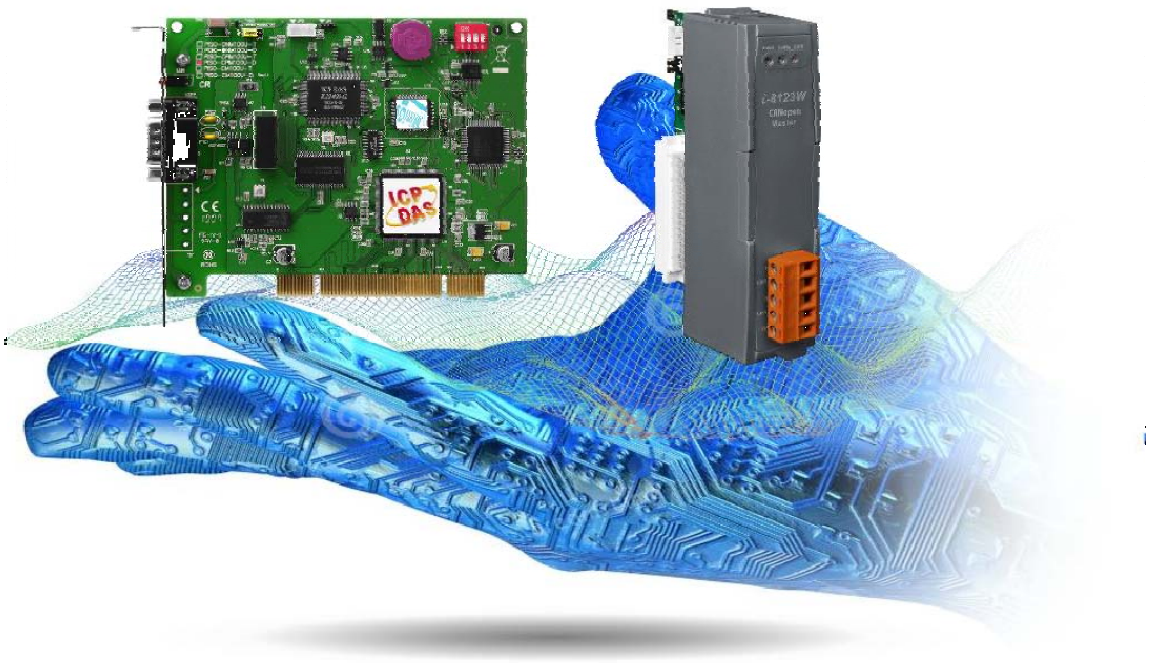
CANopen Motion

CAN bus Technology



CANopen Motion Library
for
I-8123W
I-7565-CPM
PISO-CPM100U-D/T

User's Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Revision

Version	CPMotor .DLL	Date	Author	Description
2.10	2.0.1.3	2017 09/29	Ming	Support Interpolation mode : 1. Line move 2. Any path move * I-7565-CPM not support interpolation.
2.0	2.0.0.0	2015 07/30	Johney	Support motors : 1. Delta ASDA-A2-M 2. Schneider LXM23A 3. FESTO CMMP-AS / CMMS-ST
1.0	1.0	2013 03/20	Johney	First release.

Contents

1. CANopen Motion Information

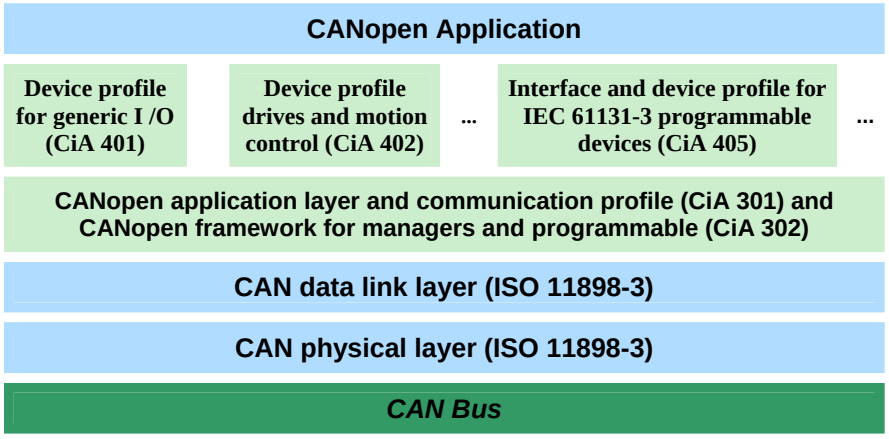
1.1 What is CANopen?

CAN (Controller Area Network) is a real-time communications protocol that is now well proven in the automobile industry. Its low costs and robust physical layer also make the protocol attractive for industrial networks. CANopen is a CAN-based higher layer protocol. It was developed as a standardized embedded network with highly flexible configuration capabilities. CANopen was designed for motion-oriented machine control networks, such as handling systems. By now it is used in many various fields, such as medical equipment, off-road vehicles, maritime electronics, public transportation, building automation, etc. The CANopen application layer and communication profile supports direct access to device parameters and transmission of time-critical process data. The CANopen network management services simplify project design, system integration, and diagnostics. In each decentralized control application, different communication services and protocols are required. CANopen defines all these services and protocols as well as the necessary communication objects.

