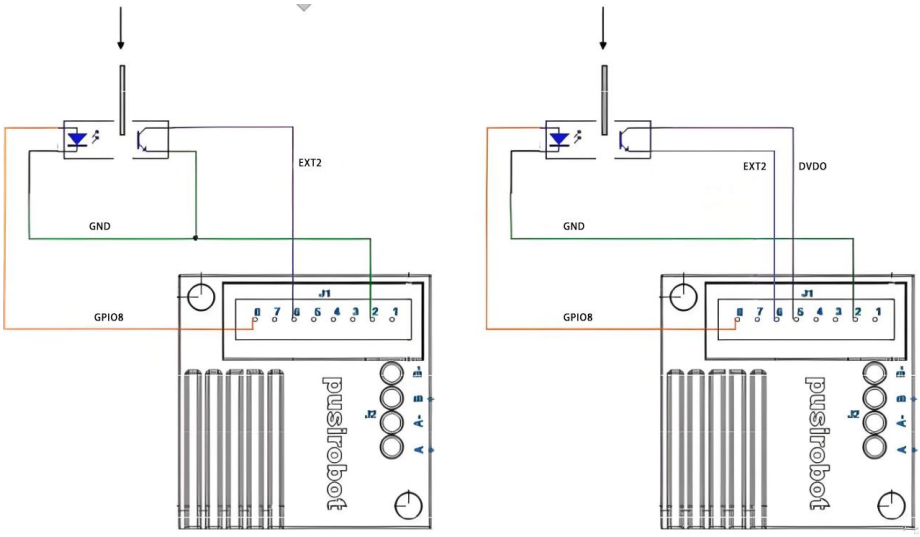
Transmission distances of up to 5000 meters can be achieved using CAN bus connections. The CAN bus is a network solution composed of multiple PMC006C2 controllers, which is compatible with CAN2.0A and CAN2.0B technical specifications, and can connect up to 127 nodes.

Note: It is recommended to use a 120 ohm shielded twisted pair cable for the CAN bus, and a 120 ohm terminal resistor needs to be connected at each end of the twisted pair.

PMC006C2 supports the standard CANopen DS301 protocol, CQPUSI provides PUSICAN, a dedicated debugging tool for networking PMC006C2, which currently supports USB2CAN modules from many mainstream brands on the market.

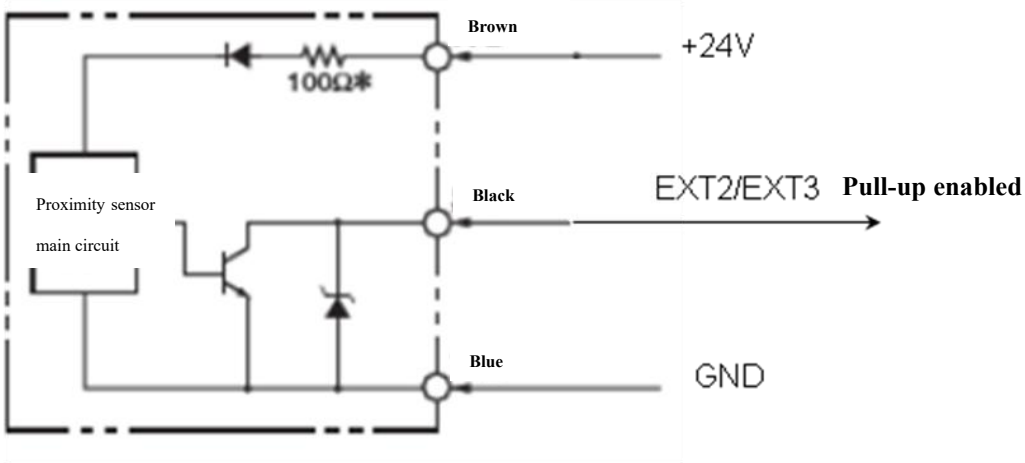
3.5 Limit switch connection

The PMC006C2 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.



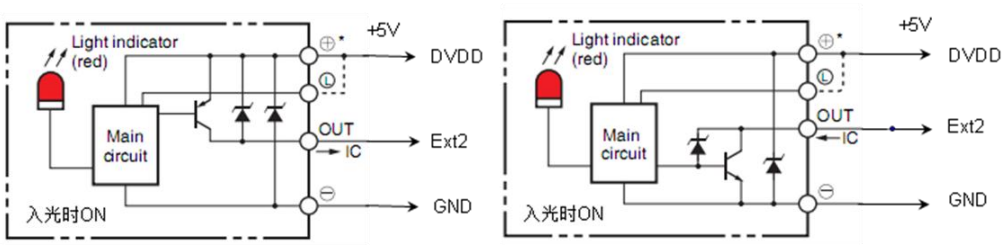
3.6 Other limit switch connections

The EXT2 pin of the PMC006C2 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.



3.7 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.



3.8 Analog Adjusting Speed

PMC006C2 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 0x6015.

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 CANopen communication

5.1 CANopen introduction

CAL provides all network management services and messaging protocols, but does not define the content of the object or the type of object being communicated (it only defines how, not what), which is where CANopen comes in. CANopen is developed on the basis of CAL, using a subset of CAL communication and service protocols to provide an implementation of a distributed control system. CANopen allows the nodes to be scaled at will: simple or complex, while ensuring the interoperability of the network nodes.

The core concept of CANopen is the Object Dictionary (OD), which is also used in other fieldbus (Profibus, Interbus-S) systems. CANopen communication provides access to all parameters of the drive via the Object Dictionary (OD). Note: The object dictionary is not part of the CAL, but is implemented in CANopen, and the PMC006C2 supported object dictionary is shown in Appendix I.

The CANopen communication model defines the following types of messages (communication objects):

Abbreviation	Full name	Description
SDO	Service Data Object	Used for non-time critical data, such as parameters.
PDO	Process Data Object	Used to transfer time critical data(Setting values, Control word, status information, etc.)
SYNC	Synchronization Message	Used to synchronize CAN nodes.
EMCY	Emergency Message	Used to transport alarm event of a driver.
NMT	Network Management	Used for CANopen network management.
Heartbeat	Error Control Protocol	Used for monitoring the life status of all nodes.

5.2 CAN frame structure

CAN Data is transmitted between the host (controller) and the bus node through the data frame. The following table is the structure of the data frame.

Header	Arbitration domain		Control domain	Data domain	Check field	Response domain	Tail frame
	COB-ID(communication object identifier)	RTR(remote request)					
1bit	11 or 29 bits	1bit	6bits	0~8byte	16bits	2bits	7bits

5.3 CAN communication configuration

PMC006C2 factory default settings: node ID is 3/5, the baud rate is 125Kbit/s. The user can modify the settings by supporting the CANOPEN master debugging tool.

5.3.1 Node ID

Object name	Node ID
SDO ID	0x2002
Object type	U8,rw
Range	1-127
Storage type	ROM
Default value	3/5

5.3.2 Baud rate

Object name	Baud rate
SDO ID	0x2003
Object type	U8,rw
Range	0,1,2,3,4,5,6,7,8
Storage type	ROM
Default value	4

Relationship between each index and the baud rate is as follows:

0：20Kbit/s

1：25Kbit/s

2：50Kbit/s

3：100Kbit/s

4：125Kbit/s

5：250Kbit/s

6：500Kbit/s

7：800Kbit/s

8：1000Kbit/s

5.3.3 Group ID

Object name	Group ID
SDO ID	0x2006
Object type	U8,rw
Range	1-127
Storage type	ROM
Default value	0

This object needs to be configured when using the synchronous run function.

5.3.4 Device node name

Object name	Device node name
SDO ID	0x1008
Object type	string,ro
Range	-
Storage type	ROM
Default value	-

5.3.5 Hardware version

Object name	Hardware version
SDO ID	0x1009
Object type	string,ro
Range	-
Storage type	ROM
Default value	-

5.3.6 Software version

Object name	Software version
SDO ID	0x100A
Object type	string,ro
Range	-
Storage type	ROM
Default value	-

5.3.7 System control

Object name	System control
SDO ID	0x2007
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(i.e. power down save command)

3: Reset factory settings

Note: the Storage type in the Object Dictionary which is ROM parameter is temporarily stored in memory after written by SDO. If you need to keep it permanently, you need to perform power down save operation for the Object Dictionary parameter.

5.4 Condition monitoring

5.4.1 Motor 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, over temperature shutdown

Bit1: AOCP, motor over-current

Bit2: BOCP, magnetic core position error

Bit3: APDF, Wrong direction of rotation when facing magnetism

Bit4: BPDF, Wrong rotation angle when facing magnetism

Bit5: UVLO, low supply voltage warning

Bit6: SERR, encoder packet error

Bit7: EERR, encoder position error

Clear the corresponding error state by writing 1 to the corresponding bit, for example, write 18 00 00 00 to clear the flags of bit3 and bit4, that is, to clear the wrong rotation direction when facing the magnet and the wrong rotation angle when facing the magnet, and write FF to be fully clear.

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

5.4.2 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

Bit5：The FIFO of PVT mode 3 is empty

Bit6：Lower FIFO limit for PVT mode 3

Bit7：Upper FIFO limit for PVT mode 3

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

5.5 Micro stepping/Current

5.5.1 Micro stepping

Object name	Micro stepping
Instruct address	0x600A
Object type	U16,rw
Range	0,2,4,8,16,32,64
Storage type	ROM
Default value	32

Only parameters within the range can be configured, and other values are outliers.

5.5.2 Maximum phase current(Operating current)

Object name	Maximum phase current
SDO ID	0x600B
Object type	U16,rw
Range	0-6000
Storage type	ROM
Default value	0

The maximum phase current is the supply current to the motor during normal operation, usually set to the rated current of the motor. In some cases, adjustments can be made appropriately, with a general range of+-20% and not exceeding 50%. Excessive current can cause severe heat generation in the motor, and prolonged operation of the motor may pose a risk of demagnetization, affecting its lifespan. Unit: mA

5.6 Working mode switching

Object name	Working mode switching

SDO ID	0x6005
Object type	U8,rw
Range	0,1,4,5
Storage type	RAM
Default value	0

The values of the motor operating mode are defined as follows:

0: Position mode

1: Velocity mode (including analog speed regulation)

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

5: PV (Profile Velocity) mode

Position 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.

Velocity Mode:

After entering the velocity 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 velocity 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 velocity 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 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

5.7 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	0

The value of the rotation direction is defined as follows:

0: forward

1: backward

5.7.2 Position mode start speed

Object name	Start speed(Unit: pps)
Instruct address	0x6006
Object type	U16,rw
Range	0-0xFFFF
Storage type	ROM
Default value	600

5.7.3 Position mode stop speed

Object name	Stop speed(Unit: pps)
Instruct address	0x6007
Object type	U16,rw
Range	0-0xFFFF
Storage type	ROM
Default value	600

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.

