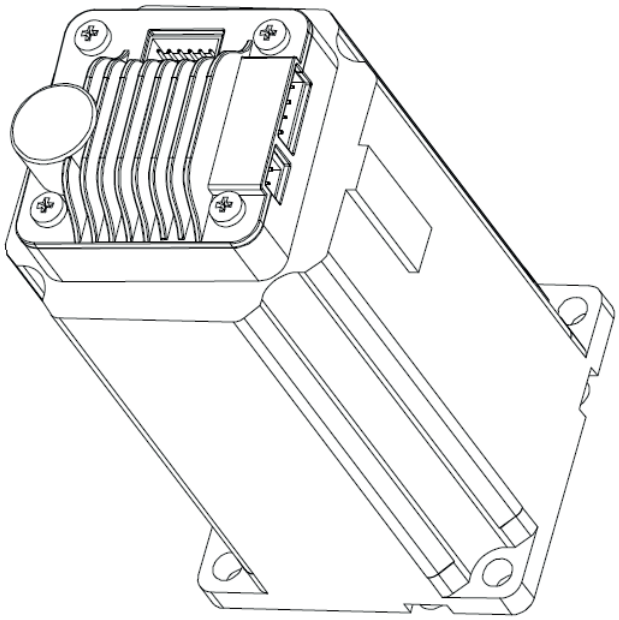
USER MANUAL
PIO007xx series



Catalog

1 Introduction

1.1 Statement of intellectual property right

PMC007xx 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.

Since PMC007xx 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. 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

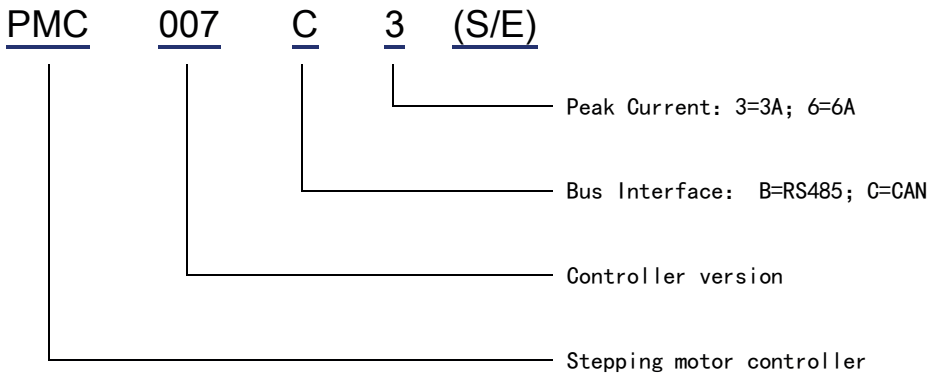
PMC007xx is a kind of miniature integrated stepper motor microstepping controller, which can be directly installed in the rear of 42/57/86 etc series stepper motor. The series controller provides a variety of models which can be chosen based on bus control of RS485/CAN and different current value. It is easy to achieve industrial control network of as many as 120 nodes, which can achieve closed-loop control based on encoder according to the requirements of user.

2.2 Features

- ✓ Wide range of 12-48V single voltage supply
- ✓ Output current 0.4A ~ 6A, Adjustable phase current by commands
- ✓ Automatic control of S curve acceleration and deceleration
- ✓ Start/stop speed can be configured by commands
- ✓ 2 external switches input ports for configurable emergency stop
- ✓ Support 0/2/4/8/16/32/64/128/256 microstepping resolution
- ✓ 7 GPIO ports, direction and pullup/pulldown register of each port can be separately configured
- ✓ Senseless stall detection.
- ✓ Burning and off-line automatic execution of custom program
- ✓ closed-loop control based on encoder(a additional module support is required)
- ✓ Configurable instruction of automatic current attenuation function
- ✓ Miniature size 42mmx42mmx18mm
- ✓ Precision aluminum shell, conducive to the protection and heat dissipation
- ✓ Control routines and the underlying driver based on VC++

2.3 Production & Ordering Information

In order to serve you quicker and better, please provide the product number in following format when ordering PMC007xx:



Remarks:

S: IO strapped down version; E: Close loop version;

Please be sure to contact the sales staff to confirm whether the required model is in a normal state of supply before placing an order.

3 Connector Description

3.1 Terminal port location

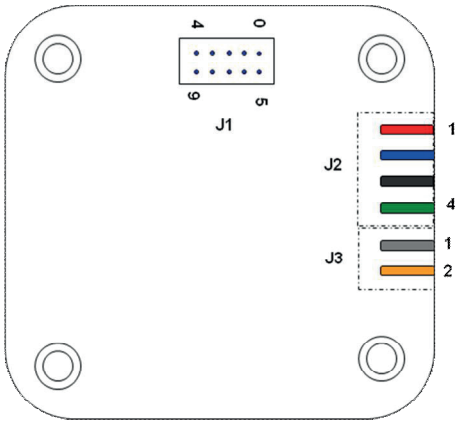


Figure 3-1

3.2 Motor connection J2

Pin no:	1	2	3	4
Designator:	M10	M11	M20	M21

Description:

M10, M11: the stepper motor phase A

M20, M21: the stepper motor phase B

WARNING: Incorrect connection of power or phase will permanently damage the controller!

3.3 Power connection J3

Pin no:	1	2
Designator:	GND	VCC

Description:

VCC: Supply voltage, 12~48VDC;

GND: Supply voltage ground;

3.4 Signal connectionJ1

Pin no:	0	1	2	3	4
Designator	GND	EXT3/ENC1	ENC2	CANH	CANL
Pin no:	5	6	7	8	9
Designator	VDD	EXT2	EXT1	GPIO8	FSET

Description:

GPIO8: Generic I/O port;

DVDD: Voltage output(+5V);

GND: Digital ground;

EXT1: External limit switch signal input 1;

EXT2: External limit switch signal input 2 (open loop);

CANH: Connect to the CAN transceiver module;

CANL: Connect to the CAN transceiver module;

ENC1-2: Encoder input; ENC1 also can be used as EXT3;

FSET: Restore factory settings, Active low level;

WARNING: The voltage of all signal ports must be between -0.3V~+5.3V.

3.5 RS485 Network Operation

It provides a network scheme which uses RS485 bus to connect multiple PMC007xx controllers in the figure 3-2. Maximum communication distance of the scheme is 1200 meters. If the transmission distance is over 50 meters when using a pair of twisted pair to connect all the nodes, both ends of the network should be terminated with 120Ω terminating resistors to prevent signal reflection and overshoot. Meanwhile, the RS485 of host and the controller of each node must be common-grounded.

Warning: The upper and lower limit of the signal threshold of RS-485 is ±200mV. That is, when A-B>200mV, the bus state should be expressed as "1"; when A-B<-200mV, the bus state should be expressed as "0". However, when the A-B is between ±200mV, the bus state is not determined. So in the actual network, it is recommended that the user set pull-up and pull-down resistance in the A, B line, to avoid this uncertainty.

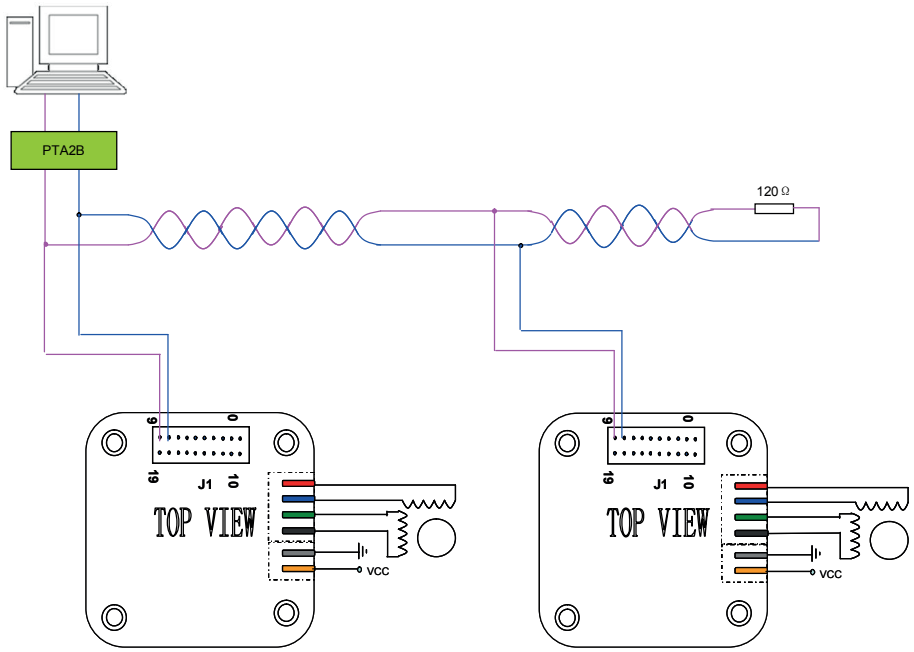


Figure 3-2

The PTA2B in the diagram is a USB-RS485 conversion module provided by the third party.

3.6 CAN Network Operation

Using the CAN bus connection can reach a maximum 5000 meters transmission

distance. It provides a network scheme that uses CAN bus to connect multiple PMC007xx controllers in the figure 3-3, which are compatible with CAN2.0A and CAN2.0B two technical specifications, and can be connected to 127 nodes.

