



Technical Note

CANopen Operation Modes Implementation

Delta Electronics (Netherlands) BV

Automotive Campus 260, 5708 JZ Helmond, the Netherlands

Technical Support contact: iatechnicalsupport@deltaww.com

www.delta-emea.com

History

Rev.	Comments	Date
V1.0	First published	26 th July 2023

Table of Contents

1	Introduction	4
1.1	Purpose of the Document	4
1.2	Scope and Limitations	4
2	CANopen Overview	5
2.1	CANopen Protocol Overview	5
2.2	Object Dictionary	5
2.3	PDOS (Process Data Objects) and SDOs (Service Data Object).....	5
3	CANopen Operation Mode.....	6
3.1	Profile Position Mode.....	6
3.2	Interpolated Position Mode	7
3.3	Homing mode	9
3.4	Profile Velocity mode	10
3.5	Profile Torque mode.....	12
4	CANopen Implementation on a Servo	13
4.1	Hardware Configuration.....	13
4.2	Parameter settings in CANopen mode	13
4.3	Configuration and Initialization Steps	14
4.3.1	<i>Network Structure</i>	15
4.3.2	<i>Servo Parameters Setup</i>	15
4.3.3	<i>Setting Master Parameters</i>	16
4.4	Mapping Process Data Objects (PDOS).....	16
4.5	Mapping Service Data Objects (SDOs).....	20
5	Practical Examples	21
5.1	Profile Position Mode.....	21
5.2	Interpolated Position Mode	24
6	Troubleshooting and FAQs	27
6.1	Common Issues and Solutions.....	27
6.2	Frequently Asked Questions.....	28

1 Introduction

CANopen is a communication protocol widely used in industrial automation for interconnecting devices within a network. It provides a standardized approach for reliable and efficient data exchange between various devices, including servos.

1.1 Purpose of the Document

The purpose of this technical note is to guide users in implementing CANopen operation mode on a servo drive. It aims to provide a comprehensive understanding of the necessary steps, configurations, and considerations involved in achieving successful integration.

1.2 Scope and Limitations

This technical note focuses on the implementation of CANopen operation modes specifically for servos. It assumes a basic understanding of CANopen protocol concepts and targets engineers and developers involved in servo control systems.

2 CANopen Overview

2.1 CANopen Protocol Overview

CANopen utilizes the **Controller Area Network (CAN)** bus as its physical layer, enabling robust communication between network nodes. It follows a master-slave architecture, where the master node manages the network and controls the behavior of slave nodes.

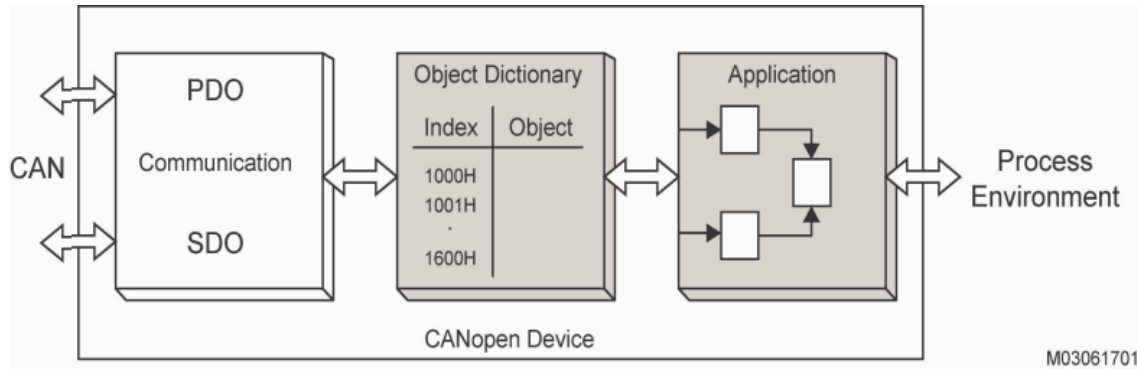


Figure 2-1 CANopen Device Model

A unified view of CANopen devices require the use of a general device model so that different devices can be described by one standard. The device model consist of three main components:

- Communication
- Object Dictionary
- Application

2.2 Object Dictionary

The Object Dictionary is a central repository that defines the data structure and functionality of each node within a CANopen network. It includes objects such as process data object (PDOs) and service data objects (SDOs), which play a crucial role in data exchange and configuration.

2.3 PDOs (Process Data Objects) and SDOs (Service Data Object)

PDOs facilitate real-time data exchange between nodes. They consist of fixed or dynamically mapped data that can be transmitted cyclically or upon event-triggered conditions. PDOs are commonly used for servo control, allowing efficient and deterministic communication.

SDOs provide a means for configuring and accessing data on a remote node. They support both expedited and segmented transfer mechanisms and are commonly used for parameterization, configuration, and diagnostics of CANopen devices.

3 CANopen Operation Mode

This section describes the mode of operation specified by CiA DS402 when the servo is in the CANopen mode. The content includes basic operation settings and related object descriptions.

3.1 Profile Position Mode

After receiving the position command transmitted from the controller, the servo drive controls the servo motor to reach the target position. In Profile Position (PP) mode, the controller only informs the servo drive of the target position, speed command, and acceleration/deceleration settings at the beginning. The motion planning from command triggering to arrival of the target position is performed by the trajectory generator in the servo drive.

The following figure shows the Profile Position mode architecture of the servo drive:

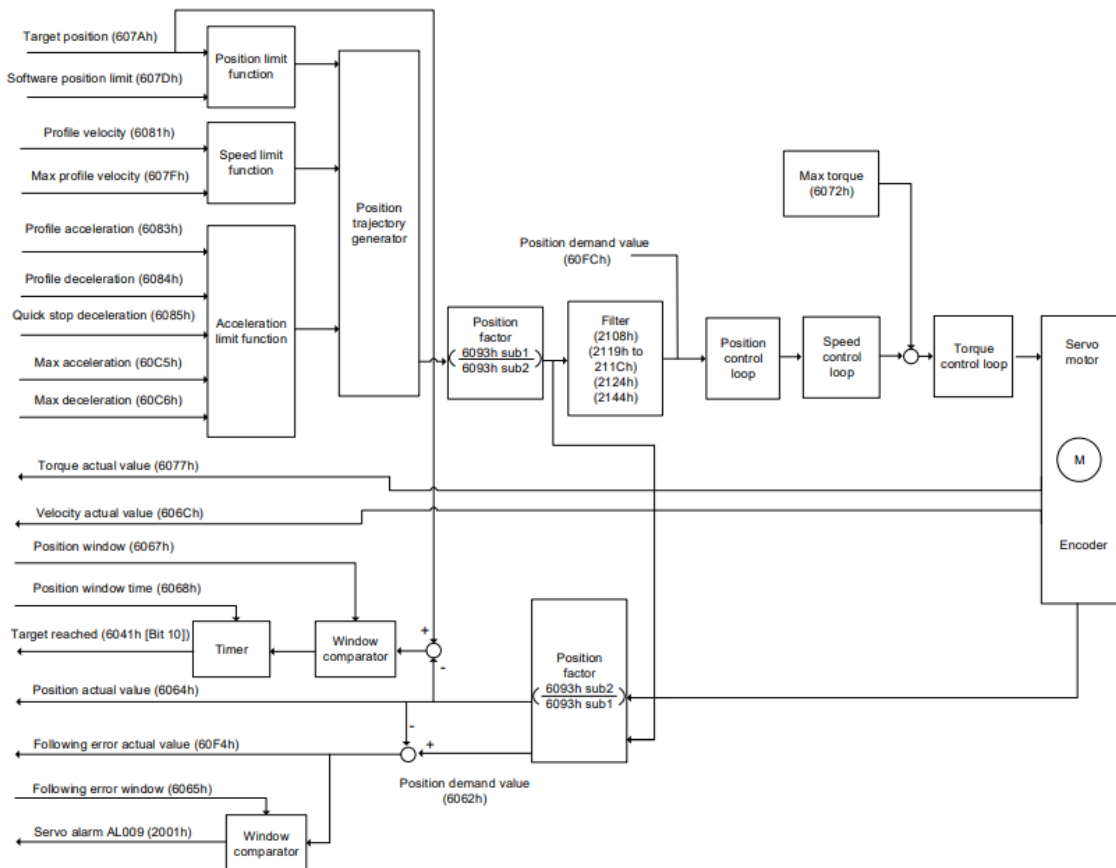


Figure 3-1 Profile Position Mode

Relevant object list

Index	Name	Data Type	Access
6040h	Controlword	UNSIGNED16	RW
6041h	Statusword	UNSIGNED16	RO