The start-stop speed of the 600 can be adapted to most operating scenarios, and it can be switched when it needs to be started faster or run at a lower speed, and the rest of the time the RPM is configured using microstepping and speed settings.

5.7.4 Position mode acceleration coefficient

Object name	Acceleration coefficient
Instruct address	0x6008
Object type	U8,rw
Range	0-8
Storage type	ROM
Default value	8

5.7.5 Position mode deceleration coefficient

Object name	Deceleration coefficient
Instruct address	0x6009
Object type	U8,rw
Range	0-8
Storage type	ROM
Default value	8

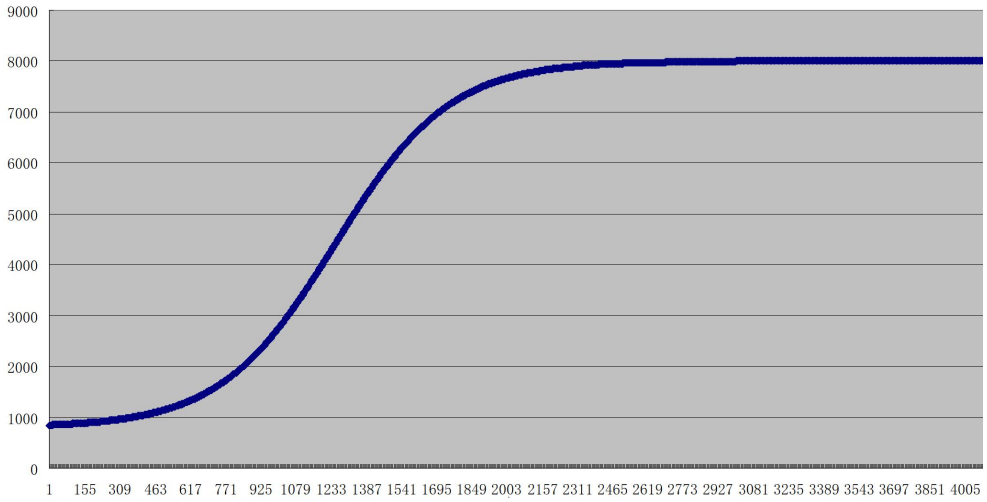


Figure 5-1

The Position mode 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.PMC006C2 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

5.7.6 Position mode operating speed

Object name	Position mode operating speed（pps）
Instruct address	0x6003
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.

The RPM per minute and the microstepping of the set, as well as the magnitude of the speed value, are as follows:

RPM = (PPS*60)/ (200*microstepping)

Similarly, the rotational speed per second is:

RPS = PPS / (200 * microstepping)

5.7.7 Position mode relative displacement command

Object name	Relative displacement command
Instruct address	0x6004
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	0x601c
Object name	S32,rw
Range	Incremental or single-turn absolute encoder:-2^31 ~ (2^31-1) Multi-turn absolute encoder:-2^31 ~ (2^31-1)
Storage type	RAM
Default value	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 to the specified position according to the set speed and acceleration.

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

The number of pulses is calculated in the same way as the relative displacement command.

5.8 Stop stepper command

Object name	Stop stepper command
Instruct address	0x6020
Object type	U8,rw
Range	0
Storage type	RAM
Default value	0

The command immediately terminates the operation of the motor, there is no deceleration process, and it can take effect by writing 0.

5.9 Commonly used parameters

5.9.1 Motor position

Object name	Motor position
Instruct address	0x600C
Object type	S32,rw
Range	Incremental or Singleturn Absolute Encoder: -2^31 ~ (2^31-1) Multiturn absolute encoder: -2^31 ~ (2^31-1)
Storage type	RAM
Default value	0

The current motor position is represented by the conversion of the number of pulses per revolution. The incremental type is a quadruple of the PPR, for example, 4000 lines are 16000 pulses for one revolution of the motor. The absolute accuracy is 2^14,16384 pulses per revolution.

In open-loop mode, the motor position is not saved, and the position information is

automatically cleared to zero after the controller is powered off.

In incremental closed-loop mode, motor positions are not saved for conventional applications. There is a switch inside the controller that can record the position of the motor before the power is off. It is generally used in conjunction with the brake to prevent the motor from moving in the event of a power failure. Otherwise, the saved location is not referential.

In the closed-loop mode of absolute single-turn value, the controller can record the change of motor position within one turn, and the value of the extra turn will be automatically discarded after the power failure. Values greater than one turn are saved, and you can refer to the same application method as the incremental type.

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 omitted.

5.9.2 Encoder position (absolute value encoder closed loop)

Object Name	Encoder position
Instruct address	0x6035
Instruct address	S32,rw
Object type	Single cycle absolute value encoder：-2^31 ~ (2^31-1) Multi-loop absolute value encoder：-2^(r+11) ~ (2^(r+11))-1)， r is encoder accuracy, such as r =14 for 14-bit encoder
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.

Currently, only absolute encoder readings are available.

5.9.3 Current reduction

Object name	Current reduction coefficient
Instruct address	0x600D
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.

5.9.4 Motor enable

Object name	Motor enable

Instruct address	0x600E
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.

5.9.5 Real-time speed(Closed-loop)

Object name	Real-time speed(pps)
Instruct address	0x6030
Object type	S32, ro
Range	-300000~+300000
Storage type	RAM
Default value	0

Registers that can only be read in closed-loop applications. This parameter is to read the feedback value of the encoder to calculate the running speed, limited by the transmission speed, so the compensation value needs to be added, the recommended compensation formula is as follows:

(PPS*60/200/microstepping/10000) +1

In practice, it is necessary to test the motor or application separately and then write the compensation value.

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

5.9.6 Temperature threshold

Object name	Temperature threshold
SDO ID	0x6014
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.

5.9.7 Motor wiring configuration

Object name	Motor wiring configuration
SDO ID	0x6033
Object type	U8,rw
Range	0-3
Storage type	ROM
Default value	3

Using the register for direction control, you can change the motor and encoder reading direction without changing the physical line sequence, where bit0 is the motor line sequence, bit1 is the encoder line sequence, and the default is 0.

5.9.8 Automatic power-down save enabled

Object name	Automatic power-down save enabled
Instruct address	0x602A
Object type	U8,rw
Range	0-1
Storage type	ROM
Default value	0

Enabled when closed loop, the controller automatically detects the power-off of the system after it is enabled, and writes the current motor position into the EEPROM, it needs to be used with a power-off save instruction.

5.10 External emergency stop

PMC006C2 controller provides a dedicated limit switch input, EXT2, which can be used as an emergency stop or zero point search function.

When the emergency stop function is enabled, if the corresponding input pin detects a valid trigger edge, the controller will immediately lock the motor and stop responding to the step 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 state bit will the controller continue to respond to new step commands.

Object name	External emergency stop
Instruct address	0x600F
Object type	Record
Storage type	ROM
Reference number	2

5.10.1 External emergency stop enabled

Subindex 0x01: External emergency stop enabled

Object type	U8,rw
Range	Bit
Default value	0

Each external emergency enable is represented by 1 bit, 0 for prohibition, and 1 for enablement, which is defined as follows:

bit0: external emergency stop 1 is enabled

bit1: External emergency stop 2 is enabled

5.10.2 The trigger mode of external emergency stop

Object type	U8,rw
Range	Bit
Default value	0

The stop trigger mode for each external emergency is represented by 1 bit, 0 for falling triggering and 1 for rising triggering, defined as follows:

bit0: External emergency stop 1 trigger mode

bit1: External emergency stop 2 trigger mode

5.10.3 Sensor type

Object type	U8,rw
Range	0-1
Default value	0

The meaning of the sensor type value is as follows:

0: When the trigger mode is configured as the rising edge, the controller is configured as an internal pull-down resistor; When configured as a falling edge, the controller configuration is a pull-up resistor inside; Typically used for NPN type sensors;

1: When the trigger mode is configured as the rising edge, the controller is configured as an internal pull-up resistor; When configured as a falling edge, the controller is configured with a pull-down resistor inside; Typically used for PNP type sensors;

The external emergency stop trigger delay can be modified by 0x601A object, and the controller will delay the time set after detecting the edge signal, and then check whether its level state is correct, and if it is correct, the motor will be triggered to stop urgently, otherwise the motor will continue to rotate.

5. 10. 4 Debouncing delay

Object name	EXT1/EXT2 stabilize delay（ms）
Instruct address	0x601a
Object type	Record
Range	0~200
Storage type	ROM

Default value	30

Similar to the keyboard debouncing delay, this parameter can set the high/low level time that needs to be maintained after triggering to prevent false triggering.

5.11 General IO port

The controller provides 2 general IO (GPIO6/7) ports, 1 external emergency stop input (EXT2) port.

5. 11. 1 General IO port setting

Object name	General IO port setting
Instruct address	0x6011
Object type	Record
Storage type	ROM
Reference number	2

Subindex 0x01: IO port direction

Object type	U16,rw
Range	bit
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：EXT1
Bit8：EXT2
Bit9：ENC1
Bit10：ENC2
Bit11：GPIO8

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.

Note: GPIO0~GPIO5 is not led to the controller interface and is only for offline programming.

Subindex 0x02: IO port configuration

Object type	U32,rw
Range	0-0x3fffff
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：EXT1

Bit17-16：EXT2

Bit19-18：EXT3/ENC1

Bit21-20：ENC2

Bit23-22：GPIO8

5.11.2 General IO port value

Object name	General IO port value
Instruct address	0x6012
Object type	U16,rw
Range	bit
Storage type	RAM
Default value	0

The value of each IO port is represented by 1bit, 1 indicates a high level, 0 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 value

Bit1：GPIO2 value

Bit2：GPIO3 value

Bit3：GPIO4 value

Bit4：GPIO5 value

Bit5：GPIO6 value

Bit6：GPIO7 value

Bit7：EXT1 value

Bit8：EXT2 value

Bit9：EXT3/ENC1 value

Bit10：ENC2 value

Bit11：GPIO8 value

5.12 Closed-loop parameter

PMC006C2 supports 200-8000PPR incremental photoelectric encoder and uses PID to realize closed loop control.

5.12.1 Encoder resolution

Object name	Encoder resolution
Instruct address	0x6021
Object type	U16,rw
Range	Incremental encoder closed loop: 200,400,500,600,800,1000,1200,1600,2000,4000,8000 Absolute encoder closed loop:14
Storage type	ROM
Default value	1000/4000,14

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.