Note: it is recommended to use the CAN bus specified 120 ohm shielded twisted pair, and the ends of the twisted pair are required to connect a 120 ohm termination resistor. In addition, The PTA2C in the diagram is a USB-CAN conversion module provided by the third party.

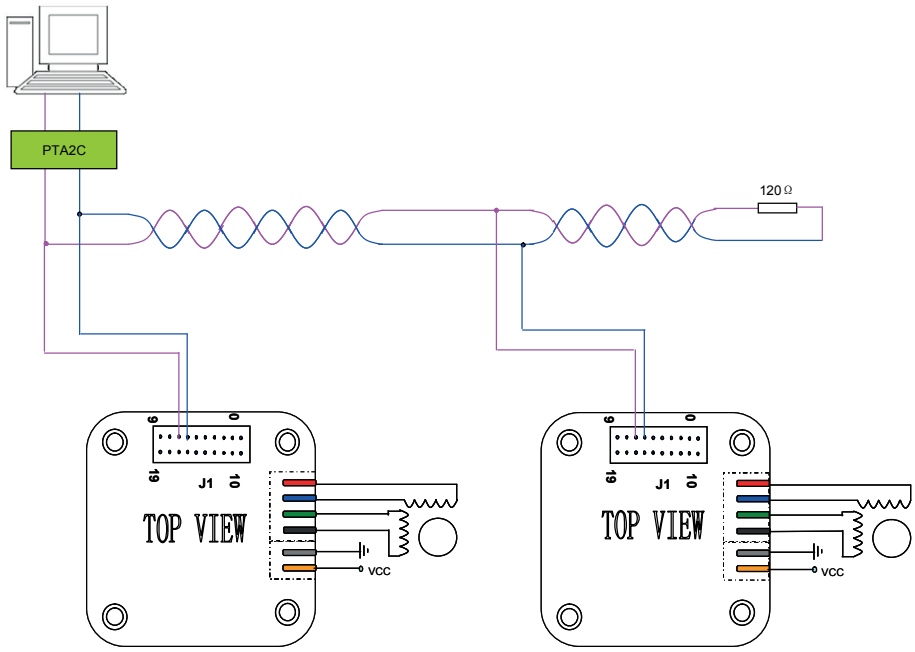


Figure 3-3

PMC007xx supports standard DS301 CANopen protocol, CQPUSI provides a dedicated debugging tool software PUSICAN for debugging PMC007xx network. The tool software currently supports a variety of USB2CAN modules which has been mainstream brands on the market. And also can add support of other CAN converters according to the needs of users.

3.7 Homing Switch Connection

There are two dedicated pins Ext1 and Ext2 in PMC007xx controller, which are used to connect the external homing(zero point, position Home) switch. The trigger mode of each pin can be selected by the instruction. Default trigger mode is falling edge trigger. The user can also open or close any of the limit switches by the instruction.

There are two kinds of trigger mode, which are the rising edge trigger and falling edge trigger. When falling edge trigger is selected, make the internal pull-up resistor enable. The input pins can be directly connected to the collector of optical coupler as shown in Figure 3-4 left; when rising edge trigger is selected, make the internal pull-down resistor enable to clamp the input port to low reliably in the absence of trigger state, such as figure 3-4 right.

The stability and accuracy of detection on the external limit switch are related to the waveform slope (Slew rate) of the sensor output. A Schmidt shaping

circuit is built into the EXT1 and EXT2 pins of PMC007xx controller for improving the precision of the trigger position. At the same time, in order to reduce the probability of false triggering, user can set the controller's trigger stable time through the PUSICAN tool software.

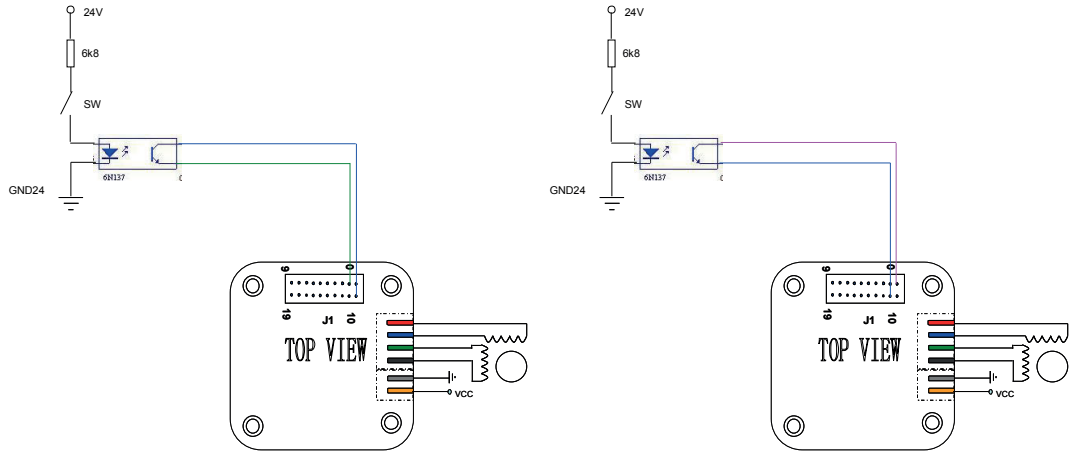


图 3-4

Above illustrated connection is suit for the case in which the optical emitter is normally open, however, in some of scenarios such as ram pump driver. The optical emitter is normally closed. When the block moves to a certain position, receiver is disconnected. So when using the falling edge trigger, configuring pull-down resistor enable, input pin can be connected directly to the emitter of opticalcoupler as Figure 3-5 left; when using the rising edge trigger, configuring pull-up resistor enable, input pin can be connected directly to the collector of opticalcoupler as Figure 3-5 right.

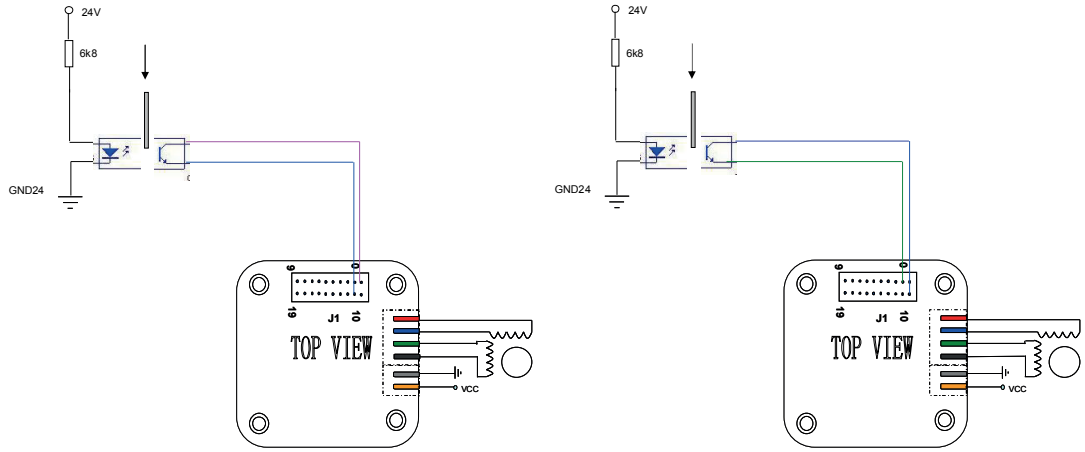


Figure 3-5

4 CANopen communication

4.1 CANopen introduction

CAL provides all network management services and message protocols, but it does not define the contents of objects or the kind of objects being communicated(it defines how, not what). This is where CANopen enters the picture. CANopen is built

on top of CAL, using a subset of CAL services and communication protocols, and providing an implementation of a distributed control system. It does this in such a way that nodes can range from simple to complex in their functionality without compromising the interoperability between the nodes in the network.

Central concept in CANopen is the Device Object Dictionary(OD), a concept used in other fieldbus systems as well(Profibus, Interbus-S). CANopen communication through the object dictionary (OD) can access all the parameters of the driver. Note that the Object Dictionary of CAL, it is an implementation aspect of CANopen. The Object Dictionary which PMC007 is supported is shown in Appendix 1.

CANopen communication model defines the following 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 for alarm event of a transport driver.
NMT	Network Management	Used for CANopen network management.
Heartbeat	Error Control Protocol	Used for monitoring the life status of all nodes.

4.2 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	11or29 bits	1bit	6 bits	0~8byte	16bits	2 bits	7bits

When the transceiver is realized through the software, except to the COB-ID and data domain, the other bits in the frame structure are completed by the hardware of CAN transceiver controller. So Users only need to limit the COB-ID and data domain.

Note: This drive uses standard frame format, COB-ID is 11 bits. Remote frame is not supported temporarily.

The distribution of COB-ID is as follows:

Function Code				NODE ID(Node address)						
10	9	8	7	6	5	4	3	2	1	0

The controller parameters are accessed by the SDO read and write objects. For the

drive, that status information is needed to report to the main station in real time can be achieved by configuring the PDO.

4.3 CAN communication configuration

PMC007 factory default settings: node ID is 5, the baud rate is 125Kbit/s. The user can modify the settings by supporting the CANOPEN master debugging tool.

