Thank you for using AC70T series crane ac drive and AC200CANPG01 fieldbus communication card from Suzhou Veichi Electric Equipment Technology Co. The communication card integrates the CAN card and PG card functions.

This CAN card is used for AC70T series lifting special ac drive, which supports CANopen protocol. AC70T ac drive can access the high-speed CAN communication network through this expansion card to realize the control of field bus. Please read this manual carefully before using this product.

# 1 Hardware Configuration

This communication card is specially configured for our AC70T series machines. The CAN bus interface is fully compliant with ISO/DIS11898 standard to realize CAN communication between multiple ac drives.

AC200CANPG01 card terminal port using terminal wiring

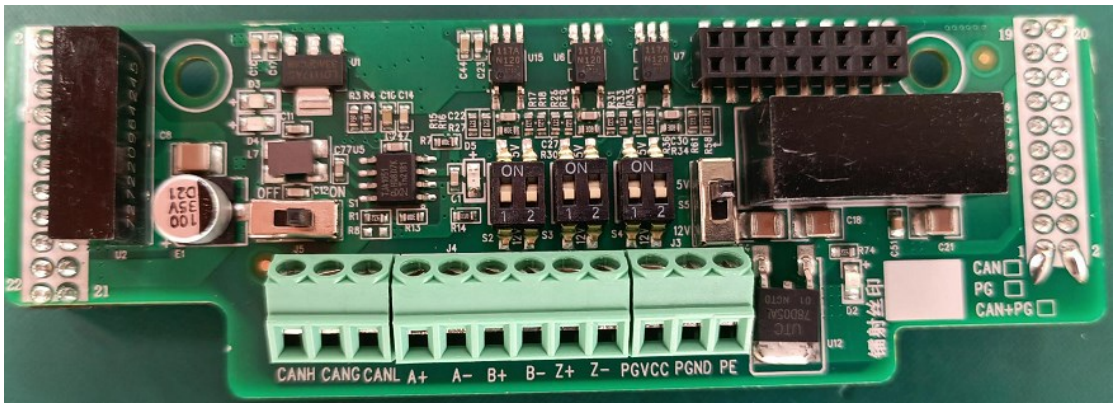



Figure 1 AC200CANPG01 front schematic

## 1.1 CAN card terminal wiring

The pinout diagram and menu are as follows:

| Name | Function |
|---|---|
| PE | Cable shield ground terminal |
| CANH | Connecting the CAN bus positive polarity port |
| CANL | Connecting the CAN bus negative polarity port |
| CANG | Connection to CAN bus signal reference ground |

## 1.2 Baud rate and transmission distance

| Communication baud rate | Communication length |
|---|---|
| 1M bit/s | 25m |

| | |
|---|---|
| 500k bit/s | 100m |
| 250k bit/s | 250m |
| 125k bit/s(Default) | 500m |
| 50k bit/s | 1000m |
| 20k bit/s | 2500m |

# 1.3 Configure dipswitches

For field use, AC200CANPG01 cards are equipped with terminal matching resistors that can be set for use via dipswitch S1. It is recommended to use termination resistors at both ends of the network topology.

The setting instructions for dipswitch S1 are as follows:

| **S1** | **Dial code position** | **Terminal resistance** |
|---|---|---|
|  | Toggle to the right end for ON | Terminal resistance access |
| | Toggle to the left end for OFF | No termination resistor used (factory default) |

# 1.4 Indicator light description

The CAN card is equipped with four LEDs to monitor its operating status, which are defined as described in Table 1.4

Table 1.4 Description of status monitoring LED indicators

| Indicator light | Status | Description | Remarks |
|---|---|---|---|
| D5（red） | ON | CAN 5V power up is normal | Power on, CAN communication 5V power supply is normal |
| | OFF | CAN 5V power up abnormal, please check if the installation is normal | Power failure, CAN communication 5V power supply is not normal |
| D2（red） | ON | PG 5V power on is normal | Power on, encoder PG 5V power supply is normal |
| | OFF | PG 5V power on abnormal, please check if the installation is normal | Power failure, encoder PG 5V power supply is not normal |
| D4（red） | ON | Internal ac drive communication timeout | Frequency converter communication timeout |
| | Fast flashing | CANopen address setting error | Address configuration is 0 |
| | Flash twice | CANopen emergency message ac drive failure | AC drive report fault |
| D3 | ON | Operation | The NMT starts the remote node and |

| | | | |
|---|---|---|---|
| （green） | | | the TPDO uploads data in a communicable state. |
| | Flash | Pre-Run | CANOPEN card is in pre-operable state. |
| | OFF | Stop | NMT stops the remote node and it is not communicable. |

## 1.5 Cautions

a. All slave stations should be connected in series, not in a star connection.

b. The master side and the last node of the slave side need to be connected to a 120 Ω termination resistor. The AC200CANPG01 communication card comes with a termination resistor, and both can be enabled by dip switch S1 to prevent signal reflection when used at the terminal.

c. To avoid interference, the CAN connection line should preferably be a shielded twisted pair, and the shield layer should generally be grounded reliably using a single point.

d. The longer the connection line, the higher the requirement for the driving capability of the CAN chip.