5.12.2 KP parameter

Object name	KP parameter
Instruct address	0x6023
Object type	U8,rw
Range	1-255
Storage type	ROM
Default value	48

This parameter affects the transient response characteristic of the system.

5.12.3 KI parameter

Object name	KI parameter
Instruct address	0x6024
Object type	U8,rw
Range	1-255
Storage type	ROM
Default value	8

This parameter affects the cumulative error characteristics of the system.

5.12.4 KD parameter

Object name	KD parameter
Instruct address	0x6025
Object type	U8,rw

Range	1-255
Storage type	ROM
Default value	8

This parameter affects the transient response characteristic of the system.

5.12.5 Pre-filtering parameter

Object name	Pre-filtering parameter
Instruct address	0x6026
Object type	U8,rw
Range	1-128
Storage type	ROM
Default value	8

This parameter affects the speed characteristics of the system, you don't need to modify the default values.

In high-speed torque mode, it can be used as a switch and setting value for the guidance correction function.

5.12.6 Post-filtering parameter

Object name	Post-filtering parameter
Instruct address	0x6027
Object type	U16,rw
Range	1-255
Storage type	ROM
Default value	8

This parameter is reserved for the time being.

5.12.7 Stall length parameter

Object name	Stall length parameter
Instruct address	0x6028
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.

5.13 High-speed torque mode switch

Object name	High-speed torque mode switch
Instruct address	0x6029
Object type	U8,rw
Range	0-1
Storage type	ROM

Default value	0

5.14 Open loop application switch

Object name	Open loop application switch
Instruct address	0x6015
Object type	U32,rw
Range	0-1
Storage type	ROM
Default value	0

Write a value to the register, and write a power-down save instruction before it takes effect, and after the effect, it will block the encoder reading, and use an open-loop method to control the motor operation. Often used in emergencies. The closed-loop version works.

5.15 PP 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.

5.15.1 PP mode parameter 1

PP mode parameter 1 is the ROM parameter, which can be saved by power off

Object name	PP mode parameter 1
Instruct address	0x602d
Object type	Record
Storage type	ROM
Reference number	4

Sub-index 0x01: pp acceleration, unit pps/s

Object type	U32,rw
Range	>150
Default value	32000

Sub-index 0x02: pp deceleration, unit pps/s

Object type	U32,rw
Range	>150
Default value	32000

Sub-index 0x03: pp start speed

Object type	U32,rw
Range	>600
Default value	600

Recommend to use the default value, the setting value should not be lower than 600.

Sub-index 0x04: pp stop speed

Object type	U32,rw
Range	>600
Default value	600

Recommend to use the default value, the setting value should not be lower than 600.

5.15.2 PP mode parameter 2

PP mode parameter 2 is the RAM parameter, which is reset to the default value after power-up.

Object name	PP mode parameter 2
Instruct address	0x602e
Object type	Record
Storage type	ROM
Reference number	4

Sub-index 0x01: control word

Object type	U16,rw
Range	0-0xFFFF
Default value	0

The function description for Control word object (602e,1)：

• Bit 4: Start the running task. When the value is converted from "0" to "1", the run task is executed. Example value: 0x10

• Bit 5: When the bit is set to "1", the run task triggered by bit 4 is executed immediately. If the bit is set to "0", the running task that is being executed will be completed before the next run task is started. Example value: 0x30

• Bit 6: The target position (602e,4) is the absolute position when the value is set to "1", and the target position is the relative position when the value is set to "0". Example value: 0x50 or 0x70

• Bit 8 (Halt): This bit is applied in PV mode, when the value of this bit changes from "1" to "0", the motor will accelerate to the target speed with a preset starting ramp. When the value of this bit changes from "0" to "1", the motor will slow down and stop moving.

• Bit 9: When the bit is set, the speed will change after reaching the first target position. That is, the braking is not performed until the first target is reached, since the motor should not stop in that position.

Sub-index 0x02: status word

Object type	U16,rw
Range	0-0xFFFF
Default value	0

The following bits in the state word object (602e,2) have special functions:

• Bit 10: When the final target has been reached, the bit will be set to "1" and the position is in place.

• Bit 12: This bit acknowledges receipt of a valid new target point. The bit will be set and reset synchronously with the bit "New Target Point" in the control word.

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

Object type	S32,rw
Range	-300000- -150,150-300000
Default value	32000

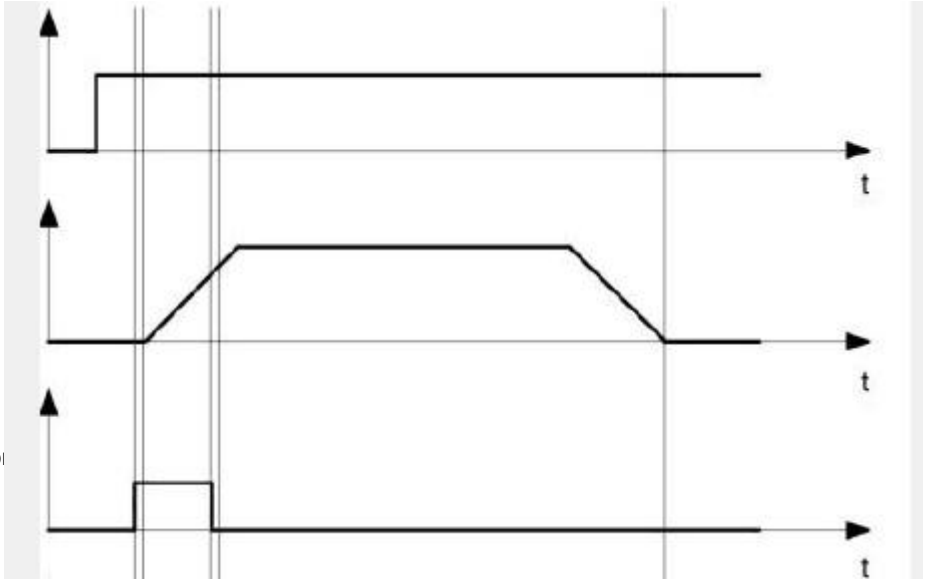
Subindex 0x04: PP Model Target Location

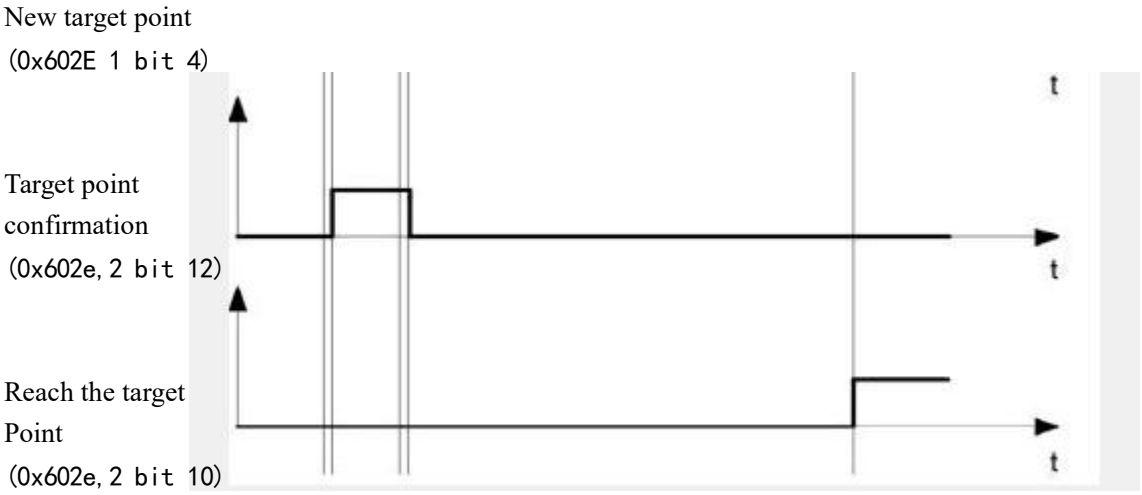
Object type	S32,rw
Range	-2^31~2^31
Default value	0

Running tasks with a speed of 0 or a run length of 0 will receive an error.

5.15.3 PP model work timing

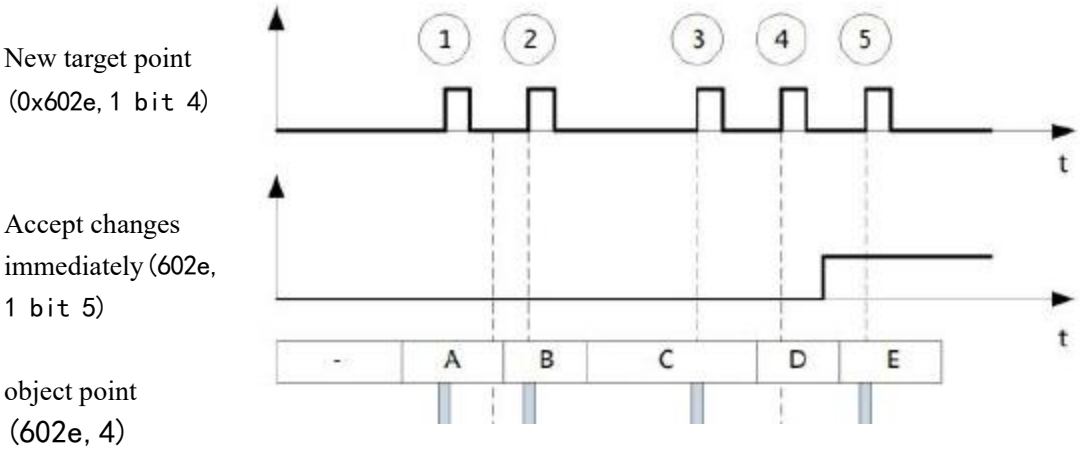
A new target position is set in the target position object (602e,4). Next, the bit 4 in the control word object (602e,1) is set for trigger the operation command. If the target location is valid, the controller will reply through the bit 12 in the object status word to locate the start of the operation. When the location is reached, the bit 10 in the status word will immediately set to a "1".

