# AC300CAN1 Communication Card User's Manual

Thank you for using AC300(310) series inverter products and AC300CAN1 fieldbus communication card. AC300CAN1 (CANopen) communication card is used for AC300(310) series inverter, and it is a special expansion card for CAN communication network, which allows the inverter to access the high-speed CAN communication network and realize the field bus control.

Please read this manual carefully before using this product.

## 1. Hardware configuration

This CAN communication card is specially configured for our AC300 and AC310 series machines. The CAN bus interface is fully compliant with ISO/DIS11898 standard to achieve CAN communication between multiple inverters.

AC300CAN1 card wiring port uses terminal wiring, the inverter only supports EX-A expansion interface in hardware, EX-B expansion port is not supported at the moment.

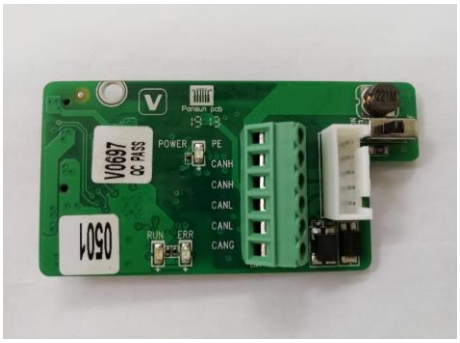



Figure 1.1 AC300CAN1 front schematic

### 1.1 Terminal Wiring

6pin terminal for CAN bus connection, number CN4, located on the front of the communication card， It is very convenient for customers to conduct parallel wiring (CANH, CANL can realize one in and one out), the pin schematic and function table are as follows：

Table 1.1 Wiring port definition description

| CN4 port menu | | |
|---|---|---|
| **Pin No.** | **Name** | **Function** |
| 1 | PE | Cable shield ground terminal |
| 2 | CANH | Connecting the CAN bus positive polarity port |
| 3 | CANH | Connecting the CAN bus positive polarity port |
| 4 | CANL | Connecting the CAN bus negative polarity port |
| 5 | CANL | Connecting the CAN bus negative polarity port |
| 6 | CANG | Connection to CAN bus signal reference ground |

### 1.2 Baud rate and transmission distance

Table 1.2 Description of communication 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 dip switches

For field use, AC300CAN1 cards are equipped with terminal matching resistors that can be set for use via dip switch S1. It is recommended to set up the use of termination resistors at both ends of the network topology. The setting instructions for dipswitch S1 are as follows：

Table 1.3 Terminal resistance dial code function description

| S1 | Dial code position | Terminal resistance |
|---|---|---|
|  | Toggle to the left side ON, i.e. pins 1 and 2 | Terminal resistance access |
| | Toggle to the right side OFF, i.e. pins 2 and 3 | No termination resistor used (factory default) |

### 1.4 Indicator light description

The CAN card is equipped with three 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 | Note |
|---|---|---|---|
| PWR（Red） | Bright | Power on normal | power up |
| | extinguish | Power on is not normal, please test the installation is normal | power off |
| ERR（Red） | Bright | Internal inverter communication timeout | Frequency converter communication timeout |
| | Fast flashing | CANopen address setting error | Address configured to 0 |
| | Flash twice | CANopen emergency message inverter failure | Inverter reported fault |
| RUN（Green） | Bright | Run | The NMT starts the remote node and the TPDO uploads data in a communicable state. |
| | Blinking | Pre-run | The CANOPEN card is in a pre-operable state. |
| | extinguish | Stop | NMT stops the remote node and is unable to communicate. |

### 1.5 Cautions

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

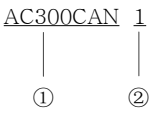
b. A 120 ohm termination resistor is required for the master and the last node of the slave. The AC300CAN1 communication card comes with a termination resistor that can be enabled by dipswitch S1 to prevent signal reflection.

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

d. The longer the connection line, the higher the requirement for the CAN chip's drive capability.