4.3.1 Node ID

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

4.3.2 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	5

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

4.4 System information acquisition

4.4.1 Device node name

Object name	Device node name
SDO ID	0x1008
Object type	string, ro

Range	-
Storage type	ROM
Default value	-

4.4.2 Hardware version

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

4.4.3 Software version

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

4.4.4 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

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.

4.5 Motor control parameters

4.5.1 Error status

Object name	Driving state

SDO ID	0x6000
Object type	U8,rw
Range	bit
Storage type	RAM
Default value	0

Driver state are defined as follows:

Bit0: UVLO, low voltage fault

Bit1: BERR, coil B error

Bit2: AERR, coil A error

Bit3: BOC, B over current

Bit4: AOC, A over current

Bit5: TSD, over temperature shutdown

Write 1 to the corresponding bit, the corresponding error state will be cleared.

4.5.2 Controller status

Object name	Controller status
SDO ID	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

Each bit except for busy state can be written 1 to clear the response state.

4.5.3 Rotation direction

Object name	rotation direction
SDO ID	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

4.5.4 Maximum speed

Object name	Maximum speed (pps)
SDO ID	0x6003
Object type	U32,rw
Range	-32000 ~ +32000
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.

4.5.5 Relative displacement command

Object name	Relative displacement command
SDO ID	0x6004
Object type	U32,rw
Range	0x0-0xFFFFFFFF
Storage type	RAM
Default value	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.

When the controller is in the busy state, the step command will be ignored. When the error state and the other bits in the controller status are valid, you need to clear up before you start the step command.

4.5.6 Absolute displacement command

Object name	Absolute displacement command
SDO ID	0x601c
Object type	U32,rw
Range	-2147483647 ~ +2147483647
Storage type	RAM
Default value	0

Absolute displacement command gives the target position, then the controller will automatically calculate the direction and the required step number, and control the motor rotate to the target position at the setting speed and acceleration.

4.5.7 Stop stepping command

Object name	Stop stepping command
SDO ID	0x6020
Object type	U32,rw
Range	0
Storage type	RAM
Default value	0

The command immediately terminates the motor running, regardless of the current mode is location mode or speed mode.

4.5.8 Operation mode

Object name	Operation mode
SDO ID	0x6005
Object type	U8,rw
Range	0,1
Storage type	RAM
Default value	0

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

0: Position mode

1: Speed mode

When the operation mode is switched from the speed mode to the position mode, the motor will stopping at the setting deceleration.

4.5.9 Start speed

Object name	Start speed(Unit: pps)
SDO ID	0x6006
Object type	U16,rw
Range	0-0xFFFF
Storage type	ROM
Default value	400

4.5.10 Stop speed

Object name	Stop speed(Unit: pps)
SDO ID	0x6007
Object type	U16,rw
Range	0-0xFFFF

Storage type	ROM
Default value	0

4.5.11 Acceleration coefficient

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

4.5.12 Deceleration coefficient

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

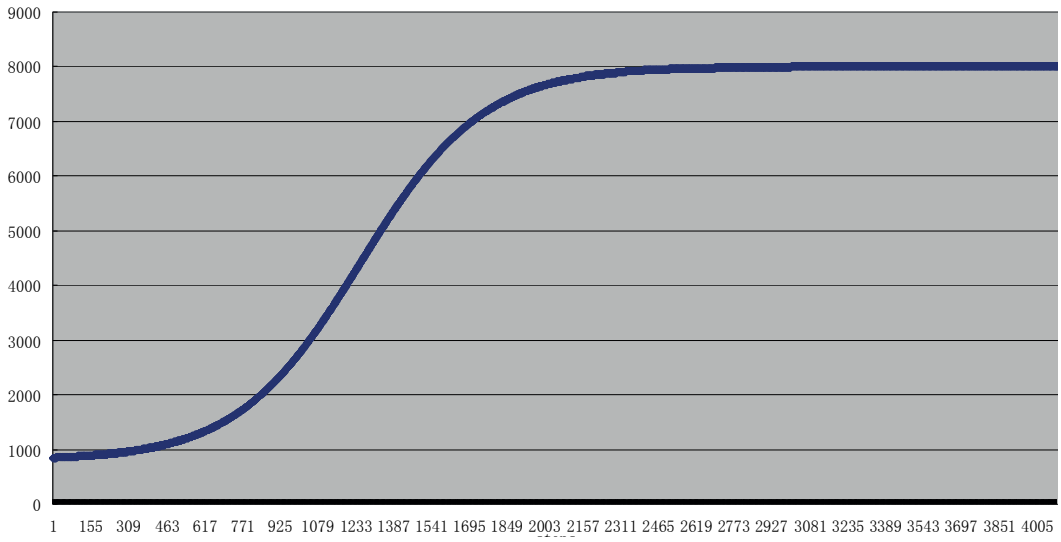


Figure 4-1

The PMC007xx controller uses S curve acceleration and deceleration. As shown in Figure 4-1, Start speed, stop speed, acceleration and deceleration can be configured separately. There are 1~8 a total of 8 gears for acceleration and deceleration. The relationship between each gear and the corresponding acceleration value is shown in the following table.

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

4.5.13 Microstepping

Object name	Microstepping
SDO ID	0x600A
Object type	U16, rw
Range	0, 2, 4, 8, 16, 32, 64, 128, 256
Storage type	ROM
Default value	0

4.5.14 Maximum phase current

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

4.5.15 Motor position

Object name	Motor position
SDO ID	0x600C
Object type	S32, rw
Range	-2147483648-2147483647
Storage type	RAM
Default value	0

When a stepping order is issued, the controller automatically records the current position which is represented by an signed integer according to the given number of steps. A positive value indicates clockwise rotation, and a negative

value indicates a counterclockwise rotation.

Current position value is calculated by the number of steps, so when users need to change the microstepping, shall read the position information firstly and then change the microstepping, in order to avoid the position conversion error.

When the controller power down, the position information is automatically cleared.

4.5.16 Current attenuation

Object name	Current attenuation coefficient
SDO ID	0x600D
Object type	U8,rw
Range	0-3
Storage type	ROM
Default value	0

4.5.17 Motor enable

Object name	Motor enable
SDO ID	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

After setting the offline, controller immediately release the control of motor and the current step command is terminated, phase current is reduced to 0. All subsequent step command issued by host computer cannot be processed, until the user reset motor enable.

4.5.18 Stall set

Object name	After set stall, whether motor stop or not.
SDO ID	0x601b
Object type	Record
Range	0-1
Storage type	ROM
Default value	0

4.5.19 Stall parameters

Object name	Set stall detection parameters
SDO ID	0x6017
Object type	U16,rw
Range	bit
Storage type	ROM
Default value	0

The values of detection parameters are defined as follows:

Bit0: Must be set to 0

Bit1~3 : Sample length

Bit4~5: Delay count

Bit6~7: Divisor

Bit8~15 : Trigger threshold

Controller PMC007xx uses two-phase winding back EMF to realize senseless stall detection. Its accuracy is affected by many factors, such as current, microstepping, voltage, motor parameters and so on, and the weight of motor speed and phase inductance is significant. The following is a reference configuration table for several typical application scenarios:

Motor type	Phase inductance mH	Current mA	Voltage V	Speed Range pps	Microstepping	Stall parameter
28	5.7	900	24	1600-16000	8	0x64e6
42	2.8	1600	24	1600-16000	8	0x64b4
57	1.1	3000	24	1600-16000	8	0xc8b2

4.6 External emergency stop

The PMC007xx controller provides two homing switches, input port EXT1 and EXT2, which can be used as an emergency stop or home point search function. The user can set any one or two enable.

When the emergency stop is enabled, if the corresponding input pin detects effective trigger edge, controller immediately lock the motor and stop response to any step command. User can read the status of controller, and check which one input pin trigger the emergency stop. The controller will continue to respond to the new step command, only after the user clears the corresponding status bit.

Object name	External emergency stop
SDO ID	0x600F
Object type	Record
Storage type	ROM
Number of parameters	2

Subindex 0x01：External emergency stop enable

Object type	U8,rw
Range	bit
Default value	0

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

bit0: External emergency stop 1 enable settings

bit1: External emergency stop 2 enable settings

bit4: External emergency stop 3 enable settings

Subindex 0x02: The trigger mode of external emergency stop

Object type	U8,rw
Range	bit
Default value	0

The 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:

bit0: The trigger mode of external emergency stop 1

bit1：The trigger mode of external emergency stop 2

bit4：The trigger mode of external emergency stop 3

Object name	EXT1/EXT2 stabilize delay (ms)
SDO ID	0x601a
Object type	Record
Range	0~200
Storage type	ROM
Default value	100

4.7 General IO port

The PMC007xx controller provides 7 general purpose IO (GPIO) ports, 2 external emergency stop input (EXT) ports and 2 encoder input (ENC) ports.

