CANopen

User manual

Website: http://www.we-con.com.cn/en
Technical Support: support@we-con.com.cn
Skype: fcwkkj
Phone: 86-591-87868869
QQ: 1043098682
Technical forum: http://wecon.freeforums.net/

1. Installation Instruction

Before the installation make sure that the PLC host and the equipment connected to BD module have been powered off. Please install the BD module in the corresponding position of the PLC, and lock the four standard screws. In case of the dust interference, please cover BD right part by PLC's cover.

Caution:

1) Install the board firmly on the PLC. Poor contact may cause malfunction.
2) The suggested tightening torque is 0.3-0.6 N.m.

Warning:

1) Disconnect the power supply before installing/removing the board and wiring in case of electric shock or product damage.
2) The After completing the installation and wiring, do not replace the PLC top cover before turning on the power.

2. Terminal description

2.1 Appearance

The expansion area number of PLC

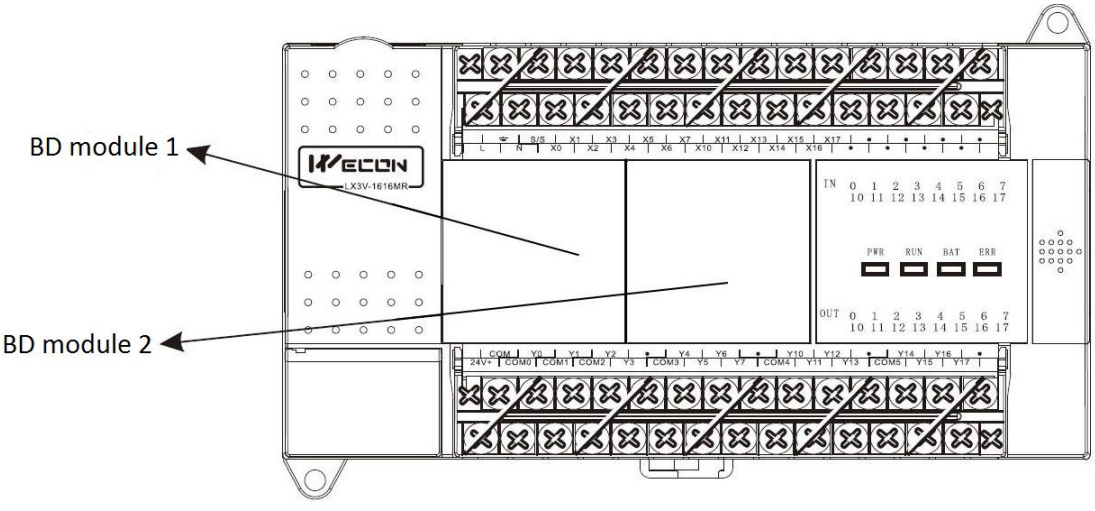


Figure 2-1

This CAN BD module has been installed in BD module 2, as the Figure 2-2 shows.

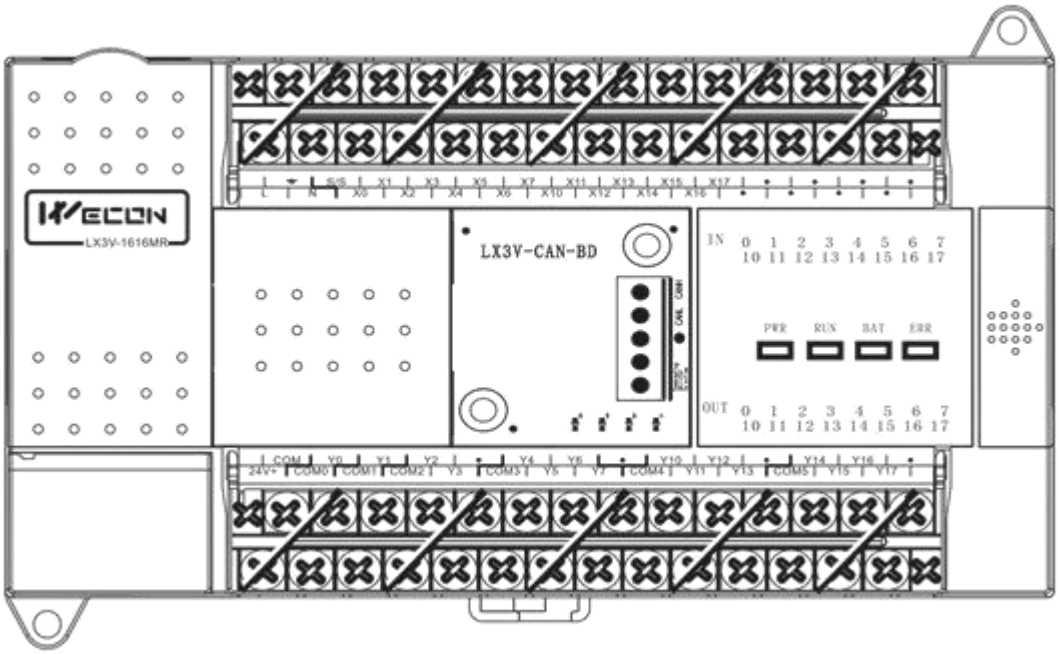


Figure 2-2

2.2 Terminal

The terminal as Figure 2-3 shows.

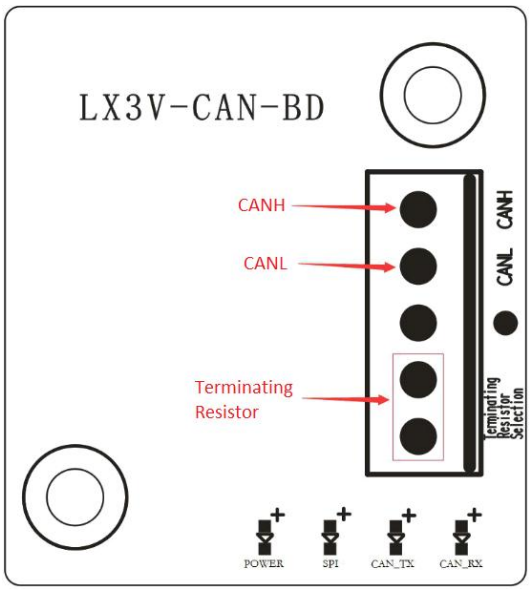


Figure 2-3

There are five ports in Terminal from top to bottom.

- 1：CANH
- 2：CANL

- 3：NC
- 4/5：Terminating resistor (short circuit is ok)

2.3 LED instruction

The lamps are power lamp, PLC communication lamp, CAN sends lamp, CAN receives lamp from left to right.

- Power lamp: power lamp will be on all the time when PLC is power on.
- PLC communication lamp: the speed of blink is depended on the number of the communication to PLC.
- CAN sends lamp: the lamp state is inverted display when CAN sends the messages (if the lamp is on/off all the time, which means that without sending any data and sending frequency can only see the constant by our eyes).
- CAN receives lamp: the lamp state is inverted display when CAN receives the messages (if the lamp is on/off all the time, which means that without receiving any data and receiving frequency can only see the constant by our eyes).

3. CANopen protocol introduction

3.1 About CANopen protocol

The CAN (controller area network) fieldbus only defines the physical layer and the data link layer. (See the ISO11898 standard.) It does not define the application layer. In the practical design, the physical layer and the data link layer are realized by the hardware. The CAN fieldbus itself is not complete. It needs the superior protocol to define the use of 11/29-bit identifier and that of 8-byte data.

The CANopen protocol is the superior protocol base on CAN. It is one of the protocols defined and maintained by CiA (CAN-in-Automation). It is developed on the basis of the CAL (CAN application layer) protocol, using a subset of the CAL communication and service protocols.

The CANopen protocol covers the application layer and the communication profile (CiA DS301). It also covers a framework for programmable devices (CiA 302), the

recommendations for cables and connectors (CiA 303-1), In the OSI model, the relation between the CAN standard and the CANopen protocol is as Figure 3-1:

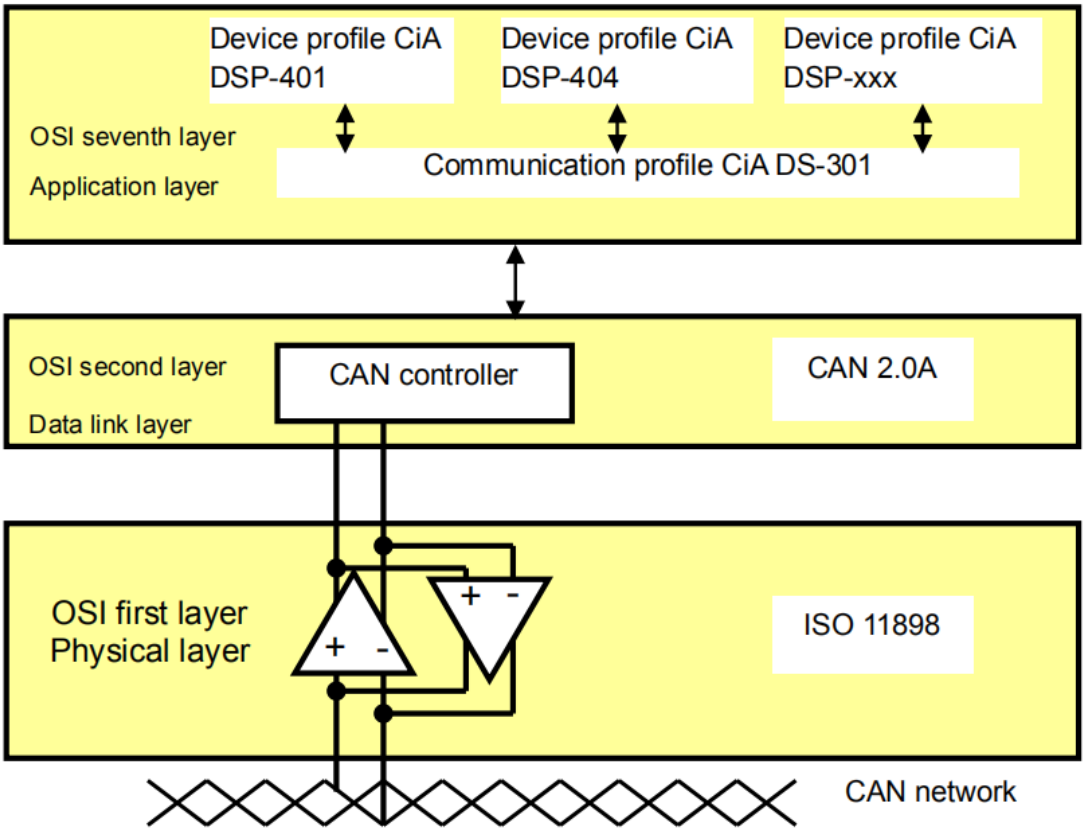


Figure 3-1

CANopen uses an object-based way to define a standard device. Every device is represented by a set of objects, and can be visited by the network. The model of the CANopen device is illustrated below. As the figure below shows, the object dictionary is the interface between the communication program and the superior application program. The core concept of CANopen is the device object dictionary (OD). It is an orderly object set. Every object adopts a 16-bit index for addressing. In order allow the visit to the single element in the data structure, it also defines, an 8-bit sub-index. Every node in the CANopen network has an object dictionary. The object dictionary includes the parameters which describe the device and the network behavior. The object dictionary of a node is described in the electronic data sheet (EDS).