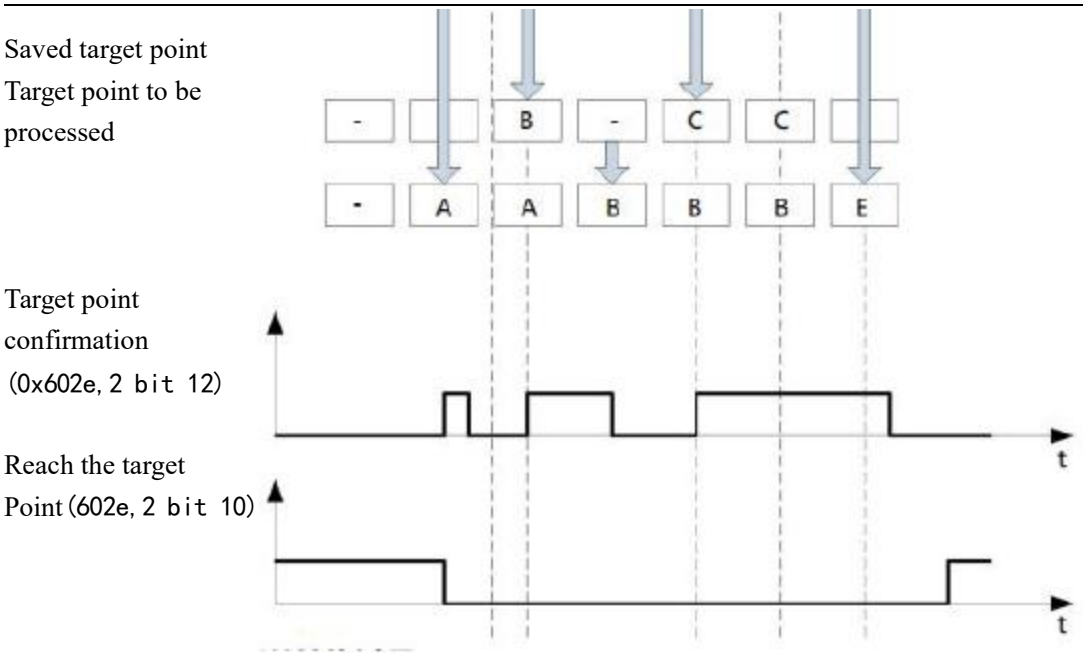




Other running commands can be stored in the cache (see point in time 1 in the figure below), and bit 12 in the status word object (602e, 2 sets the target point response) will be set to "0". During the motion to the target position, a second target position can be sent to the controller to prepare for 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).

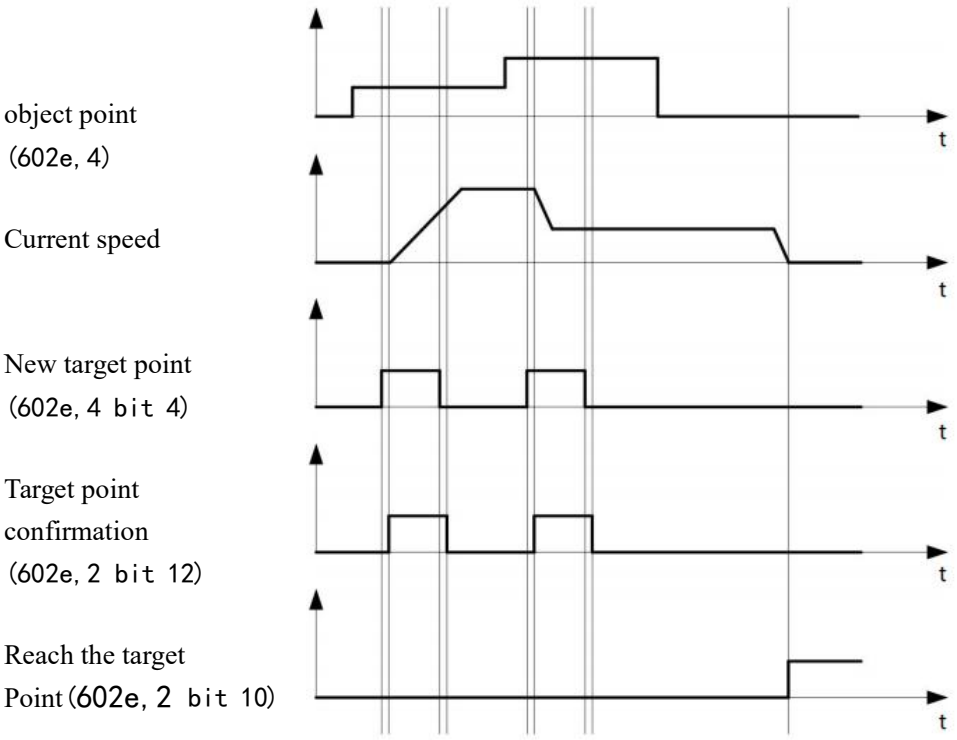
If the cache is full, the new target point will be ignored (point in time 4). If bit 5 in the control word object (602e, 1 bit: "change the target point now") is set, the controller will not use cache when working, and the new running command will be executed directly (point in time 5).





The conversion process of the second target position:

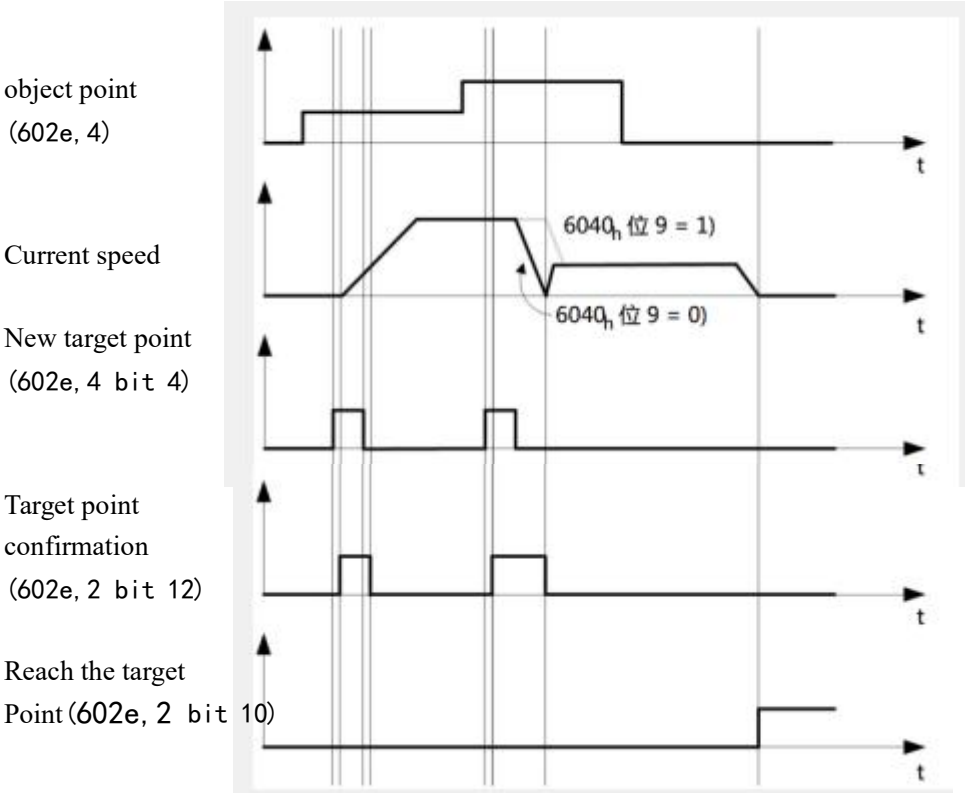
The following figure shows the conversion process of the second target position when moving to the first target position. In this figure, the bit 5 of the control word object (602e, 1) is set to "1" and the new target value will be accepted immediately.



The method of moving to the target position:

If the bit 9 in the control word object (602e,1) 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.



5.16 Synchronous position Motion mode

Object name	Synchronous position Motion control
Instruct address	0x601D
Object type	Record
Storage type	RAM
Default value	2

The synchronous positioning motion mode is the method of absolute immediate execution of the PP mode, by sending broadcast commands, so that the control in the specified group has set the task, and the synchronous stop is also used in the same way.

Before use, you need to switch the running mode to PP mode, set the group (group ID, 2006H) of the controller to be run to a unified value, and then write the running speed and absolute position in the registers of the synchronous positioning speed and synchronous positioning position, that is, set the motion task. After the setting is completed, the NMT broadcast command to let the specified group of controllers run synchronously.

5.16.1 SP speed

Sub-index 0x01: Synchronized location speed

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

5.16.2 SP position

Sub-index 0x02: Synchronized location position

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

5.16.3 Synchronous start and stop

PMC006C2 is an extension of the NMT instructions of the standard CANOpen to realize the synchronous start and stop of two or more nodes in a network.

Standard NMT format

COB-ID	Byte0	Byte1
0x000	CS	Node-ID

The expanded NMT directive adds new definitions of Byte0 and Byte1 without affecting the standard protocol.

Byte0 is defined as follows:

Command	Function
10	Start the synchronous positioning movement

Byte1 is the group ID, and the corresponding command operation will be executed only when the group ID received by the controller matches its own group ID. Start using NMT command to send, the first data is the function command, 10 is the start synchronous positioning, so the first parameter is 0A, the second parameter is the group id, if it is 1, it is 01.cob-id is 000.

Taking site 3 as an example, the sending instruction is set to run synchronously:

Set group ID	603	2f 06 20 00 01 00 00 00
Set sport mode	603	2f 05 60 00 04 00 00 00
Set the synchronous positioning speed	603	23 1d 60 01 00 7d 00 00
Set the synchronous location	603	23 1d 60 02 00 7d 00 00
Start the synchronous motion	000	0A 01

5.17 Analog parameters

The PMC006C2 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.

Object name	Analog settings
Instruct address	0x602f
Object type	Record
Storage type	ROM
Reference number	6

5.17.1 Analog enabled

Subindex 0x01: Analog Enable, 1 On, 0 Off

Object type	U8, rw
Range	0,1
Default value	0

5.17.2 Analog start_adc

Subindex 0x02: Analog start_adc: Adjust when the analog signal curve drifts

Object type	U16, rw
Range	0-4095
Default value	0

5.17.3 Analog end_adc

Subindex 0x03: Analog end_adc: Adjust when the analog signal curve drifts

Object type	U16,rw
Range	0-4095
Default value	0

5.17.4 Analog adjustment lower limit

Subindex 0x04: 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 type	U16,rw
Range	0-4095
Default value	0

5.17.5 Analog adjustment upper limit

Subindex 0x05: 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 type	U16, rw
Range	0-4095
Default value	0

5.17.6 Analog Adjustment Interval

Subindex 0x06: Analog adjustment interval, unit ms.

The controller checks the analog input value at this interval. If the difference between the current AD input value and the previous input value exceeds the trigger threshold, it will do a position adjustment.

Object type	U16, rw
Range	0-65535
Default value	100

5.17.7 Analog adjustment trigger value

Subindex 0x07: Analog adjustment trigger value. When the difference between the current AD code and the previous AD code, converted into position, is greater than this value, the controller will do a position adjustment.