The CANopen system offers the following performance features:

- Transmission of time-critical process data according to the producer consumer principle.
- Standardized device description (data, parameters, functions, programs) in the form of the so-called "object dictionary". Access to all "objects" of a device with standardized transmission protocol according to the client-server principle.
- Standardized services for device monitoring (node guarding / heartbeat), error signalization (emergency messages) and network coordination ("network management").

- Standardized system services for synchronous operations (synchronization message), central time stamp message.
- Standardized help functions for configuring baud rate and device identification number via the bus.
- Standardized assignment pattern for message identifiers for simple system configurations in the form of the so-called "predefined connection set".



1.2 Why CANopen Motion (CiA 402)?

Today CANopen is established as a communication network in many different application fields. Even it might not be the fastest network looking only at the bit speed. It has a lot of performance advantages due to the underlying CAN protocol and the event driven communication principle. In the last 15 years a large community of CANopen users has developed more and more devices and motors for CANopen motion application. This is the largest number of profiles a single protocol can offer. The CiA 402 device profile for frequency inverters, servo controllers and stepper motors has been already standardized in the IEC 61800-7 series.



This proven CANopen technology has been integrated to the industrial automation arena, and enhanced to meet the needs of motion control. Standard motion commands are now incorporated into the CANopen plan, enabling multi-axis system designers to invoke "prefabricated" functions, instead of writing the software from scratch.

--Eliminate Motion Controller & Cabling Complexity

CANopen motion control works with motor drivers (servo amplifiers) that close the motor-amplifier feedback loop locally. Traditional analog systems must send motor/amplifier feedback data to a distant motion controller for computation of the ±10 V position control signals. CANopen operation, which works with servo amplifiers that close the feedback loop locally, means that no separate motion controller is needed.

In a typical ±10 V servo amplifier system, up to 12 wires are involved in communicating with the motion controller. For a multi-axis system with 6 motors, the wires of each drive stage expand six fold, implying significant noise, crosstalk and wiring errors. This would result in the serious system setup and commissioning angst. CANopen systems not only eliminate the complexity of

the motion controller. The daunting wire bundles were replaced by a rugged, low-cost and robust CAN bus. Under the aegis of the CANopen protocol, all servo motors, limit switches, and other I/Os, are connected via the CAN bus. The control computer merely requires CANopen master with a simple connection to the servo motors. The control computer sends simple positioning instructions over the CAN bus to individual drive stages.

--Light Loading Permits Embedded Processor

The distributed CANopen servo motor is ability to close its feedback loop locally. This feature lightens the control computer's computation loading. In fact, instead of requiring a high performance computer, the CANopen motion system could be operated by a simple and low-cost embedded microprocessor. The embedded microprocessor could deal with only the mathematical algorithm and CANopen instructions. The development of CANopen motion project becomes easy and quick.

--Portable Control Algorithm

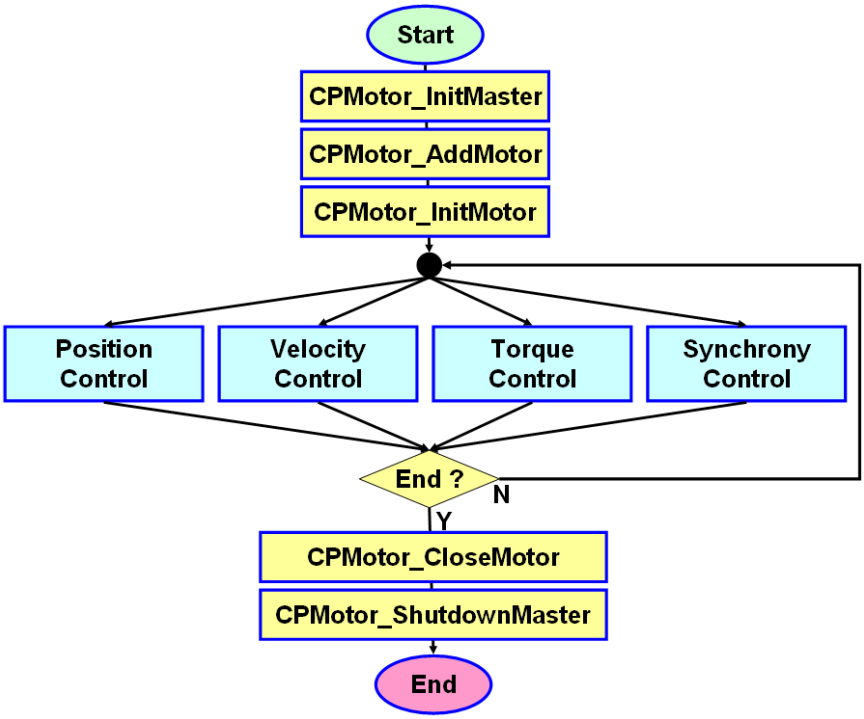
Wiring simplification and motion controller elimination are the grate benefits of CANopen control. Similarly, the CANopen motion libraries and motion control software also reduce a lot of expense indirectly, while speeding new equipment designs to market.

While controlling of multi-axis automation, the control software could focus on the mechanism algorithm without handling the heavy PID computation of the motors. Individual servo motor make their own closed PID loop in response to the CANopen instructions which sent over the CAN bus. Computation demands on the control computer are therefore quite modest.