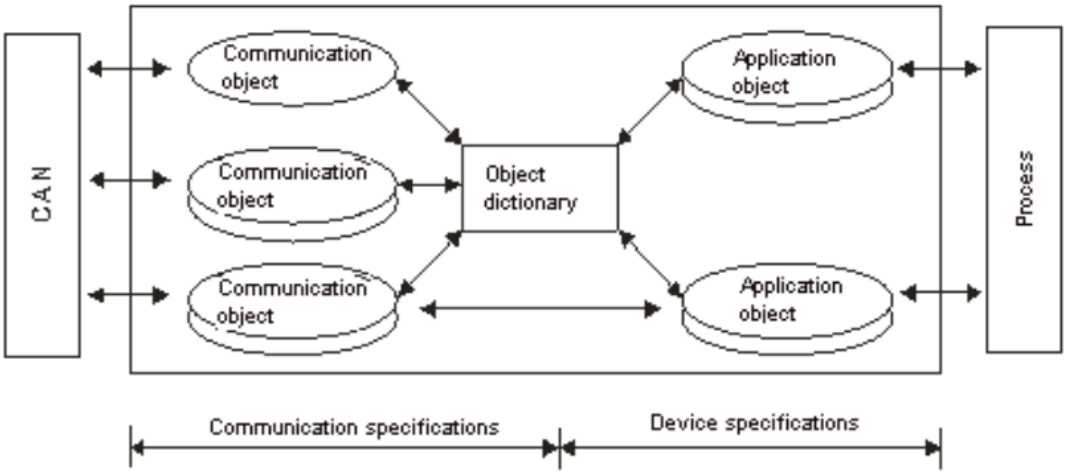
3.2 The object dictionary

The core concept of CANopen is the device object dictionary (OD). It is an orderly

object set. Every object adopts a 16-bit index for addressing. In order allow the visit to the single element in the data structure, it also defines, an 8-bit sub-index. Every node in the CANopen network has an object dictionary. The object dictionary includes the parameters which describe the device and the network behavior. The object dictionary of a node is described in the electronic data sheet (EDS).

The items in the CANopen object dictionary are described by a series of sub-protocols. The sub-protocol describes each function in the object dictionary its function, name, index, sub-index, data type, and whether the object is required, read and write attributes, etc., so as to ensure compatibility with devices of the same type from different vendors.

The core description sub-protocol of the CANopen protocol is DS301, which includes the CANopen protocol application layer and communication structure description. Others complement and extend the DS301 protocol. A CANopen device sub-protocol is drafted in different application industries. The sub-protocol number is generally DS4xx, such as DS402 for motor control.



3.3 CANopen COB-ID

In order to reduce the SCADA workload of simple networks, CANopen defines a assignment chart which is default identifier. In the pre-defined connection settings, the identifier structure BIT10-BIT7 with 11 bits is defined as the function code to distinguish the communication object type of the message. BIT6-BIT0 is defined as a node number (NODE-ID) and it cannot be 0, Details as following:

The broadcast object in the predefined connection setting

Object	Function code	COB-ID
NMT	0000	0
SYNC	0001	128(80h)
Time stamp	0010	256(100h)

The corresponding object in the predefined connection setting

Object	Function code	COB-ID
EMCY	0001	129(81h) – 255(FFh)
PDO1(TX)	0011	385(181h) – 511(1FFh)
PDO1(RX)	0100	513(201h) – 639 (27Fh)
PDO2(TX)	0101	641(281h) – 767 (2FFh)
PDO2(RX)	0110	769 (301h) – 895 (37Fh)
PDO3(TX)	0111	897 (381h) – 1023 (3FFh)
PDO3(RX)	1000	1025 (401h) – 1151 (47Fh)
PDO4(TX)	1001	1153 (481h) – 1279 (4FFh)
PDO4(RX)	1010	1281 (501h) – 1407 (57Fh)
SDO(TX)	1011	1409 (581h) – 1535 (5FFh)
SDO(RX)	1100	1537 (601h) – 1663 (67Fh)
NMT Error control	1110	1793 (701h) – 1919(77Fh)

3.4 The CANopen Communication Object

3.4.1 SDO (Service Data Object)

The SDO message contains the index information and the sub-index information which can be used to position the objects in the object dictionary, and the composite data structure can easily pass the SDO visit.

The SDO can transmit the data in any length. If the data length is more than 4 bytes, the data has to be transmitted by segment. The last segment of the data contains an end flag.

Protocol is confirmed service type: Generate a response for each message (one SDO

requires two IDs). The SDO request and response messages always contain 8 bytes (the meaningless data length is represented in the first byte and the first byte contains the protocol information).

The structures of the SDO requested message and reply message are as follows. The format of the requested message:

COD-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
600 (hex)	Requeste	Object index		Object	Requested data			
+Node-ID	d code	LSB	MSB	sub-index	bit7-0	bit 15-8	bit23-16	bit31-24

The definition of the requested code in the requested message:

Request code (Hex)	Description
23	Writing the 4-byte data
2B	Writing the 2-byte data
2F	Writing the 1-byte data
40	Reading the data
80	Stopping the current SDO function

The format of the reply message:

COD-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
580 (hex)	Reply	Object index		Object	Reply data			
+Node-ID	code	LSB	MSB	sub-index	bit7-0	bit 15-8	bit23-16	bit31-24

The definition of the reply code in the reply message:

Request code (Hex)	Description
43	Reading the 4-byte data
4B	Reading the 2-byte data
4F	Reading the 1-byte data
60	Writing the 1/2/4-byte data
80	Stopping the SDO function

3.4.2 PDO (process data object)

The PDO is described by means of the “producer/consumer mode”. The data is transmitted from one producer to one or many consumers. The data which can be transmitted are limited to 1-byte data to 8-byte data. (For example, a PDO can

transmit up to 64 digital I/O values, or 4 16-bit AD values.)

There is no protocol on PDO communication. The PDO data content is only defined by its CAN ID, assuming that the producer and consumer know the data content of this PDO.

Every PDO is described by two objects in the object dictionary

- The PDO communication parameters: The COB-ID which will be used by PDO, the transmission type, the prohibition time, and the cycle of the counter.
- The PDO mapping parameters: They include the object list in an object dictionary. These objects are mapped into the PDO, including the data length (in bits). To explain the contents of the PDO, the producer and the consumer have to understand the mapping.

The content of the PDO message is pre-set (or configured at network startup)

- Mapping application objects to PDO is described in the device object dictionary. If the device (producer and consumer) supports variable PDO mapping, the PDO mapping parameters can be configured using SDO messages.

The PDO transmission mode: synchronous and asynchronous

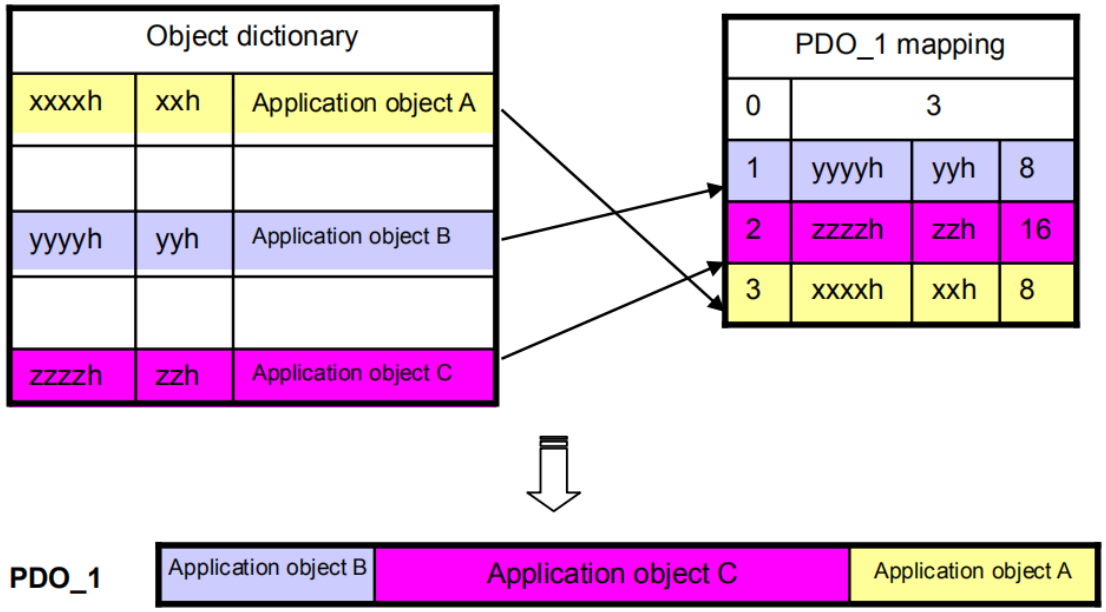
- Synchronous: Synchronous periodic and synchronous non-periodic
- Asynchronous: The PDO is transmitted when the data changes or it is transmitted after a trigger.