Object type	U16, rw
Range	0-65535
Default value	30

5.17.8 Analog speed/position 1

Subindex 0x08: Speed/position corresponding to the starting AD code of the analog quantity

Object type	S32, rw
Range	-2^31~2^31
Default value	0

5.17.9 Analog speed/position 2

Subindex 0x09: Speed/position corresponding to AD code 4095

Object type	S32, rw
Range	-2^31~2^31
Default value	0

5. 18 offline programming

5.18.1 Offline programming parameter 1

Object name	Offline programming parameter 1
SDO ID	0x6018
Object type	Record
Storage type	ROM
Default value	2

Sub-index 0x01: Number of offline programming data instructions

Object type	U8, rw

Range	0-100
Default value	0

Sub-index 0x02: Offline automatic operation enable

Object type	U8, rw
Range	0,1
Default value	0

Offline auto operation value definition:

0: Do not enable offline automatic operation

1: Enable offline automatic operation

5.18.2 Offline programming parameter 2

Object name	Offline programming parameter 2
SDO ID	0x6019
Object type	Record
Storage type	RAM
Default value	5

Sub-index 0x01: Offline program pointer

Object type	U8, rw
Range	0-100
Default value	0

Sub-index 0x02: Offline command

Object type	U32, rw
Range	-
Default value	-

Offline instruction definitions are described in the User-Defined Programs section.

Sub-index 0x03: Save offline instructions

Object type	U8, rw
Range	0,1
Default value	0

Write 1 to save all the offline instructions

Sub-index 0x04: Run instructions

Object type	U16, rw
Range	0,1
Default value	0

Write 1 to the instruction pointed to by the pointer of the offline program

5.19 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	0x602B
Object type	U16,rw
Range	0~4095
Storage type	RAM
Default value	0

5.20 Step notification

The controller can set the step notification in position mode or speed mode, that is, when the motor movement reaches a certain set position in a step, the controller can report the step to position notification through TPDO, and supports two step notification position points.

Object name	Step notification
SDO ID	0x602C
Object type	Record
Storage type	RAM
Default value	3

Sub-index 0x01: Step notification status

Object type	U8, rw
Range	bit
Default value	0

The direction of each IO port is represented by 1 bit, 0 is the input, 1 is the output, and the meaning is as follows:

Bit0: Step notification position point 1 is valid;

Bit1: Step notification position point 2 is valid;

The object can be mapped to the TPDO, and when the object value changes, it will be automatically reported to the host computer.

Sub-index 0x02: Step notification position1(absolute setting) setting

Object type	S32, rw
Range	−2147483647 ~ +2147483647
Default value	0

Sub-index 0x03: Step notification position2(absolute setting) setting

Object type	S32, rw
Range	−2147483647 ~ +2147483647
Default value	0

5.21 Low-voltage protection

The controller detects the power loss of the system and can set the corresponding low voltage protection. The following describes the objects related to the low voltage protection settings in detail.

Object name	low voltage protection setting
SDO ID	0x6031
Object type	Record
Storage type	ROM
Reference number	3

5.21.1 Low-voltage protection control word

Sub-index 0x01: low voltage protection control switch

Object type	U16, rw
Range	bit
Default value	0

Bit0: Offline when the input supply voltage is detected to be below the set value

Bit1: Brake tightening when the input supply voltage is detected to be below the set value

If the corresponding bit bit is 1, it will be enabled, and if it is 0, it will be turned off.

5.21.2 Offline voltage threshold

Sub-index 0x02: Offline threshold voltage, unit mV

Object type	U16, rw
Range	0–65535
Default value	0

When the low-voltage offline enable is turned on, the motor goes off when the supply voltage is detected below this voltage.

6 User-defined programs

PMC006C2 controller can be configured to work offline, in which the controller automatically executes user-defined program code after powering on, which is pre-compiled and programmed into EEPROM by CQPUSI tool software.

When the PMC006C2 controller is working offline, the CAN communication interface can still respond to the user's online instructions.

The maximum number of user instructions supported by PMC006CX controller is 100.

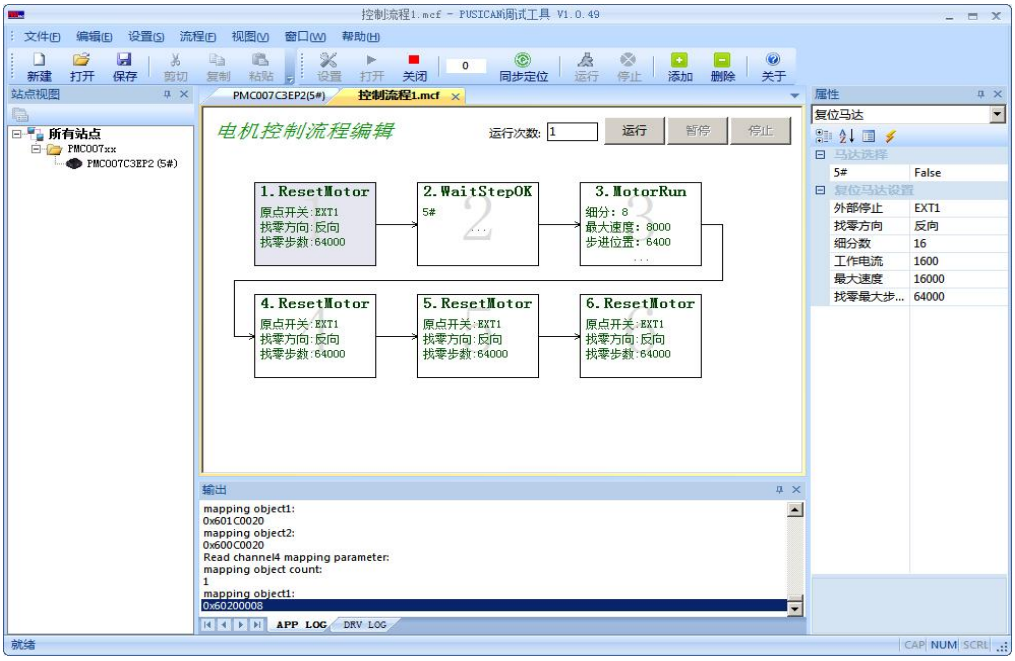
For detailed examples of user-defined programs, please refer to the Controller Custom Programming Guide.

7 Tool software operation introduction

PMC006C2 can be used for command debugging, IO port setting detection, stepper motor parameter debugging and custom programming through the CQPUSI tool software PUSICAN. The tool software usage process is tested, and details can be found in the "Pusican Operation Guide".

7.1 Graphic programming support

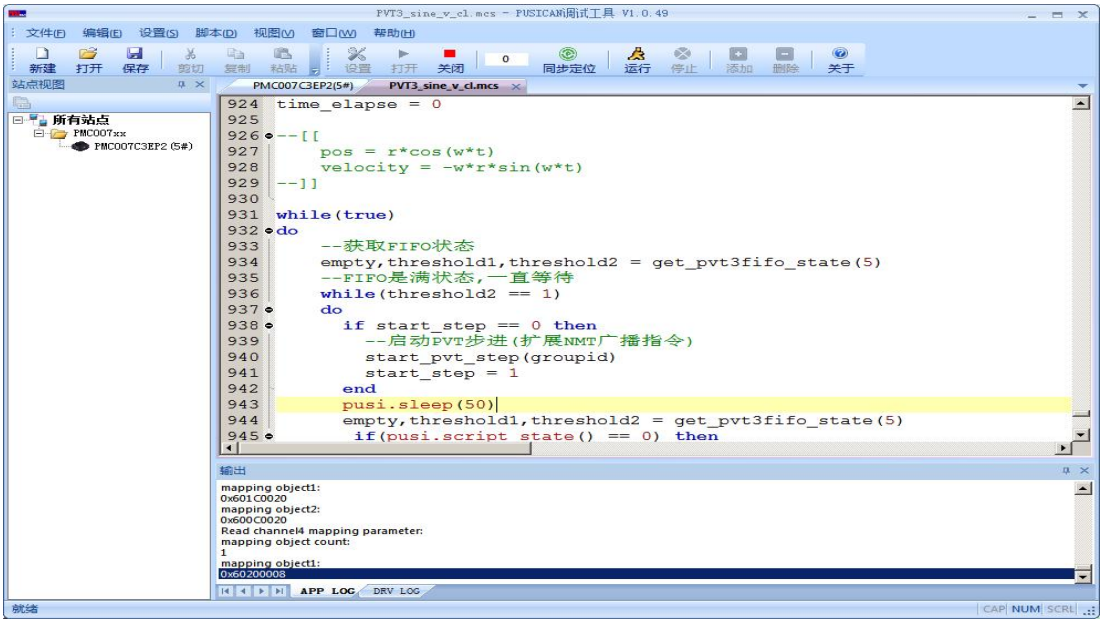
The PUSICAN debugging tool software supports graphical programming, adding process items through the interface, and setting relevant motion parameters to complete simple movements.



7.2 Scripting language support

PUSICAN debugging tool software supports LUA scripting language and has built-in CANOPEN SDO operation functions, which users can call directly in the script program. You can create or open a script file by clicking the "New" and "Open" icons in the upper left corner, and once the script program is written, you can control the program execution through the "Run" and "Stop" icons in the upper right corner. Figure 6-9 below.

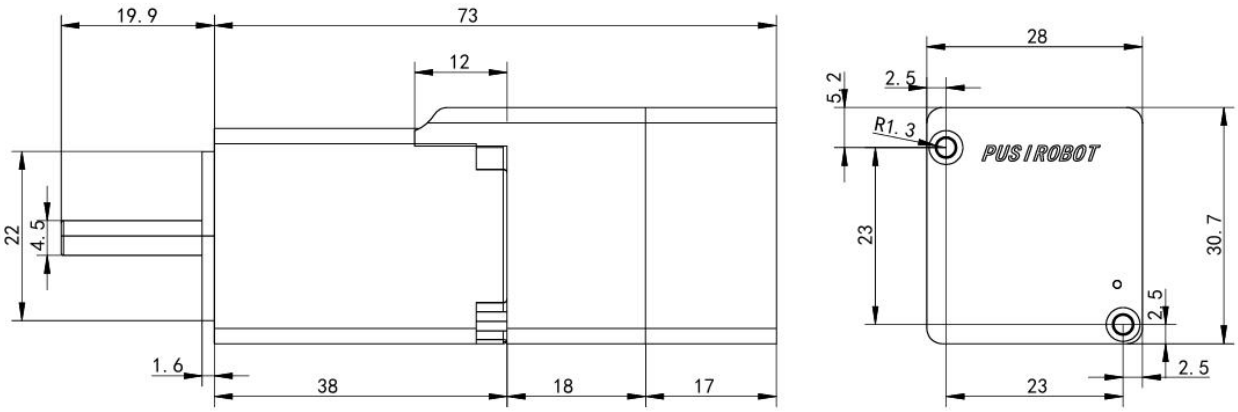
The syntax of LUA scripting language is similar to that of C language, and in application scenarios without special UI interface requirements, users can easily complete complex control loop tasks with the help of the powerful functions of LUA scripting without developing the CANOPEN master program in the host computer.