# 2 Software Configuration

## 2.1 Agreement Description

### 2.1.1 Software Features

The AC200CANPG01 card supports the following protocols:

support for the Heartbeat protocol, with the slave station reporting the current status to the master station at regular intervals;

Support for SDO accelerated transmission mechanism;

support for 4 TPDO, 4 RPDO;

Support emergency objects；

### 2.1.2 Communication object COB-ID

CANopen provides a variety of communication objects, each with different characteristics. This card uses a predefined COB-ID, which is planned as follows:

1）NMT object：0x000

2）SYNC object：0x080

3）SDO object：

Send SDO：0x600+NodeID

Receive SDO：0x580+NodeID

4）PDO object：

RPDO1：0x200+NodeID

RPDO2：0x300+NodeID

RPDO3：0x400+NodeID

RPDO4：0x500+NodeID

TPDO1：0x180+NodeID

TPDO2：0x280+NodeID

TPDO3：0x380+NodeID

TPDO4：0x480+NodeID

5）EMCY object：0x80+NodeID

where NodeID is the device ID (station address), set by the function code parameter.

### 2.1.3 AC drive parameter operation

1）Baud rate

The AC70T ac drive sets the baud rate through the ten bits of function code Fd.02, which needs to be re-powered after changing. The setting values are listed in the following table:

| Fd.02 10 bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| CAN Baud rate | 20K | 50K | 100K | 125K | 250K | 500K | 1M |

2）Device Address

The AC70T sets the device address (node station number) via Fd.12, and needs to be re-powered after changing this parameter.

3）Mapping Instructions

The ac drive function codes are mapped to the 2000h group index of CANopen. 2000h group index and function codes directly satisfy the following relationship:

① Index value = 2000h + function code group number;

② Sub-index value=function code parameter number+1.

AC drive function codes include group F, group C (monitoring parameters) and group T (communication control parameters)

**F Group（FX. YZ）:**

X is the function code group number, expressed in hexadecimal, i.e. X=0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F;

YZ is the function code parameter number, expressed in decimal, which needs to be converted to hexadecimal when calculating the subindex.

Take FA.25 function code as an example, its function code group number is A, function code parameter number is 25, index value is 2000h+Ah=200Ah, sub index value is 19h+1=1Ah.

The correspondence between group F and the index is shown in the following table:

| **Function code sets** | **CANopen Index** |
|---|---|
| F0-FF | 0x2000-0x200F |

In addition, if a power-down save write operation is performed for the group F parameters, the corresponding index values are shown in the following table:

| Function code sets | CANopen index value |
|---|---|
| F0-FF | 0x20F0-0x20FF |

**C Group（C-XY）:**

Group C is the monitoring parameters, which can be viewed on the panel by pressing and holding "PRG" for more than 2 seconds, the range is C-00~C-39, please refer to the monitoring parameters in the AC70T manual (Section 8) for details.

Group C is the monitoring function code group number, fixed at 21h, so the index value is 2000h+21h=2021h;

XY is the function code parameter number, expressed in decimal, which needs to be converted to hexadecimal when calculating the subindex.

Take C-02 as an example, the index value is 2000h+21h=2021h, and the subindex value is 02h+1=03h

**T Group（T-XY）:**

T group is the communication control function code group number, fixed at 30h, so the index value is 2000h + 30h = 2030h;

XY is the function code parameter number, expressed in decimal, which needs to be converted to hexadecimal when calculating the sub-index.

The communication control group parameters are as follows:

| Parameter Code | Parameter Description | Data range and description | R/W Characteristics R: Read-only W: Writable |
|---|---|---|---|
| T-00 | Communication given frequency | 0～32000 to 0.00Hz～320.00Hz | W/R |
| T-01 | Communication command setting | 0000H：No command<br>0001H：Forward running<br>0002H：Reverse running<br>0003H：Forward rotation pointing<br>0004H：reverse rotation pointing<br>0005H：Deceleration stop<br>0006H：Free stop<br>0007H：Fault reset<br>0008H：Run prohibit command<br>0009H：Run allow command | W/R |
| T-02 | AC drive status | bit0:0-stop, 1-run<br>bit1:0-Non-acceleration, 1-acceleration<br>bit2:0-Non-deceleration,<br>1-Deceleration | R |

| | | | |
|---|---|---|---|
| | | bit3:0-Forward, 1-Reverse<br>bit4:0-Normal, 1-Faulty<br>bit5:0-unlocked state, 1-locked state<br>bit6:0-No warning, 1-Warning<br>bit7:Reserved<br>bit8:Brake failure protection status<br>0-not protected, 1-protection in progress<br>bit9:Brake torque detection status<br>0-Not detected, 1-Detecting<br>bit10:Motor self-learning status<br>0-Not self-learning, 1-Self-learning in progress | |
| T-16 | Fault and warning codes | 1-63 are fault codes, others are warning codes<br>See the fault section in the AC70T manual (section 9) for details | R |

## 2.1.4 SDO read/write ac drive parameter operation

### 1 SDO Read/Write Operation Protocol Explanation

SDO Read operation

The master uses the CANopen Service Data Object (SDO) to read the ac drive, and the master sends the data in the format shown in the following table:

| CAN | CANopen data | Description |
|---|---|---|
| ID | 0x600+NodeID | NodeID device address |
| RTR | 0 | - |
| Data0 | Command code (0x40) | 0x40 indicates read command |
| Data1 | Index low byte | - |
| Data2 | Index high byte | - |
| Data3 | Subindex | - |
| Data4 | Data1 | Reserved (0) |
| Data5 | Data2 | Reserved (0) |
| Data6 | Data3 | Reserved (0) |
| Data7 | Data4 | Reserved (0) |

Note: In the read operation, data 1,2,3,4 are recommended to be set to 0, but set to other values will not affect the read result.