In summary, the CANopen motion system greatly eliminates the complex cabling along with noise and erroneous connections and simplifies the CANopen master with the embedded computer rather than a high-end PC.

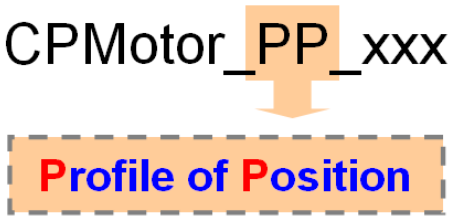
1.3 Flow Chart of motion control

Here shows the general flow char of the motion control.

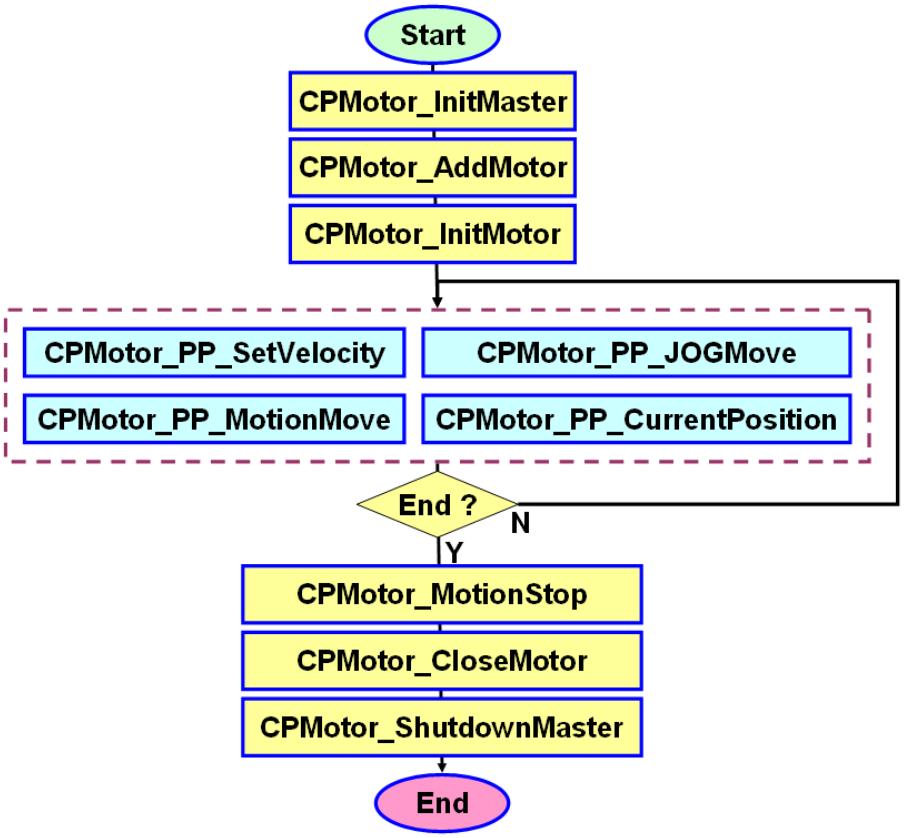


1.3.1 Positon Control

There is a set of position instructions. The user could figure out easily by the function name. Here shows the naming rule of the position instructions.

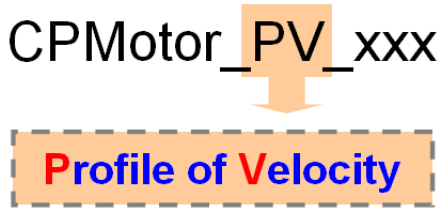


The function name with “PP” means the instruction of the position control. Here shows the position functions and its control flow chart.

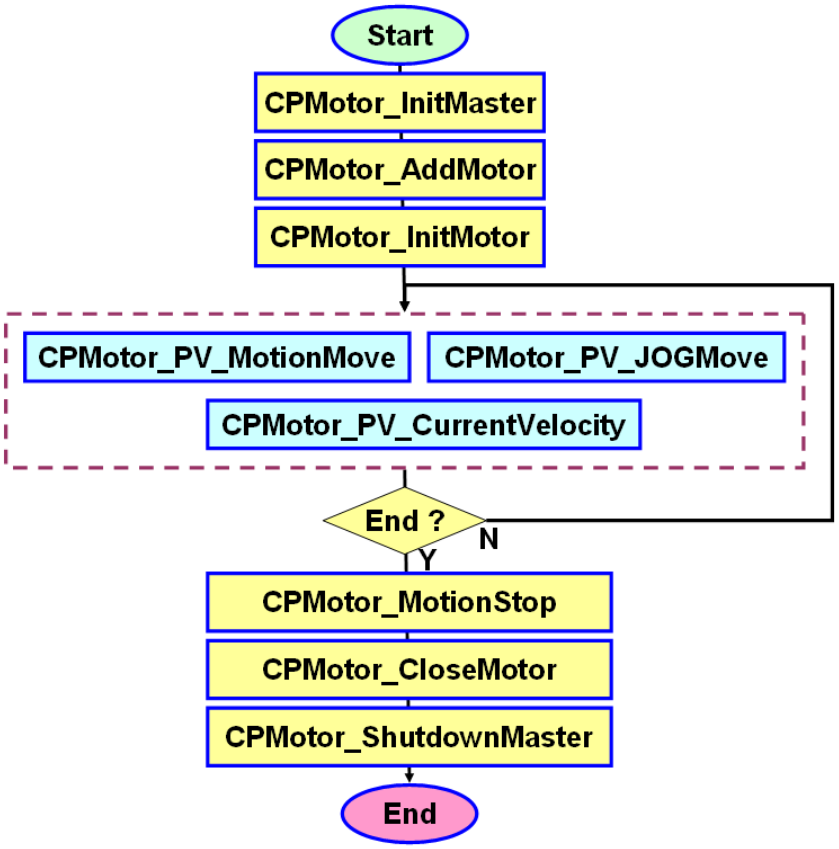


1.3.2 Velocity Control

There is a set of velocity instructions. The user could figure out easily by the function name. Here shows the naming rule of the velocity instructions.