4.7.1 General IO port set

Object name	General IO port set
SDO ID	0x6011
Object type	Record
Storage type	ROM
Number of parameters	2

Subindex 0x01: the direction of IO port

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

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

Sub index 0x02: IO port configuration

Object type	U32,rw
Range	0-0x3fffff
Default value	0

Each port is configured with 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

4.7.2 General IO port value

Object name	General IO port value
SDO ID	0x6012
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 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

4.8 Offline programming

4.8.1 Offline programming parameter 1

Object name	Offline programming parameter 1
SDO ID	0x6018
Object type	Record
Storage type	ROM
Number of parameters	2

Subindex 0x01: Number of offline programming command

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

Subindex 0x02: Offline automatic operation enable

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

The values of Offline automatic operation are definite as follows:

0: Disable offline automatic operation

1: Enable offline automatic operation

4.8.2 Offline programming parameter 2

Object name	Offline programming parameter 2
SDO ID	0x6019
Object type	Record
Storage type	RAM
Number of parameters	5

Subindex 0x01: Off-line program pointer

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

Subindex 0x02: offline command

Object type	U32,rw
Range	-
Default value	-

For details about the definition of offline command, please refer to User-defined program section.

Subindex 0x03: Save offline command

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

If 1 is written into, all offline command will be saved.

Subindex 0x04: GPIO mask

Object type	U16,rw
Range	bit
Default value	0

Subindex 0x05: run command

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

If 1 is written into, run the command which the program pointer is pointing to.

4.9 Close Loop

PMC007xx supports 400-1600CPR incremental encoder for the close loop function by employing PID. Following are relative registers.

4.9.1 Encoder CPR

Object name	Encoder CPR
SDO ID	0x6021
Object type	U16, rw
Range	400, 500, 600, 800, 1000, 1200, 1600
Storage type	ROM
Default value	500

Note: the controller must be re-power after the CPR is changed.

4.9.2 KP

Object name	Encoder CPR
SDO ID	0x6023
Object type	U8, rw
Range	1-255
Storage type	ROM
Default value	48

This parameter affects the transient response.

4.9.3 KI

Object name	KI
SDO ID	0x6024
Object type	U8, rw
Range	1-255
Storage type	ROM
Default value	48

This parameter affects accumulative error.

4.9.4 KD

Object name	KD
SDO ID	0x6025
Object type	U8, rw
Range	1-255
Storage type	ROM
Default value	32

This parameter affects dynamic response.

4.9.5 Filter para 1

Object name	Filter 1
SDO ID	0x6026
Object type	U8, rw
Range	1-128
Storage type	ROM
Default value	8

This parameter affects system speed character, the bigger value is preferred in high speed scene, but the value should not be greater than microstepping value.

4.9.6 Filter para2

Object name	Filter 2
SDO ID	0x6027
Object type	U16, rw
Range	1-255
Storage type	ROM
Default value	8

This parameter affects recovery character after stall occurred.

4.9.7 Stall Length

Object name	Stall len
SDO ID	0x6028
Object type	U16, rw
Range	1-255
Storage type	ROM
Default value	64

4.9.8 Torque loop enable

Object name	Torque en
SDO ID	0x6029
Object type	U8, rw
Range	0-1
Storage type	ROM
Default value	1

The PID will be invalid if Torque loop en is disabled.

4.9.9 Power off auto save

Object name	Power off auto save
SDO ID	0x602A
Object type	U8,rw
Range	0-1
Storage type	ROM
Default value	0

If this parameter is enabled, the controller detect the system power off automatically, the current position will be written into EEPROM and restored after re-power on.

5 User-defined program

PMC007xx can be configured into offline mode. In this mode, controller automatically execute custom user code after powered on, the code is compiled and in advance burned to the EEPROM through CQPUSI tool software.

When the PMC007xx controller works in offline mode, the CAN communication interface is still responsive to the user's online instruction.

The maximum number of instructions that PMC007xx controller supports for user is 100.

Please refer to the "controller offline programming guide" for details about detailed example of user defined program.

5.1 User Instruction Set Summary

PMC007xx controller supports the following user defined instructions, these commands are provided by the CQPUSI tool software to interact with the controller automatically, and users do not need to write their own programs, only need to operate the command in the "custom programming interface".

Command	Function	Options	Data Range
ROT	Rotate a given number of steps	0	1~65536
MIC	Set microstepping	0	0/2/4/8/16/32/64/128/256
DIR	Set direction of rotation	0	0：backward,1：forward.
EXTEN	Set ext_stop enable	0	0/1
FREE	Set freerun enable	0	0/1
CLR2	clean ext_stop2 flag	0	----
CLR1	Clean ext_stop1 flag	0	----
VSET	Set speed(pps)	0	0-6000PPS
ACC	Set Acceleration	0	3-6
TRIG	Set external trigger mode	0	0-3

CNTI	Internal counter plus 1	0	----
CNTC	Internal counter reset	0	----
JMP	Unconditional jump	0	0-50
JNE	Unequal jump	0	0-50
JEQ	Equal jump	0	0-50
WAIT	Waiting condition	1-7	0/1
OUT	Port output	1-5	0/1
CMP	Comparison	1-6	0~65536
RESET_EN	Set GPI1 reset enable	0-1	0-50
PAUSE_EN	Set GPI2 pause enable	0-1	0

5.2 User command description

The detail information for some of command is described as following.

5.2.1 CNTI，CNTC command

These two instructions are used to add and reset the internal counter, and the internal counter can be used as a function of the cycle count in the user's custom program. The value of the counter can be used as a comparison condition in the CMP command.

5.2.2 JMP command

Unconditional jump instruction, the program jumps to the specified location.

5.2.3 JNE，JEQ command

Conditional jump instruction. Based on the flag which is generated by CMP instruction, jump to the specified position. If the flag bit is 1, the program will jump to the specified position by the JEQ instruction; if the flag is 0, the program will jump to the specified position by the JEQ instruction.

5.2.4 WAIT command

Pause the program execution. Execute the next instruction until the condition of the option is satisfied. A total of 9 options can be selected, please refer to the CQPUSI tool software for "custom programming" interface settings for details.

Note: when using the ROT command to launch the motor rotation, the next instruction is executed immediately and don't wait until the rotation command is completed. So generally there should be a WAIT instruction following the rotation command.

5.2.5 OUT command

Output value to GPO1~5. The instruction can only output a port value at a time.

5.2.6 CMP command

Compare the value of option with the setting value. Option can be the value of the internal counter, or any one input port or an external stop status or all input ports as a bus data comparison, a total of 9 options can be selected. After comparing the internal flag will be set, if the result of comparison is equal, the flag is set to 1, otherwise set to 0.

5.2.7 RESET_EN 和 PAUSE_EN command

The controller can choose to bind GPI1 as the external reset stop key input, GPI2 as external pause/start key input. These functions can only be enabled by the offline program and only need to be done once in the program to take effect globally, so the user should try to put these two statements in the beginning of the offline program. When the external suspension / launch function is enabled, the low level pulse on the GPI2 will alternately start or pause the execution of the offline program, but the rotation command which has been issued will not be stopped. When the external reset stop function is enabled, the low level pulse on the GPI1 will immediately stop all operation instructions, including the rotation command being executed, and put the program pointer to the set position.

6 Introduction to Debug Tool

Users can use the CQPUSI tool software Tool Debug to debug command, IO port setting detection, set control parameters of motor, custom programming.

6.1 Installation preparation

Tool software PUSICAN requires CAN adapter (USB2CAN or PCI2CAN) support. Currently, the tool software has supported a wide variety of common USB2CAN adapter on the market. If you need to support other types of adapter, please contact with sales staffs.

6.2 Software installation

6.2.1 Driver installation

Install the driver of adapter, please follow the instructions on the user manual of adapter.

6.2.1.1 Tool software installation

Debug tool PUSICAN is green free installation software. After download, extract it into a special folder. Double-click the pusican.exe, then the software is running.

6.3 Software instructions

6.3.1 Preparation for using

Connect the PMC007xx and CAN adapter to the computer by way shown in Figure 3-3. Then power PMC007xx up. After power up normally, the LED lights will be flashing at 2.5Hz frequency.

6.3.2 Main interface

Double click on the desktop PUSICAN shortcut icon to enter the main interface. As shown in Figure 6-1:

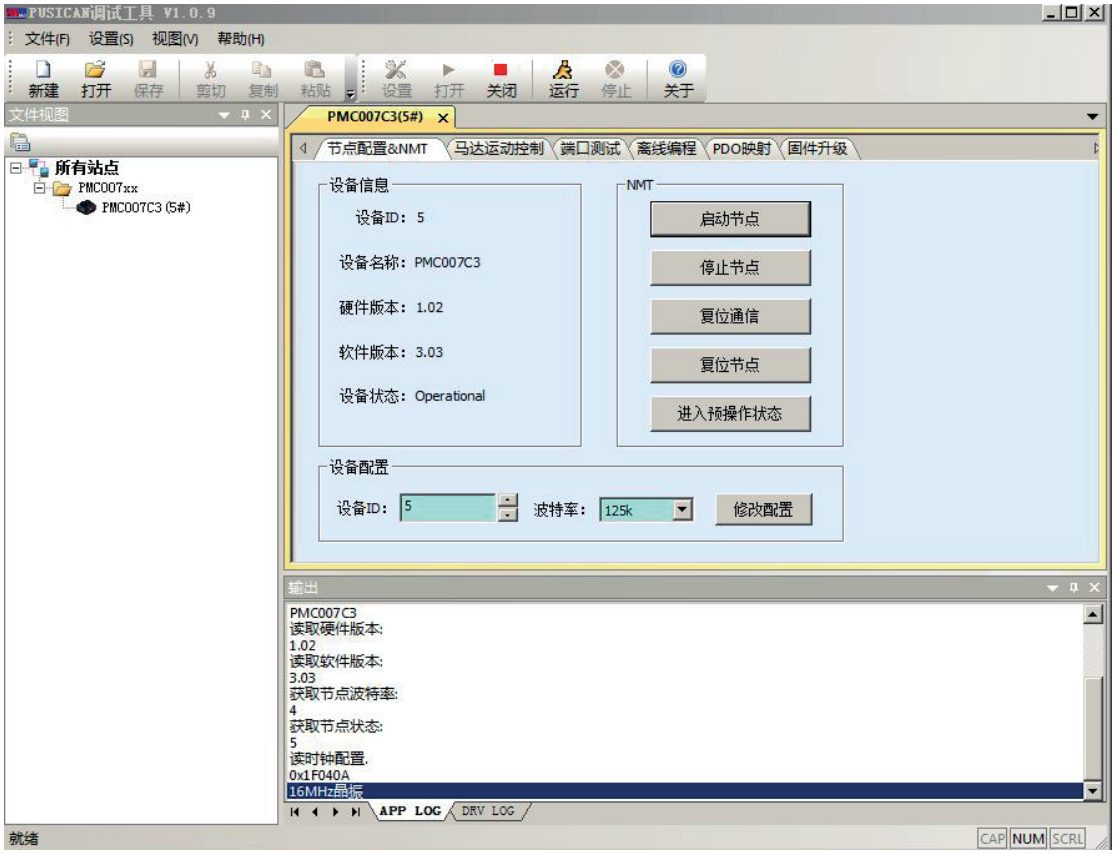


Figure 6-1

In the main interface, click "Settings" icon to select the adapter and baud rate. PMC007xx controller factory default baud rate is 125kHz. Click "open" icon, debugging tools will boot the adapter scanning site, and all online activity equipment are listed in the left side tree list. Double-click on operation site and the right workspace will display the control interface of the device.

Start node: Make PMC007xx into the operational state;

Stop node: Make PMC007xx into stopped state. The node will not respond to any SDO command;

Reset communication: The communication parameters take effect immediately by this operation, after SDO modifies the communication parameters;

Reset node: Notify the node reenter power on reset process;

Pre operation state: In this state, the node waits for the network command of the main station, receives the configuration request of the main station, so it can receive and send all messages except PDO;

6.3.3 Motor motion control interface

In the main interface, click "motor drive settings" into the motor motion control interface. As shown in Figure 6-2:

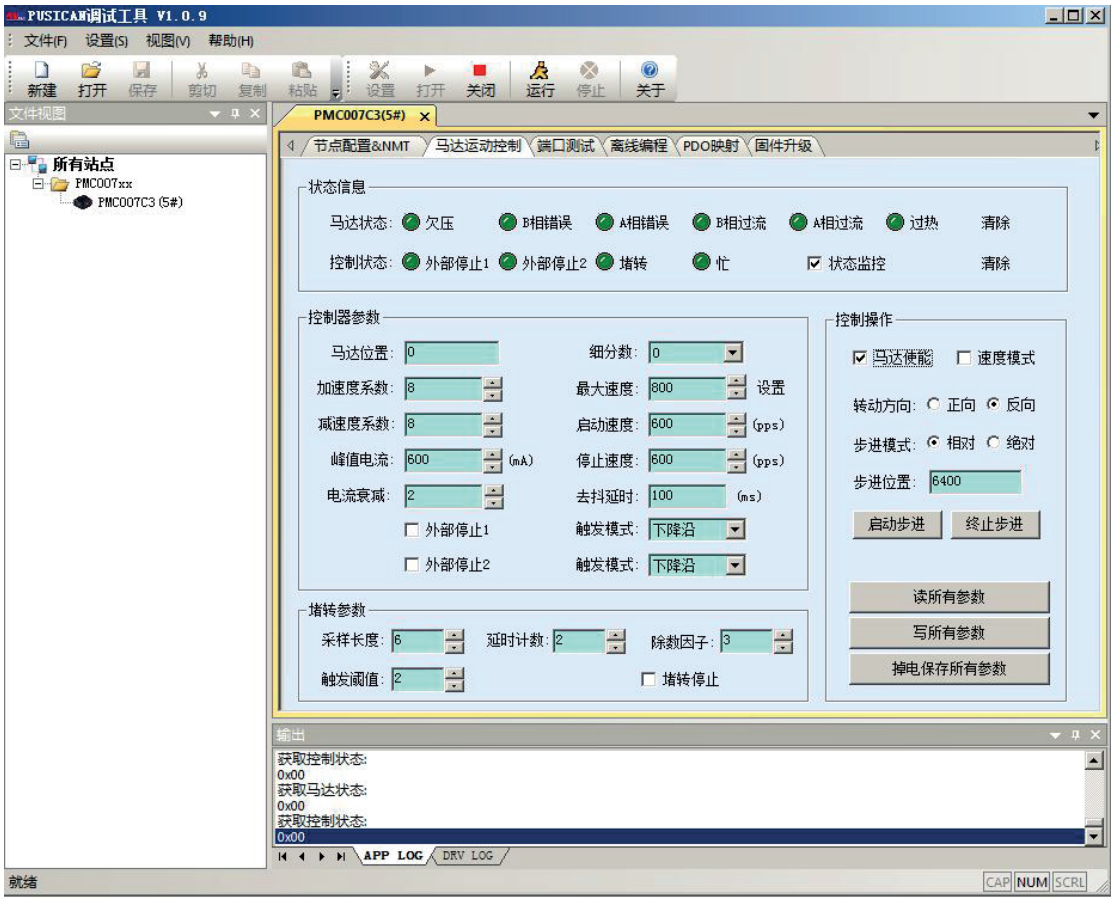


Figure 6-2

Before modifying the parameters, it is recommended to first click on "read all the parameters", update the current display value from the node. After set the control parameters, click "write all parameters" to write the setting parameters to the controller and make it effect. Click "power down to save all parameters" to write the parameter to the FLASH of the device in order to save permanently.

When an error occurs in the motor status, it must be cleared before motor can be started again.

6.3.4 Offline programming interface

In the main interface, click on "offline programming" into the Offline programming interface, As shown in Figure 6-3.

After entering the interface, the software will send command to close offline automatic operation, then read all offline programming command of the device, and display them on the interface.

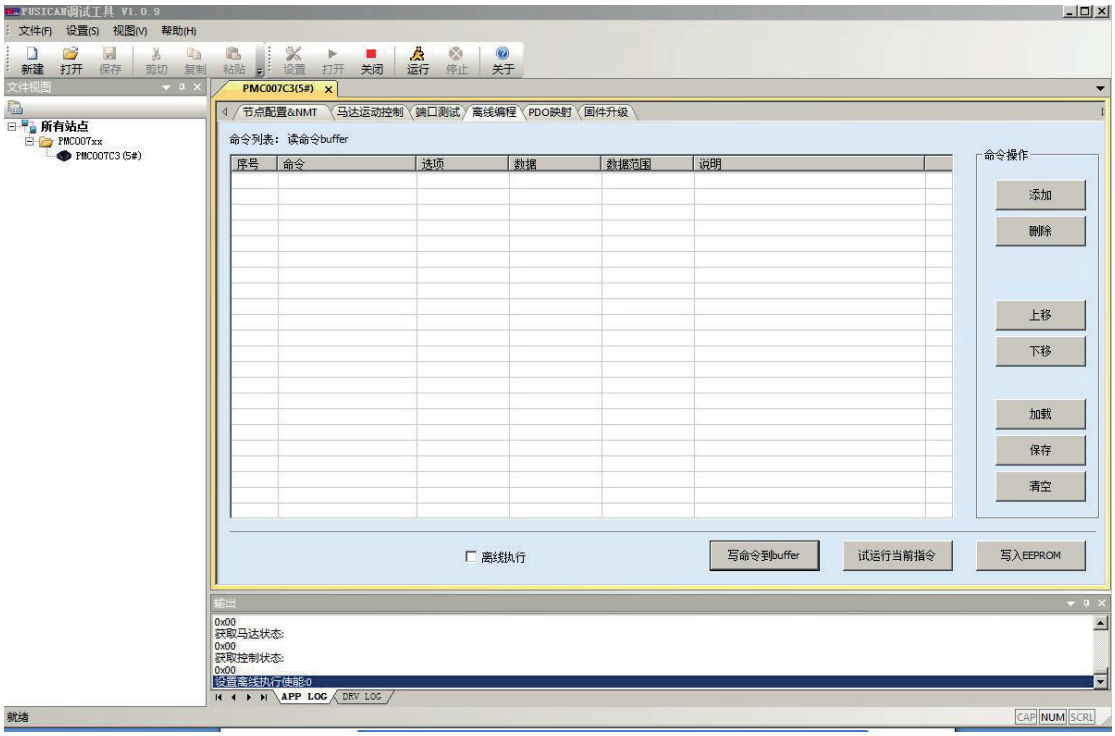


Figure 6-3

Users can complete the edit of offline command through the "add", "delete", "move", "down" button on the right according to the actual demand. Commands can be saved as a local disk file, and the offline command file on the local disk can also be loaded into interface to display.