6060h	Modes of operation	INTEGER8	RW
6061h	Modes of operation display	INTEGER8	RO
6062h	Position demand value [PUU]	INTEGER32	RO
6063h	Position actual internal value [Pulse]	INTEGER32	RW
6064h	Position actual value [PUU]	INTEGER32	RO
6065h	Following error window	UNSIGNED32	RO
6067h	Position window	UNSIGNED32	RO
6068h	Position window time	UNSIGNED16	RO
606Ch	Velocity actual value	INTEGER32	RW
6072h	Max torque	UNSIGNED16	RW
6077h	Torque actual value	INTEGER16	RW
607Ah	Target position	INTEGER32	RO
607Dh	Software position limit	INTEGER32	RW
607Fh	Max profile velocity	UNSIGNED32	RO
6081h	Profile velocity	UNSIGNED32	RW
6083h	Profile acceleration	UNSIGNED32	RW
6084h	Profile deceleration	UNSIGNED32	RW
6085h	Quick stop deceleration	UNSIGNED32	RW
6093h	Position factor	UNSIGNED32	RW
60C5h	Max acceleration	UNSIGNED32	RW
60C6h	Max deceleration	UNSIGNED32	RW
60F4h	Following error actual value	INTEGER32	RO
60FCh	Position demand value	INTEGER32	RO

3.2 Interpolated Position Mode

Interpolated Position (IP) mode requires a series of position data to complete the interpolation for positioning. Different from PP (Profile Position) mode, all the motion command paths in IP modes

are issued by the controller. The servo drive only follows each position that the controller issues and finally completes a motion command. Delta servo drives only support synchronous operation in which the controller periodically sends the SYNC object (COB-ID = 0x80). The interpolation time period can be set with OD 60C2h. And the controller issues the position command to the interpolation position of OD 60C1h.

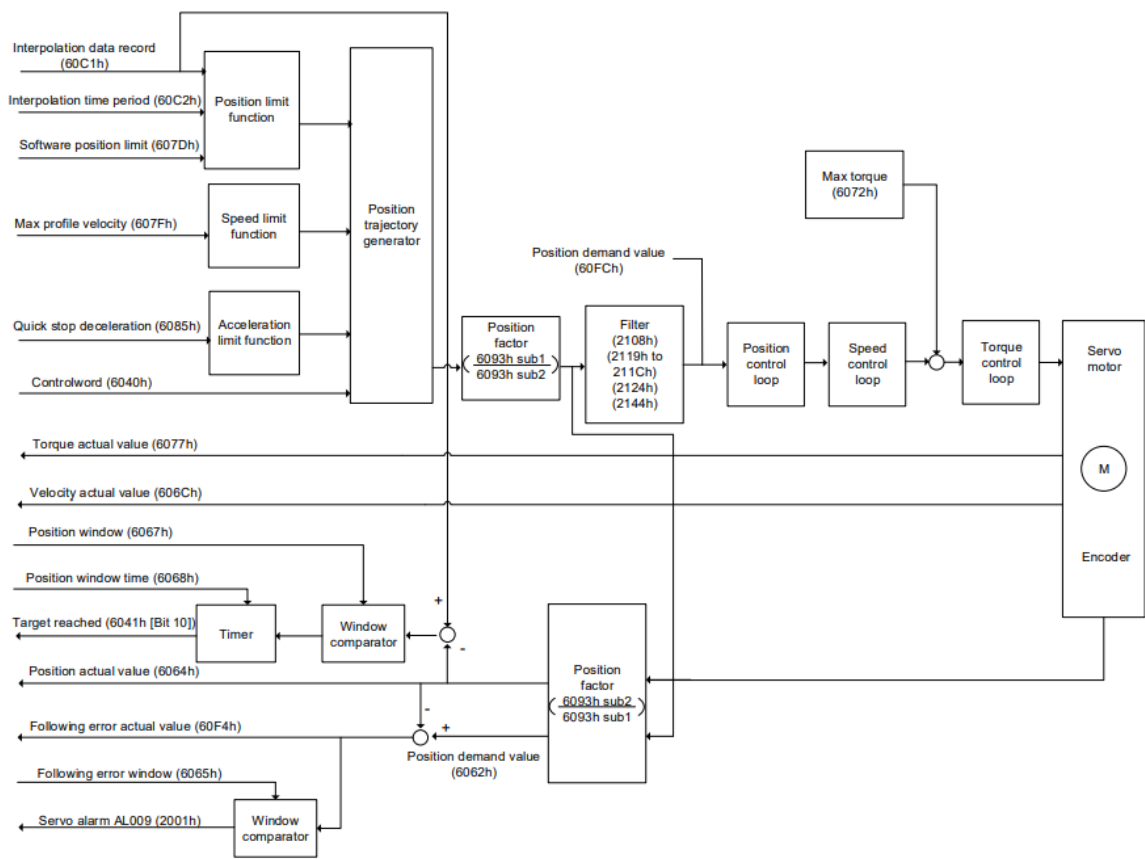


Figure 3-2 Interpolated Position Mode

Relevant object list

Index	Name	Data Type	Access
6040h	Controlword	UNSIGNED16	RW
6041h	Statusword	UNSIGNED16	RO
6060h	Modes of operation	INTEGER8	RW
6061h	Modes of operation display	INTEGER8	RO
6093h	Position factor	UNSIGNED 32	RW
60C0h	Interpolation sub mode select	INTEGER16	RW

60C1h	Interpolation data record	INTEGER32	RW
--------------	---------------------------	-----------	----

3.3 Homing mode

After homing is complete, the position system of the servo drive is established and the driven can start executing the position command issued by the controller. The Delta servo drive offers 39 homing methods, including homing on the home switch, positive or negative limit, motor Z pulse, and hard stop.

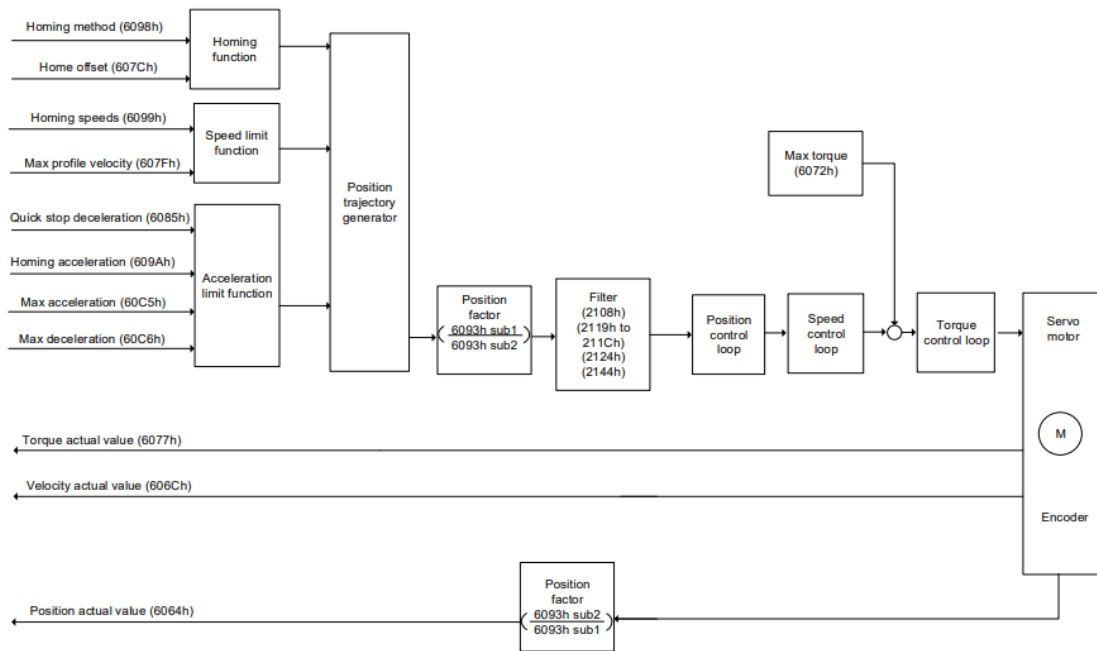


Figure 3-3 Homing Mode

Relevant object list

Index	Name	Data Type	Access
6040h	Controlword	UNSIGNED16	RW
6041h	Statusword	UNSIGNED16	RO
6060h	Modes of operation	INTEGER8	RW
6061h	Modes of operation display	INTEGER8	RO
6064h	Position actual value [PUU]	INTEGER32	RO
606Ch	Velocity actual value	INTEGER32	RW
6072h	Max torque	UNSIGNED16	RW

607Ch	Home offset	INTEGER32	RW
607Fh	Max profile velocity	UNSIGNED32	RW
6085h	Quick stop deceleration	UNSIGNED32	RW
6093h	Position factor	UNSIGNED32	RW
6098h	Homing method	INTEGER8	RW
6099h	Homing speeds	UNSIGNED32	RW
609Ah	Homing acceleration	UNSIGNED32	RW
60C5h	Max acceleration	UNSIGNED32	RW
60C6h	Max deceleration	UNSIGNED32	RW

3.4 Profile Velocity mode

In Profile Velocity (PV) mode, the controller specifies the speed command and acceleration /deceleration settings, and then the trajectory generator of the servo drive plans the motion path according to these conditions.