The function name with “PV” means the instruction of the velocity control. Here shows the velocity functions and its control flow chart.



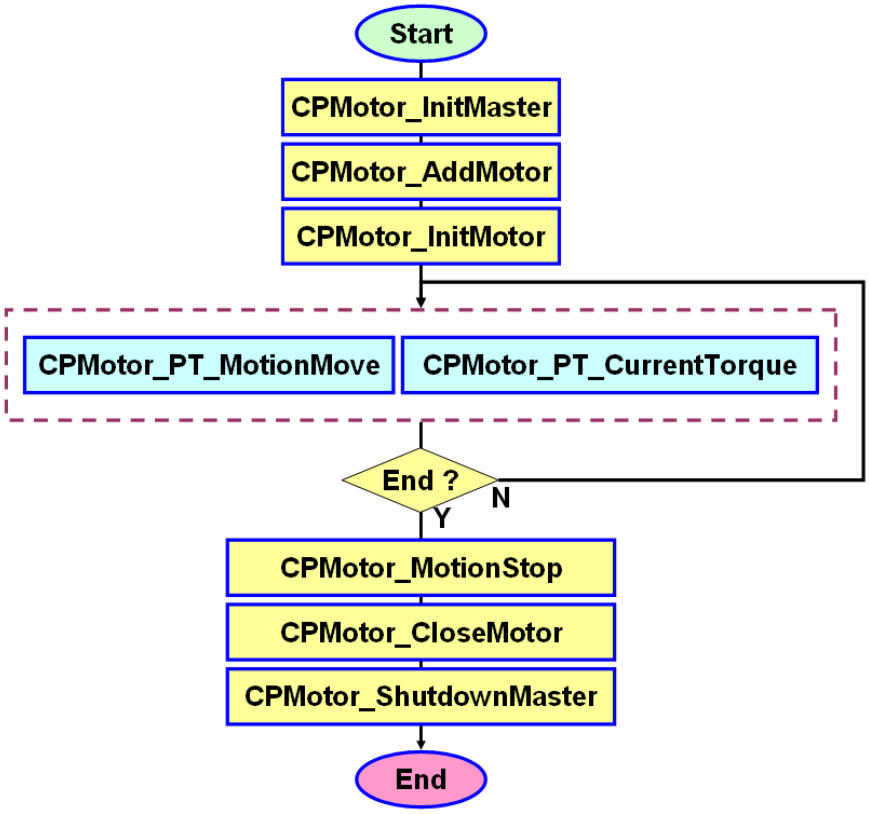
1.3.3 Torque Control

There is a set of torque instructions. The user could figure out easily by the function name. Here shows the naming rule of the torque instructions.

CPMotor_PT_xxx

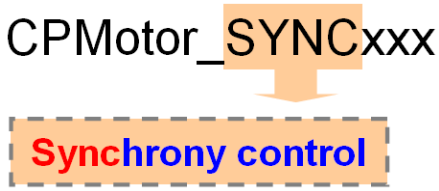
Profile of Torque

The function name with “PT” means the instruction of the torque control. Here shows the torque functions and its control flow chart.

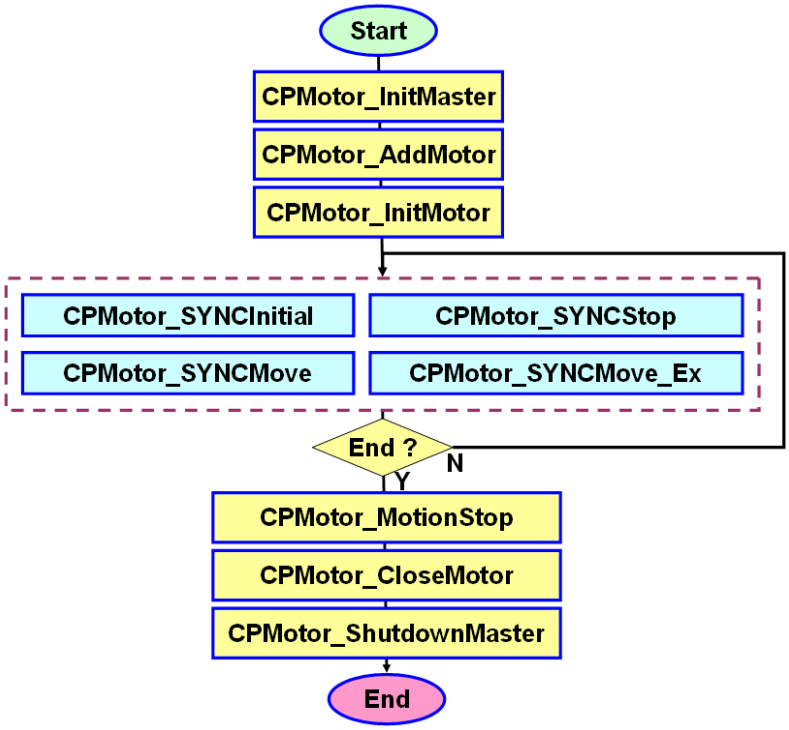


1.3.4 Synchrony Control

There is a set of synchrony instructions. The user could figure out easily by the function name. Here shows the naming rule of the synchrony instructions.