## 2. Order Model

Product order model introduction




**Description：**

① Product Series

AC300(AC310) series CAN communication expansion card

② Wiring port method

1：European type terminal block　　2：RJ45 Network Port

## 3. software configuration

### 3.1 Agreement Description

#### 3.1.1 Software Features

The AC300(310) CANopen card supports the following protocols：

Support Heartbeat protocol, the slave reports the current status to the master at regular intervals；

Support SDO accelerated transmission mechanism；

Support 4 TPDO, 4 RPDO；

Support for emergency objects；

#### 3.1.2 Newsletter recipients 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

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

### 3.1.3 Inverter parameter operation

1）Baud rate

AC300 sets the baud rate by the ten bits of function code F13.02, which needs to be re-powered after changing. The setting values are listed below：

Table 3.1 AC300 Baud Rate Settings

| F13.02 Ten bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| CAN Baud rate | 20K | 50K | 100K | 125K | 250K | 500K | 1M |

The AC310 sets the baud rate via function code F12.42, which needs to be re-powered after the change. The setting values are listed in the following table：

Table 3.2 AC310 Baud Rate Settings

| F12.42 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| CAN Baud rate | 20K | 50K | 100K | 125K | 250K | 500K | 1M |

2）Device Address

AC300 sets the device address (node station number) via F13.12; AC310 sets the device address via F12.41. After changing this parameter, you need to reapply power

3）Mapping Instructions

The inverter 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 Subindex Value = Function code parameter number +1。

For example, F01.05 function code corresponds to index 2001h, subindex 06h (object dictionary 2001_06h)。The correspondence between function codes and indexes is shown in the following table：

| **Function Code** | **CANopen Index** |
|---|---|
| F0-FF | 0x2000-0x200F |

| | |
|---|---|
| F16-F31 | 0x2010-0x201F |

If a write parameter power-down save operation is to be performed, the corresponding index is shown in the following table：

| **Function Code** | **CANopen Index** |
|---|---|
| **F0-FF** | **0x20F0-0x20FF** |
| **F16-F31** | **0x20A0-0x20AF** |

### 3.1.4 SDO read/write inverter parameter operation

1) SDO Read/Write Operation Protocol Explanation

SDO read operation

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

Table 3.3 SDO Read

| CAN | CANopen Data | Description |
|---|---|---|
| ID | 0x600+NodeID | NodeID Device Address |
| RTR | 0 | - |
| Data0 | Command Code（0x40） | 0x40 Indicates a read command |
| Data1 | Index Low Byte | - |
| Data2 | Index High Byte | - |
| Data3 | Sub-index | - |
| Data4 | Data1 | Reserved（0） |
| Data5 | Data2 | Reserved（0） |
| Data6 | Data3 | Reserved（0） |
| Data7 | Data4 | Reserved（0） |

Read the inverter SDO slave response data as shown in the table below：

Table 3.4 SDO Read Response

| 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 | Data 1 | Read back data if correct, SDO error code if incorrect |
| Data5 | Data 2 | |
| Data6 | Data 3 | |
| Data7 | Data 4 | |

SDO write operations

The inverter uses the CANopen Service Data Object (SDO) to write to the inverter, and the master sends the data in the format shown in the table below：

Table 3.5 SDO write

| CAN | CANopenData | 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 | Data 1 | Data to be written |
| Data5 | Data 2 | |
| Data6 | Data 3 | |
| Data7 | Data 4 | |

Inverter Response

Table 3.6 SDO Write 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 | Data 1 | 0 in case of correct, SDO error code in case of error |
| Data5 | Data 2 | |
| Data6 | Data 3 | |
| Data7 | Data 4 | |

SDO Error Code

Table 3.7 SDO error codes

| SDO Error Code | Error Description |
|---|---|
| 0x05040000 | Protocol Timeout |
| 0x05040001 | Invalid command for SDO messages |
| 0x06010002 | Write operations on read-only object dictionaries |
| 0x06020000 | Can't find object dictionary |
| 0x06040041 | Objects cannot be mapped to PDO |
| 0x06040042 | PDO's mapping length 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 errors |

| | |
|---|---|
| 0x08000020 | Data cannot be changed in the current state |

The following is an example of reading and writing function codes for inverters with inverter node address 2

2）Example of reading group F function code

Take the example of reading the value of F02.02 (object dictionary corresponds to 2002_03H), the message sent by the master is as follows:

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

The response message of the inverter 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 function codes

Writing value 3 to F02.02, the master sends the following message:

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

The response message of the inverter is as follows：

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

4）Example of reading power group C parameters

Group C parameter function code corresponds to inverter start address 0x2100, so when sending SDO telegrams when reading group C parameters，

Index is: 0x2021+Group C number, subindex is: function code parameter number plus +1。

If you read C00.26 (inverter rated voltage), the index is 0x2021 and the sub-index is 0x1B。

Then the master sends a message as：

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

The response message of the inverter is as follows：

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x582 | 0 | 4b 21 20 1B DC 00 |

5）Example of reading and writing 0x3000 address group parameters

0x3000 address group parameter function code corresponds to inverter starting address 0x3000，So when sending SDO messages when reading this set of parameters，Index is: 0x2030, subindex is: 0x3000 group address of the last 8 bits + 1.