The slave response data is shown in the following table:

| CAN | CANopen data | Description |
|---|---|---|
| ID | 0x580+NodeID | NodeID device address |
| RTR | 0 | - |

| | | |
|---|---|---|
| Data0 | Command Code Return | Correct "0x43/4b/4f"; error "0x80" |
| Data1 | Index Low Byte | - |
| Data2 | Index High Byte | - |
| Data3 | Subindex | - |
| Data4 | Data1 | The data is read back when correct, and the SDO error code when incorrect<br>Data 1 - Data 4 constitutes a 32-bit data, where Data 4 is the high eight bits (bit 31 - bit 24), Data 3 is bit 23 - bit 16, Data 2 is bit 15 - bit 8, and Data 1 is the low eight bits (bit 7 - bit 0) |
| Data5 | Data2 | |
| Data6 | Data3 | |
| Data7 | Data4 | |

SDO Write operation

The master uses the CANopen Service Data Object (SDO) to write to the ac drive. The master sends data in the format shown in the following table:

| **CAN** | **CAN open Data** | **Description** |
|---|---|---|
| ID | 0x600+NodeID | NodeID Device Address |
| RTR | 0 | - |
| Data0 | Command Code | "0x23/2b/2f" is to write "32/16/8" bits of data respectively |
| Data1 | Index Low Byte | - |
| Data2 | Index High Byte | - |
| Data3 | Subindex | - |
| Data4 | Data1 | Data to be written<br>Data 1-Data 4 constitutes a 32-bit data, where Data 4 is the high eight bits (bit31-bit24), Data 3 is bit23-bit16, Data 2 is bit15-bit8, and Data 1 is the low eight bits (bit7-bit0) |
| Data5 | Data2 | |
| Data6 | Data3 | |
| Data7 | Data4 | |

**Description**：AC drive parameter codes are 16-bit data, so the Data0 command code needs to be set to "2b", otherwise it will report error 0x05040001, see SDO error code description for details.

AC drive Response

| **CAN** | **CANopen Data** | **Description** |
|---|---|---|
| ID | 0x580+NodeID | NodeID Device Address |
| RTR | 0 | - |
| Data0 | Command Code Return | Correct "0x60"; Error "0x80" |
| Data1 | Index Low Byte | - |
| Data2 | Index high byte | - |
| Data3 | Subindex | - |
| Data4 | Data1 | It is 0 when correct and SDO error code when wrong.<br>The SDO error code is a 32-bit data, which consists of data 1 - data 4. Where data 4 is the high eight bits (bit31-bit24), data 3 is bit23-bit16, data 2 is |
| Data5 | Data2 | |
| Data6 | Data3 | |
| Data7 | Data4 | |

| | | |
|---|---|---|
| | | bit15-bit8, and data 1 is the low eight bits (bit7-bit0) |

SDO Error code

| **SDO Error Code** | **Error Description** |
|---|---|
| 0x05040000 | Protocol timeout |
| 0x05040001 | Invalid command for SDO messages |
| 0x06010002 | Write operation on read-only object dictionary |
| 0x06020000 | Object dictionary not found |
| 0x06040041 | Objects cannot be mapped to PDO |
| 0x06040042 | The mapping length of PDO is out of range |
| 0x06070010 | Data type mismatch |
| 0x06090011 | Subindex does not exist |
| 0x06090030 | Write data outside the scope of the object dictionary |
| 0x08000000 | General error |
| 0x08000020 | Data cannot be changed in the current state |

The following is an example of reading and writing function code for ac drive with ac drive node address (Fd.12) as 2.

## 2 Example of reading group F function code

Reading the value of F2.02 with index 2002h and subindex 03h, the master sends the following message:

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x602 | 0 | 40 02 20 03 00 00 00 00 |

The ac drive response message is as follows:

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x582 | 0 | 4b 02 20 03 00 00 00 00 |

## 3 Example of writing group F parameters

Writing value 3 to F2.02 with index 2002h and subindex 03h, the master sends the following message.

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x602 | 0 | 2b 02 20 03 03 00 00 00 |

The ac drive response message is as follows:

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x582 | 0 | 60 02 20 03 00 00 00 00 |

## 4 Example of reading group C parameters

Read C-26 (ac drive rated voltage) with index 0x2021 and subindex 0x1B.

Then the master sends a message as follows

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x602 | 0 | 40 21 20 1B 00 00 00 00 |

The response message of the ac drive is as follows：

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|

| | | |
|---|---|---|
| 0x582 | 0 | 4b 21 20 1B DC 00 00 00 |

### 5 Example of reading and writing T group parameters

When reading T-00, the index is 2030h and the subindex 01h

then the master sends a message as follows:

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x602 | 0 | 40 30 20 01 00 00 00 00 |

The ac drive response message is as follows:

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x582 | 0 | 4b 30 20 01 00 00 00 00 |

Writing 100 to T-00 with index 2030h and subindex 01h, the master sends the message as follows:

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x602 | 0 | 2b 30 20 01 64 00 00 00 |