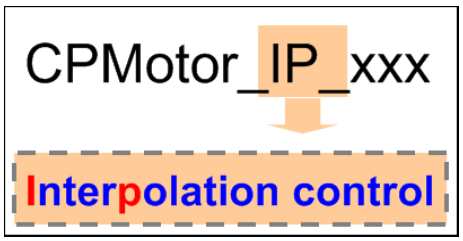


The function name with “SYNC” means the instruction of the synchrony control. Here shows the synchrony functions and its control flow chart.



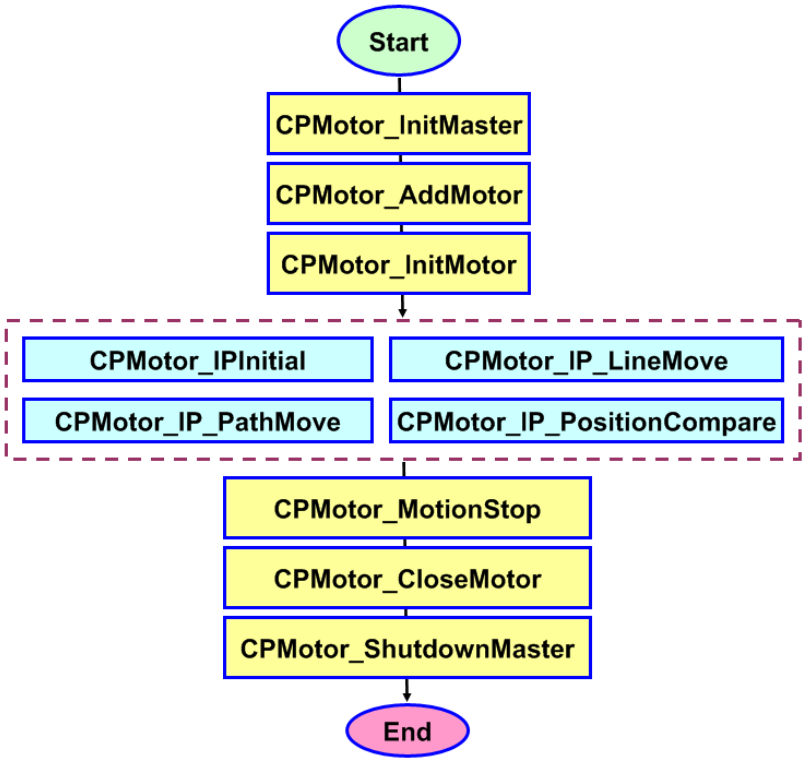
1.3.5 Interpolation Control

There is a set of interpolation instructions. The user could figure out easily by the function name. Here shows the naming rule of the interpolation instructions.



The function name with “IP” means the instruction of the interpolation control. Here shows the interpolation functions and its control flow chart.

*** I-7565-CPM does not support interpolation mode function.**



2. Windows Driver Installation

The motion library is based on the CANopen master driver. First, the users need to install the window driver of the PISO-CPM100U, I-7565-CPM or the I-8123W. After installing the window driver, the users could install the CANopen motion library.

2.1 Driver of PISO-CPM100U, I-7565-CPM and I-8123W

Below are PISO-CPM100U, I-7565-CPM and I-8123W web and driver link:

PISO-CPM100U Web Link
PISO-CPM100U Driver Link
I-7565-CPM Web Link
I-7565-CPM Driver Link
I-8123W Web Link
I-8123W Driver Link

2.2 Driver of the CANopen Motion Library

Below is the CANopen motion library web and driver link:

CANopen Motion Library Web Link
CANopen Motion Library Driver Link

3. Function Description

3.1 CPMotor_DLLVersion

- **Description:**

  This function is used to obtain the version information of CPMotor.DLL.

- **Syntax:**

  WORD CPMotor_DLLVersion (WORD *Ver)

- **Parameter:**

  Ver: [output] the version of the CPMotor.DLL.
  If the Ver is 0x0021 (old display), it means that the version is 2.1.
  If it is 0x2013 (new display), it means that the version is 2.0.1.3.

- **Return:**

  The same as Ver parameter.

3.2 CPMotor_InitMaster

- **Description:**

  The function would initialize the CANopen master firmware and the motion library. Before the users control the motors, they must call this function once.

- **Syntax:**

  WORD CPMotor_InitMaster (BYTE MasterModule,
  BYTE ModuleID, BYTE BaudRate)

- **Parameter:**

  MasterModule: [input] The name of the CANopen master product.