Transmission type	**PDO trigger condition (B=both need, O=one or both)**			**POD transmission**
	SYNC	**RTR**	**EVENT**	
0	B	-	B	Synchronous, non-periodic
1-240	O	-	-	Synchronous, periodic
241-251	-	-	-	Reserved
252	B	B	-	Synchronous, after RTR
253	-	O	-	Asynchronous, after RTR
254	-	O	O	Asynchronous Factory-defined event
255	-	O	O	Device sub-protocol specific event
Description: SYNC – receive SYNC-object; RTR – receive remote frame;				

Event – data changes or timer interrupt;
Transmission type: From 1 to 240, this number represents the number of SYNC objects between two PDOs.

Mode 0: The PDO information is transmitted only when the PDO data changes and the synchronous signal comes.
Modes 1~240: One piece of PDO information is transmitted every 1~240 synchronous signals.
Mode 254: The trigger is defined the manufacturer. The definition of the PLC is the same as mode 255.
Mode 255: PDO is transmitted when the data changes, or it is transmitted after a trigger.

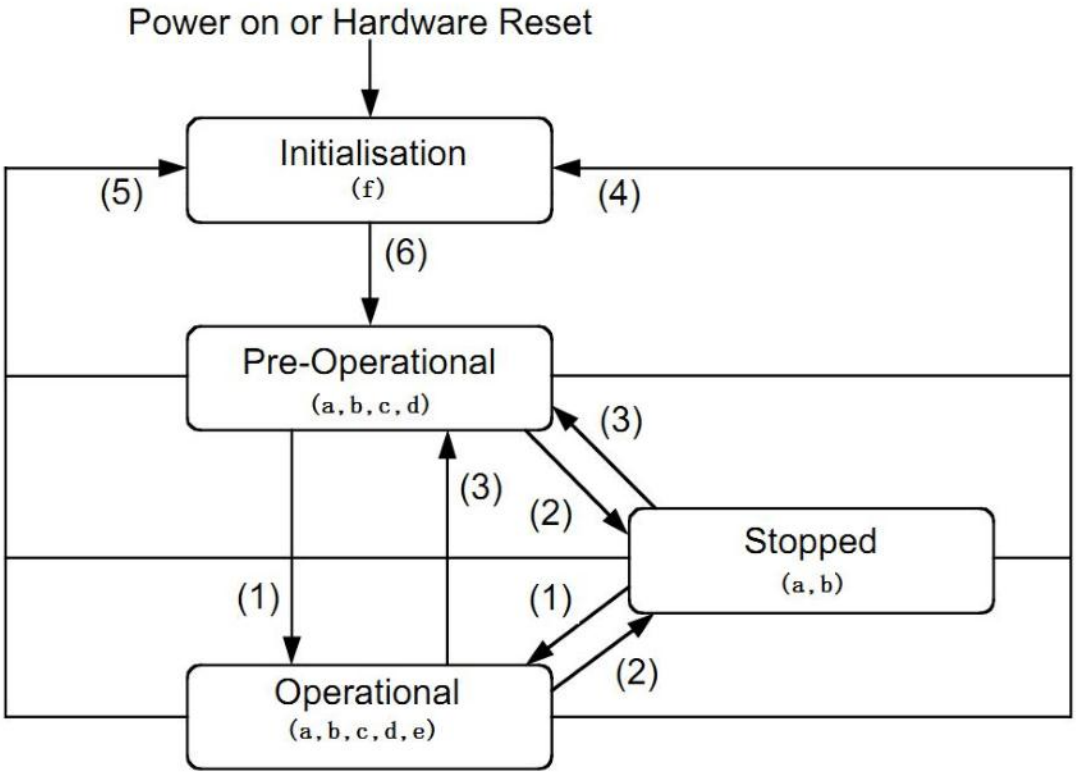
All the data in the PDO has to be mapped from the object dictionary. The following is an example of the PDO mapping.



3.4.3 NMT (network management object)

The CANopen network management conforms to the “master/slave” mode. Only one NMT master exists in the CANopen network, and other nodes are considered slaves. NMT realized three services. They are module control services, error control services, and boot-up services.

Module control services



a. NMT, b. Node Guard, c. SDO, d. Emergency, e. PDO, f. Boot-up

State transition (1-5 initiated by the NMT service), NMT command word (in parentheses)

1: Start_Remote_node (0x01)
2: Stop_Remote_node (0x02)
3: Enter_Pre-operational_State (0x80)
4: Reset_node (0x81)
5: Reset_Communication (0x82)
6: After the device initialization is complete; it automatically enters the pre_operational state and sends a boot_up message.

Node state control service

The format of the control message for the node state:

COB-ID	Byte 0	Byte 1
0	Command specifier (CS)	Slave address (0: Broadcast)

NMT Module Control messages do not need to be acknowledged. When Node-ID=0, all NMT slaves are addressed.

The command specifiers are listed below.

Command specifier	Function

1	Start the remote node
2	Stop the remote node
128 (0x80)	Enter the pre-operation state
129 (0x81)	Reset the application layer
130 (0x82)	Reset the communication

Error control services

The error control service is used to detect the disconnection of the node in the network. The error control services can be classified into two types, i.e. Heartbeat and Node Guarding. The PLC only supports Heartbeat. The Hearbeat producer transmits the Heartbeat message according to the Heartbeat producing time which is set. One or many Heartbeat consumers detect the message transmitted by the Heartbeat producer. If the consumer does not receive the message transmitted by the producer within the timeout period, the CANopen communication is abnormal.

For the format sent by the heartbeat message, the CANID is the same as the node online message is 700 h + Node-ID, and the data is 1 byte as shown in the figure below.

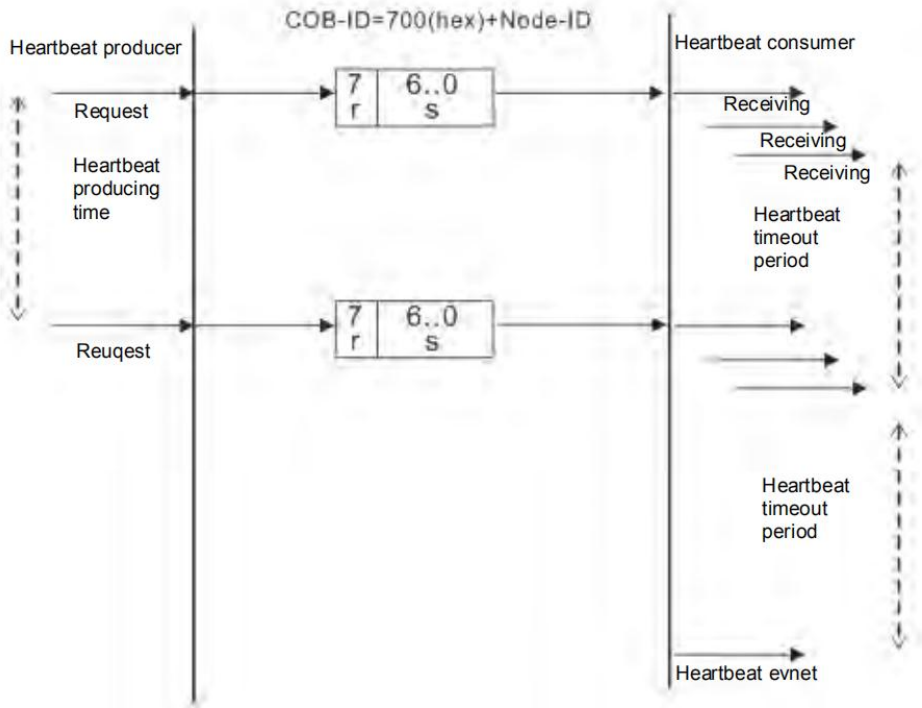


Figure 3-2

Heartbeat producer->consumer

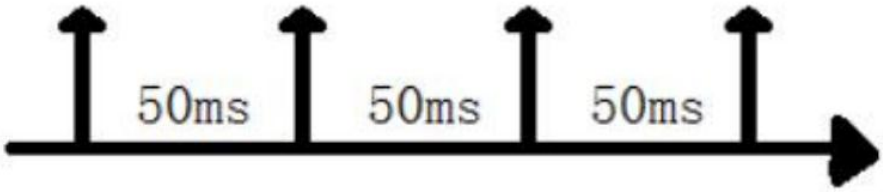
COB-ID	BYTE0
0x700+Node-ID	State

State meaning is as follows:

State	Meaning
0	Boot-up
4	Stopped
5	Operational
127 (0x7F)	Pre-operational

Heartbeat package (fixed once for a period of time, such as 50MS, determined by 0x1017 in the object dictionary)

	COB-ID	BYTE0
<- Slave	0x702	0x05



3.4.4 Other predefined CANopen communication objects (SYNC and EMCY)

SYNC Object (Synchronous object)

The synchronous object is the message broadcasted periodically by the master node in the CANopen network. This object is used to realize the network clock signal. Every device decides whether to use the event and undertake the synchronous communication with other network devices according to its configuration. For example, when controlling the driving device, the devices do not act immediately after they receive the command sent by the master. They do act until they receive the synchronous message. In this way, many devices can act synchronously.

The message object mainly implements synchronous transmission of the entire network, and each node uses the synchronization message as a PDO trigger parameter, so the COB-ID of the synchronization message has a relatively high priority and a short transmission time. 0x80 is generally selected as the CAN-ID of the synchronization message.

Emergency object

The emergency object is used by the CANopen device to indicate an internal error. When an emergency error occurs in the device, the device sent the emergency message (including the emergency error code), and the device enters the error state. After the error is eliminated, the device sends the emergency message, the emergency error code is 0, and the device enters the normal state.