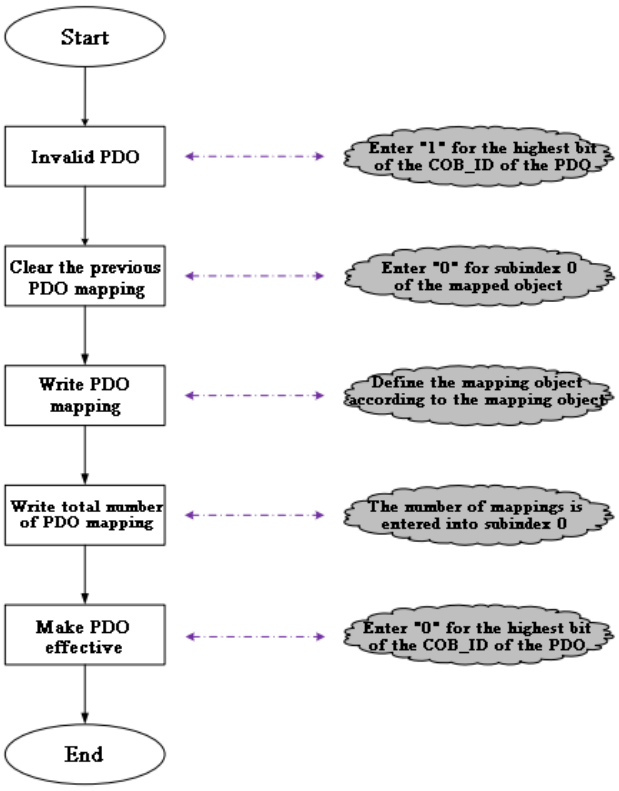
The ac drive response message is as follows:

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x582 | 0 | 60 30 20 01 00 00 00 00 |

## 2.1.5 PDO ac drive operation

The PDO is a process data service object that communicates periodically between the master and the slave via PDO messages to ensure real-time data interaction.

### 1 Mapping process of PDO



## 2 PDO Object

PDO can be divided into Receive PDO (RPDO) and Transmit PDO (TPDO). PDO is defined by both communication parameters and mapping parameters to determine the transmission The PDO is determined by both communication parameters and mapping parameters to determine the transmission mode and content. This ac drive is designed with 4 RPDOs and 4 TPDO to realize the data transmission of PDO, and the list of related objects is Shown as follows:

| NAME | | COB_ID | communication object | Mapped objects |
|---|---|---|---|---|
| RPDO | RPDO1 | 200h + NodeID | 1400h | 1600h |
| | RPDO2 | 300h + NodeID | 1401h | 1601h |
| | RPDO3 | 400h + NodeID | 1402h | 1602h |
| | RPDO4 | 500h + NodeID | 1403h | 1603h |
| TPDO | TPDO1 | 180h + NodeID | 1800h | 1A00h |
| | TPDO2 | 280h + NodeID | 1801h | 1A01h |
| | TPDO3 | 380h + NodeID | 1802h | 1A02h |
| | TPDO4 | 480h + NodeID | 1803h | 1A03h |

## 3 PDO communication parameters

The COB_ID of the PDO contains control bits and identification data to determine the bus priority of this PDO. COB_ID is located on sub-index 01 of the communication parameters (RPDO: 1400h~1403h; TPDO: 1800h~1803h), and the highest bit determines whether this PDO is valid or not.

MSB LSB

| 31 | 30 0 |
|---|---|
| 0: ON | 1400h~1403h + Node_ID |
| 1: OFF | 1800h~1803h + Node_ID |

For example, for a site with Node_ID 1, the COB_ID of RPDO1 in the invalid state is "80000201h"; writing "00000201h" to this COB_ID will activate RPDO1.

## 4 PDO Transfer Type

Depending on the type configured, the PDO corresponds to the corresponding transmission conditions and data validity conditions. The following table shows the PDO types and transmission conditions.

| Type | Data sending conditions | Data validity conditions |
|---|---|---|
| Cyclic synchronization (Type0) | Receive synchronized frames and send data | Effective immediately |
| Cyclic synchronous (Type1-240) | Receive frames with the corresponding | Effective immediately |

| | | |
|---|---|---|
| | synchronization steps and send data frames | |
| Asynchronous (Type252) | Not supported | Not Supported |
| Asynchronous (Type253) | Not supported | Not supported |
| Asynchronous - vendor specified (Type254) | Send data after data change | Effective immediately |
| Asynchronous (Type255) | Data changes or meets the event time and the change rate is less than the suppression time | Effective immediately |

## 5 Prohibited time

The prohibition time is set for TPDO and stored on sub-index 03 of the communication parameters (1800h~1803h) to prevent the CAN network from being continuously occupied by PDO with lower priority. The time unit of this parameter is 100us. After setting the value, the transmission interval of the same TPDO must not be smaller than the time corresponding to this parameter.

For example, if the forbidden time of TPDO1 is 300, the minimum transmission interval of TPDO1 is 30ms.

## 6 Event timer

For TPDO with asynchronous transmission (transmission type 254 or 255), define the event timer, located on sub-index 05 of the communication parameter (1800h~1803h). The event timer can also be seen as a trigger event, which also triggers the corresponding TPDO transfer. If other events such as data changes occur during the timer run cycle, TPDO will also be triggered and the event counter will be reset immediately.

## 7 PDO mapping parameters

The PDO mapping parameter contains a pointer to the process data corresponding to the PDO that the PDO needs to send or receive, including the index, subindex, and length of the mapped object. Each PDO data length can be up to 8 bytes and can map one or more objects at the same time. The subindex 0 records the number of objects mapped by the PDO, and the subindexes 1 to 8 are the mapping contents. The mapping parameters are defined as follows.

| Bit | 31 | …… | 16 | 15 | …… | 8 | 7 | …… | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Definition | Index | | | Sub-index | | | Object length | | |

The index and subindex together determine the location of the object in the object dictionary, and the object length specifies the specific bit length of the object, expressed in hexadecimal:

| Object length | Bit length |
|---|---|
| 08h | 8 |

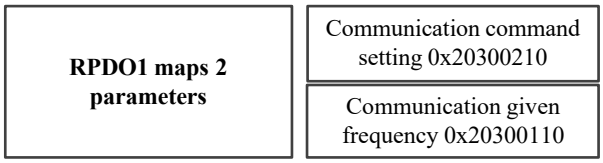
| | |
|---|---|
| 10h | 16 |
| 20h | 32 |