8 Electrical Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Input Voltage	Normal 25℃	9	24	36	V
Operation Temperature	24V input voltage	-40		85	℃
IO maximum current	source/sink current	0		20	mA
Output current	Normal 25℃	0	2.5	3	A

9 Dimensions (Unit: mm)



10 Appendix 1 PMC006C2 Object dictionary table

Index	Sub index	Object type	Name	Type	Attr.	PDO	Storage type
1000h	--	VAR	Device type	UINT32	RO	NO	ROM
1001h	--	VAR	Error registor	UINT8	RO	Optional	RAM
1002h	--	VAR	manufacturer status register	UINT32	RO	Optional	RAM
1003h	--	ARRAY	pre-defined error field	--	--	--	RAM
	0h		number of errors	UINT8		NO	
	1h-7h		standard error field			Optional	
1005h	--	VAR	COB-ID SYNC	UINT32	RW	NO	ROM
1006h	--	VAR	communication cycle period	UINT32	RW	NO	ROM
1007h	--	VAR	synchronous window length	UINT32	RW	Optional	ROM
1008h	--	VAR	manufacturer device name	Visible String	const	NO	ROM
1009h	--	VAR	manufacturer hardware version	Visible String	const	NO	ROM
100ah	--	VAR	manufacturer software version	Visible String	const	NO	ROM
1014h	--	VAR	COB-ID Emergency message	UINT32	RO	NO	ROM
1015h	--	VAR	Inhibit Time EMCY	UINT16	RW	NO	ROM
1016h	--	ARRAY	Consumer Heartbeat Time	--	--	--	ROM
	0h		number entries	UINT8	RO	NO	
	1h-3h		Consumer Heartbeat Time	UINT32	RW	NO	
1017h	--	VAR	Producer Heartbeat Time	UINT16	RW	NO	ROM
1018h	--	RECORD	Identity Object	--	--	--	ROM
	0h		number of entries	UINT8	RO	NO	
	1h		Vendor ID	UINT32	RO	NO	
	2h		Product code	UINT32	RO	NO	
	3h		Revision number	UINT32	RO	NO	
	4h		Serial number	UINT32	RO	NO	
1200h	--	RECORD	Server SDO parameter	--	--	--	ROM
	0h		number of entries	UINT8	RO	NO	
	1h		COB-ID Client->Server (rx)	UINT32	RO	NO	

	2h		COB-ID Server -> Client (tx)	UINT32	RO	NO	
	3h		Node-ID of the SDO client	UINT32	RW	NO	
1280h	--	RECORD	Client SDO parameter	--	--	--	RAM
	0h		number of entries	UINT8	RO	NO	
	1h		COB-ID Client->Server (tx)	UINT32	RW	NO	
	2h		COB-ID Server -> Client (rx)	UINT32	RW	NO	
	3h		Node-ID of the SDO server	UINT32	RW	NO	
1400h	--	RECORD	receive PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		compatibility entry	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1401h	--	RECORD	receive PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		compatibility entry	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1402h	--	RECORD	receive PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		compatibility entry	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1403h	--	RECORD	receive PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	

	4h		compatibility entry	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1600h	--	RECORD	receive PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the nth application object to be mapped	UINT32	RW	NO	
1601h	--	RECORD	receive PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the nth application object to be mapped	UINT32	RW	NO	
1602h	--	RECORD	receive PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the nth application object to be mapped	UINT32	RW	NO	
1603h	--	RECORD	receive PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the nth application object to be mapped	UINT32	RW	NO	
1800h	--	RECORD	transmit PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		reserved	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1801h	--	RECORD	transmit PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	

	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		reserved	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1802h	--	RECORD	transmit PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		reserved	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1803h	--	RECORD	transmit PDO parameter	--	--	--	ROM
	0h		largest sub-index supported	UINT8	RO	NO	
	1h		COB-ID used by PDO	UINT32	RW	NO	
	2h		transmission type	UINT8	RW	NO	
	3h		inhibit time	UINT16	RW	NO	
	4h		reserved	UINT8	RW	NO	
	5h		event timer	UINT16	RW	NO	
1a00h	--	RECORD	transmit PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the n-th application object to be mapped	UINT32	RW	NO	
1a01h	--	RECORD	transmit PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the n-th application object to be mapped	UINT32	RW	NO	
1a02h	--	RECORD	transmit PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	

	1h-8h		PDO mapping for the n-th application object to be mapped	UINT32	RW	NO	
1a03h	--	RECORD	transmit PDO mapping	--	--	--	ROM
	0h		number of mapped application objects in PDO	UINT8	RO	NO	
	1h-8h		PDO mapping for the n-th application object to be mapped	UINT32	RW	NO	
2002h	--	VAR	Node ID	UINT8	RW	NO	ROM
2003h	--	VAR	Baud rate	UINT8	RW	NO	ROM
2006h	--	VAR	Group ID	UINT8	RW	NO	ROM
2007h	--	VAR	System control	UINT8	RW	NO	RAM
6000h	--	VAR	Error state	UINT8	RW	Optional	RAM
6001h	--	VAR	Controller status	UINT8	RW	Optional	RAM
6002h	--	VAR	Rotation direction	UINT8	RW	NO	RAM
6003h	--	VAR	Max speed	UINT32	RW	NO	RAM
6004h	--	VAR	Step command	UINT32	RW	NO	RAM
6005h	--	VAR	Operation mode	UINT8	RW	NO	RAM
6006h	--	VAR	Start speed	UINT16	RW	NO	ROM
6007h	--	VAR	Stop speed	UINT16	RW	NO	ROM
6008h	--	VAR	Acceleration coefficient	UINT8	RW	NO	ROM
6009h	--	VAR	Deceleration coefficient	UINT8	RW	NO	ROM
600ah	--	VAR	Microstepping	UINT16	RW	NO	ROM
600bh	--	VAR	Max phase current	UINT16	RW	NO	ROM
600ch	--	VAR	Motor position	UINT32	RW	Optional	RAM
600dh	--	VAR	Current attenuation	UINT8	RW	NO	ROM
600eh	--	VAR	Motor enable	UINT8	RW	NO	RAM
600fh	--	RECORD	External emergency stop	UINT8	RW	NO	ROM
	0h		The number of parameters	UINT8	RO	NO	ROM
	1h		External emergency stop enable	UINT8	RW	NO	RAM
	2h		Trigger mode of external emergency stop	UINT8	RW	NO	RAM
6011h	--	RECORD	GPIO parameter	--	--	--	ROM

	0h		The number of GPIO parameters	UINT8	RO	NO	
	1h		GPIO direction	UINT16	RW	NO	
	2h		GPIO configuration	UINT32	RW	NO	
6012h	--	VAR	GPIO value	UINT16	RW	Optional	RAM
6015h	--	VAR	Open loop application switch	UINT8	RW	NO	ROM
6016h	--	VAR	Brake control	UINT8	RW	NO	RAM
6017h			Stall parameter(open loop)	UINT8	RW		ROM
6018h	--	RECORD	Off-line programming parameter 1	--	--	--	ROM
	0h		Number of Off-line programming parameter 1	UINT8	RO	NO	
	1h		Total number of offline programming command	UINT8	RW	NO	
	2h		Offline operation enable automatically	UINT8	RW	NO	
6019h	--	RECORD	Off-line programming parameter 2	--	--	--	RAM
	0h		Number of Off-line programming parameter 2	UINT8	RO	NO	
	1h		Off-line parameter pointer	UINT8	RW	NO	
	2h		Off-line command	UINT32	RW	NO	
	3h		Offline command preservation	UINT8	RW	NO	
	4h		Run current command	UINT8	RW	NO	
601ah	--	VAR	Jitter delay of external emergency stop	UINT16	RW	NO	ROM
601bh	--	VAR	Locked-Rotor configuration	UINT8	RW	NO	ROM
601ch	--	VAR	Absolute position step	INT32	RW	NO	RAM
601dh	--	RECORD	PV step	--	--	--	RAM
	0h		The number of PV step parameter	UINT8	RO	NO	
	1h		PV speed	INT32	RW	NO	
	2h		PV position	INT32	RW	NO	
6020h	--	VAR	Termination step	UINT8	RW	NO	RAM
6021h	--	VAR	Encoder CPR	UINT16	RW	NO	ROM

6022h	--	VAR	Position saving value power-down	INT32	RO	NO	ROM
6023h	--	VAR	Closed-loop parameter KP	UINT8	RW	NO	ROM
6024h	--	VAR	Closed-loop parameter KI	UINT8	RW	NO	ROM
6025h	--	VAR	Closed-loop parameter KD	UINT8	RW	NO	ROM
6026h	--	VAR	Closed-loop Pre-filter parameter	INT8	RW	NO	ROM
6027h	--	VAR	Closed-loop post-filter parameter	INT16	RW	NO	ROM
6028h	--	VAR	Closed-loop stall length	INT16	RW	NO	ROM
6029h	--	VAR	Enable closed-loop torque loop	UINT8	RW	NO	ROM
602Ah	--	VAR	Enable saving automatically when power is off	UINT8	RW	NO	ROM
602Bh	--	VAR	Analogue input	UINT16	RW	Optional	RAM
602Ch	--	RECORD	Step notification	--	--	--	RAM
	0h		Number of parameters	UINT8	RO	NO	
	1h		Step notification status	UINT8	RW	Optional	
	2h		Step notification position 1	INT32	RW	Optional	
	3h		Step notification position 2	INT32	RW	Optional	
602Dh	--	RECORD	PP/PVmode parameters 1	--	--	--	ROM
	0h		Number of parameters	UINT8	RO	NO	
	1h		PP Accelerated speed	UINT32	RW	Optional	
	2h		PP Dccelerated speed	UINT32	RW	Optional	
	3h		PP Initial speed	UINT32	RW	Optional	
	4h		PP Stop speed	UINT32	RW	Optional	
602Eh	--	RECORD	PP/PVmode parameters 2	--	--	--	RAM
	0h		Number of parameters	UINT8	RO	NO	
	1h		PP control word	UINT16	RW	Optional	
	2h		PP status word	UINT16	RW	Optional	
	3h		PP running speed	INT32	RW	Optional	
	4h		PP target location	INT32	RW	Optional	
602Fh	--	RECORD	Analog parameters	--	--	--	ROM
	0h		Number of parameters	UINT8	RO	NO	