Product	Define Code	Value
I-7565-CPM	MASTER_MODULE_I7565CPM	1
PISO-CPM100U	MASTER_MODULE_PISOCPM100U	2
I-8123W	MASTER_MODULE_I8123W	3

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  BaudRate: [input] The baud rate of the CAN bus. Refer to the list below.

```
#define CANBaud_10K   0
#define CANBaud_20K   1
#define CANBaud_50K   2
#define CANBaud_125K  3
#define CANBaud_250K  4
#define CANBaud_500K  5
#define CANBaud_800K  6
#define CANBaud_1M    7
```

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.3 CPMotor_ShutdownMaster

- **Description:**

  The function would shutdown the CANopen master. It could remove all the motors which have been added and stop all the communicating functions between CANopen master and all CANopen motors. The function must be called before exit the users' application programs.

- **Syntax:**

  WORD CPMotor_ShutdownMaster (BYTE ModuleID)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.4 CPMotor_AddMotor

- **Description:**

The function is used to add CANopen motors into the CANopen master module. It must be called once before using the motors. The motor information will not save in the CANopen master products. The users should call this function every time when starting the user's applications.

- **Syntax:**

```
WORD CPMotor_AddMotor (BYTE ModuleID, BYTE Node,
                       WORD DelayTime, WORD ResTimeout,
                       WORD Motor_Manufacturer)
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

Node: [input] The node ID of the CANopen motor.

DelayTime: [input] The parameter is the shortest time interval between sending two messages from the CANopen master. It can help users to control the timing between two CAN messages. The unit of the parameter is millisecond.

ResTimeout: [input] The timeout value of the responded messages from the CANopen motors. The unit of this parameter is millisecond.

Motor_Manufacturer: [input] The manufacturer of the CANopen motor. There are some differences among various CANopen motors.

Manufact_GeneralMotor	0	generic CANopen motor
Manufact_Delta_A2_Motor	1	Delta ASDA-A2-M motor
Manufact_SANYO_R2_Motor	2	SANYO R2 motor
Manufact_Schneider_LXM23A_Motor	3	Schneider LXM23A motor
Manufact_Festo_Motor	4	FESTO CMMP-AS / CMMS-ST

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.5 CPMotor_CloseMotor

- **Description:**

  The function can reset all the motors to the original status and stop all the communicating functions between CANopen master and all CANopen motors. The function must be called before exit the users' application programs.

- **Syntax:**

  WORD CPMotor_CloseMotor (BYTE ModuleID, BYTE Node)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.6 CPMotor_InitMotor

- **Description:**

  The function can initialize the CANopen motor including the PDO mapping, guarding configuration and etc.

- **Syntax:**

  WORD CPMotor_InitMotor (BYTE ModuleID, BYTE Node,
  WORD PDOTimer = 10)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  PDOTimer: [input] The event timer of the TxPDO. The default event timer is 10(ms).

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.7 CPMotor_Servo_ON_OFF

- **Description:**

  The function is used to enable or disable the servo state.

- **Syntax:**

  WORD CPMotor_Servo_ON_OFF (BYTE ModuleID, BYTE Node,
  BYTE ServoFlag)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  ServoFlag: [input] The switch command of the servo.
  The value “1” is to make the servo ON.
  The value “0” is to make the servo OFF.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.8 CPMotor_IsServoON

- **Description:**

  The function is used to read the servo state.

- **Syntax:**

  WORD CPMotor_IsServoON (BYTE ModuleID, BYTE Node,
  BYTE *IsServoON)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  IsServoON: [output] The servo ON flag.
  The flag “1” means the servo is ON.
  The flag “0” means the servo is OFF.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.9 CPMotor_Servo_Reset

- **Description:**

  The function can reset the servo motor.

- **Syntax:**

  WORD CPMotor_Servo_Reset (BYTE ModuleID, BYTE Node ,
  BYTE ResetType)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  ResetType: [input] The type of the reset command. There are two reset commands to reset the CANopen motor. Some motors support only [NMT Reset] command and others support only [Fault Reset] command. **If the users do not know which command to be used, you can use the [NMT Reset] command.**
  The value “0” means the [Fault Reset] command.
  The value “1” means the [NMT Reset] command.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.10 CPMotor_MotorStatus

- **Description:**

The function is able to get the operating status of the motor which defined as the [statusword] (0x6041) in the CiA 402. The [statusword] includes a lot of useful real-time information from the motor.

- **Syntax:**

```
WORD CPMotor_MotorStatus(BYTE ModuleID, BYTE Node,
                         BYTE *CurrMode, BYTE *IsServoReady,
                         BYTE *IsFault, BYTE *IsWarning,
                         BYTE *IsDone)
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

Node: [input] The node ID of the CANopen motor.

CurrMode: [output] The operation mode of the motor now.

0	Reserved	1	Profile position mode
2	Velocity mode	3	Profile velocity mode
4	Profile torque mode	5	Reserved
6	Homing mode	7	Interpolation position mode

If the users need not to get this information, just pass the NULL pointer into the parameter.

IsServoReady: [output] The servo-ready flag which is retrieved from the bit-0(ready-to-switch-on bit) of the [statusword]. The value “0” means that the servo is not ready. The value “1” means that the servo is ready. If the users need not to get this information, just pass the NULL pointer into the parameter.

IsFault: [output] The [Fault] flag which is retrieved from the bit-3(fault bit)

of the [statusword]. The value “0” means that the servo is normal. The value “1” means the servo is abnormal. If the users need not to get this information, just pass the NULL pointer into the parameter.