Example:

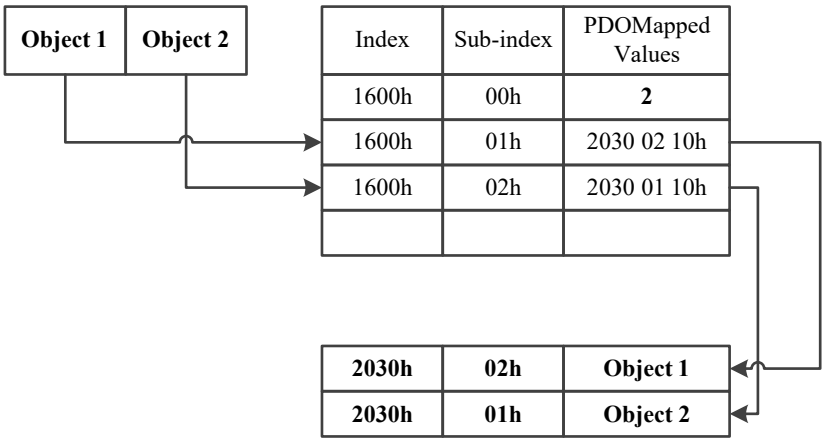
RPDO1 maps 2 parameters, T-00 (communication given frequency) and T-01 (communication command setting).

T-00 (communication given frequency), index is 0x2030, subindex is 0x01, and its mapping parameter is 0x20300110

T-01 (communication command setting), index is 0x2030, subindex is 0x02, and its mapping parameter is 0x20300210.




Then the total length of the mapping is 4 bytes, that is, RPDO1 has 4 bytes in the data segment during transmission, and its mapping relationship is as follows:




According to the above mapping example, if the site number is 3, the PLC sends the PDO data as follows:

| 11 byte ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
|---|---|---|---|---|---|---|---|---|
| 0x203 | Run command L | Run command H | Given frequency L | Given frequency H | | | | |

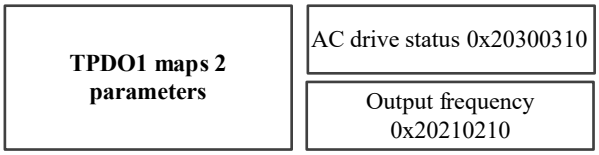
Note: This example only needs to process Byte0~Byte3 because only two variables are mapped and RPDO1 has 4 bytes in the data segment during transmission. TPDO is mapped in the same way as RPDO, but in the opposite direction; RPDO decodes the input according to the mapping relationship, while TPDO adds codes to the output according to the mapping relationship.

Examples:

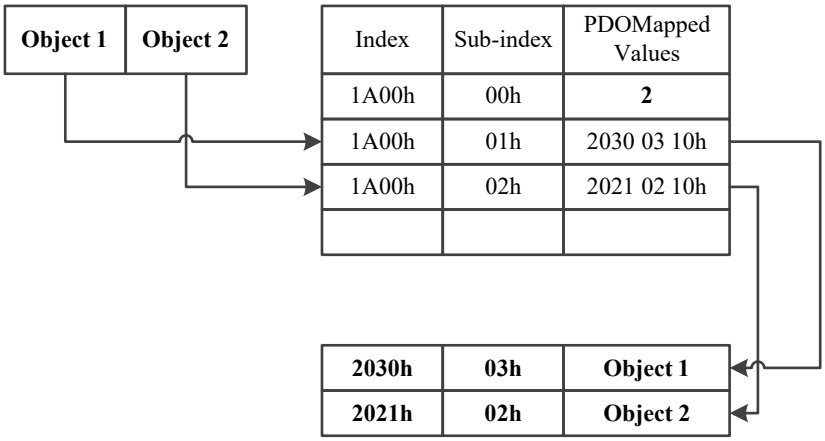
TPDO1 maps 2 parameters, which are ac drive status (T-02) and output frequency (C-01).

T-02 (ac drive status), indexed as 0x2030, subindexed as 0x03, whose mapping parameter is 0x20300310

C-01 (output frequency), indexed as 0x2021, subindexed as 0x02, and its mapping parameter is 0x20210210.




Then the total length of the mapping is 4 bytes, that is, the data segment of TPDO1 is 4 bytes during the transmission, and the mapping relationship is:




According to the above mapping example, if the site number is 3, the PDO data sent by the ac drive is:

| 11 byte ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 |
|---|---|---|---|---|---|---|---|---|
| 0x183 | Operation status L | Operation status H | Feedback frequency L | Feedback frequency H | | | | |

## 2.1.6 Emergency Message

The byte data of 8 of the emergency message is shown in the table below.

| COB-ID | Emergency error code | Error Register | Vendor-specified error code |
|---|---|---|---|
| 0x80+Node_ID | Byte0~Byte1 | Byte2 | Byte3~Byte7 |

The emergency error code in this card is the same as the vendor specified error code, both are fault codes for the ac drive. Error register please refer to the DS301 document related section object dictionary 1001H data value, 1001H bit0 for error generation flag, bit4 for communication error, bit7 vendor specified error.