If the address 0x3006 (torque control forward maximum frequency limit) is read, the index is 0x2030 and the sub-index is 0x06+1=0x07

Then the master sends a message as：

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x602 | 0 | 40 30 20 07 00 00 00 00 |

The response message of the inverter is as follows：

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x582 | 0 | 4b 30 20 07 00 00 |

When writing the data at this address, the index remains 0x2030 and the subindex remains 0x07. For example, if 100 is written to this address, the master sends a message as：

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x602 | 0 | 2b 30 20 07 64 00 |

The response message of the inverter is as follows：

| Message identifier（Hex） | RTR | Data（Hex） |
|---|---|---|
| 0x582 | 0 | 60 30 20 07 |

**3.1.6 PDO inverter 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）RPDO inverter operation

RPDO is the process data received by the inverter. 4 RPDOs, RPDO1, RPDO2, RPDO3 and RPDO4, are supported. Each RPDO can map objects in address group 0x3000, and the objects mapped in the RPDO must be writable addresses in the 0x3000 group，Otherwise, an error may be reported. The total data length of each RPDO mapping variable cannot exceed 64 bits.

2）TPDO Inverter Operation

TPDO is the process data sent by the inverter. The inverter maps the real-time variables to be sent to TPDO and sends the data to the master station periodically. Support 4 TPDO, TPDO1, TPDO2, TPDO3, TPDO4, each TPDO can map common objects in C00 group monitoring (such as C00.01, C00.02, C00.03, etc.). The total length of each TPDO mapped variable cannot exceed 64 bits.

3）PDO data sending conditions

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。

Table 3.8 PDO types and transmission conditions

| **Type** | **Data sending conditions** | **Data validity conditions** |
|---|---|---|
| Cyclic synchronization（Type0） | Receive synchronization frame, send data | Effective immediately |
| Cyclic synchronization（Type1-240） | After receiving the frame with the corresponding synchronization step, send the data frame | Effective immediately |
| 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） | The data changes or meets the event time and the rate of change is less than the suppression time | Effective immediately |

4）Default PDO Mapping

The variables of PDO mapping can be selected arbitrarily according to the actual application. The following table shows the factory default mapping variables for practical application reference。The control command and run status are mandatory mapping variables.

Table 3.9 Default PDO Mapping

| PDO | Transfer Protocol | Periodicity | Parameters | | | |
|---|---|---|---|---|---|---|
| RPDO1 | Asynchronous Timing Transfer | 100ms | Control commands (0x3001) | Frequency command（0.01Hz）(0x3000) | Torque Feeding（0.1%）(0x3005) | |
| TPDO1 | Asynchronous Timing Transfer | 100ms | Operation Status (0x3002) | Output Current (0.1A) (C00.02) | Output Frequency（0.01Hz）(C00.1) | Mechanical speed 1RPM (C00.05) |
| TPDO2 | Asynchronous Timing Transfer | 100ms | Output power (C00.10) (0.1%) | Input Voltage 0.1V (C00.03) | Fault Code (0x3003) | |
| TPDO3 | Asynchronous Timing Transfer | 100ms | Output torque（C00.07）(0.1%) | DC bus voltage (0.1V)(C00.11) | Motor voltage (0.1V)(C00.04) | Module Temperature (0.1℃)(C00.12) |

The following table shows the control commands and the bit definitions of the operation status