IsWarning: [output] The [Warning] flag which is retrieved from the bit-7(Warning bit) of the [statusword]. The value “0” means that the servo is normal. The value “1” means the servo is abnormal. If the users need not to get this information, just pass the NULL pointer into the parameter.

IsDone: [output] The “Done” flag which is retrieved from the bit-10(target-reached bit) of the [statusword]. The value “0” means that the servo does not reach the target. The value “1” means the servo reach the target. If the users need not to get this information, just pass the NULL pointer into the parameter.

- **Return:**

Please refer to the chapter 4 for the function return code.

3.11 CPMotor_MotionStop

- **Description:**

  This function makes the servo motor stop.

- **Syntax:**

  WORD CPMotor_MotionStop(BYTE ModuleID, BYTE Node,
  DWORD DecVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  DecVal: [input] The time (in ms) from 3000 rpm to 0 rpm.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.12 CPMotor_ClearPulse

- **Description:**

  The method is used to set the current position value as the zero position value.

- **Syntax:**

  WORD CPMotor_ClearPulse (BYTE ModuleID, BYTE Node,
  long ZeroPosition)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  ZeroPosition: [input] The zero position value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.13 CPMotor_ServoAlarm

- **Description:**

  This function is used to get the alarm number. If the servo motor is in the alarm state, the function will retrieve the alarm number.

- **Syntax:**

  WORD CPMotor_ ServoAlarm (BYTE ModuleID, BYTE Node,
  WORD *Alarm)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  Alarm: [output] The alarm number. If no alarm occurs, the value is zero.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.14 CPMotor_ResetAlarm

- **Description:**

  This function will clear the alarm notification when the servo motor is in the alarm state.

- **Syntax:**

  WORD CPMotor_ResetAlarm (BYTE ModuleID, BYTE Node)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.15 CPMotor_ServoDOStatus

- **Description:**

This function is only for Delta A2 and Schneider LXM23A series servo. It can help you to obtain the real-time DO status of the servo. The DO value comes from the P0-46 object (Servo Digital Output Status Display).

- **Syntax:**

WORD CPMotor_ServoDOStatus (BYTE ModuleID, BYTE Node,
DWORD *DO_Status)

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

Node: [input] The node ID of the CANopen motor.

DO_Status: [output] The DO value of the servo.

Bit No.	Descriptation
0	SRDY (Servo is ready)
1	SON (Servo ON)
2	ZSPD (Zero speed detection)
3	TSPD (Target speed reached)
4	TPOS (Target position reached)
5	TQL (Torque limit)
6	ALRM (Servo alarm)
7	BRKR (Brake control output)
8	HOME (Homing finished)
9	OLW (Early warning for overload)
10	WARN (When Servo warning, CW, CCW, EMGS, under voltage, Communication error, etc, occurs, DO is ON)
11 ~ 15	Reserved

- **Return:**

Please refer to the chapter 4 for the function return code.

3.16 CPMotor_TorqueLimit

- **Description:**

  This function is only for Delta A2 and Schneider LXM23A series servo. This function will write the maximum torque value into the servo motor. The function uses the parameters which are the P1-02, P3-06 and P4-07. The user could refer to the ASDA-A2 manual for detail.

- **Syntax:**

  WORD CPMotor_TorqueLimit (BYTE ModuleID, BYTE Node,
  WORD TorqueLimit)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  TorqueLimit: [input] The maximum torque value of the servo motor.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.17 CPMotor_HM_HomeMove

- **Description:**

  This function is used to search the home sensor position.

- **Syntax:**

  WORD CPMotor_HM_HomeMove (BYTE ModuleID, BYTE Node,
  BYTE HomeMethod, DWORD MaxV,
  DWORD FinalV, DWORD Acceleration)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  HomeMethod: [input] There are 0 ~ 35 methods to search home sensor.

  MaxV: [input] The Max. speed when searching the home sensor.

  FinalV: [input] The speed when motor is in the home sensor.

  Acceleration: [input] The acceleration value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.18 CPMotor_PP_CurrentPosition

- **Description:**

  This function is used to obtain the current position of the servo motor. The function is valid in the position mode.

- **Syntax:**

  WORD CPMotor_PP_CurrentPosition (BYTE ModuleID, BYTE Node,
  long *PositionVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  PositionVal: [output] The current position of the servo motor.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.19 CPMotor_PP_MotionMove

- **Description:**

  This function makes the servo motor to move to the specific position (absolute mode) or move the specific distance (relative mode).

- **Syntax:**

  WORD CPMotor_PP_MotionMove (BYTE ModuleID, BYTE Node,
  BYTE IsChangeImmediately, BYTE IsAbsolute, long TargetPosition,
  DWORD AccVal, DWORD Velocity, DWORD DecVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  IsChangeImmediately: [input] Indicate whether move right now or not.
  0: When the motor is moving, this command changes nothing.
  0: When the motor is static, the motor will start moving.
  1: When the motor is moving, it will move to the new position right now.
  1: When the motor is static, the motor will start moving.

  IsAbsolute: [input] The position mode. The value is “0” means the [TargetPposition] value is relative. The [TargetPosition] value could be views as the moving distance. The value is “1” means the [TargetPosition] is absolute.

  TargetPosition: [input] The target position. When the [IsAbsolute] is absolute position mode, The [TargetPosition] is the absolute position. When the [IsAbsolute] is relative position mode, The [TargetPosition] is the distance value.