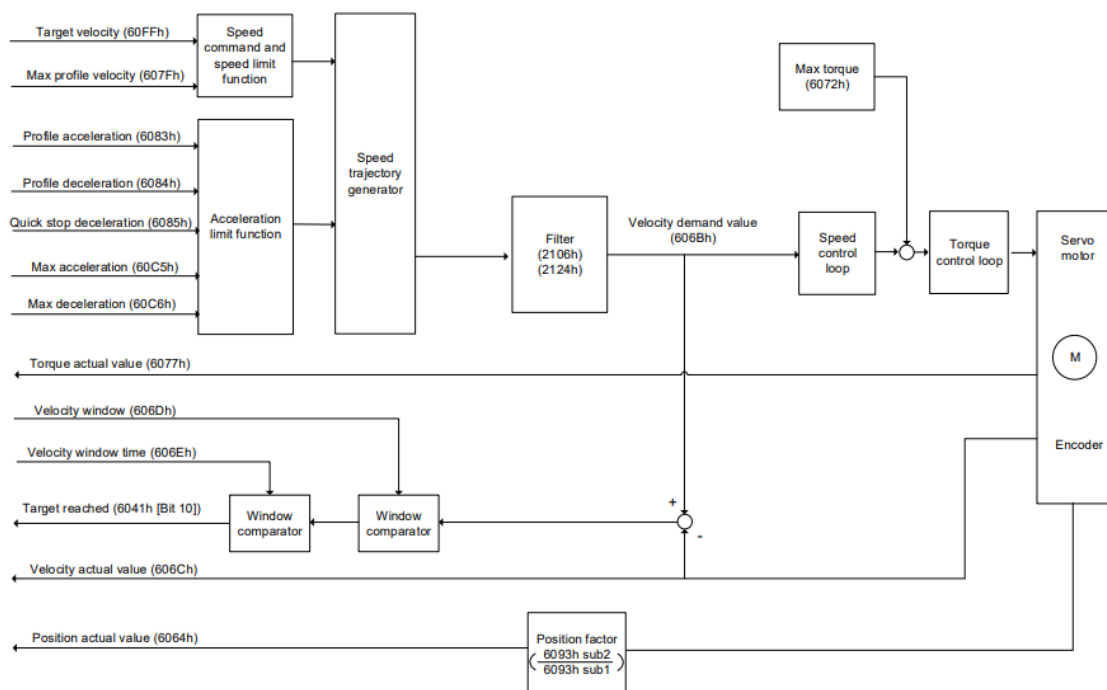


Figure 3-4 Profile Velocity Mode

Relevant object list

Index	Name	Data Type	Access
6040h	Controlword	UNSIGNED16	RW
6041h	Statusword	UNSIGNED16	RO
6060h	Modes of operation	INTEGER8	RW
6061h	Modes of operation display	INTEGER8	RO
6064h	Position actual value [PUU]	INTEGER32	RO
606Bh	Velocity demand value	INTEGER32	RO
606Ch	Velocity actual value	INTEGER32	RO
606Dh	Velocity window	UNSIGNED16	RW
606Eh	Velocity window time	UNSIGNED16	RW
606Fh	Velocity threshold	UNSIGNED16	RW
6072h	Max torque	UNSIGNED16	RW
6077h	Torque actual value	INTEGER16	RO
607Fh	Max profile velocity	UNSIGNED32	RO
6083h	Profile acceleration	UNSIGNED32	RW
6084h	Profile deceleration	UNSIGNED32	RW
6085h	Quick stop deceleration	UNSIGNED32	RW
6093h	Position factor	UNSIGNED32	RW
60C5h	Max acceleration	UNSIGNED32	RW
60C6h	Max deceleration	UNSIGNED32	RW
60FFh	Target velocity	INTEGER32	RW

3.5 Profile Torque mode

In Profile Torque (PT) mode, the controller specifies the torque command and filtering conditions, and then the trajectory generator of the servo drive plans the torque slope according to these conditions.

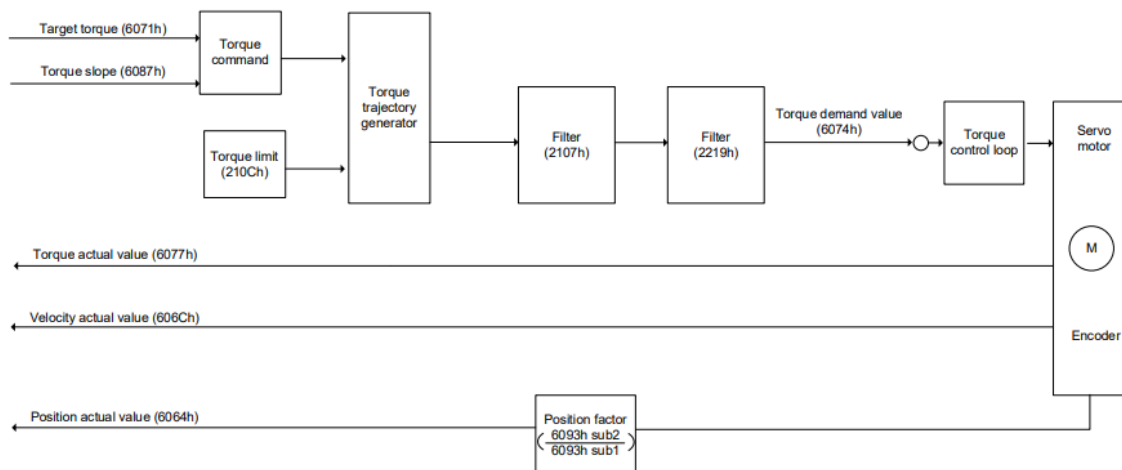


Figure 3-5 Profile Torque Mode

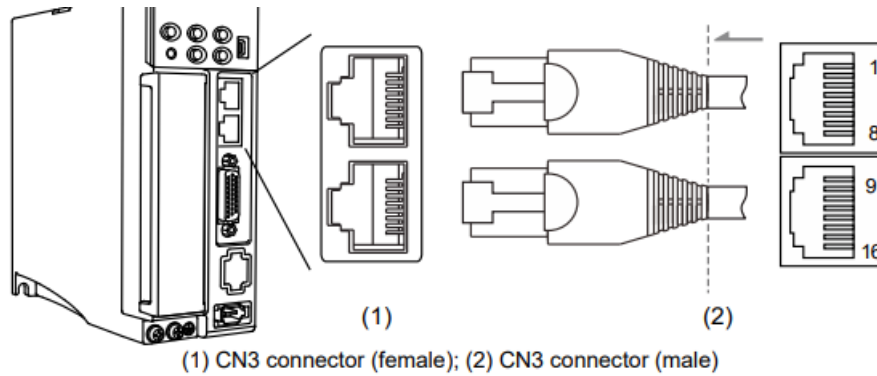
Relevant object list

Index	Name	Data Type	Access
6040h	Controlword	UNSIGNED16	RW
6041h	Statusword	UNSIGNED16	RO
6060h	Modes of operation	INTEGER8	RW
6061h	Modes of operation display	INTEGER8	RO
6064h	Position actual value [PUU]	INTEGER32	RO
606Ch	Velocity actual value	INTEGER32	RO
6071h	Target torque	INTEGER16	RW
6074h	Torque demand value	INTEGER16	RO
6075h	Current actual value	INTEGER16	RO
6087h	Torque slope	UNSIGNED32	RW
6093h	Position factor	UNSIGNED32	RW

4 CANopen Implementation on a Servo

4.1 Hardware Configuration

Pin assignment (RJ-45) for CAN bus wiring on ASDA-B3-E, for example:



Pin assignment:

Pin No.	Signal	Description
1, 9	CAN_H	CAN_H bus line (dominant high)
2, 10	CAN_L	CAN_L bus line (dominant low)
3, 11	GND_ISO	Signal GND
4, 12	RS-485-	For the servo drive to transmit the data to differential terminal (-).
5, 13	RS-485+	For the servo drive to transmit the data to differential terminal (+).
6, 14	-	Reserved
7, 15	GND_ISO	Signal GND
8, 16	-	Reserved

4.2 Parameter settings in CANopen mode

Following these instructions to connect the CANopen controller and the servo drive:

1. Set to CANopen mode: set P1.001.YX to 0C
2. Set the node ID: set P3.000 to 0x0001 – 0x007F
3. Set the transmission rate (baud rate): set P3.001.Z to 4
4. It is suggested that you change the settings value of P3.012.Z from 0 (default) to 1 to enable the non-volatile setting for the parameter. Note that the default E-Gear ratio varies with the set value of P3.012.Z

Function	P3.012 = 0x0100 (Z = 1)		P3.012 = 0x0000 (Z = 0)	
	Servo parameter	Default	OD address	Default
Motor stop mode	P1.032	0x0000	605Bh	0
S-curve acceleration constant	P1.034	200	6087h	200
Zero speed range	P1.038	100 (0.1 rpm)	606Fh	100 (0.1 rpm)
E-Gear ratio - numerator N1	P1.044	16777216	6093h sub1	1
E-Gear ratio - denominator M	P1.045	100000	6093h sub2	1
Speed reached (DO.SP_OK) range	P1.047	10 (rpm)	606Dh	100 (0.1 rpm)
Accumulated time to reach desired speed	P1.049	0	606Eh	0
Maximum speed limit	P1.055	Depending on the motor (rpm)	607Fh	Depending on the motor (0.1 rpm)
			6080h	Depending on the motor (rpm)
Excessive deviation warning condition of Position command	P2.035	50331648	6065h	50331648
Positive software limit (PP / CSP / CSV / CST mode)	P5.008	2147483647	607Dh sub2	2147483647
Negative software limit (PP / CSP / CSV / CST mode)	P5.009	-2147483648	607Dh sub1	-2147483648
Origin definition (HM mode)	P6.001	0	607Ch	0

- It is suggested that you enable the dynamic brake function (P1.032 = 0x0000)

4.3 Configuration and Initialization Steps

Here is an example setup utilizing the DVP15MC11T controller with CANopen Builder and ASDA-A2-M servo drive. Please note that you have the flexibility to switch to any other CANopen controller and compatible servo drive of your choice. The practical examples that follow will be based on this initial setup.