Once editing is completed when users need to do online debugging, first press the "write cmd to the buffer" button to download the program to on-chip memory of PMC007xx Controller, and then press the "try run current command" button, and then device will run the current selected command. After confirmation, press the "Write to Flash" button to burn all program to non-volatile Flash memory.

If you select the "offline exec", PMC006xx controller will automatically run the offline program. Controller will read offline program from Flash and automatically run next time power is on.

6.3.5 PDO mapping

In the main interface, click "PDO map" into this interface. after entering the interface, the software will automatically read the current mapping object from the device and display them in the interface, as shown in Figure 6-.

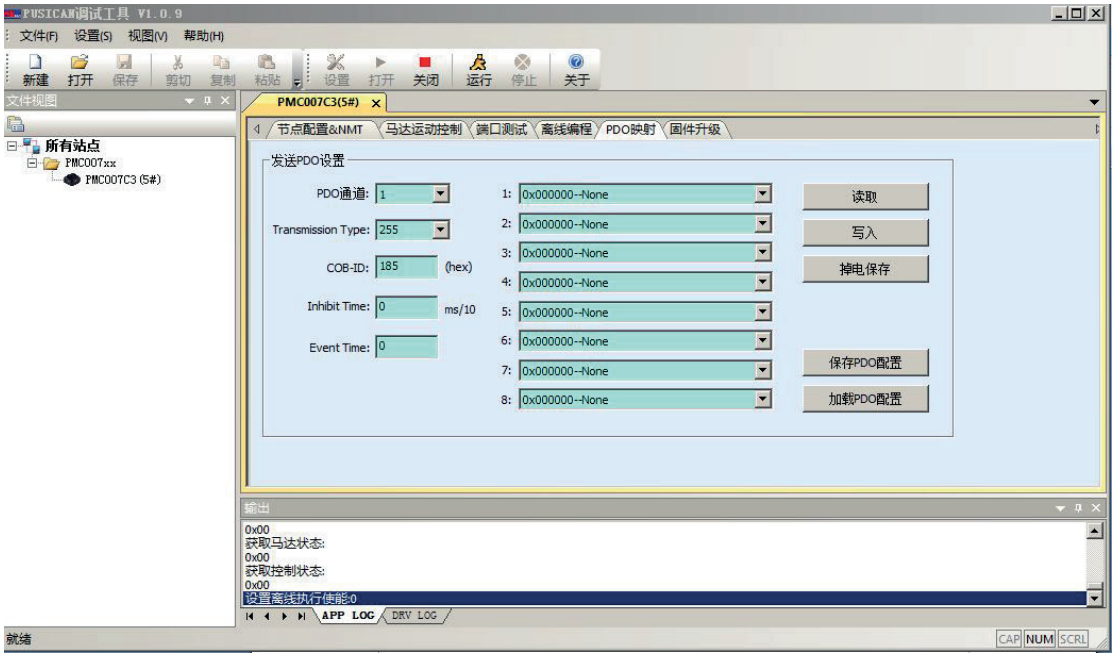


Figure 6-5

PMC007xx supports sending and receiving PDO of 4 channels, and each channel can map up to 8 objects.

For more details about related parameters configuration, please refer to 2.1 chapter of Appendix 2.

6.3.6 Firmware upgrade

Firmware upgrade of PMC007xx can be carried out in bootloader mode through the CAN bus. In "firmware upgrade" interface, click "into the bootloader/ application", PMC007xx will enter bootloader mode. The node ID and baud rate are set in the application model.

After entering the bootloader mode, LED lights will be double flash. In the "application path" column select the upgrade file, click the "upgrade" button to start upgrading, as shown in figure 6-7. After firmware upgrade is completed, tool will prompt a dialog box, then you can click on "into the bootloader/ application" (or repower on the controller), and then the controller will switch to the normal application mode.

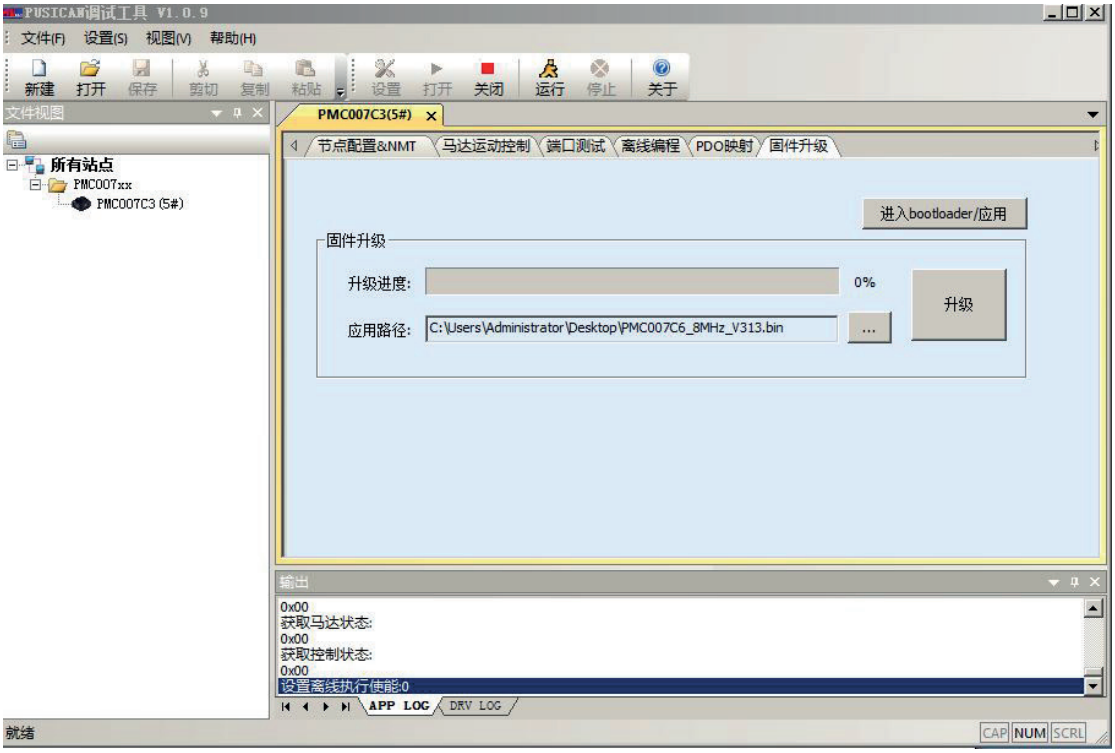


Figure 6-7

6.3.7 Scripting language support

PUSICAN debugging tool software supports LUA scripting language, and has built SDO CANOPEN operating function which user can call directly in the script program. User can create or open a script file through click the "new" and "open" icon on the top left. Once the script was written completely, you can control program execution through click the "run", "stop" icon on the top right. As shown in figure 6-8.

The syntax of the LUA scripting language is similar to C language. In application scenarios without special requirements of UI interface, the user can complete the complex control loop task with the powerful function of LUA script easily, and no need to develop the CANOPEN master control program in the host computer.

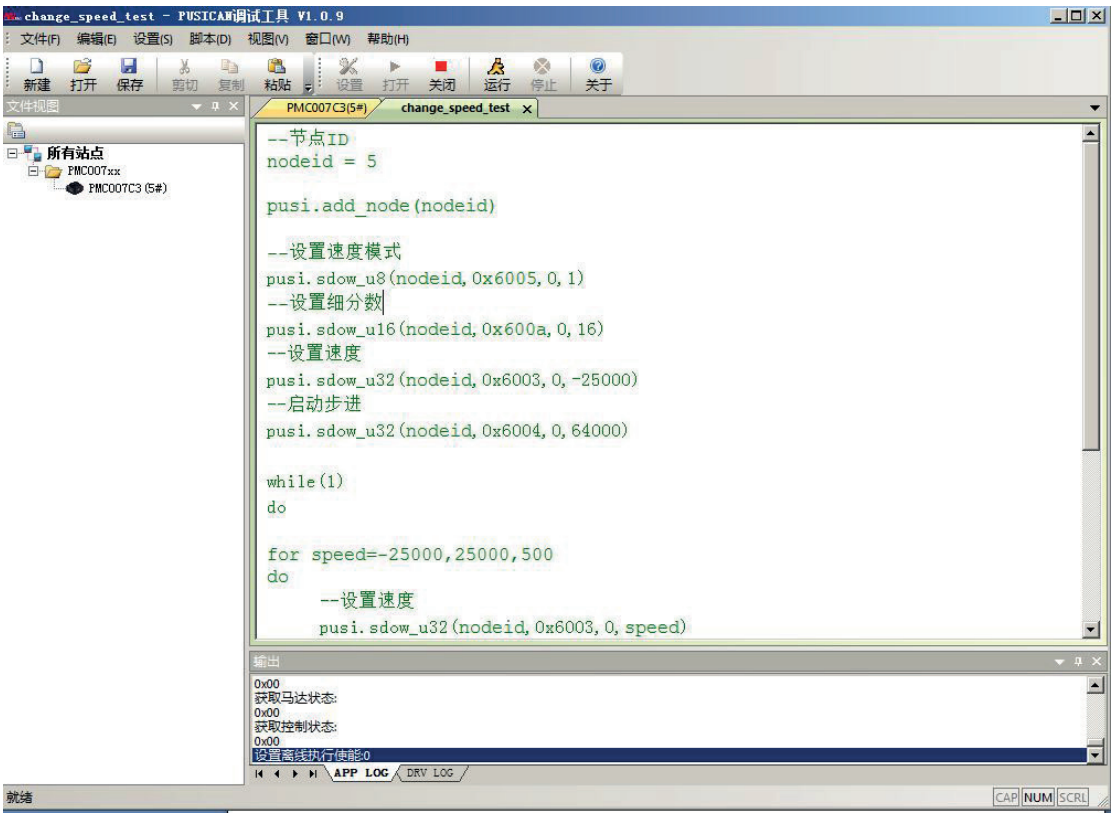
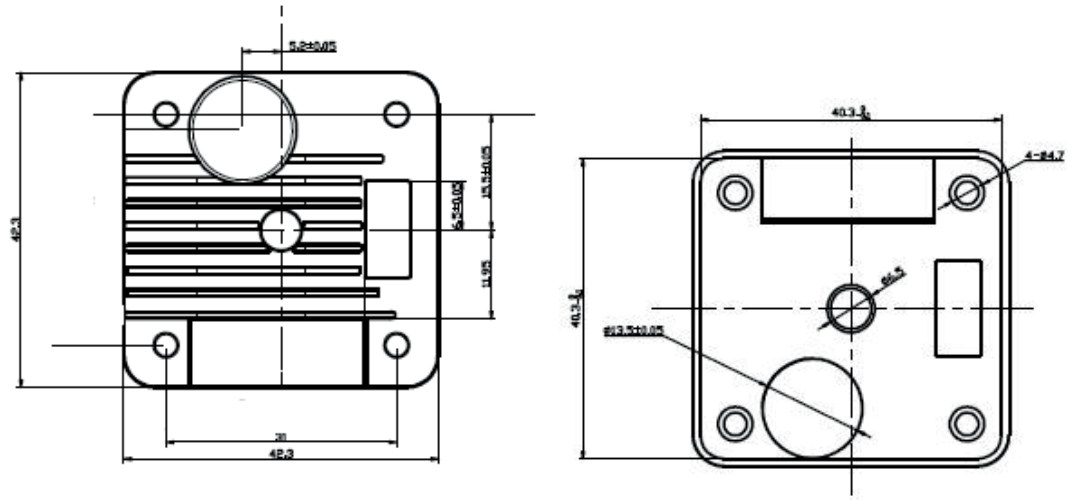