	1h		Analog enabled	UINT8	RW	NO	
	2h		Analog start_adc	UINT16	RW	NO	
	3h		Analog end_adc	UINT16	RW	NO	
	4h		Analog adjustment lower limit	UINT16	RW	NO	
	5h		Analog adjustment upper limit	UINT16	RW	NO	
	6h		Analog Adjustment Interval	UINT16	RW	NO	
	7h		Analog adjustment trigger value	UINT16	RW	NO	
	8h		Analog speed/position1	INT32	RW	NO	
	9h		Analog speed/position2	INT32	RW	NO	
6030h	--	VAR	real-time speed	INT32	RW	Optional	RAM
6031h	--	RECORD	power-down behavior control	--	--	--	ROM
	0h		Number of parameters	UINT8	RO	NO	
	1h		Power-down behavior control word	UINT16	RW	Optional	
	2h		Power off motor enabling threshold	UINT16	RW	Optional	
	3h		Brake lock threshold	UINT16	RW	Optional	
6033h	--	VAR	Motor wiring control	U8	RW	NO	ROM
6034h	--	VAR	Calibration zero position	INT32	RW	NO	ROM
6035h	--	VAR	Encoder position	INT32	RW	Optional	RAM

11 Appendix 2 CANopen Communication Examples

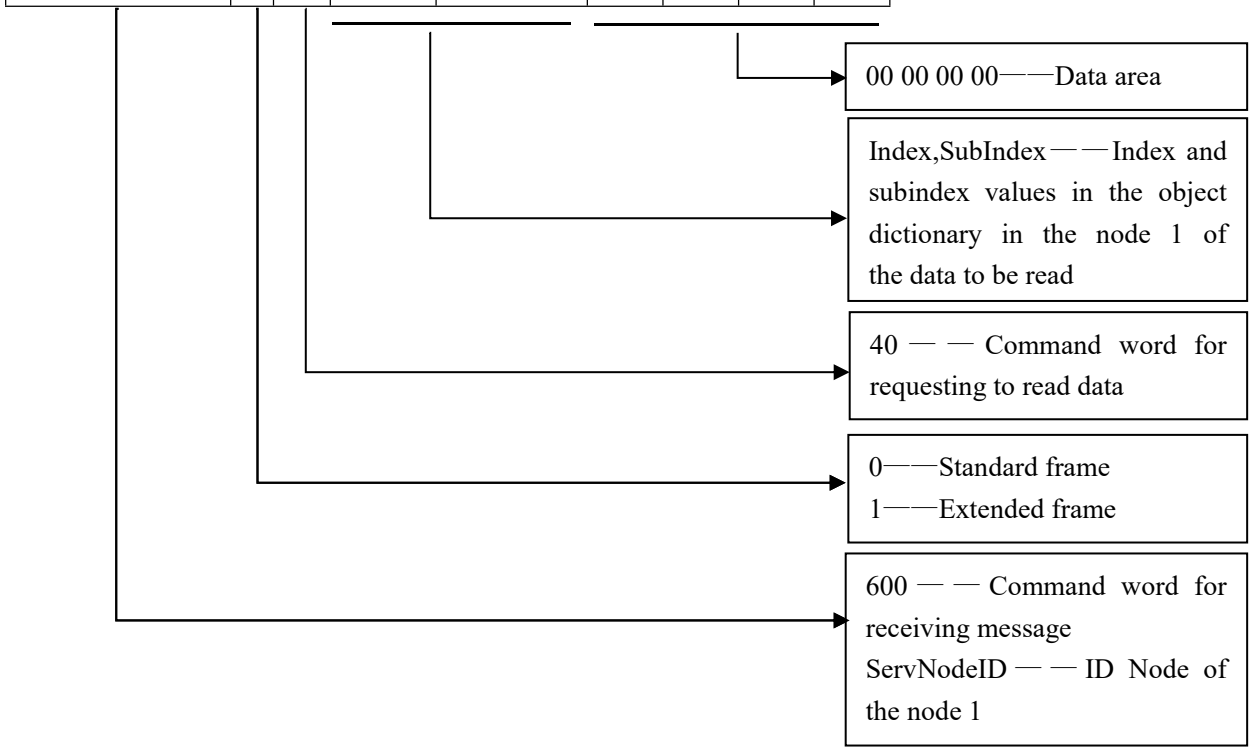
11.1 SDO Read/Write Examples

11.1.1 SDO Read

11.1.1.1 Data frame format

Master send:

600+ServNodeID	0	40	Index	Subindex	00	00	00	00



Slave response:

When the data length is 1 byte								
580+ServNodeID	0	4F	Index	Subindex	d0	0	0	0
When the data length is 2 bytes								
580+ServNodeID	0	4B	Index	Subindex	d0	d1	0	0
When the data length is 3 bytes								
580+ServNodeID	0	47	Index	Subindex	d0	d1	d2	0
When the data length is 4 bytes								
580+ServNodeID	0	43	Index	Subindex	d0	d1	d2	d3

11.1.1.2 SDO Read example

Master send: 605 40 01 60 00 00 00 00 00

Slave response: 585 4F 01 60 00 08 00 00 00

The master initiated a read request to the device whose node ID is 5. The index and subindex of the request are 0x6001 and 0x00 respectively, which corresponds to controller status parameter in the PMC007 Object Dictionary. The slave response 4F indicates that the parameter length is one byte, the data is 0x08 and the device is in busy state.

11.1.2 SDO Write in

11.1.2.1 Data frame format

Master send:

When the data length is 1 byte								
600+ServNodeID	0	2F	Index	Subindex	d0	0	0	0
When the data length is 2 byte								
600+ServNodeID	0	2B	Index	Subindex	d0	d1	0	0
When the data length is 3 byte								
600+ServNodeID	0	27	Index	Subindex	d0	d1	d2	0
When the data length is 4 byte								
600+ServNodeID	0	23	Index	Subindex	d0	d1	d2	d3

Correct response from the slave station:

580+ServNodeID	0	60	Index	Subindex	00	00	00	00

00 00 00 00——Data area

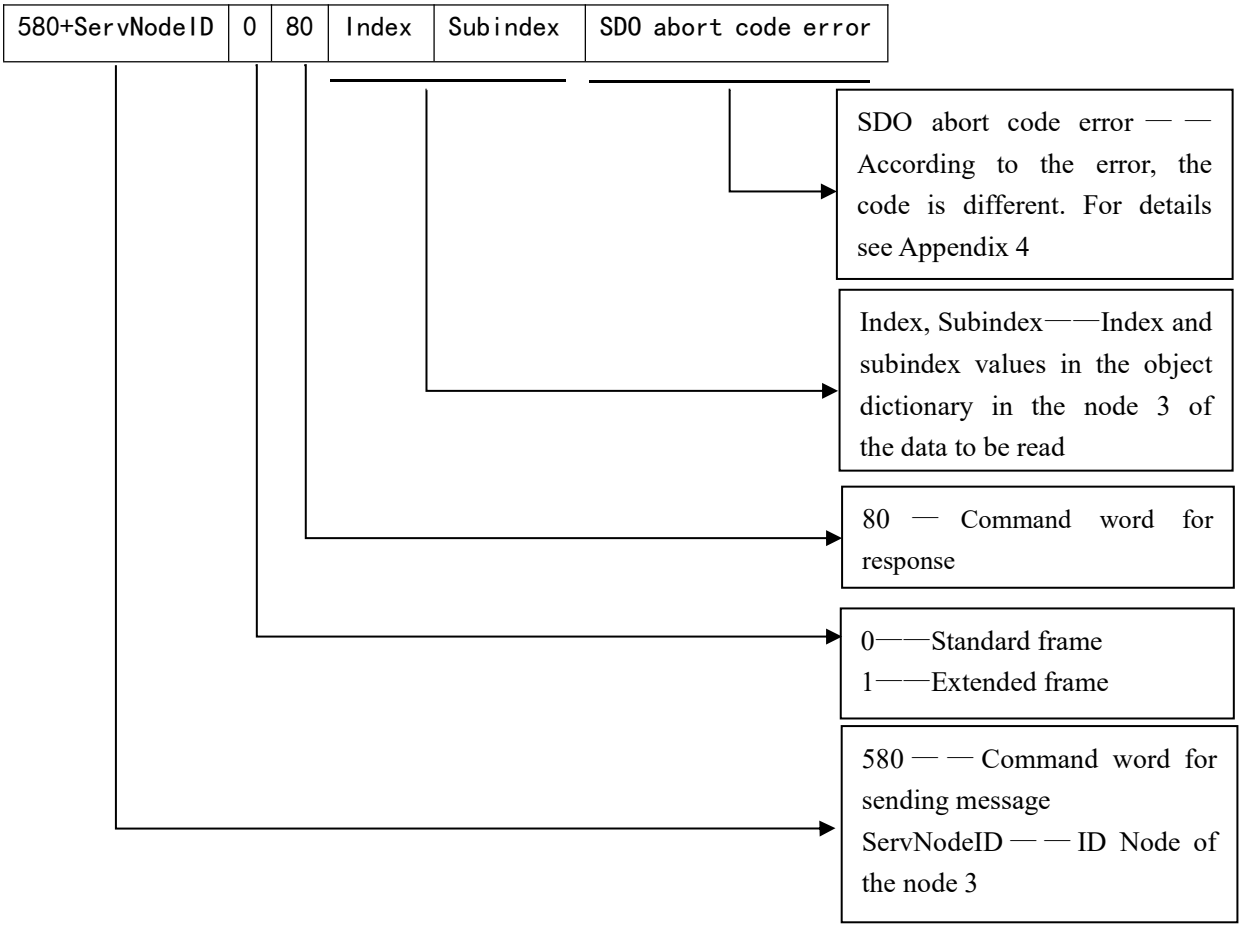
Index, Subindex——Index and subindex values in the object dictionary in the node 3 of the data to be read

60 — Command word for response to write data successfully

0——Standard frame
1——Extended frame

580——Send SDO
ServNodeID — — ID Node of the node 3

Error response from the slave station:



Note: Abort code error SDO returns the corresponding parameters according to the specific error. The specific parameters are shown in Appendix 4.

11.1.2.2 SDO Write example

Master send: 605 2F 03 20 00 07 00 00 00
Slave response: 585 60 03 20 00 00 00 00 00

The master initiated a write request to the device whose node ID is 5. The index and subindex of the request are 0x2003 and 0x00 respectively, which corresponds to the baud rate setting parameter in the PMC007 Object Dictionary. and the write data is 7, which indicates the baud rate is set to 800Kbit/s. The slave response 60 indicates the data is written successfully.