4.3.1 Network Structure

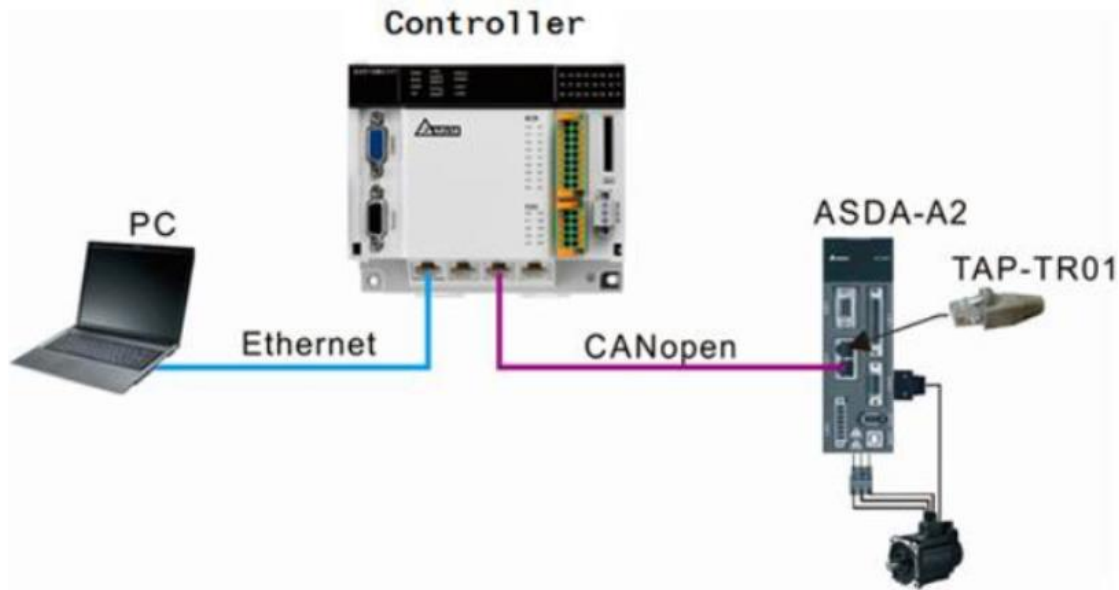


Figure 4-1 Network Structure

Construct a network to control the start, run and stop of a servo through DVP15MC series controller.

Note:

1. Delta standard CANopen communication cable is recommended for wiring
2. The motion port of the controller is embedded with a terminal resistor and the end of servo drive should be connected with a terminal resistor: TAP-TR01

4.3.2 Servo Parameters Setup

Parameter	Setting value	Description
P1-01	b	Set servo work mode to CANopen mode.
P03-00	1	The CANopen station address of ASDA-A2 servo
P03-01	0400	The CANopen baud rate of ASDA-A2 servo is 1Mbps. The 3 rd bit of P03-01 sets the CANopen baud rate of the servo. The relationship between the value and corresponding baud rate is shown as below. 0: 125Kbps 1: 250Kbps 2: 500Kbps 3: 750Kbps 4: 1M bps

Figure 4-2 Servo Parameters Setup

Note: If use third-party controller, please set P1-01=C as in Chapter 4.2.

4.3.3 Setting Master Parameters

Open the CANopen Builder software of version 6.0 or above. After the controller and PC are connected successfully, refer to CANopen under Network Configuration in the software help and set up master parameters according to the requirement.