Figure 6-8

7 Electrical Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Supply Power Voltage	Normal 25℃	12		48	V
Operation Temperature	12V DC	-20		55	℃
IO maximum current	source/sink current	0		20	mA
Output current	Normal 25℃	0		6	A
IO low Voltage	12V DC	-0.5		1.0	V
IO High Voltage	12V DC	3.0		5.5	V

8 Dimensions



9 Appendix 1 PMC007xx Object dictionary table

Index	Subindex	Object 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	UINT32	RW	NO	

			to be mapped				
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
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	
	2h		Trigger mode of external emergency stop	UINT8	RW	NO	
6010h	--	VAR		UINT8	RW	NO	ROM
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
6013h	--	RECORD	OCP parameter(ROM)	--	--	--	
	0h		The number of OCP parameters	UINT8	RO	NO	
	1h		OCT1	UINT8	RW	NO	
	2h		OCD1	UINT8	RW	NO	
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	UINT8	RW	NO	

			automatically				
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
6020h	--	VAR	Termination step	UINT8	RW	NO	RAM
6021h	--	VAR	Encoder CPR	UINT16	RW	NO	ROM
6022h	--	VAR	Position save value	INT32	RO	NO	ROM
6023h	--	VAR	Close loop KP	UINT8	RW	NO	ROM
6024h	--	VAR	Close loop KI	UINT8	RW	NO	ROM
6025h	--	VAR	Close loop KD	UINT8	RW	NO	ROM
6026h	--	VAR	Close loop filter1	INT8	RW	NO	ROM
6027h	--	VAR	Close loop filter2	INT16	RW	NO	ROM
6028h	--	VAR	Close loop stall len	INT16	RW	NO	ROM
6029h	--	VAR	Close loop torque en	UINT8	RW	NO	ROM
602Ah	--	VAR	Power off auto save	UINT8	RW	NO	ROM

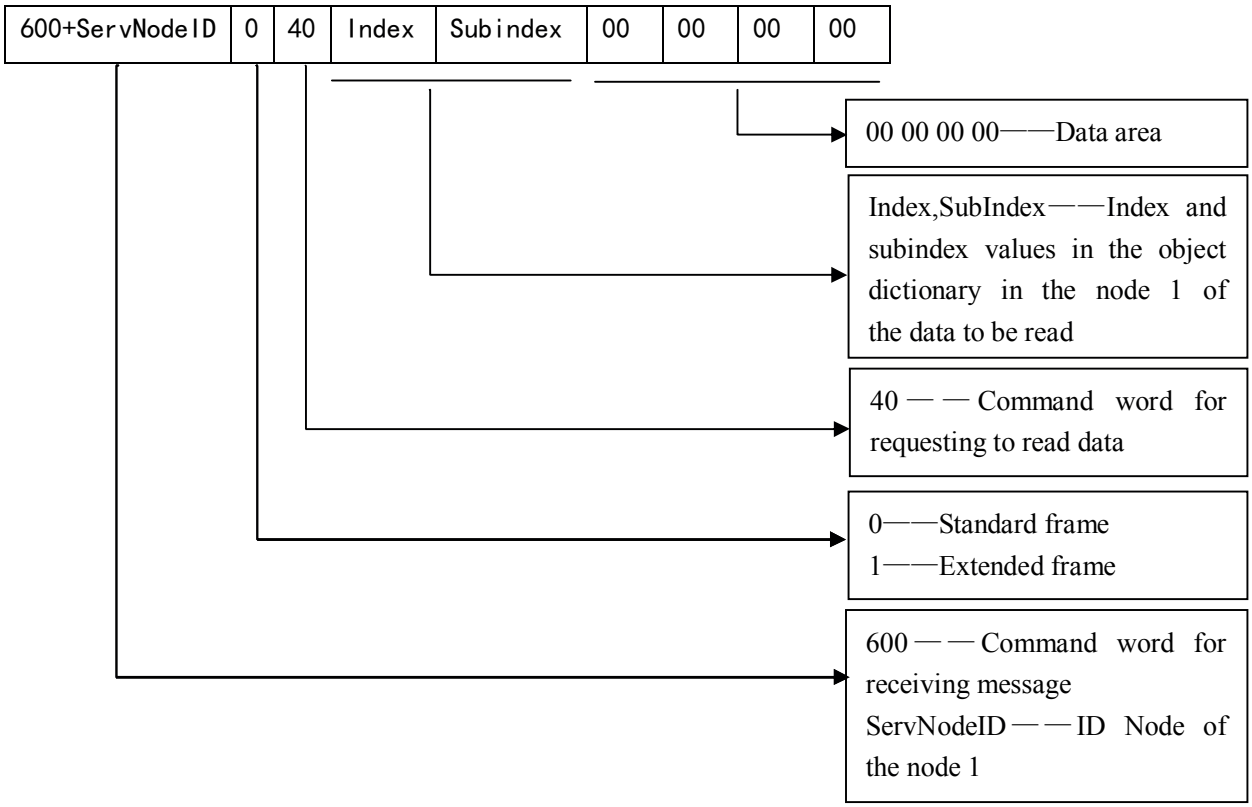
10 Appendix 2 CANOPEN Communication example

10.1 SDO Reading and writing example

10.1.1 SDO Read

10.1.1.1 Data frame format

Master send:



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

10.1.1.2 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.

10.1.2 SDO Write in

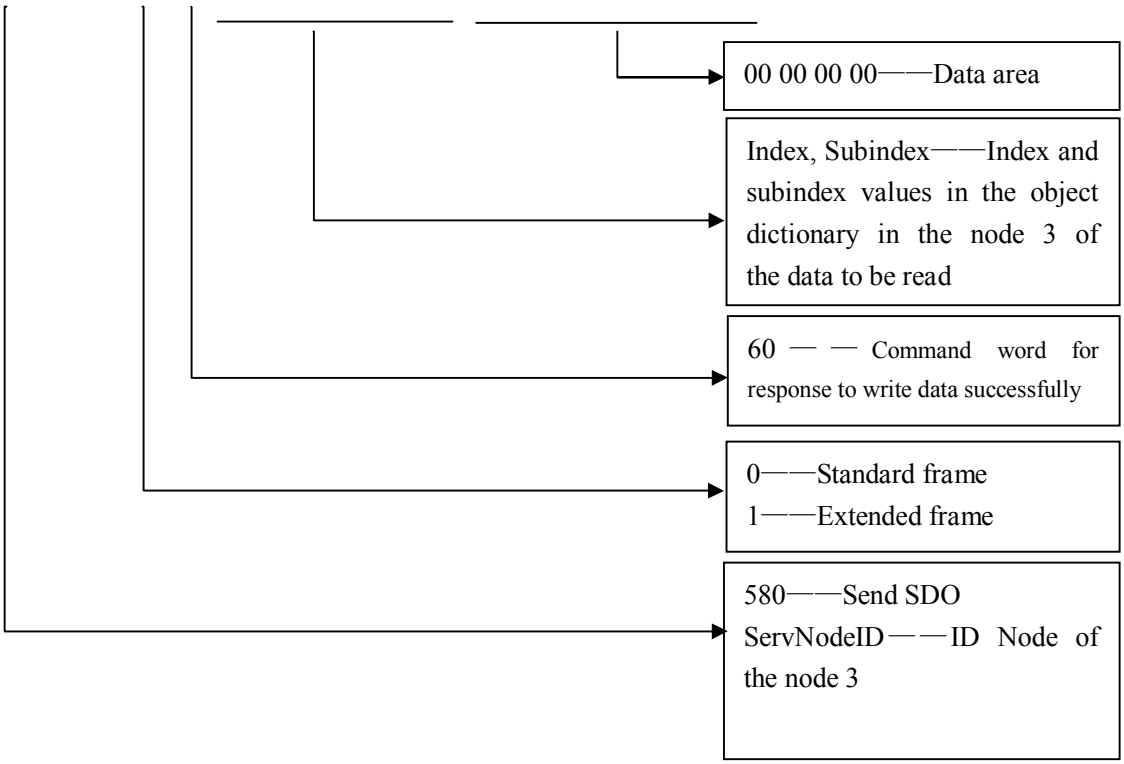
10.1.2.1 Data frame format

Master send:

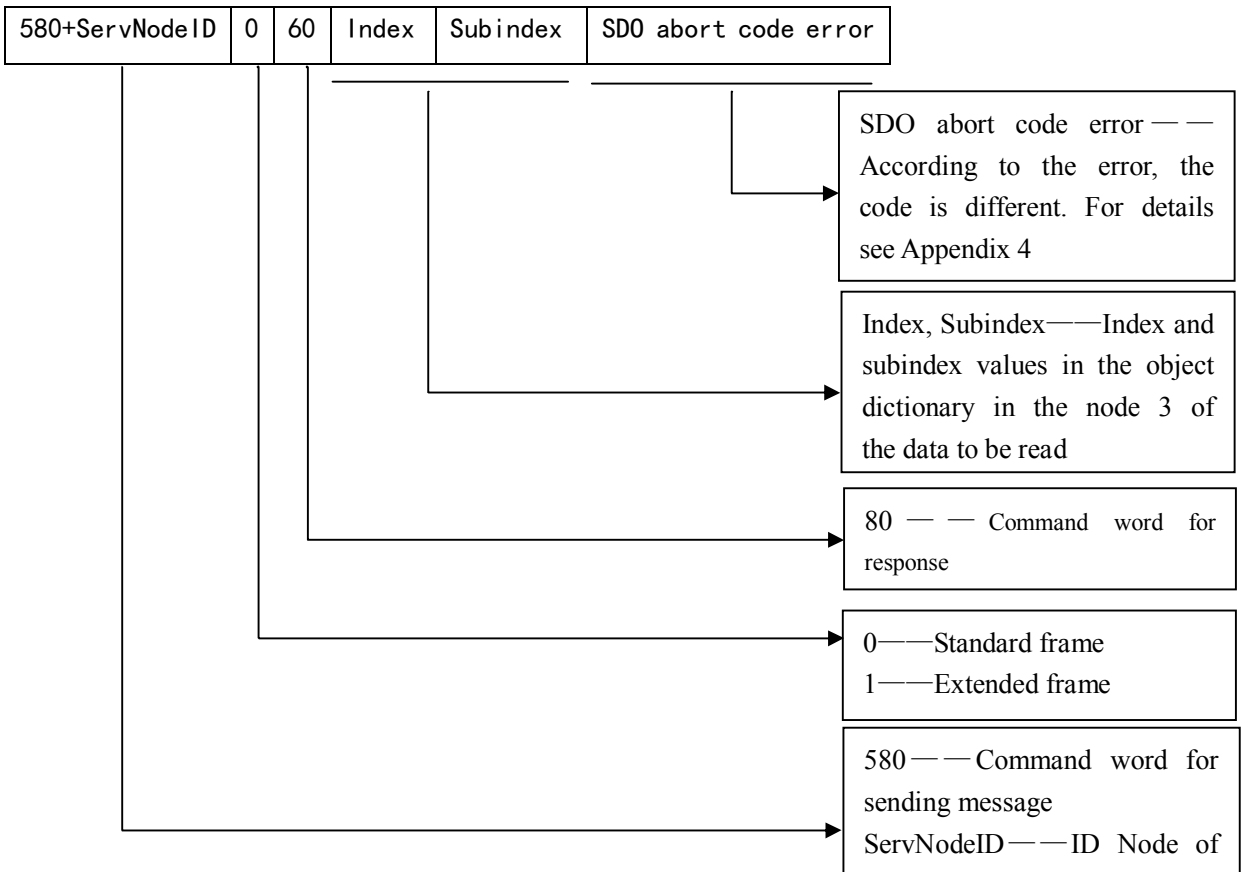
When the data length is 1 byte								
600+ServNodeID	0	2F	Index	Subindex	d0	0	0	0
When the data length is 2 bytes								
600+ServNodeID	0	2B	Index	Subindex	d0	d1	0	0
When the data length is 3 bytes								
600+ServNodeID	0	27	Index	Subindex	d0	d1	d2	0
When the data length is 4 bytes								
600+ServNodeID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

Correct response from the slave station:

580+ServNodeID	0	60	Index	Subindex	00	00	00	00



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.

10.1.2.2 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.

11 Appendix 3 PDO configuration example

11.1 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.

11.1.1 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.

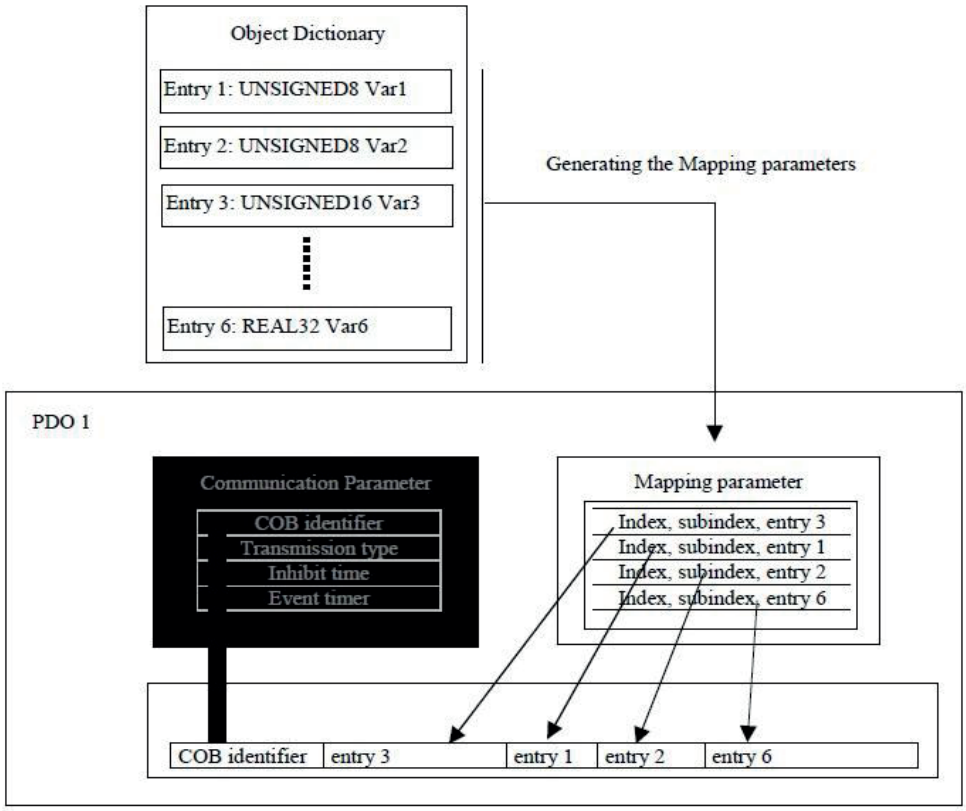


Figure 3: Mapping of Object Dictionary entries into a PDO

11.1.2 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.

11.1.2.1 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	000000000000000000	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

11.1.3 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.

11.2 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 communication COB-ID to be 187, that is, the device whose node ID is 7 receives the PDO

Master send: 605 23 00 18 01 87 01 00 00

Event trigger is set

Master send: 605 2F 00 18 02 FF 00 00 00

Set Inhibit time as 5ms

Master send: 605 2B 00 18 03 32 00 00 00

Set Event time as 1000ms

Master send:605 2B 00 18 05 E8 03 00 00

Set the number of map entries to be 1

Master send:605 2F 00 1A 00 01 00 00 00

Set the mapping parameters to map 0x6012 to TPDO1

Master send:605 23 00 1A 01 10 00 12 60

After 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.

12 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

www.purelogic.ru

8 800 555-63-74 бесплатные звонки по РФ

Контакты

+7 (495) 505-63-74 - Москва
+7 (473) 204-51-56 - Воронеж
+7 (812) 425-17-35 - Санкт-Петербург
purelogic.ru

394033, Россия, г. Воронеж,
Ленинский пр-т, 160, офис 149

Пн-Чт: 8.00–17:00
Пт: 8.00–16.00
Перерыв: 12.30–13.30

info@purelogic.ru