The format of the emergency message:

COD-ID	Byte0-1	Byte 2	Byte 3-7
80 (hex)+Node-ID	Emergency error	Error register	Factory-defined error code

The emergency error code function is as follows:

Emergency error code	Description
00xx	Error reset or No error
10xx	Generic error
20xx	Current
21xx	Current, device input side
22xx	Current, inside the device
23xx	Current, device output side
30xx	Voltage
31xx	Mains voltage
32xx	Voltage inside the device
33xx	Output voltage
40xx	Temperature
41xx	Ambient temperature
42xx	Device temperature
50xx	Device hardware
60xx	Device software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Additional modules
80xx	Monitoring
81xx	Communication
8110	CAN overrun
8120	Error passive

8130	Life guard error or heartbeat error
8140	Recovered from BUS-OFF
82xx	Protocol error
8210	PDO no processed Due to length error
8220	Length exceeded
90xx	External error
F0xx	Additional functions
FFxx	Device specific

The error register functions as follows:

Bit	Error type
0	Generic
1	Current
2	Voltage
3	Temperature
4	Communication
5	Device profile specific
6	Reserved (=0)
7	Manufacturer specific

4. The main features

WECON CAN BD can be master station and slave station.

4.1 When act as Master Station

1) Supporting CANopen standard protocol DS301 V4.02
2) Supporting NMT(Network Management Object) service
3) Supporting NMT state control
   NMT status control can be used to control the status of slaves in a CANopen network
4) Supporting NMT error control
   NMT error control can be sued to monitor whether slave station is dropped. NMT error control is divided to two types: Heartbeat and Node Guarding. Wecon

CAN BD support Heartbeat , not support Node Guarding

5) Supporting PDO (Process Data Object) service:
   PDO message can be used to transmit real-time input and output data.
   Supporting up to 128 RxPDO , Supporting up to 400 bit data
   Supporting up to 128 TxPDO , Supporting up to 400 bit data
   PDO transmit types: Synchronous mode, asynchronous mode
   Supporting PDO service RTR mode data in PLC Ladder program
6) Supporting SDO (Service Data Object) service client terminal
   SDO can be used to read/write the parameters in slave station or modify them.
   Supporting SDO standard transmit mode.
   Supporting auto DCF function, Slave station power on configuration via SDO
   Supporting use SDO service read/write slave station data in PLC Ladder program
7) Supporting read emergency message from slave station
   The emergency messages from slave station can be use for read slave station errors or alarm messages.
   Support reading emergency message from PLC program.
8) Supporting SYNC Object service
   Synchronous messages can be used to synchronize multiple devices (up to one sync generator in the network) Supported mapping data types:

Storage Space	Data format
8-Bit	SINT USINT BYTE
16-Bit	INT UINT WORD
32-Bit	DINT UDINT REAL DWORD
64-Bit	LINT ULINT LREAL LWORD

4.2 When act as slave station

1) Supporting CANopen standard protocol DS301 V4.02
2) Supporting NMT(Network Management Object) service
   The states is controled by master station
   Supporting Heartbeat error control, not support Node Guarding error control.
3) Supporting PDO (Process Data Object) service.
   PDO message can be used to transmit real-time input and output data.
   Supporting up to 8 RxPDO , Supporting up to 8 TxPDO ,
   PDO transmit types: Synchronous mode, asynchronous mode
4) Supporting SYNC Object service
   Synchronous messages can be used to synchronize multiple devices (up to one

sync generator in the network)

5) Supported mapping data types:
   Master station access ability
6) Supporting read emergency message service.
   It can be set via PLC ladder.

5. Instruction description

5.1 Connection number description

BD module 1	Connection No.	BD module 2	Connection No.
CPAVL	100	CPAVL	101

5.2 CPAVL instruction

Name	Function	bits	Pulse type	Instruction format	Step
CPAVL	Communication port setting	16	No	CPAVL S D M	11

Operand	Bit device				Word device										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	V	Z
S													√		
D			√												
M					√	√									

S: The starting address of the D device parameter table. (For BD module 1, the default value of S is D4500, for BD module 2, it is D5000)

D: The starting address of “M” device parameter table (not used)

M: The connection No. (For BD module 1, the connection No. is K100, for BD module 2, it is K101)

6. D device description

6.1 The usage of the D device parameter table

The parameter table starts from the D address:
0-31 are used as a request mapping registers. (SD0, PD0, NMT , EMERGENCY)
32-63 are used as a response mapping registers.

Offset (0~31)	High byte	Low byte	Offset (32~63)	High byte	Low byte
0	Request ID	Command code	32	Request ID	Command code/reserve
1	Node address	Data length	33	Node address	Data length
2	Reserve	Reserve	34	Reserve	Reserve
3	type	Reserve	35	type	Reserve
4	Data1	Data0	36	Data1	Data0
5	Data3	Data2	37	Data3	Data2
6	Data5	Data4	38	Data5	Data4
7	Data7	Data6	39	Data7	Data6
8	Reserve		40	Reserve	
…			…		
31			63		

Command code: Command code is fixed as 1.
Request ID: Each request must be assigned a request ID which ranges from 0-255. When changing the request ID, the request will be sent. When the request ID is matched, the reply will be sent.
Data length: The number of bytes of the request instruction is 4-31. The number of bytes of reply instruction is 36-63.
Node address: The node address of the target device in the CANopen network.
Type:

Service	Request type(hex)	Reply type(hex)
SDO READ	1	41
SDO WRITE	2	42
NMT	3	43

PDO RTR	4	44
EMCY inquire	5	45
EMCY set	6	46
EMCY erase	7	47

State code:

State code(Hex)	Description
0	No error
1	SDO message is sent successfully
2	Error message-Termination code (Refer to the error code of the SDO response message)
3	ERROR-BD fails to call SDO read
4	ERROR-SDO fails to read
5	ERROR-BD fails to call SDO write
6	ERROR-SDO fails to write
7	ERROR-Command data length error
8	ERROR-node error
9	SD0 unknown error
A	PDO number error / PDO not supported
B	Master/slave error (only master/slave supports this command)
C	The sending queue of CAN is full
09~FF	Reserved

6.1.1 The SDO format is as below (only valid when serves as the master station):

	SD0		
Offset of request register	**Request**	**Respond**	**Offset of response register**
0	Command code	State code	32
	Request ID	Request ID	
1	Data length	Data length	33
	Node address	Node address	
2	reserved	reserved	34
	reserved	reserved	

3	reserved	reserved	35
	type	type	
4	Index L	Index L	36
	Index H	Index H	
5	Subindex	Subindex	37
	reserved	reserved	
6	Data 0	Data 0	38
	Data 1	Data 1	
7	Data 2	Data 2	39
	Data 3	Data 3	

Command code: fixed as H1

Request ID: Each request must be assigned with a request ID (range:0-255). A request occurs when the request ID changes, and the response occurs when the request ID in the reply coincide with the request ID of the request.

Data length: the length is fixed as H4 when reading or requesting, when writing, the length (max length is H8) is the sum of H4 and the the byte number of the sub-index. When writing, if the index and sub-index data type is word-type data, the data length is H6; if the index and sub-index data type is byte-type data, the data length is H5.

Type: The SDO read request type is H1, the SDO read response type is H41, the SDO write request type is H2, and the SDO write response type is H42.

SDO read node 2 object Index 1000H sub-index 0H		
Request	**High byte**	**Low byte**
0	Request ID H01	Command code H01
1	Node address H02	Data length H04
2	Reserved H00	Reserved H00
3	Type H01	Type H00
4	Index high byte H10	Index LOW byte H00
5	Reserved H00	Subindex H00
Response	High byte	Low byte
32	Request ID H01	State code H01
33	Node address H02	Data length H08
34	Reserved H00	Reserved H00
35	Type H41	Reserved H00
36	Index high byte H10	Index LOW byte H00

37	Reserved H00	Subindex H00
38	Data1 H01	Data0 H02
39	Data3 H00	Data2 H02

SDO write node 2 object Index 2000H sub-index 0H value H45		
request	**High byte**	**Low byte**
0	Request ID H01	Command code H01
1	Node address H02	Data length H05
2	Reserved H00	Reserved H00
3	Type H01	reserved H00
4	Index high byte H20	Index LOW byte H00
5	Reserved H00	Subindex H00
5	Reserved H00	Value H45
response	High byte	Low byte
32	Request ID H01	State code H01
33	Node address H02	Data length H04
34	Reserved H00	Reserved H00
35	Type H42	Reserved H00
36	Index high byte H10	Index LOW byte H00
37	Reserved H00	Subindex H00

6.1.2 NMT format is as below (only valid when plc serves as master)

Request register offset	**NMT**		**Response register offset**
	Request	**Response**	
0	Command code	State code	32
	Request ID	Request ID	
1	Data length	Data length	33
	Node address	Node address	
2	Reserved	Reserved	34
	Reserved	Reserved	
3	Reserved	Reserved	35
	type	type	
4	NMT service code		36
	null		

Command code: fixed as H1

Request ID: Each request must be assigned with a request ID (range:0-255). A request occurs when the request ID changes, and the response occurs when the request ID in the reply coincide with the request ID of the request.

Data length: the length is fixed as H1 when requesting, it is H0 when responsing.

Type: The request type of NMT is H3, the response type is H43.

NMT service code:

NMT service code (HEX)	Description
1	Start the remote node
2	Stop the remote node
80	Enter pre-operation state
81	Application reset
82	Communication reset

NMT start node 2		
Request	**High byte**	**Low byte**
0	Request ID H01	Command code H01
1	Node address H02	Data length H01
2	Reserved H00	Reserved H00
3	Type H03	reserved H00
4	Reserved H00	NMT service code H01
Response	High byte	Low byte
32	Request ID H01	State code H00
33	Node address H02	Data length H00
34	H00	H00
35	Type H43	H00

6.1.3 PDO RTR format is as below (Not recommended to use)

Request register offset	PDO		Response register offset
	Request	Response	
0	Command code	State code	32
	Request ID	Request ID	
1	Data length	Data length	33

	Reserved	Reserved	
2	Reserved	Reserved	34
	Reserved	Reserved	
3	Reserved	Reserved	35
	type	type	
4	PDO No. L		36
	PDO No. H		

Command code: fixed as H1

Request ID: Each request must be assigned with a request ID (range:0-255). A request occurs when the request ID changes, and the response occurs when the request ID in the reply coincide with the request ID of the request.