Table 3.10 Control commands and operation status definitions

| | Bit | Name | Description | Signal Status |
|---|---|---|---|---|
| Control command | | Auto control（Address 0x3000） | Master Control | 0：No order 5：Deceleration stop<br>1：Forward run 6：Free stop<br>2：Reverse Run 7：Fault Reset<br>3：Positive point movement 8：Run the ban command<br>4：Reversing point movement 9：Run the allow command |
| Operation Status | **Bit** | **Name** | **Description** | **Signal Status** |
| | 0 | Operation Status | | 0：Downtime status 1：Operation Status |
| | 1 | Accelerated Status | | 0：Non-accelerated status 1：Accelerated Status |
| | 2 | Deceleration status | | 0：Non-deceleration status 1：Deceleration status |
| | 3 | Direction | | 0：Positive 1：Reverse |
| | 4 | Fault | With or without fault | 0：No faults 1：Inverter fault |
| | 5 | GPRS status | | 0：GPRS Unlock 1：GPRS Locked status |
| | 6 | Early Warning | With or without warning | 0：No warning 1：Inverter warning |

| | | | | |
|---|---|---|---|---|
| | 7 | Ready | | 1：Inverter ready |
| | 14 | Heartbeat | With heartbeat messages alternating between 0 and 1 | |
| | 15 | Motor braking feedback | | |

### 3.1.7 Emergency message and fault code description

1）Description of the emergency message

The byte data of the emergency message is shown in the following table：

| COB-ID | Emergency error code | Error Register | Vendor-specified error code |
|---|---|---|---|
| 0x80+Node_ID | 0~1 | 2 | 3~7 |

The emergency error code in this card is the same as the manufacturer's assigned error code, both are fault codes for the inverter。Error register, please refer to the DS301 documentation of the relevant chapter object dictionary 1001H data value, 1001H bit0 for error generation flag, bit4 for communication errors, bit7 vendor specified error.

2）Fault code (0x3003) definition

AC300 inverter and AC310 inverter fault definition is different, please pay a little attention to the application, please refer to the following two tables for specific fault meaning。

Table 3.11 AC300 fault message codes

| **Inverter fault information** | **Inverter fault information** | **Inverter fault information** |
|---|---|---|
| 0x0000：No faults | 0x0012 ： RS485 communication exception | 0x0027：Short circuit to ground |
| 0x0001：System anomaly | 0x0013：Current detection fault | 0x0028：Fan short circuit |
| 0x0004：Overcurrent in acceleration | 0x0014：Motor detection fault | 0x0029：Motor overheat |
| 0x0005：Deceleration overcurrent | 0x0015：Storage failure | 0x002A：Extension card disconnection |
| 0x0006：Constant speed overcurrent | 0x001A：Parameter copy exception | 0x0040：Low voltage at shutdown |
| 0x0007：Accelerated overpressure | 0x001B：PG card connection abnormality | 0x0041：Input side out of phase |
| 0x0008：Deceleration overpressure | 0x001C：Overvoltage at shutdown | 0x0042 ： PID feedback failure |
| 0x0009：Constant speed overpressure | 0x001D：PID feedback failure | 0x0043：Load protection 1 |
| 0x000A：Undervoltage in operation | 0x001F：Initial position angle learning failure | 0x0044：Load protection 2 |
| 0x000B：Motor overload | 0x0020：Excessive speed deviation | 0x0045：Storage failure |
| 0x000C：Inverter overload | 0x0021：Flying Protection | 0x0046：Excessive speed deviation |
| 0x000D：Input side phase loss | 0x0022：Load protection 1 | 0x0047：Flying Protection |
| 0x000E：Output side phase loss | 0x0023：Load protection 2 | 0x0048：GPS Locking Machine |
| 0x000F：Rectifier bridge overheat | 0x0024 ： CPU timeout | 0x0049 ： GPSDisconnected |
| 0x0010：Inverter overheat | 0x0025 ： OTP verification failure | 0x004a：RS485 Communication abnormalities |
| 0x0011：Inverter external fault | 0x0026：Synchronizer out of step | 0x004c：Motor overheat |