Name	Node address	Mode	Baud rate
DVP15MC11T	127	Master mode	1M bps

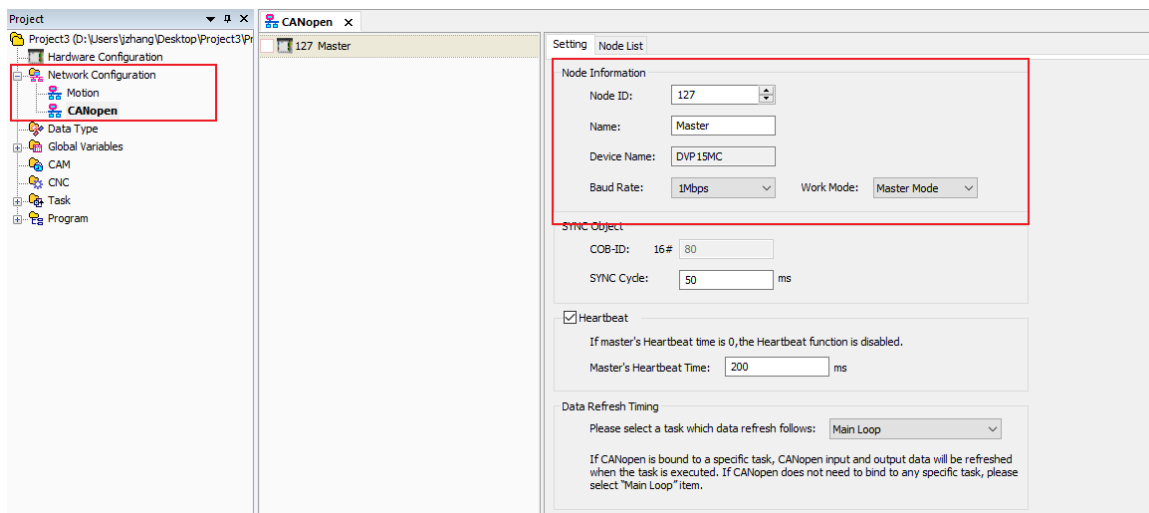


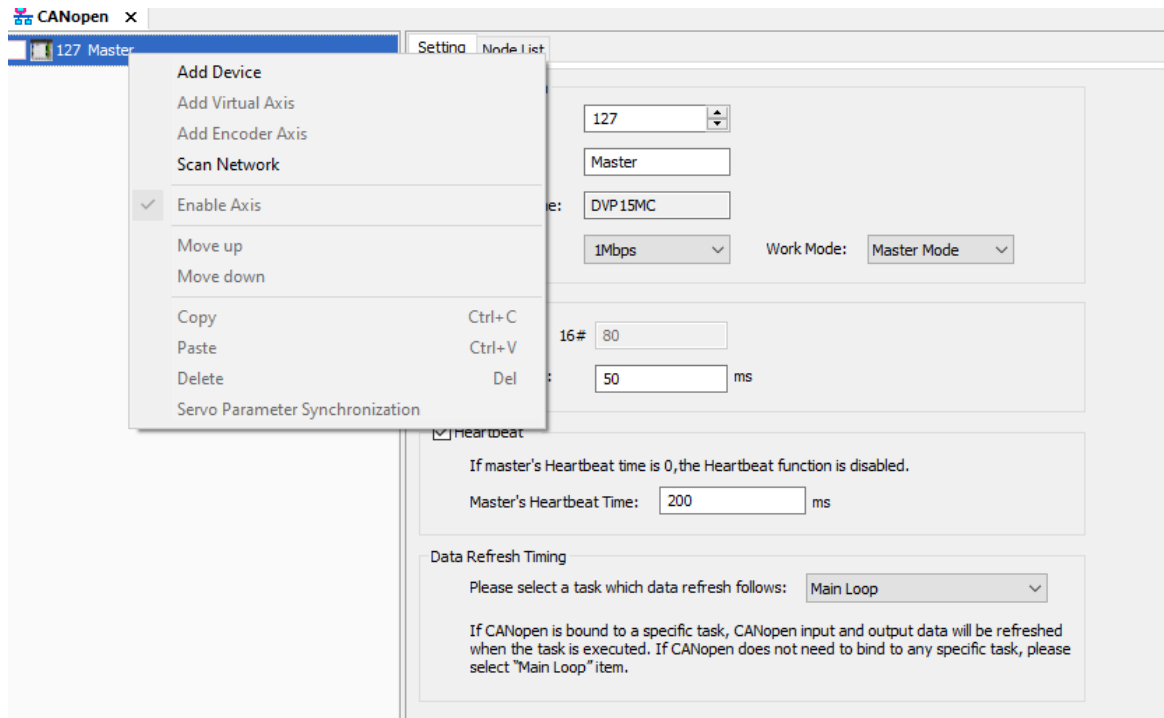
Figure 4-3 Setting Master Parameters

4.4 Mapping Process Data Objects (PDOs)

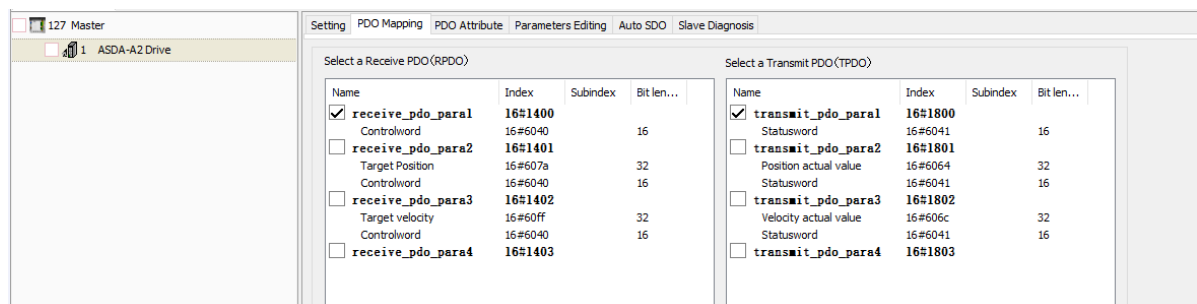
Mapping PDOs allows the exchange of critical servo data, such as position, velocity, and torque, between the servo and the CANopen network. Understanding the PDO mapping process and configuring the necessary objects are crucial for accurate and efficient data transmission.

Example:

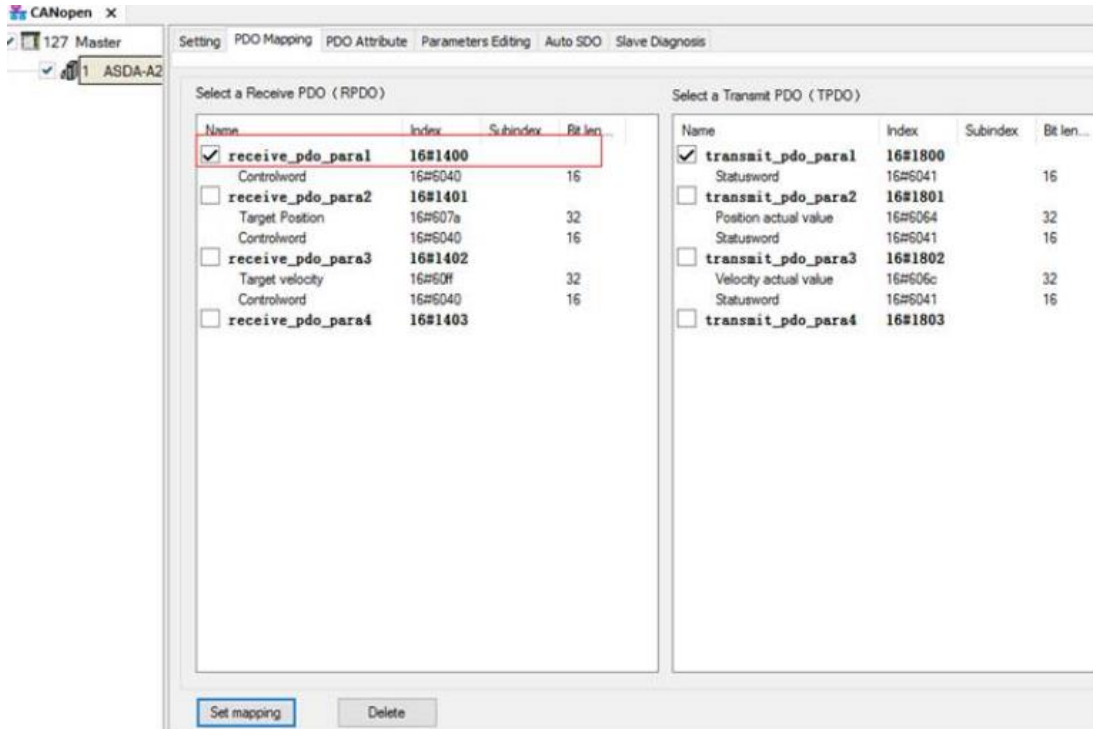
1. After opening the CANopen Builder software, under **Network Configuration** CANopen, right click “**127 Master**” and then select **Scan Network** from the dropdown menu, which pops up. You can also manually add device by selecting **Add Device**.



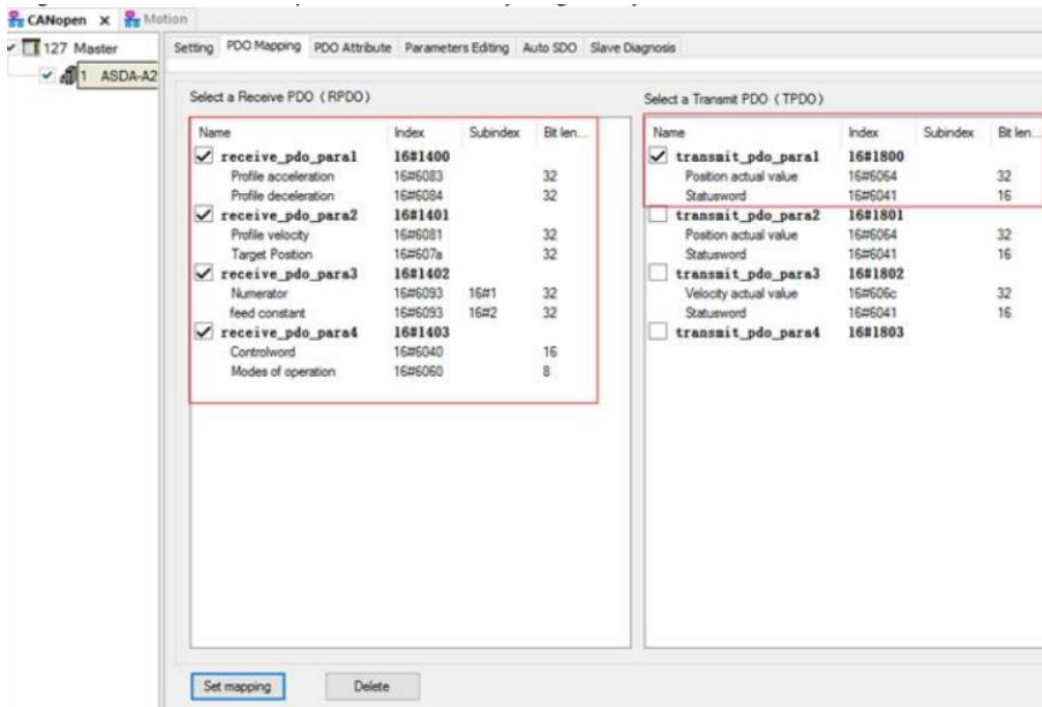
2. Double click the slave ASDA-A2 servo and then select PDO Mapping tab and then corresponding PDO configuration interface will show up.



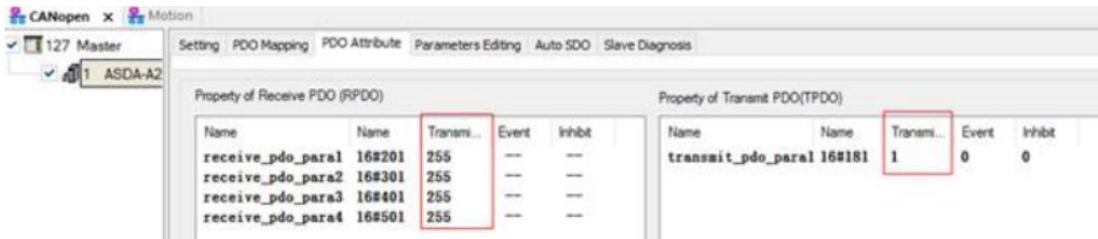
3. Double click the part in the following read box. Then the Add Map window pops up for configuration of RXPDO1 parameters.