Data length: the length is fixed as H2 when requesting, it is H0 when responsing.

Type: The request type of PDO is H4, the response type is H44.

PDO No. : the PDO No. starts from 0, for example 1400H's PDO NO. is 0, 1405H is 5.

Request target PDO response message corresponding to the second PDO		
Request	**High byte**	**Low byte**
0	Request ID H01	Command code H01
1	Reserved H00	Data length H01
2	Reserved H00	Reserved H00
3	Type H04	reserved H00
4	High byte H00 of PDO NO.	LOW byte H02 of PDO NO.
Response	High byte	Low byte
32	Request ID H01	State code H00
33	Reserved H00	Reserved H00
34	H00	H00
35	Type H44	H00

6.1.4 EMCY inquire format is as below (only valid when plc serves as master):

Request register offset	**EMCY**		**Response register offset**
	Request	**Response**	
0	Command code	State code	32

	Request ID	Request ID	
1	Data length	Data length	33
	Node address	Node address	
2	Reserved	Reserved	34
	Reserved	Reserved	
3	Reserved	Reserved	35
	type	type	
4	Reserved	Error code L	36
	Reserved	Error code H	
5	Reserved	Error code register	37
	Reserved	EMCY1	
6	Reserved	EMCY2	38
	Reserved	EMCY3	
7	Reserved	EMCY4	39
	Reserved	EMCY5	

Command code: fixed as H1

Request ID: Each request must be assigned with a request ID (range:0-255). A request occurs when the request ID changes, and the response occurs when the request ID in the reply coincide with the request ID of the request.

Data length: the length is fixed as H0 when requesting, it is H8 when responsing.

Type: The request type is H5, the response type is H45.

Error code: as shown above

Error code register: as shown above

EMCY: the error that is customized by the manufacturer.

EMCY information of the node address 2		
Request	**High byte**	**Low byte**
0	Request ID H01	Command code H01
1	Node address H02	Data length H00
2	Reserved H00	reserved H00
3	Type H05	reserved H00
Response	High byte	Low byte
32	Request ID H01	State code H00
33	Node address H02	Data length H08
34	Reserved H00	Reserved H00
35	Type H45	Reserved H00

36	Error code high byte	Error code low byte H00
37	EMCY1 H00	Error code register H00
38	EMCY3 H00	EMCY2 H00
39	EMCY5 H00	EMCY4 H00

6.1.5 EMCY setting format is as below (only valid when plc serves as slave):

Request register offset	EMCY		Response register offset
	Request	Response	
0	Command code	State code	32
	Request ID	Request ID	
1	Data length	Data length	33
	Node address	Node address	
2	Reserved	Reserved	34
	Reserved	Reserved	
3	Reserved	Reserved	35
	type	type	
4	Error code L	reserved	36
	Error code H	reserved	
5	Error code register	reserved	37
	EMCY1	reserved	
6	EMCY2	reserved	38
	EMCY3	reserved	
7	EMCY4	reserved	39
	EMCY5	reserved	

Command code: fixed as H1

Request ID: Each request must be assigned with a request ID (range:0-255). A request occurs when the request ID changes, and the response occurs when the request ID in the reply coincide with the request ID of the request.

Data length: the length is fixed as H8 when requesting, it is H0 when responsing.

Type: The request type is H6, the response type is H46.

Error code: cannot be repeated

Error code register: as shown above

EMCY: the error that is customized by the manufacturer.

Setting the EMCY information for this node(current alarm)

Request	High byte	Low byte
0	Request ID H01	Command code H00
1	Reserved H00	Data length H08
2	Reserved H00	reserved H00
3	Type H06	reserved H00
4	Error code high byte H20	Error code low byte H00
5	EMCY1 H00	Error code register H02
6	EMCY3 H00	EMCY2 H00
7	EMCY5 H00	EMCY4 H00
Response	High byte	Low byte
32	Request ID H01	command code H00
33	Reserved H00	Data length H00
34	Reserved H00	Reserved H00
35	Type H46	Reserved H00

6.1.6 EMCY erasing format is as below (only valid when plc serves as slave):

Request register offset	EMCY		Response register offset
	Request	Response	
0	Command code	State code	32
	Request ID	Request ID	
1	Data length	Data length	33
	Node address	Node address	
2	Reserved	Reserved	34
	Reserved	Reserved	
3	Reserved	Reserved	35
	Type	Type	
4	Error code L	reserved	36
	Error code H	reserved	

Command code: fixed as H1

Request ID: Each request must be assigned with a request ID (range:0-255). A request occurs when the request ID changes, and the response occurs when the request ID in the reply coincide with the request ID of the request.

Data length: the length is fixed as H2 when requesting, it is H0 when responsing.

Type: The request type is H7, the response type is H47.

Error code: One of the set EMCY error codes to be eliminated, set to 0 means to eliminate all EMCY.

Deleting the EMCY information for this node(current alarm)		
Request	**High byte**	**Low byte**
0	Request ID H01	Command code H00
1	Reserved H00	Data length H02
2	Reserved H00	reserved H00
3	Type H07	reserved H00
4	Error code high byte H20	Error code low byte H00
Response	High byte	Low byte
32	Request ID H01	command code H00
33	Reserved H00	Data length H00
34	Reserved H00	Reserved H00
35	Type H47	Reserved H00

6.2 PDO address mapping description

Starting from the start address of the D device parameter table specified by the CPAVL instruction:

64-263 are the PDO output mapping data

264-463 are the PDO input mapping data

TPDO mapping	RPDO mapping
64	264
…	…
263	463

6.3 State of slave node address

Starting from the start address of the D device parameter table specified by the CPAVL instruction:

470~477 are the state of the slave device (only valid when plc serves as slave)

Slave state	High byte	Low byte
470	Node 2	Node 1
471	Node 4	Node 3
472	Node 6	Node 5

473	Node 8	Node 7
474	Node 10	Node 9
475	Node 12	Node 11
476	Node 14	Node 13
477	Node 16	Node 15

Starting from the start address of the D device parameter table specified by the CPAVL instruction: 478~491 are the information of the can bd.

6.4 CAN BD module information

Module information	
478	BD module version
479	Baud rate
480	Node id
481	The number of the slave device
482	Master-slave condition
483	Bd board type
484	SPI protocol version
485	Can address version
486	User interface version of can
487	SPI error
488	Reserved
489	Reserved
490	Reserved
491	Reserved
492	Reserved
493	Reserved

SPI error is as below:

0xE0	SPI time out
0xE1	Data length error
0xE2	Command error
0xE3	Check error

7. Example

It uses Delta ASDA-A2 servo as example to set the communication

7.1 Servo configuration

First configure the servo, select CAN communication, configure the baud rate 1M, node ID=2.
(See the servo manual for the configuration method)

7.2 CANopen configuration and download

Operating procedures

1) Open the upper computer CANopenTool software, the new master station is as Figure 7-1 shows:

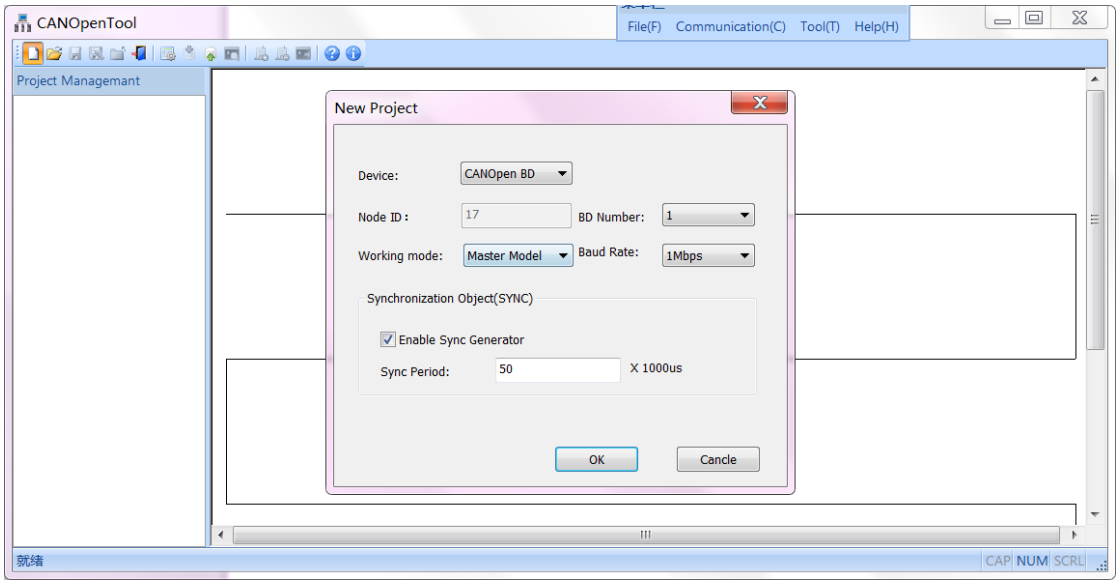


Figure 7-1

2) Find the “ASDA-A2 Drive” in “Device list”, as Figure 7-2 shows:

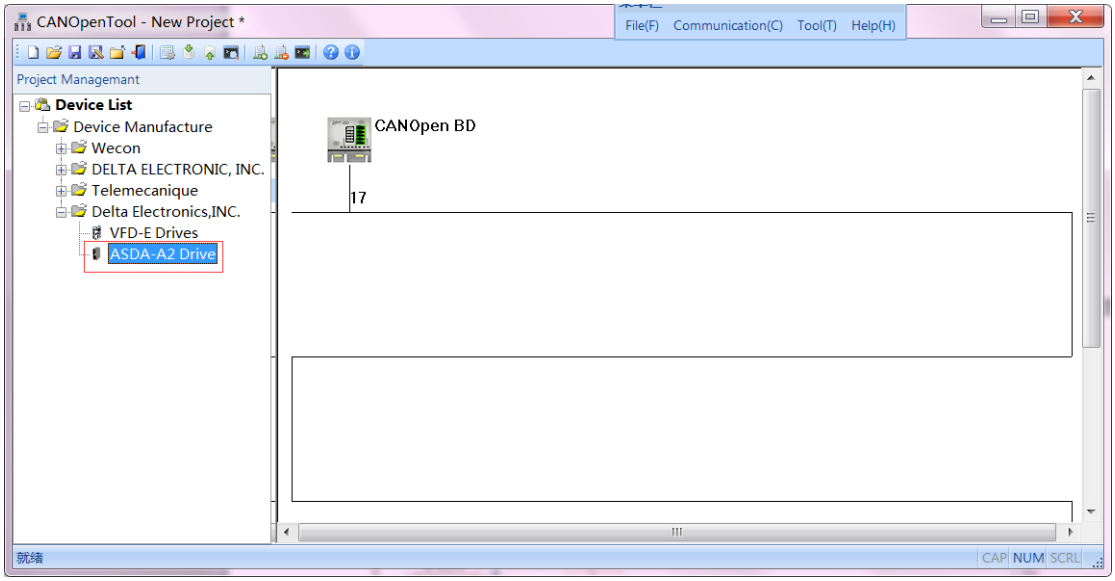


Figure 7-2

3) Double click “ASDA-A2 Drive” in “Device list “for selecting it;

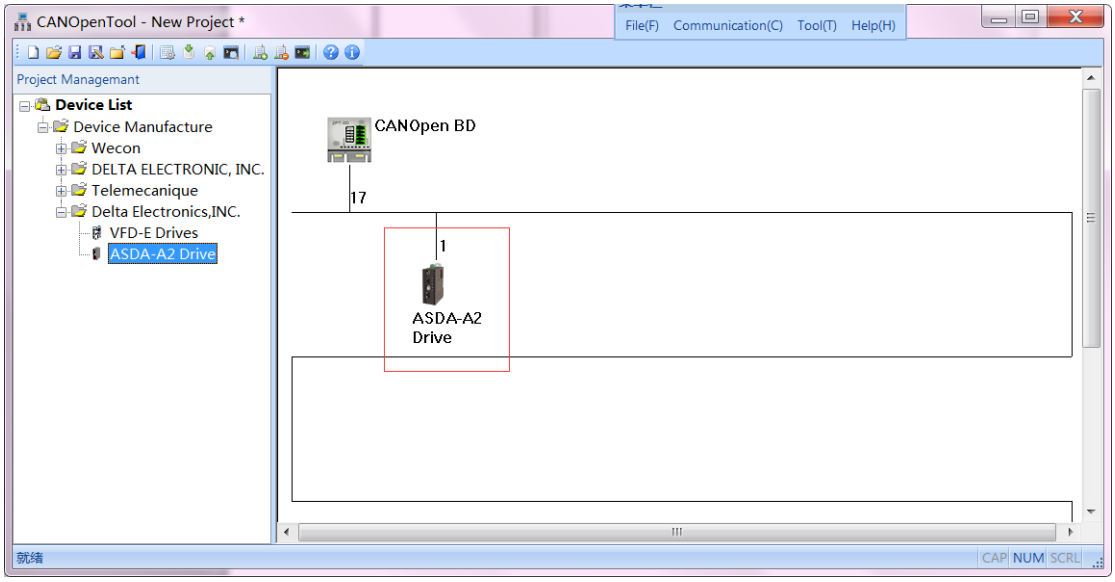


Figure 7-3

4) Double click “ASDA-A2” in main window to open the “slave setting” windows;
5) Modify the “Node number”, “configuration heartbeat time” and “monitoring time” as Figure 7-4 shows

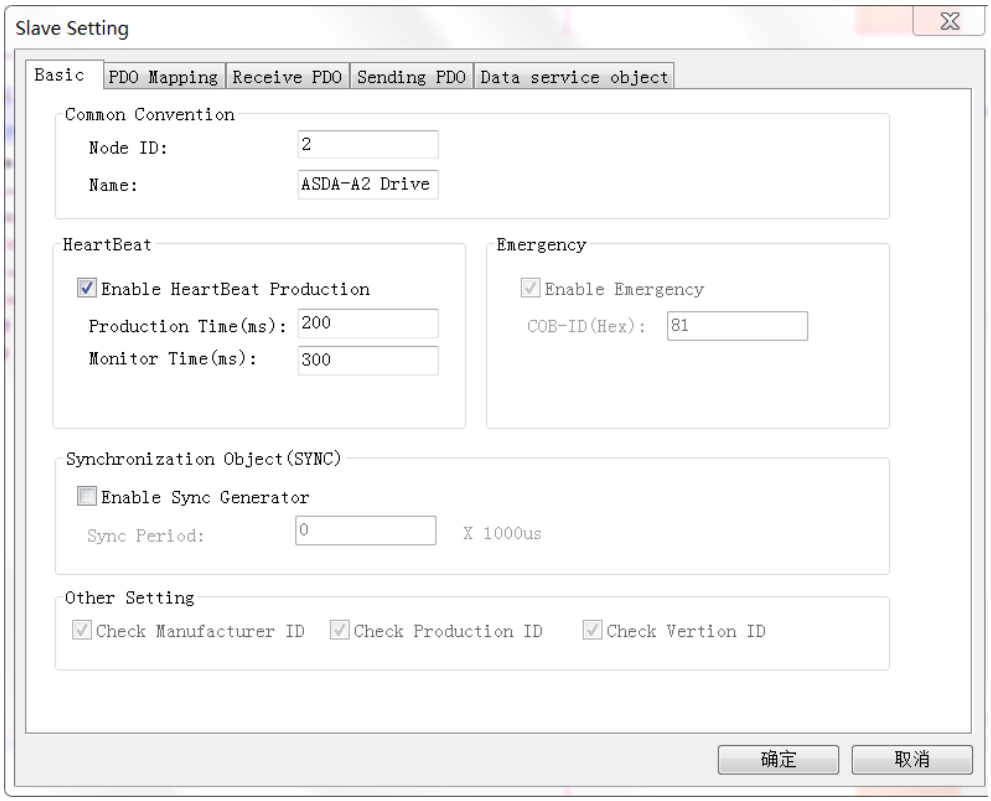


Figure 7-4

6) Configure the slave to receive the PDO and add the mapping parameter index as Figure 7-5 shows;

Add the object as follows:

Added RPDO	**Setting value**	**Description**
P3-06	0x3FFF	Input contact source control switch -> internal register
P4-07	Controlled by M	Make M corresponding to CN1 pin for control
P6-03	100000	Path-1 -> 100000(position)
P6-05	200000	Path-2 -> 200000(position)
Added TPDO	**Setting value**	**Description**
Velocity actual value	##	Velocity actual value
Position actual value	##	Position actual value
Statusword	Mapping to output Y	statusword