表 3.12 AC310 故障信息代码

| 变频器故障信息 | 变频器故障信息 | 变频器故障信息 |
|---|---|---|
| 0x0000：No faults | 0x0024：V-phase zero drift is large | 0x005C：Extension card B disconnected |
| 0x0001：Module failure in acceleration | 0x0025：Three phase current sum is not 0 | 0x005D：CAN expansion card fault |
| 0x0002：Module failure in deceleration | 0x0026：W-phase zero drift is large | 0x005E：Other expansion card failures |
| 0x0003：Module failure in constant speed | 0x0028：Short circuit to ground | 0x005F：Other card disconnection |
| 0x0004：Stop module failure | 0x0029：Fan shorted to ground | 0x0060：Other card disconnection |
| 0x0005：Overcurrent in acceleration | 0x002A：PID disconnection fault | 0x0061：Comparator 1 failure |
| 0x0006：Overcurrent in deceleration | 0x002B：Parameter copy exception | 0x0062：Comparator 2 failure |
| 0x0007：Overcurrent in constant speed | 0x002C：PG parameters are not suitable | 0x0063：Parameter setting error |
| 0x0008 ：AC310 software overcurrent | 0x002D： Z pulse fault | 0x0080：Shutdown undervoltage |
| 0x0009：Overvoltage in acceleration | 0x002E：ABZ encoder failure | 0x0082：Input side out of phase |
| 0x000A：Overpressure in deceleration | 0x002F：Rotation check error | 0x0083：PID disconnection |
| 0x000B：Overvoltage in constant speed | 0x0030：Rotation change broken line | 0x0084：Storage Alert |
| 0x000C：Reserved | 0x0031：Other encoder failures | 0x0085：Large speed deviation |
| 0x000D：Busbar undervoltage | 0x0032：Brake unit failure | 0x0086：Flying Protection |
| 0x000E：Motor overload | 0x0034：Motor self-learning fault | 0x0087：Lockout Alert |
| 0x000F：Inverter current overload | 0x0047：Initial angle learning failure 1 | 0x0088：GPRS lockout warning |
| 0x0010：Inverter CBC overload | 0x0048：Initial angle learning failure 2 | 0x0089：485 communication exception warning |
| 0x0011：Inverter overload 3 | 0x0049：Initial angle learning failure 3 | 0x008A：Load protection 1 |
| 0x0012：Input phase loss | 0x004A：PM_ST | 0x008B：Load protection 2 |
| 0x0013：Three-phase output missing | 0x004B：PM_ST | 0x008C：Extension card disconnect warning |
| 0x0014：U-phase output missing | 0x004C：PM_ST | 0x008D：Heat sink overheat warning |
| 0x0015： V-phase output missing | 0x004D：Excessive speed deviation | 0x008E：Motor overheating warning |
| 0x0016： W-phase output missing | 0x004E：Flying Protection | 0x008F：Running warning |
| 0x001E：Rectifier overheat | 0x004F：Load protection 1 | 0x0090：External keyboard disconnection warning |
| 0x001F：Inverter overheat | 0x0050：Load protection2 | 0x0091：Parameter copy exception warning |
| 0x0020：Motor overheat | 0x0051：CPU Timeout | 0x0092：Comparator 1 Warning |
| 0x0021：External faults | 0x0055：Program Protection | 0x0093：Comparator 2 Warning |
| 0x0022：Communication failure | 0x0056：Write function code error | |
| 0x0023 ：U-phase zero drift is large | 0x005B：Extension card A disconnected | |

## 3.2 CANopen Related function codes

### 3.2.1 Set function code

Inverter-related function codes must be set to use CANopen expansion cards。

1）AC300 related function code setting

Table 3.13 AC300-related function codes

| Function Code | Description |
|---|---|
| F00.02=2 | Run command communication selection RS485 given |
| F00.03=6 | RS485 frequency selection |
| F07.01=6 | In torque mode, the torque feed selects RS485 feed |
| F07.10=6 | In torque mode, RS485 is selected for torque forward speed limit *F07.12 |
| F07.11=6 | Torque mode, torque reversal speed limit selection RS485 given *F07.13 |
| F07.12 | Torque control forward maximum speed limit Default 100.0%, relative to maximum frequency (F00.09) |
| F07.13 | Torque control reversing maximum speed limit Default 100.0%, relative to maximum frequency (F00.09)) |
| F13.02 Ten bit | Set baud rate, need to re-power after change |
| F13.12 | Set node number, need to re-power after change |

2）AC310 related function code setting

Table 3.14 AC310 related function codes

| Function Code | Description |
|---|---|
| F01.01=3 | Run command communication selection option card |
| F01.02=10 | Frequency Feeding Select Optional Card Feeding |
| F03.41=7 | In torque mode, torque command selection option card |
| F03.54=7 | In torque mode, the torque forward speed limit is given by the optional card selection |
| F03.55=7 | In torque mode, the torque reversal speed limit is given by the optional card selection |
| F03.56 | Default 100.0%, relative to maximum output frequency (F01.10) |
| F03.57 | Default 100.0%, relative to maximum output frequency (F01.10) |
| F12.42 | Set baud rate, need to re-power after change |
| F12.41 | Set node number, need to re-power after change |

## 3.3 Overview of the CANopen protocol

### 3.3.1 CANopen Brief introduction

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. It can be used in a multi-master-slave mode, where any node on the network can initiate messages to other nodes. Network nodes can be divided into different priority levels according to the system real-time requirements, which can reduce the bus arbitration time in case of bus conflicts. The CAN network abolishes the traditional partial address coding and replaces it with the coding of communication data blocks. This not only makes the number of nodes in the network theoretically unlimited, but also allows different nodes to receive the same data at the same time, and has the characteristics of short transmission bytes, fast speed, good fault tolerance and reliable data transmission, making it very suitable for industrial control and distributed real-time control.