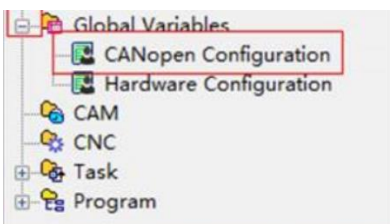
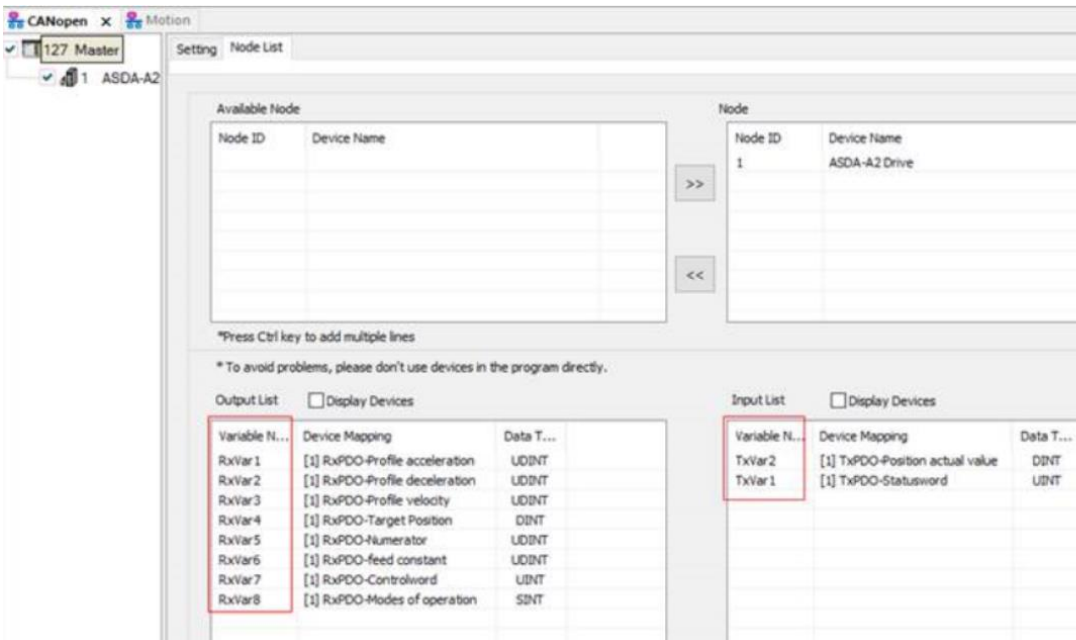
4. Configuration the RXPDO, TXPDO parameters in the slave by using the way above.



5. Click PDO Attribute tab and select PDO transmission type. In this example, the asynchronous mode 255 for RXPDO is selected. For TXPDO, synchronization mode 1 is selected.



6. After the configuration is finished, click “127 Master” and select **Node List** tab. The variable names in the **Input List** and **Output List** can be modified by users. The global variables that the PDO mapping of the slave corresponds to can be seen in the **CANopen Configuration** under **Global Variables**.



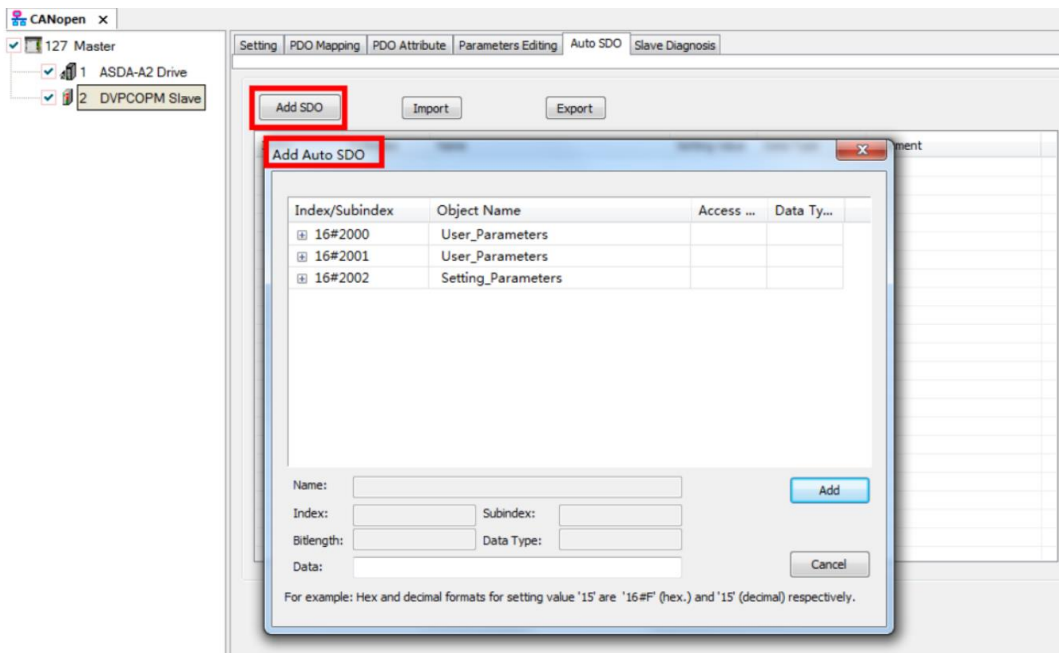
Index	Scope	Name	Address	Data Type	Initial Value	Comment
1	VAR_GLOBAL	RxVar1	%MWS500	UDINT		[1] RxPDO-Profile acceleration
2	VAR_GLOBAL	RxVar2	%MWS502	UDINT		[1] RxPDO-Profile deceleration
3	VAR_GLOBAL	RxVar3	%MWS504	UDINT		[1] RxPDO-Profile velocity
4	VAR_GLOBAL	RxVar4	%MWS506	DINT		[1] RxPDO-Target Position
5	VAR_GLOBAL	RxVar5	%MWS508	UDINT		[1] RxPDO-Numerator
6	VAR_GLOBAL	RxVar6	%MWS510	UDINT		[1] RxPDO-feed constant
7	VAR_GLOBAL	RxVar7	%MWS512	UINT		[1] RxPDO-Controlword
8	VAR_GLOBAL	RxVar8	%MWS513	SINT		[1] RxPDO-Modes of operation
9	VAR_GLOBAL	TxVar2	%MWS000	DINT		[1] RxPDO-Position actual value
10	VAR_GLOBAL	TxVar1	%MWS002	UINT		[1] RxPDO-Statusword

4.5 Mapping Service Data Objects (SDOs)

SDOs enable remote configuration and access to servo parameters. This section explains how to map SDOs to the Object Dictionary, allowing seamless parameterization and diagnostics of the servo via the CANopen network.

Example:

In the following Auto SDO window, click Add SDO button to add SDO. Then the Add Auto SDO window comes out for you to select corresponding slave parameters. Fill the value which is required in the Data field. Then click Add button to add the parameter to the auto SDO list.



After adding SDO, you will see the following window. A maximum of 30 auto SDO can be added for each slave. For auto SDO, its parameters only have the attribute of "only write" rather than "read" attribute. After the controller makes the connection with the slave, perform the write action for the parameters in auto SDO in the slave once.

If the controller is repowered on or the slave is offline, the controller will make the connection with the slave once again.

Index	Subindex	Name	Setting Value	Data Type	Comment
16#2000	16#1	Rx_DATA0	10	UINT	User Added

5 Practical Examples


In this chapter, we will delve into two practical examples utilizing the aforementioned setup to illustrate the usage of different CANopen operation modes. These examples aim to provide a hands-on understanding of how to leverage the versatility of CANopen to achieve specific control objectives. By following along with these practical scenarios, you will gain valuable insights into implementing and configuring various CANopen operation modes in your servo control system.

5.1 Profile Position Mode

In this example, we will focus on utilizing the Position Mode operation of CANopen to precisely control the position of the servo motor. We will cover the steps required to configure the necessary PDO mappings, set the target position, and monitor the actual position feedback from the servo drive. By the end of this example, you will have a clear understanding of how to leverage Position Mode in your application and achieve accurate positioning control.


- **IO data mapping between Master PLC and Slave**

Controller > Slave device

Master variable name	CANopen data transmission	Slave parameter index	Slave parameter subindex	Explanation of slave parameters
RxVar1		16#6083	16#0	Acceleration time of the servo drive
RxVar2		16#6084	16#0	Deceleration time of the servo drive
RxVar3		16#6081	16#0	Target velocity of the servo drive
RxVar4		16#607A	16#0	Target position of the servo drive
RxVar5		16#6093	16#1	Numerator of servo e-gear ratio
RxVar6		16#6093	16#2	Denominator of servo e-gear ratio
RxVar7		16#6040	16#0	Control word of the servo drive
RxVar8		16#6060	16#0	Motion mode of the