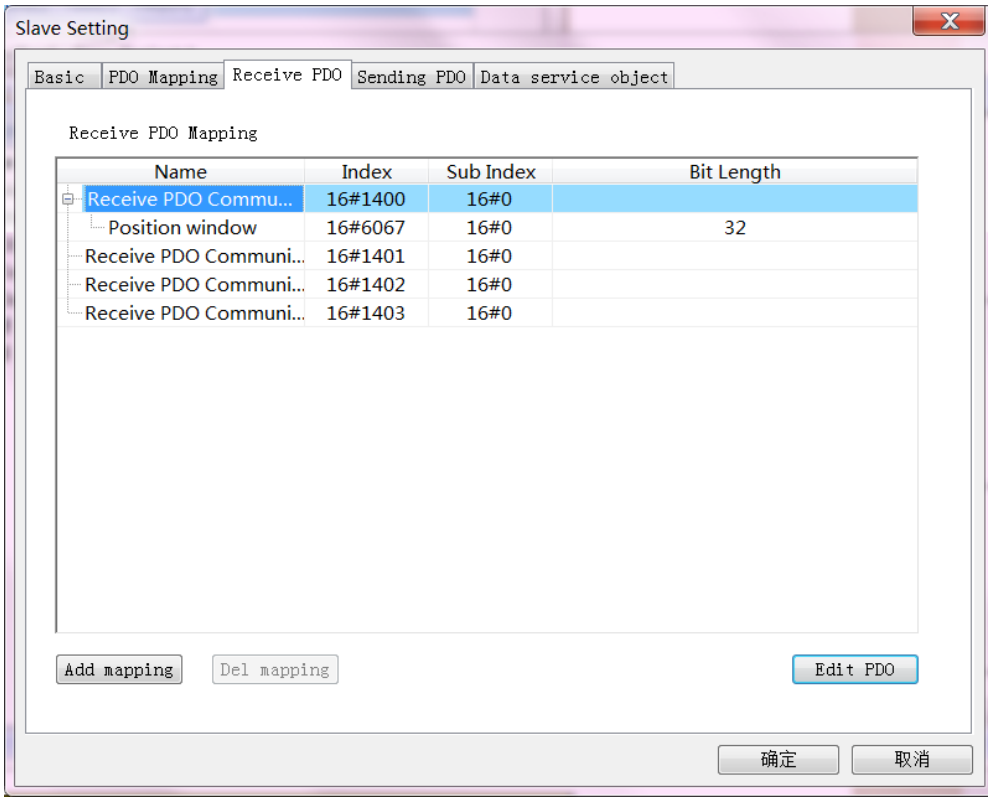


Figure 7-5

7) Select to enable the configured PDO

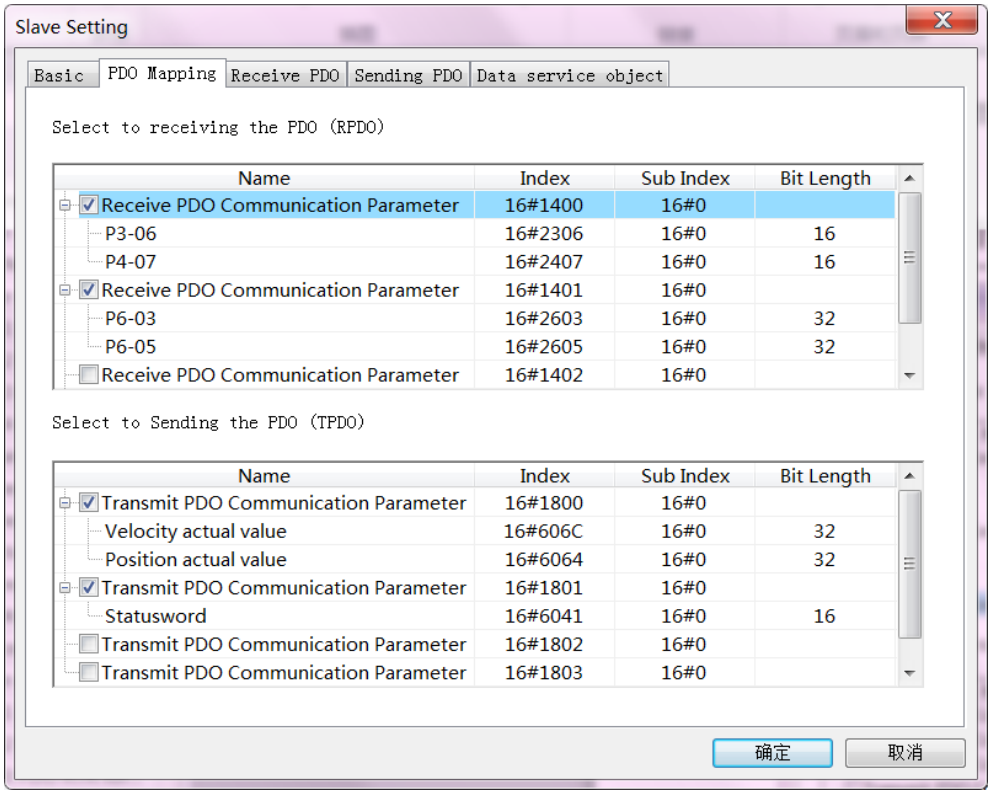


Figure 7-6

8) Select New Service Data Object

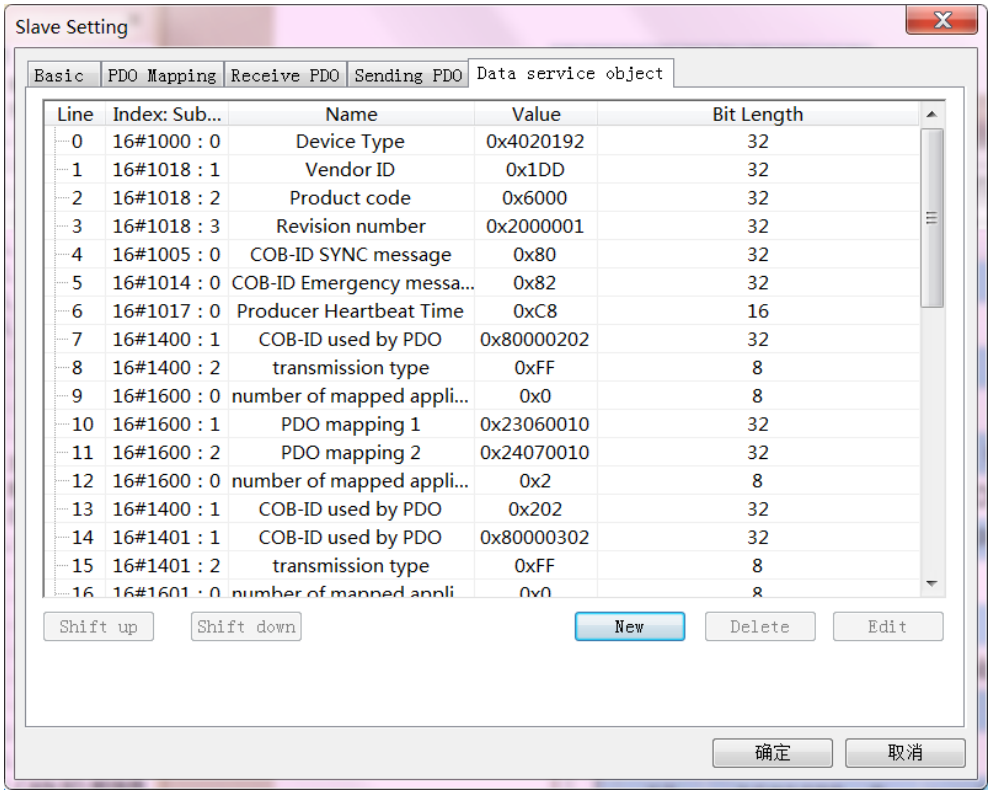


Figure 7-7

9) Click “New” to add a new object

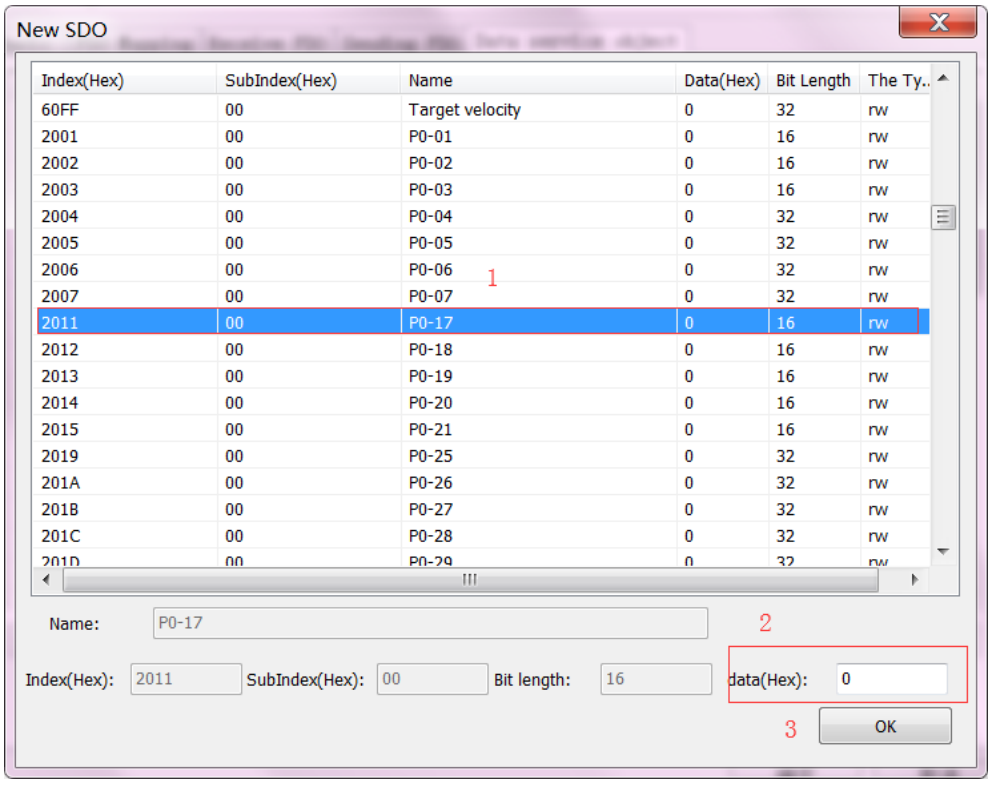


Figure 7-8

10) Please follow the steps as Figure 7-8 shows;

11) Set the value as Figure 7-9 shows;

37	16#1801 : 1	COB-ID used by PDO	0x282	32
38	16#2011 : 0	P0-17	0x0	16
39	16#2012 : 0	P0-18	0x1	16
40	16#220A : 0	P2-10	0x101	16
41	16#220B : 0	P2-11	0x108	16
42	16#220C : 0	P2-12	0x111	16
43	16#220D : 0	P2-13	0x112	16
44	16#2211 : 0	P2-17	0x137	16
45	16#2515 : 0	P5-21	0x12C	16
46	16#253D : 0	P5-61	0x64	16
47	16#2600 : 0	P6-00	0x0	32
48	16#2601 : 0	P6-01	0x0	32
49	16#2602 : 0	P6-02	0x32	32
50	16#2604 : 0	P6-04	0x32	32

Figure 7-9

12) Select the master station(double click “CANOpen BD” in main window);

13) Configure the baud rate synchronization;

14) Map the slave PDO to the PLC register;