Master send: 605 23 04 60 00 80 0C 00 00
Slave response: 585 80 04 60 00 22 00 00 08

The master initiated a write request to the device whose node ID is 5. The index and subindex of the request are 0x6004 and 0x00 respectively, which corresponds to the step command parameter in the PMC007 Object Dictionary. and the write data is C80(3200), which indicates that the motor performs 3200 steps. The slave response 60 indicates the data is written unsuccessfully, and error code is 0x08000022. See Appendix 4 we can know that the error code indicates that the data cannot be transferred or saved to the application due to the current

device status. Check whether the controller status parameter is that the external stop is enabled and whether there is an error in the error state.

12 Appendix 3 PDO configuration example

12.1 PDO overview

PDO communication is based on the Producer/Consumer model, which is mainly used to transfer real-time data. The node which generated data puts the data with its own node ID on the bus, and nodes which need the data can be configured to receive the data sent by the node. The transmission of PDO is triggered by the event, which can represent a change in a PDO variable and can also be a time of expiration or a specific message to be received. Process data is transmitted directly in an CAN message without a protocol header file. The length of a PDO is between 0 and 8 bytes.

PDOs is included in the mapping parameter and communication parameter. PMC007xx supports 4 PDOs.

12.1.1 The structure PDO——Mapping parameter

A PDO in the Object Dictionary consists of adjacent items. The mapping parameters define the connection of these items. A mapping parameter defines a data source through an index, a subindex, and a number of bits.

For example:

Index	*Sub-index*	*Object Data*	*Description*
0x1A00	0	4	Number of mapped entries
	1	0x20000310	The entry at index 0x2000, sub-index 3, with a length of 16 bit, is mapped to bytes 0 and 1 within the CAN message.
	2	0x20000108	The entry at index 0x2000, sub-index 1, with a length of 8 bit, is mapped to byte 2 within the CAN message.
	...	...	

Table 1: Example for mapping parameters for the first TPDO

A CAN message has not more than 8 bytes. This means that there can send 8 object items at most when there is only one PDO used.

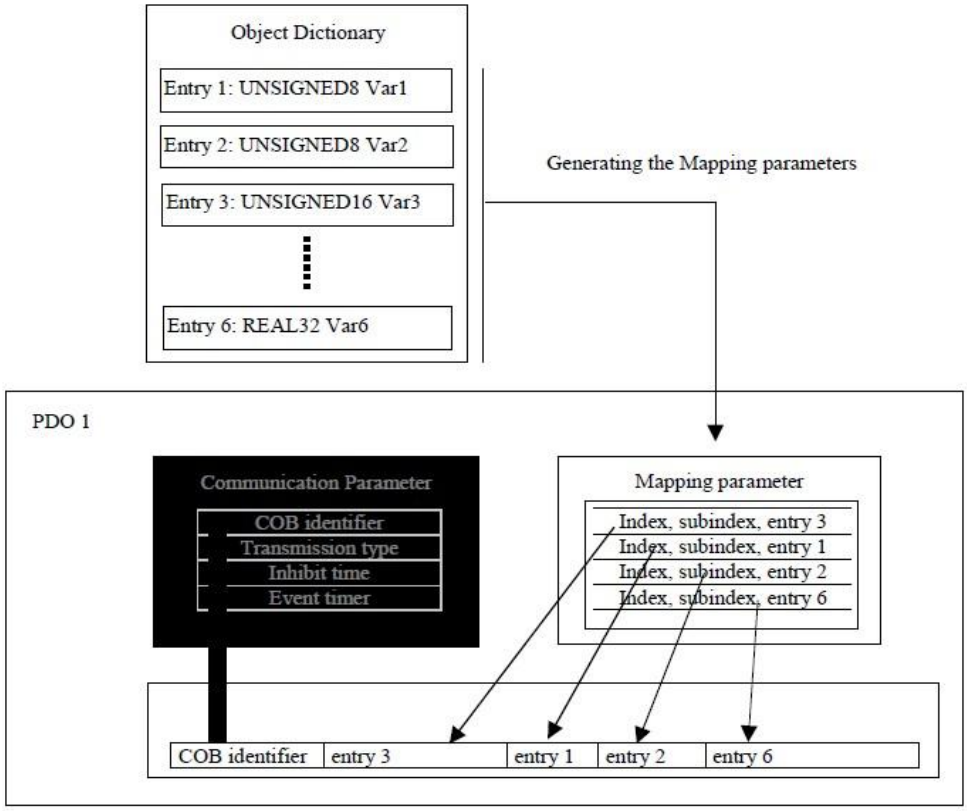


Figure 3: Mapping of Object Dictionary entries into a PDO

12.1.2 The structure PDO——Communication parameter

In order to transmit a PDO, the communication parameter defines the nature of the transport and the CAN identifier.

Index	Sub-index	Object Data	Description
1800h	0	Number on entries	
	1	COB-ID	CAN identifier for the PDO
	2	Transmission Type	transmission type of the PDO
	3	Inhibit Time	minimum inhibit time for a TPDO
	4	reserved	reserved
	5	Event Time	maximum time between two TPDOs

Table 4: Communication parameter for the first TPDO

PDO communication parameter is an item in the Object Dictionary.

(RPDOs: index 0x1400 – 0x15FF,TPDOs: 0x1800-0x19FF)

If allowed, the communication parameter can be modified by the CAN with the help of the data service object.

COB-ID(CAN identifier. Subindex 1)

COB-ID as proof of identity, the priority of PDO is before the bus access. For every CAN message, there is only one sender (producer). However, it allows multiple recipients (consumers)

for the existing message.

Bit	*31*	*30*	*29*	*28 – 11*	*10 - 0*
11-bit-ID	0/1	0/1	0	00000000000000000	11-bit identifier
29-bit-ID	0/1	0/1	1	29-bit identifier	

Table 5: Structure of a COB-ID for PDOs

The thirtieth bit is 0, which indicates that a remote transmission request (RTR) is allowed for this PDO.

PDO COB-ID distribution:

PDO1(send)	181H-1FFH
PDO1(receive)	201H-27FH
PDO2(send)	281H-2FFH
PDO2(receive)	301H-37FH
PDO3(send)	381H-3FFH
PDO3(receive)	401H-47FH
PDO4(send)	481H-4FFH
PDO4(receive)	501H-57FH

12.1.3 PDO Trigger mode

Sending of PDO can be triggered by the following methods:

1) Event trigger.

2) Time trigger.

3) Single query.

4) Synchronization.

When only use event to trigger the sending of PDO, once the event process is changed, the PDO is sent. It may bring very serious consequences, that is, when the frequency of a process data change is very high, the PDO is sent uninterruptedly , that will cause the message of other nodes is not sent out, which seriously affect the efficiency of the bus.

CANopen uses the " Inhibit time " mechanism to solve this problem. Inhibit time is a configurable time period in units of 100 us. The same PDO sends at least this time interval, so it can determine the maximum transmission frequency of an event triggered PDO.

Generally, the sending of PDO can be triggered by a combination of any of the trigger mode. But the most common way is to combine the Event trigger and Time trigger. In the case of single event trigger, since process data did not change for a long time (such as temperature variables), the PDO have not been triggered for a long time. It will affect the nodes just joined the network. So if plus time triggered mode, PDO is forced to send again within the stipulated time. For

example, for a PDO, the inhibition time is configured as 5, event timer is configured as 250, so the PDO can be sent when process data changes. The minimal interval of sending is 5ms, on the other hand, no matter whether there is no change in the data, the PDO will be sent every 250ms.

Configuration of PDO trigger mode is realized through setting subindex 2 in the Object Dictionary of PDO communication parameter. The range of the subindex is 0-255 The following lists the different values for different trigger modes.

0: PDO is sent after SYNC is received, but not cycle.

1-240: PDO is sent periodically after SYNC is received . The value is the number of SYNC between two send of PDO .

255：Event trigger.

12.2 PDO Configuration example

PMC007xx supports PDO mapping by SDO configuration. To configure the GPIO value to be TPDO1 as an example, the SDO is sent as:

Set the COB-ID of communication to 187, which means that the device with node ID 7 receives this PDO

Sent by the main station: 607 23 00 18 01 87 01 00 80

Clear channel information

Main station sending: 607 2F 00 1A 00 00 00 00 00 00

Set as event triggered

Main station sending: 607 2F 00 18 02 FF 00 00 00 00

Set the Inhibit time to 5ms

Sent by the main station: 607 2B 00 18 03 32 00 00 00

Set the Event time to 1000ms

Sent by the main station: 607 2B 00 18 05 E8 03 00 00

Set mapping parameters to map 0x6012 to TPDO1

Main station sending: 607 23 00 1A 01 10 00 12 60

Set the mapping channel TPDO2 parameter to 0

Sent by the main station: 607 23 00 1A 02 00 00 00 00 00

Set the mapping channel TPDO3 parameter to 0

Main station sending: 607 23 00 1A 03 00 00 00 00 00

Set the number of mapping entries 1

Main station sending: 607 2F 00 1A 00 01 00 00 00

Resend communication site information

Sent by the main station: 607 23 00 18 01 83 01 00 00After the configuration is completed, PMC007 will send PDO message every 1s. When the of value GPIO port is changed, PMC007 will also issue the message.

187 03 00

The message indicates that GPIO1 and GPIO2 are both high level.

13 Appendix 4 SDO abort code error

Abort code	Code function description
05030000	No alternation of trigger bits
05040000	SDO protocol timeout
05040001	Illegal or unknown Client/Server command word
05040002	Invalid block size (only Transfer Block mode)
05040003	Invalid serial number (only Transfer Block mode)
05030004	CRC error (only Transfer Block mode)
05030005	Out of memory
06010000	Access is not supported for the Object.
06010001	Try to read write-only objects
06010002	Try to write read-only objects
06020000	Object does not exist in the Object Dictionary
06040041	Object cannot be mapped to PDO
06040042	The number and length of the mapped object exceeds the PDO length
06040043	General parameters are not compatible
06040047	General equipment is not compatible
06060000	Hardware error causes the object access failure
06060010	Data type does not match, and service parameter length does not match
06060012	Data type does not match, the service parameter is too large
06060013	Data type does not match, the service parameter is too small
06090011	The subindex does not exist
06090030	Beyond the range of the parameter values (when write access)
06090031	Parameter value is written too large
06090032	Parameter value is written too small
06090036	The maximum value is less than the minimum value
08000000	General error
08000020	Data cannot be transferred or saved to applications
08000021	Due to local control, data cannot be transferred or saved to applications
08000022	Due to the current device status, data cannot be transferred or saved to applications
08000023	The dynamic condition of Object dictionary generates error or Object Dictionary does not exist