				servo drive
--	--	--	--	-------------

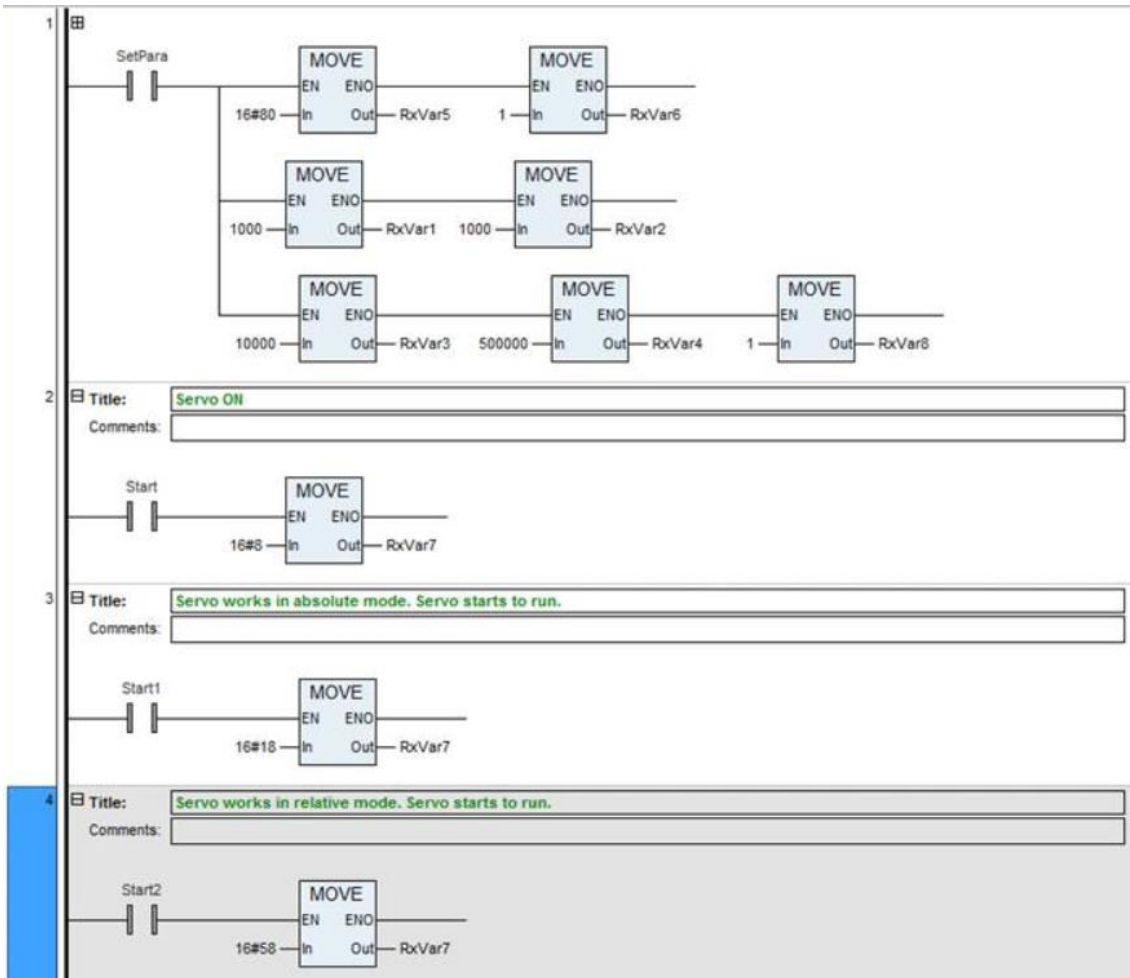
Slave device > Controller

Master variable name	CANopen data transmission	Slave parameter index	Slave parameter subindex	Explanation of slave parameters
TxVar1		16#6064	16#0	Current position of the servo drive
TxVar2		16#6041	16#0	Status word of the servo drive

- CANopen Network Control

Control program from master PLC

Index	Scope	Name	Address	Data Type	Initial
1	VAR	start		BOOL	
2	VAR	SetPara		BOOL	
3	VAR	Start1		BOOL	
4	VAR	Start2		BOOL	



Program Explanation

- When SetPara changes to TRUE, setting target velocity, position, acceleration time and deceleration for the servo is started.
- When Start changes to TRUE, Start1 and Start2 change to FALSE, and ASDA-A2 is servo on.
- When Start1 changes to TRUE, Start and Start1 change to FALSE, the servo drive works in absolute position mode. The servo will run until the target position is reached.
- When Start2 changes to TRUE, Start and Start1 change to FALSE, the servo drive works in relative position mode. The servo will run until the target position is reached.

The status word of the servo can be read via TxVar1 and actual position of the servo motor can be read via TxVar2.

Bit	Definition in each operation mode		
	Profile Position mode	Homing mode	Profile Velocity mode Profile Torque mode Interpolated Position mode
Bit 4	Command triggering (rising-edge triggered)	Homing (rising-edge triggered)	-
Bit 5	Function for the command to take immediate effect	-	-
Bit 6	0: absolute position command 1: relative position command	-	-


5.2 Interpolated Position Mode

In this example, we will explore the Interpolated Position Mode operation of CANopen, which enables coordinated motion control of multiple axes. We will demonstrate how to configure the CANopen network to achieve synchronized motion among several servo drives, allowing precise interpolated positioning.

- IO data mapping between Master PLC and Slave

Controller > Slave device

Important note: As interpolated position mode receives position commands from the controller in a cyclical manner, it is crucial to ensure that the PDO transmission type is also set to cyclical.

Master variable name	CANopen data transmission	Slave parameter index	Slave parameter subindex	Explanation of slave parameters
RxVar1		16#60C1	16#1	Interpolated position command
RxVar2		16#60C2	16#1	Interpolation time units
RxVar3		16#60C2	16#2	Interpolation time index
RxVar4		16#6040	16#0	Control word of the servo drive
RxVar5		16#6060	16#0	Motion mode of the servo drive

Slave device > Controller

Master variable	CANopen data	Slave parameter	Slave parameter	Explanation of
-----------------	--------------	-----------------	-----------------	----------------

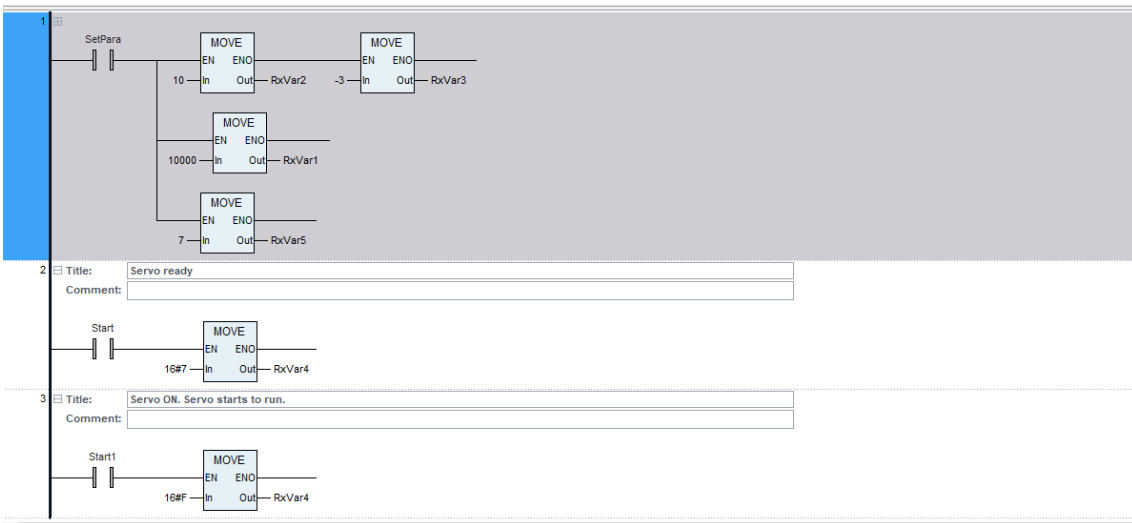
name	transmission	index	subindex	slave parameters
TxVar1	←	16#6064	16#0	Current position of the servo drive
TxVar2		16#6041	16#0	Status word of the servo drive