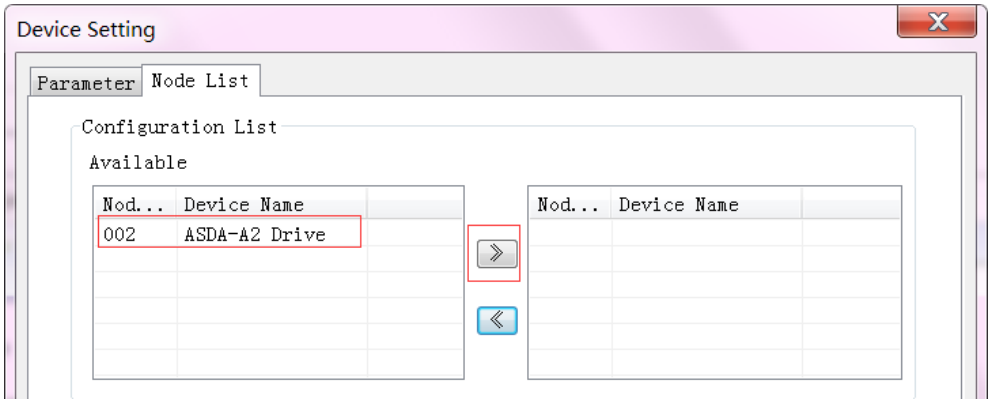


Figure 7-10

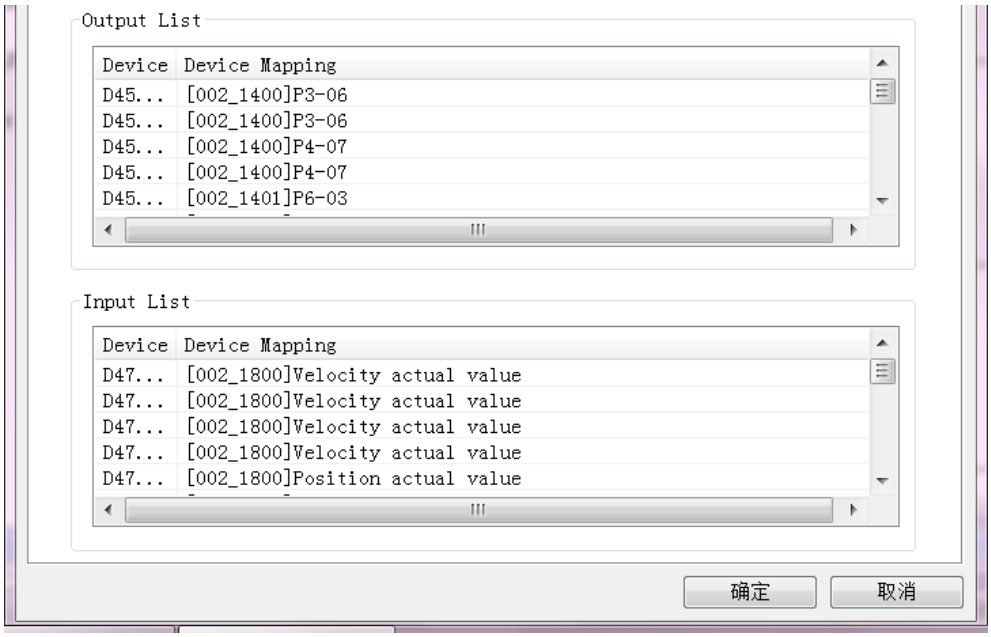


Figure 7-11

15) Click “Download” to start download

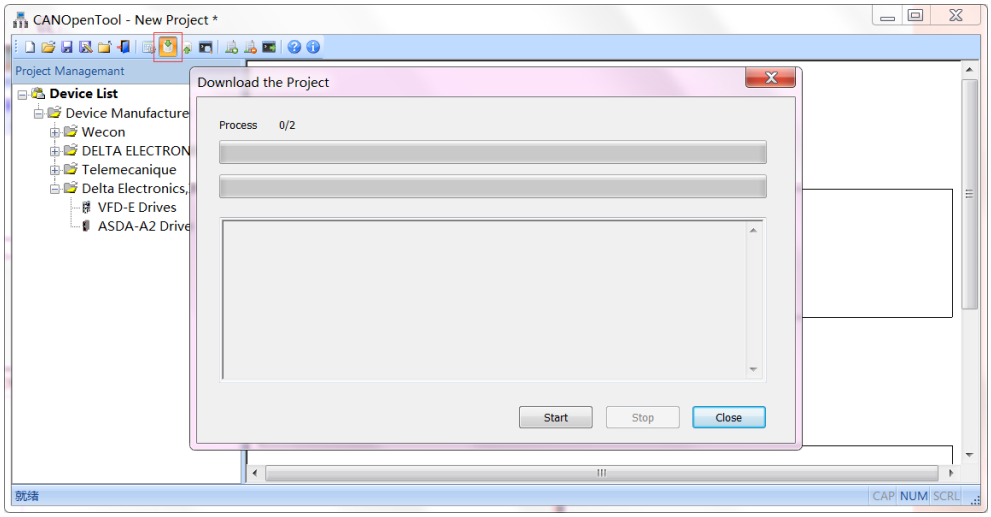
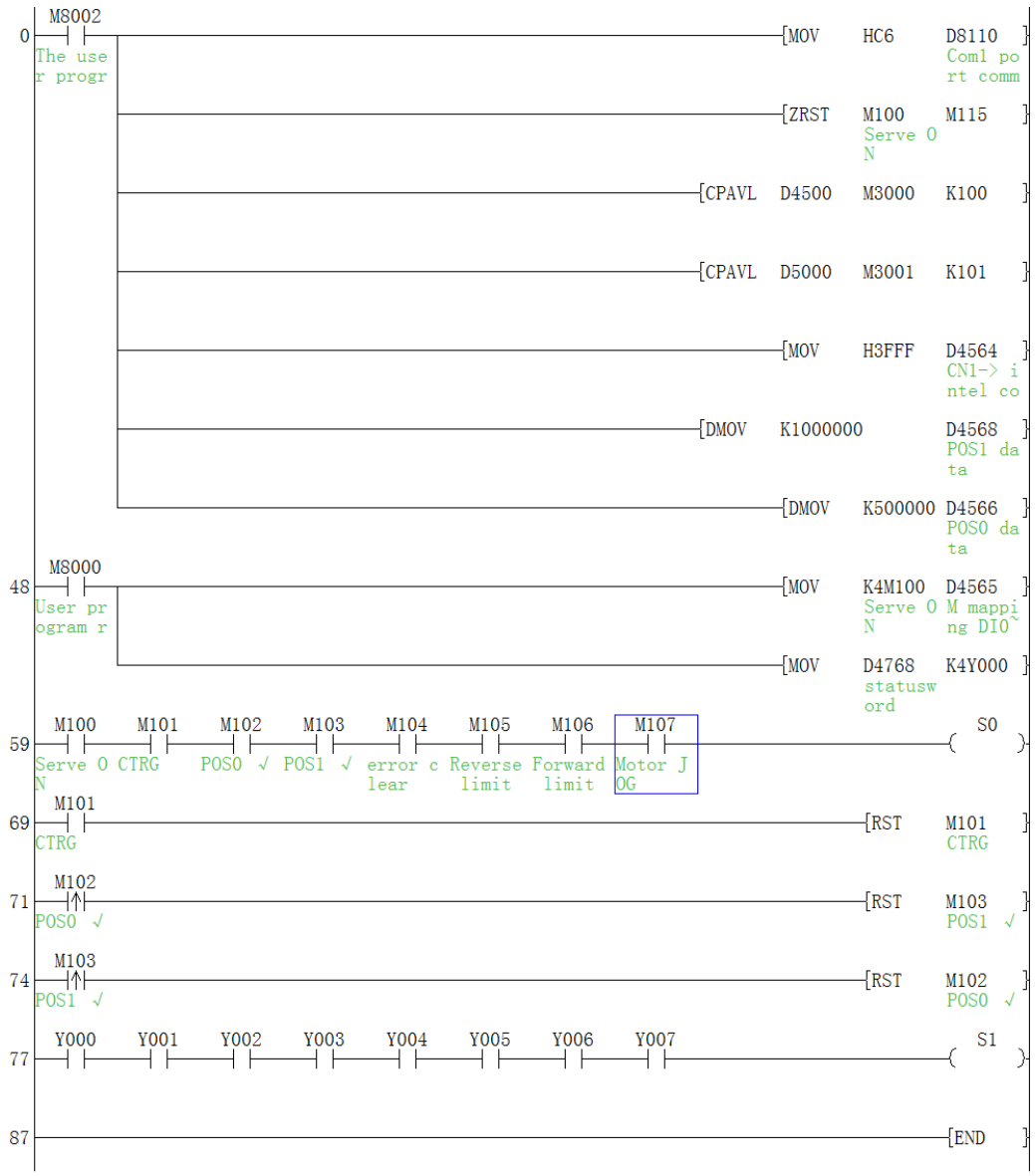


Figure 7-12

7.3 PLC and HMI program

7.3.1 PLC ladder

Mainly complete the BD power-on configuration and servo control, monitoring function facilitation (internal M mapping control bit, status bit map output Y; for CAN BD board in BD2, users can use CPAVL to select D register)



7.3.2 HMI screen

Control words, control bits, status words, status bits are listed for visualization.

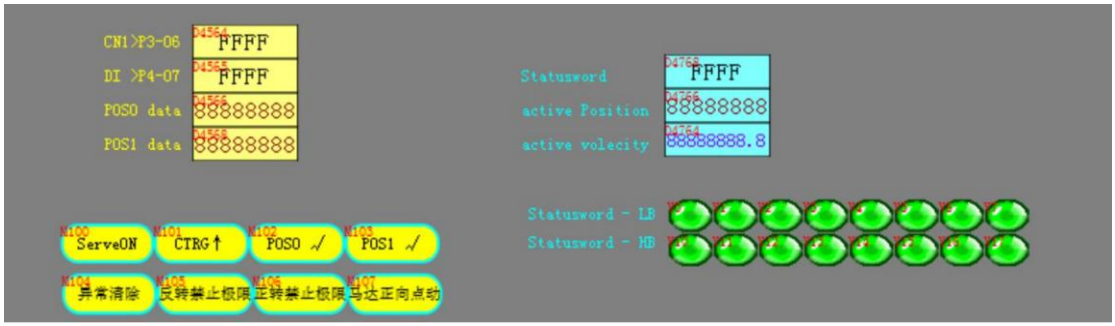


Figure 7-13

7.4 Turn ON servo

After completing the above steps, start the servo motor in the following four steps, as Figure 7-13 shows:

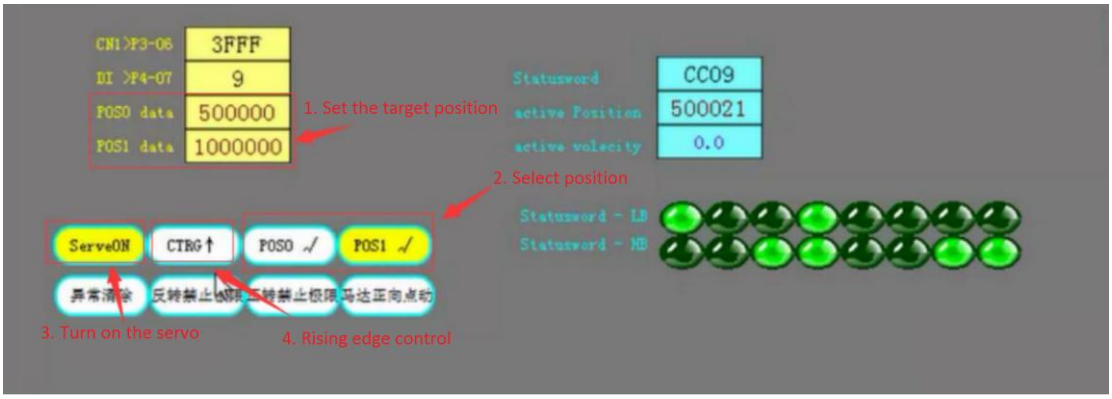


Figure 7-14