The CANopen device model is shown in the following figure：

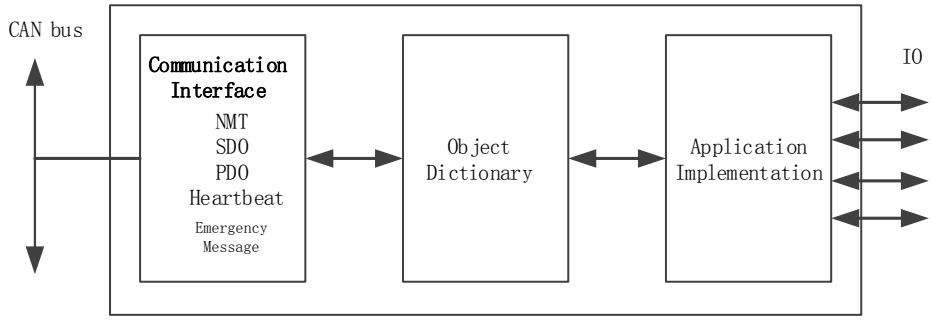



Figure 3.1 CANopen device model

### 3.3.2 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 through the network in an ordered and predefined way。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.

Table 3.15 Object Dictionary

| Index | Object |
|---|---|
| 0000 | Not used |
| 0001-000F | Static data types |
| 0020-003F | Complex Data Types |
| 0040-005F | Complex data types specified by the manufacturer |
| 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 sub-protocol area (e.g. device type, error register, number of PDO's supported) |
| 2000-5FFF | Manufacturer-specific sub-protocol areas |
| 6000-9FFF | Standard equipment sub-protocol area |
| A000-FFFF | Reserved |

### 3.3.3 NMTControl 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 | Entering pre-run status |
| 0x81 | Reset Node |
| 0x82 | Reset communication |

### 3.3.4 Service Data Objects (SDO)

SDO enables clients to access variables in the device object dictionary through the use of indexes and sub-indexes. 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 3.1.4 and are not described here in detail

### 3.3.5 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. the PDO contains both synchronous and asynchronous transmission methods, determined by the communication parameters corresponding to that PDO.

### 3.3.6 Heartbeat Message

A node can be configured to generate periodic messages called heartbeat telegrams that 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：

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

### 3.4.Structure size and installation schematic

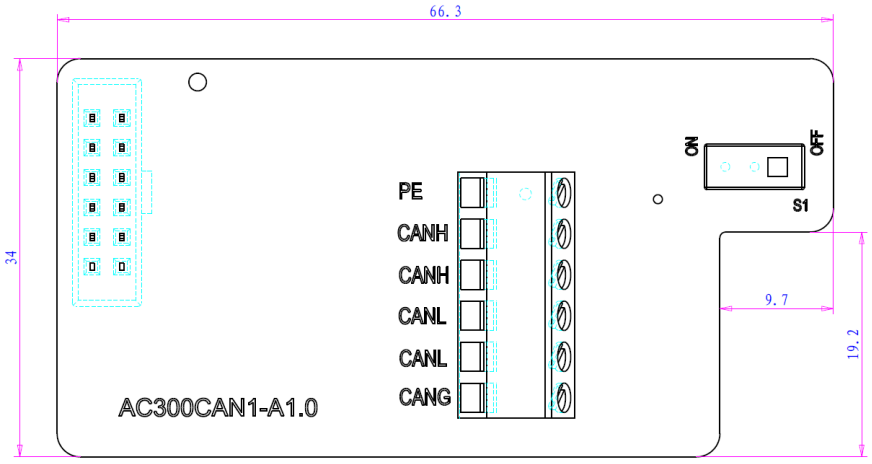



Figure 4.1 AC300CAN1 External Dimensions