- CANopen Network Control

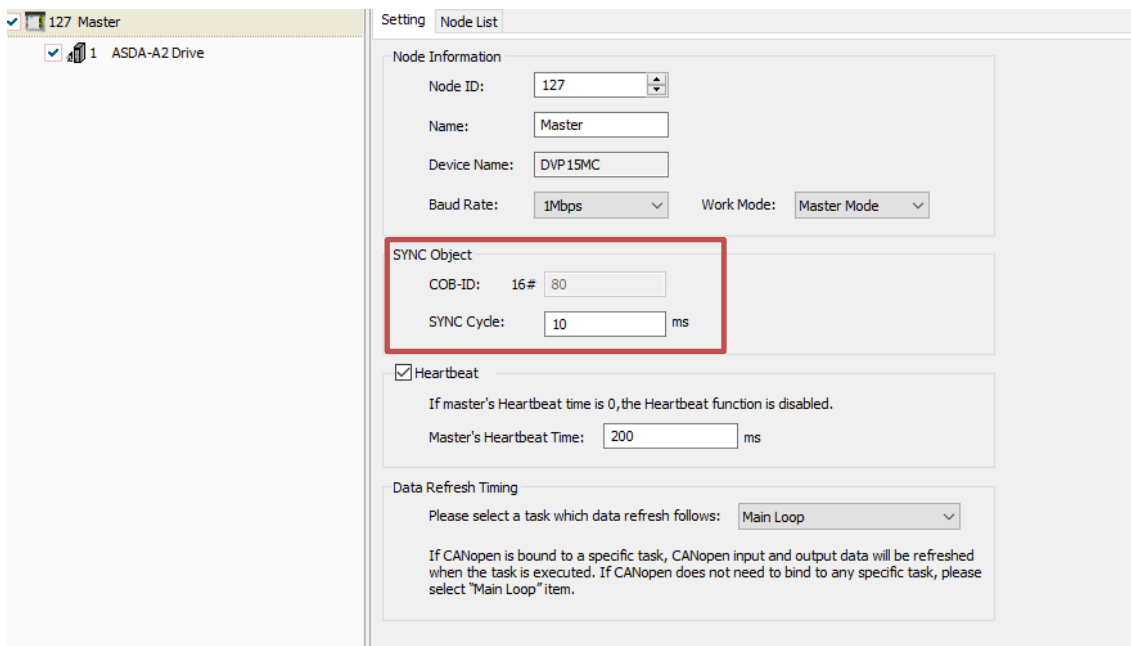
Control program from master PLC

Index	Scope	Name	Address	Data Type	Initial Value	Comment
1	VAR_GLOBAL	RxVar1	%MW500	DINT		[1] RxPDO-Parameter1 of ip function
2	VAR_GLOBAL	RxVar2	%MB11004	USINT		[1] RxPDO-Interpolation time units
3	VAR_GLOBAL	RxVar3	%MB11005	SINT		[1] RxPDO-Interpolation time index
4	VAR_GLOBAL	RxVar4	%MW503	UINT		[1] RxPDO-Controlword
5	VAR_GLOBAL	RxVar5	%MB11008	SINT		[1] RxPDO-Modes of operation
6	VAR_GLOBAL	TxVar1	%MW5000	DINT		[1] TxPDO-Position actual value
7	VAR_GLOBAL	TxVar2	%MW5002	UINT		[1] TxPDO-Statusword

Index	Scope	Name	Address	Data Type	Initial Value	Comment
1	VAR	SetPara		BOOL		
2	VAR	Start		BOOL		
3	VAR	Start1		BOOL		



It is crucial to note that the value for OD60C2 must be set to match the cycle time of the CANopen controller's synchronization (SYNC) signal. The SYNC cycle time represents the interval at which the controller sends synchronization messages to the servo drives in the network. To ensure proper synchronization and coordination among the drives, the value of OD60C2 should be aligned with the SYNC cycle time of the controller.



It is important to note that while these examples are based on the initial setup using the DVP15MC11T controller with CANopen Builder and ASDA-A2-M servo drive, you can adapt the concepts and steps to your specific CANopen controller and supported servo drive. The principles and procedures discussed in these examples will be applicable across various CANopen-enabled devices, allowing you to expand your knowledge and apply it to different hardware configurations.

By working through these practical examples, you will gain hands-on experience in implementing and utilizing different CANopen operation modes, enabling you to leverage the full potential of your servo control system and tailor it to your specific application requirements.

6 Troubleshooting and FAQs

6.1 Common Issues and Solutions

Addressing common implementation and providing troubleshooting solutions can greatly assist users. This section covers potential challenges encountered during the CANopen implementation on a servo and suggests troubleshooting approaches.

1. Communication Problems:

- Issue: Difficulty establishing communication between the CANopen controller and servo drives.
- Solution: Verify the network configuration, including node IDs, baud rate, and network topology. Ensure proper termination and wiring of the CAN bus. Check for any communication errors or conflicts in the network.

2. PDO Mapping Errors:

- Issue: Incorrect mapping of Process Data Objects (PDOs) between the controller and servo drives, resulting in data inconsistency or improper control.
- Solution: Review the Object Dictionary (OD) of both the controller and servo drives to ensure accurate PDO mapping. Verify the PDO assignments and parameters, such as index, sub-index, and data length. Check for any conflicting or missing PDO mappings.

3. Synchronization Issues:

- Issue: Lack of proper synchronization among multiple servo drives in a network, leading to unsynchronized motion or erratic behavior.
- Solution: Confirm that the SYNC signals from the controller are being transmitted at the desired cycle time. Adjust the SYNC cycle time settings in both the controller and servo drives to ensure synchronization. Verify proper synchronization of motion commands, position feedback, and other critical data.

4. Firmware and Software Compatibility:

- Issue: Compatibility issues between the firmware/software versions of the CANopen controller and servo drives, resulting in operational inconsistencies or limited functionality.
- Solution: Ensure that the firmware/software versions of the controller and servo drives are compatible and up to date. Check for any firmware/software updates or patches provided by the manufacturers. Verify the compatibility matrix provided by the vendors for supported versions.

6.2 Frequently Asked Questions

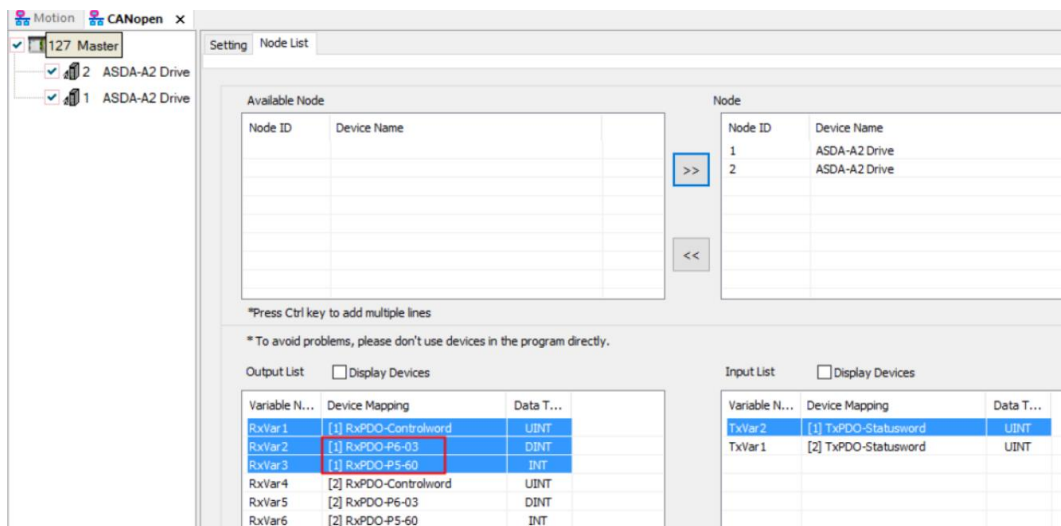
This section addresses frequently asked questions related to CANopen implementation on servos, providing concise answers to common queries and concerns.

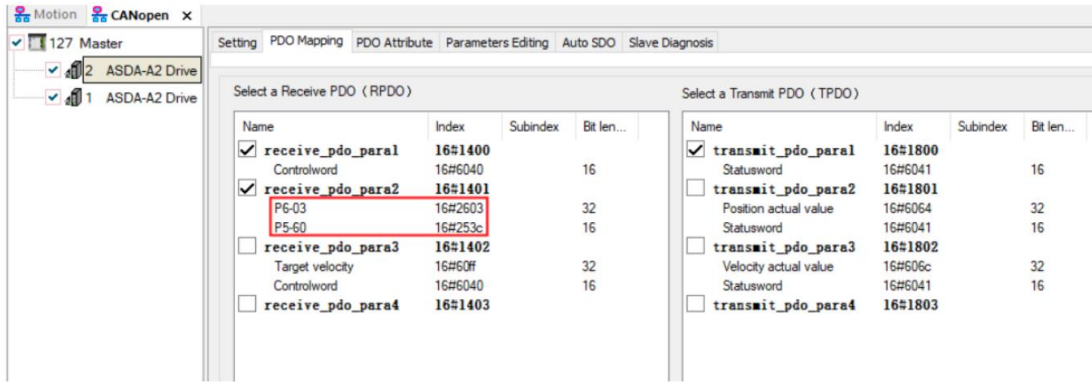
Q: How to deal with the servo alarm of AL303/AL302/AL301 when controller controls servos through CANopen?

- A:
1. Check if CAN cable is Delta standard cable and both ends of CAN cable are equipped with TAP-TR01 terminal resistors.
 2. Check if the shielded wire of CAN bus is grounded properly.
 3. Ensure that the value of servo parameter P3-09 is set to 5055 (hex).
 4. Make sure that the synchronization cycle period is set properly.

Q: How to deal with the servo alarm of AL124 when the controller controls Delta servo through CANopen port?

A: As shown in the following red box, the same PDO is configured with two parameters: P6-03 and P5-60. The transmission type for the PDO is 255 the asynchronous mode. P6-03 and P5-60 correspond to RxVar2 and RxVar3 variables in the master respectively. The initial value of RxVar3 is 0. If a value is written in RxVar2 and no value is written in RxVar3, the values in RxVar2 and RxVar3 would be sent to servo parameters P6-03 and P5-60 respectively. The value of P5-60 can not be 0 and therefore, the servo would release the alarm fault of 124. In this configuration, both of the two parameters should be assigned values. The assignment for both parameters should be done only when the value of RxVar2 is modified for the first time after the controller is powered on. After that, you can modify either of the two parameter values in no need of assigning values for both parameters at the same time.





The program of the two servo parameters is modified as follows.