### 1 Emergency Message Configuration Description

Object 1001h: Emergency message COB-ID（COB-ID Emergency Message）

| Index | 1001h | | | | |
|---|---|---|---|---|---|
| Name | Emergency Message COB-ID | | | | |
| Object structure | VAR | Data Type | Unit16 | Data Range | Unit16 |
| Can you map | NO | Accessibility | RW | Factory setting | 0x80+Node_ID |

Example of error code:

| COB-ID | Data Information | Remarks |
|---|---|---|
| 081 | 3E 00 01 00 3E 00 00 00 | Byte0:3E corresponds to the following error code description: current or torque detection fault before releasing the gate |

### 2 Fault code definition

AC70T ac drive fault and alarm definition, please refer to the following table for specific fault meaning.

| AC drive fault information | AC drive fault information |
|---|---|
| 0x0000: No fault<br>0x0001: Module failure<br>0x0004: Overcurrent in acceleration<br>0x0005: Overcurrent in deceleration<br>0x0006: Constant speed overcurrent<br>0x0007: Accelerating overvoltage<br>0x0008: Deceleration overvoltage<br>0x0009: Constant speed overvoltage<br>0x000A: Bus undervoltage<br>0x000B: Motor overload<br>0x000C: AC drive overload<br>0x000D: Input side out of phase<br>0x000E: Output side out of phase<br>0x000F: Rectifier overheating<br>0x0010: AC drive overheating<br>0x0011: AC drive external fault<br>0x0012: RS485 communication abnormality<br>0x0013: Current detection fault<br>0x0014: Motor self-learning fault<br>0x0015: Storage fault<br>0x0017: Brake unit fault | 0x001A: Parameter copy exception<br>0x001B: PG card connection abnormality<br>0x001C: Overvoltage at shutdown<br>0x001D: PID feedback fault<br>0x001E: Keyboard unrecognized fault<br>0x001F: Initial position angle learning failure<br>0x0020: Excessive speed deviation<br>0x0021: Flying speed protection<br>0x0022: Load protection 1<br>0x0023: Load protection 2<br>0x0024: CPU timeout<br>0x002A: PG card phase AB disconnected<br>0x002B: PG card phase B disconnected<br>0x002C: PG card phase A disconnected<br>0x002D: PG card phase Z disconnected<br>0x003B: Anti-hanging fault<br>0x003D: Brake failure detection fault<br>0x003E: Current or torque detection fault before release<br>0x003F: Current detection fault during operation |

## 2.2 CANopen related function codes

To use CANopen expansion card, you must set the ac drive related function code, AC70T related function code as follows

| Function Code | Description |
|---|---|
| F0.02=3 | Run command communication selection expansion card |
| F00.03=10 | Select expansion card for frequency setting |
| Fd.02 Ten positions | Set baud rate, need to re-power after change |
| Fd.12 | Set the node number and re-power after change |

Among them, the baud rate setting corresponds to the following table：

| Fd.02 Ten positions | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| CAN Baud rate | 20K | 50K | 100K | 125K | 250K | 500K | 1M |

# 3 Overview of the CANopen protocol

## 3.1 Introduction to CANopen

CANopen is an application layer protocol for network transmission systems based on the CAN serial bus, which defines the data link layer and part of the physical layer of the OSI model. Multiple master-slave mode can be used, and any node on the network can take the initiative to send information to other nodes. Network nodes can be divided into different priority levels according to system real-time requirements, which can reduce the bus arbitration time in case of bus conflicts. The CAN network abolishes the traditional part-address encoding and replaces it with the encoding of communication data blocks. This not only allows for a theoretically unlimited number of nodes in the network, but also allows different nodes to receive the same data at the same time. It also has the characteristics of short transmission bytes, fast speed, good fault tolerance and reliable data transmission, which makes it very suitable for industrial control and distributed real-time control.

CANopen the device model is shown in the following figure:

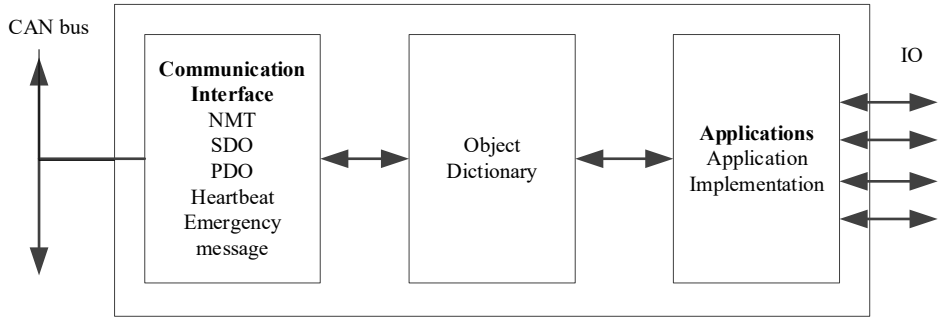



Figure 3.1 CANopen device model

## 3.2 Object Dictionary

An object dictionary is an ordered set of parameters and variables that contains all parameters for the device

description and the device network state. The set of objects can be accessed in an ordered and predefined way through the network. The CANopen protocol uses an object dictionary with a 16-bit index and an 8-bit sub-index. The structure of the object dictionary is shown in the table below, and a master node or configuration tool can access all values in the object dictionary of a slave node.

| Index | Object |
|---|---|
| 0000 | Not used |
| 0001-000F | Static data types |
| 0020-003F | Complex data types |
| 0040-005F | Manufacturer-specified complex data types |
| 0060-007F | Static data types specified by device sub-protocols |
| 0080-009F | Complex data types specified by device sub-protocols |
| 00A0-0FFF | Reserved |
| 1000-1FFF | Communication subprotocol area (e.g., device type, error register, number of PDO's supported) |
| 2000-5FFF | Manufacturer-specific subprotocol area |
| 6000-9FFF | Standard device subprotocol area |
| A000-FFFF | Reserved |

## 3.3 NMT Control message

Only the master node can generate NMT messages. NMT is used to manage and monitor each node in the network, and mainly implements node status control, error control and node startup. Where the frame ID is fixed to 0x000, Data0 is the command word and Data1 is the node number.

NMT Message

| COB-ID | RTR | Data0 | Data1 |
|---|---|---|---|
| 0x000 | 0 | Command word | Node ID |

NMT Message command

| command | Description |
|---|---|
| 0x01 | Start Node |
| 0x02 | Stop node |
| 0x80 | Enter pre-run state |
| 0x81 | Reset node |
| 0x82 | Reset communication |

## 3.4 Service Data Objects (SDO)

By using indexes and sub-indexes, SDO enables clients to access variables in the device object dictionary. the SDO protocol is to acknowledge the service type and generate an answer for each message. SDO request and answer operations can be found in Section 2.1.4 and will not be described here in detail.

## 3.5 Process Data Objects (PDO)

Used to transfer real-time data, which is passed from one creator to one or more receivers. Data transfer is limited to 1 to 8 bytes. Each CANopen device contains 8 default PDO channels, 4 transmit PDO channels and 4 receive PDO channels. PDO contains both synchronous and asynchronous transmission methods, which are determined by the corresponding communication parameters of this PDO。

## 3.6 Heartbeat messages

The heartbeat model uses a producer-consumer model.

CANopen devices can send heartbeat messages according to the period set by the producer heartbeat interval object 1017h, in ms. The node with the consumer heartbeat function in the CAN network always monitors this producer according to the consumer time set in object 1016h, and once the producer heartbeat of the corresponding node is not received within the consumer heartbeat time, the node is considered to be faulty.

After configuring the producer heartbeat interval 1017h, the node heartbeat function is activated and starts to generate heartbeat messages. After configuring a valid subindex of consumer heartbeat 1016h, monitoring starts after receiving a frame of heartbeat from the corresponding node.

The host sends heartbeat telegrams according to its producer time, and the slave monitoring the host does not receive the heartbeat telegrams within the object 1016 subindex time, then the host is considered to have dropped the station. The object 1016h subindex time ≥ host producer time × 2, otherwise it causes the slave to mistakenly think that the host is dropped.

Each object of the slave sends a heartbeat telegram at 1017h time, and the host that monitors the slave and does not receive the heartbeat telegram within the consumer time is considered to have dropped the slave.

A node can be configured to generate periodic messages called heartbeat telegrams, which reflect the state of the node itself. Heartbeat telegrams are optional, i.e. the master can choose to enable or disable heartbeat telegrams.

Heartbeat message structure:

| COB-ID | RTR | Data0 |
|---|---|---|
| 0x700+NodeID | 0 | Status word |

Heartbeat message status word:

| date | Description |
|---|---|
| Data0 | 4: Stop running<br>5：Run<br>127: Pre-run |

### 1 Heartbeat frame configuration

Object 1016h：Consumer Heartbeat Time

| Index | 1016h | | | | |
|---|---|---|---|---|---|
| Name | Consumer Heartbeat Time | | | | |
| Object structure | ARR | Data Type | Unit16 | Data Range | Unit16 |

| | | | | | |
|---|---|---|---|---|---|
| Can be mapped or not | NO | Accessibility | RW | Factory setting | |

Function description: The parameters include the address of the monitored node and the actual consumer time, which must be greater than the heartbeat producer of the corresponding node

Time (in ms). It is not possible to set two consumer times for the same node.

| | | | | | |
|---|---|---|---|---|---|
| Index | 00h | | | | |
| Name | Number of items (number entries) | | | | |
| Object structure | - | Data Type | - | Data Range | 1 |
| Can be mapped or not | NO | Accessibility | RO | Factory setting | 1 |

Function description: Only 0 can be written, and all error records are cleared at this time

| | | | | | |
|---|---|---|---|---|---|
| Index | 01h | | | | |
| Name | （Consumer Heartbeat Time ） | | | | |
| Object structure | - | Data Type | Unit16 | Data Range | Unit16 |
| Can be mapped or not | NO | Accessibility | RW | Factory setting | 0 |

Function Description: Save all parameters in the object dictionary list

Object 1017h: Producer Heartbeat Time (Producer Heartbeat Time)

| | | | | | |
|---|---|---|---|---|---|
| Sub-index | 01h | | | | |
| Name | Consumer Heartbeat Time | | | | |
| Object structure | VAR | Data Type | Unit16 | Data Range | Unit16 |
| Can be mapped or not | NO | Accessibility | RW | Factory setting | 0 |

Function Description: The producer heartbeat time defines the cycle time of the heartbeat.

Heartbeat message configuration example:

| | | | |
|---|---|---|---|
| 61B | **STANDARD** | 8 | 2B 17 10 00 2C 01 00 00 |
| 59B | **STANDARD** | 8 | 60 17 10 00 00 00 00 00 |

# 4 PG card wiring instructions

## 4.1 Product technical parameters

| **Product Model** | **Power supply** | **Function** | **Input Signal Characteristics** | | **Output signal characteristics** | |
|---|---|---|---|---|---|---|
| | | | **Frequency range** | **Input Impedance** | \ | **Output Current** |
| AC200CANPG01 | 24V±5% 100mA | Differential input wire break | Differential0～100kHz | 1kΩ | \ | ≤200mA |

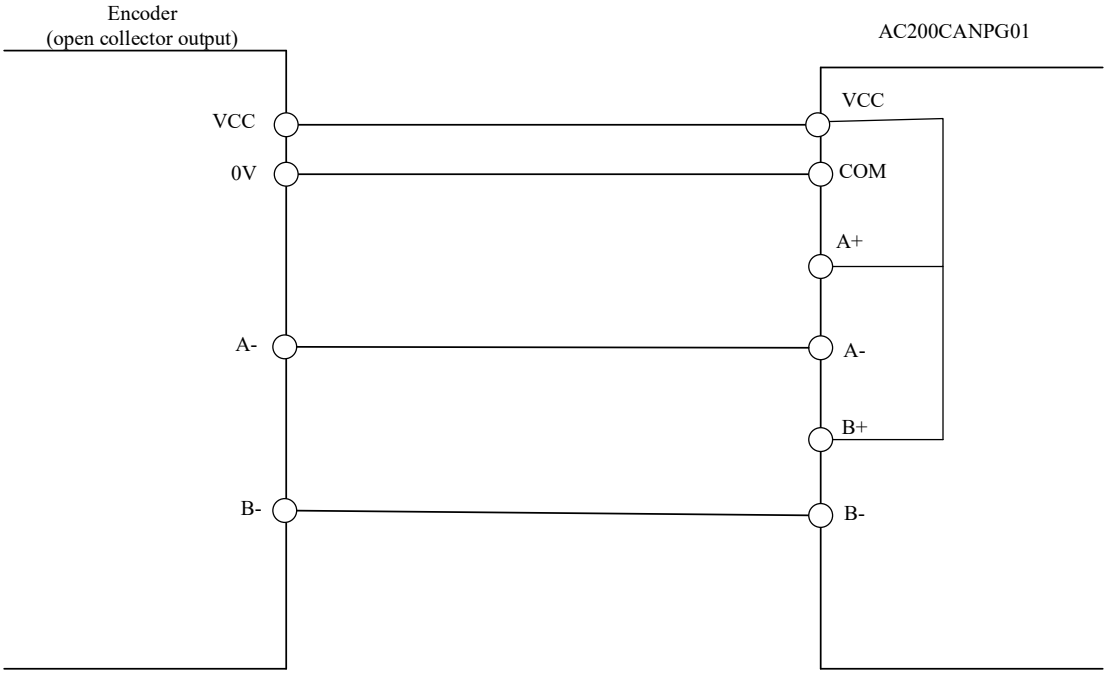
| | | | | | | |
|---|---|---|---|---|---|---|
| | | detection | | | | |

## 4.2 PG card terminal function description

| Terminal Name | Function |
|---|---|
| PE | Ground |
| PGVCC | +5V/12V output power supply (can supply power to encoder) |
| PGND | 5V power ground |
| +24V | +24V output power (can supply power to encoder) |
| COM | 24V power ground |
| A+ | Differential A+ signal |
| A- | Differential A- signal/OC type A signal |
| B+ | Differential B+ signal |
| B- | Differential B-signal/OC type B-signal |
| Z+ | Differential Z+ signal |
| Z- | Differential Z-signal/OC-type Z-signal |

## 4.3 Open collector output type (OC) encoder wiring method

When wiring the open collector output (OC) encoder, note that A+ and B+ on the PG card must be left open, as shown in the diagram below.



# 5 Block frequency setting (FF.00 is not 0)

## 5.1 Corresponding block relationship for a given frequency

The operation object is T-00 (communication given frequency) with index 2030h and subindex 01h. See 2.1.3 AC drive parameter operation for details of index and subindex calculation. Assuming the ac drive node address (Fd.12) is 2, write 1000 (3E8h) to T-00, which means that a block operation is performed

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x602 | 0 | 2b 30 20 01 E8 03 |

Correspondence between communication given frequency and gear position:

| Gear | Host writes values to T-00 | Corresponding to group F parameters | Default value (Hz) |
|---|---|---|---|
| First gear | 1000 | F0.08 | 10.00 |
| Second gear | 2000 | FC.00 | 25.00 |
| Third gear | 3000 | FC.02 | 40.00 |
| Fourth gear | 4000 | FC.06 | 60.00 |
| Fifth gear | 5000 | FC.14 | 80.00 |

## 5.2 Ant speed frequency corresponds to gear relationship

Correspondence of communication given frequency

| Ant-Speed gears | Host writes values to T-00 | Corresponding to group F parameters | Default value (Hz) |
|---|---|---|---|
| Ant speed first gear | 100 | FC.46 | 3.00 |
| Second gear | 200 | FC.47 | 5.00 |
| Third gear | 300 | FC.48 | 10.00 |
| Fourth gear | 400 | FC.49 | 15.00 |
| Fifth gear | 500 | FC.50 | 20.00 |

# 6 Running operation

Firstly, to control the ac drive start/stop by communication requires setting command source F0.02=3;

Second, the frequency source F0.03=10 needs to be set.

The following is an example of how to control the operation of the ac drive with one gear positive transit line.

1, the host writes 1000 (3E8h) to T-00

The master sends the message as

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x602 | 0 | 2b 30 20 01 e8 03 00 00 |

2, the host writes 0001h to T-01

The host sends a message as

| Message identifier（Hex） | RTR | DATE（Hex） |
|---|---|---|
| 0x602 | 0 | 2b 30 20 02 01 00 00 00 |

# Description：

1, Ac drive parameter information, such as decimal points, ranges and attributes, refer to the parameter section of the AC70T manual;

2, About the frequency given

FF.00=0 (general purpose mode), the ac drive will operate with the value of T-00 (communication given frequency) written by the host as the given frequency in 0.01Hz, i.e. 1.00Hz when written to 100 and 10.00Hz when written to 1000;

When FF.00 is not 0, the ac drive is given the frequency as described in Section 5 (Block Setting Frequency).

3, Please refer to this document and eds document for CANopen address related descriptions, the communication address in AC70T manual refers to 485 communication address.