  AccVal: [input] The acceleration value.

  Velocity: [input] The max. speed of this motion.

DecVal: [input] The deceleration value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.20 CPMotor_PP_SetVelocity

- **Description:**

  When the motor is moving, this function can change the velocity immediately. This function is valid in position mode.

- **Syntax:**

  WORD CPMotor_PP_SetVelocity (BYTE ModuleID, BYTE Node,
  DWORD AccVal, DWORD Velocity, DWORD DecVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  AccVal: [input] The acceleration value.

  Velocity: [input] The new speed of servo motor.

  DecVal: [input] The deceleration value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.21 CPMotor_PP_JOGMove

- **Description:**

  This function is used for JOG moving. The users can run a specific short distance to adjust the position of the servo motor.

- **Syntax:**

  WORD CPMotor_PP_JOGMove (BYTE ModuleID, BYTE Node,
  long JogPulse, DWORD ACCVal,
  DWORD JogVelocity, DWORD DECVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  JogPulse: [input] The jog distance. If the [JogPulse] is positive value, the motor will move forward. If the [JogPulse] is negative value, the motor will move backward.

  ACCVal: [input] The acceleration value.

  JogVelocity: [input] The max. speed of this JOG motion.

  DECVal: [input] The deceleration value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.22 CPMotor_PV_JOGMove

- **Description:**

  This function is used for moving with specific JOG speed. This function will start moving the motor until the “CPMotor_MotionStop” was called. The users can move the motor with the specific small speed.

- **Syntax:**

  WORD CPMotor_PV_JOGMove (BYTE ModuleID, BYTE Node,
  DWORD ACCVal, long JogVelocity, DWORD DECVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  ACCVal: [input] The acceleration value.

  JogVelocity: [input] The maximum speed of this JOG motion. The unit is 0.1 rpm. If it is positive value, the motor will move forward. If it is negative value, the motor will move backward.

  DECVal: [input] The deceleration value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.23 CPMotor_PV_CurrentVelocity

- **Description:**

  This function is used to obtain the current velocity of the servo motor. The function is valid in the velocity mode.

- **Syntax:**

  WORD CPMotor_PV_CurrentVelocity (BYTE ModuleID, BYTE Node,
  long *VelocityVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  VelocityVal: [output] The current velocity of the servo motor. The unit is 0.1 rpm. If it is positive value, the motor is moving forward. If it is negative value, the motor is moving backward.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.24 CPMotor_PV_MotionMove

- **Description:**

  This function makes the servo motor to move in the specific velocity. The function will make the servo motor into the velocity mode.

- **Syntax:**

  WORD CPMotor_PV_MotionMove (BYTE ModuleID, BYTE Node, DWORD AccVal, long TargetVelocity, DWORD DecVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  ACCVal: [input] The acceleration value.

  TargetVelocity: [input] The maximum speed of this motion. The unit is 0.1rpm.

  DECVal: [input] The deceleration value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.25 CPMotor_PT_CurrentTorque

- **Description:**

  This function is used to obtain the current torque of the servo motor. The function is valid in the torque mode.

- **Syntax:**

  WORD CPMotor_PT_CurrentTorque (BYTE ModuleID, BYTE Node,
  long *TorqueVal)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  TorqueVal: [output] The torque value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.26 CPMotor_PT_MotionMove

- **Description:**

  This function makes the servo motor to move in the specific torque. The function will make the servo motor into the torque mode.

- **Syntax:**

  WORD CPMotor_PT_MotionMove (BYTE ModuleID, BYTE Node,
  DWORD TorqueSlope, long TargetTorque)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  TorqueSlop: [input] The acceleration of the torque.

  TargetTorque: [input] The target torque value.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.27 CPMotor_SYNCInitial

- **Description:**

  The “SYNC move” means all the specific motors will start moving at the same time. Before the synchronic motion, the user must call this function once. This function will initialize the parameters for the synchronic motion.

- **Syntax:**

  WORD CPMotor_SYNCInitial (BYTE ModuleID, BYTE Node)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.28 CPMotor_SYNCMove_Ex

- **Description:**

After calling “CPMotor_SYNCInitial” with each servo motors, this function will make all servo motors move at the same time and reach their target position at the same time. The motor which moves the longest distance will move with the “MaxVelocity” velocity. Other motors will reduce their velocity by their moving distance.

- **Syntax:**

```
WORD CPMotor_SYNCMove_Ex (BYTE ModuleID,   BYTE NodeNum,
                          BYTE* Node_IDs, long* TargetPositions,
                          DWORD MaxVelocity, BYTE IsAbsolute)
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

NodeNum: [input] The amount of all servo motors in this synchronic motion.

Node_IDs: [input array] The node IDs of all the CANopen servo motor. Provide the node ID array like below.
Node_IDs[0] = the ID of motor #1.
Node_IDs[1] = the ID of motor #2.
…
Node_IDs[NodeNum-1] = the ID of motor #n.

TargetPositions: [input array] The target positions of all the CANopen motor. Provide the target position array like below.
TargetPositions[0] = the target position of motor #1
TargetPositions[1] = the target position of motor #2
…
TargetPositions[NodeNum-1] = the target position of motor #n

MaxVelocity: [input] The max. velocity for the longest moving distance.

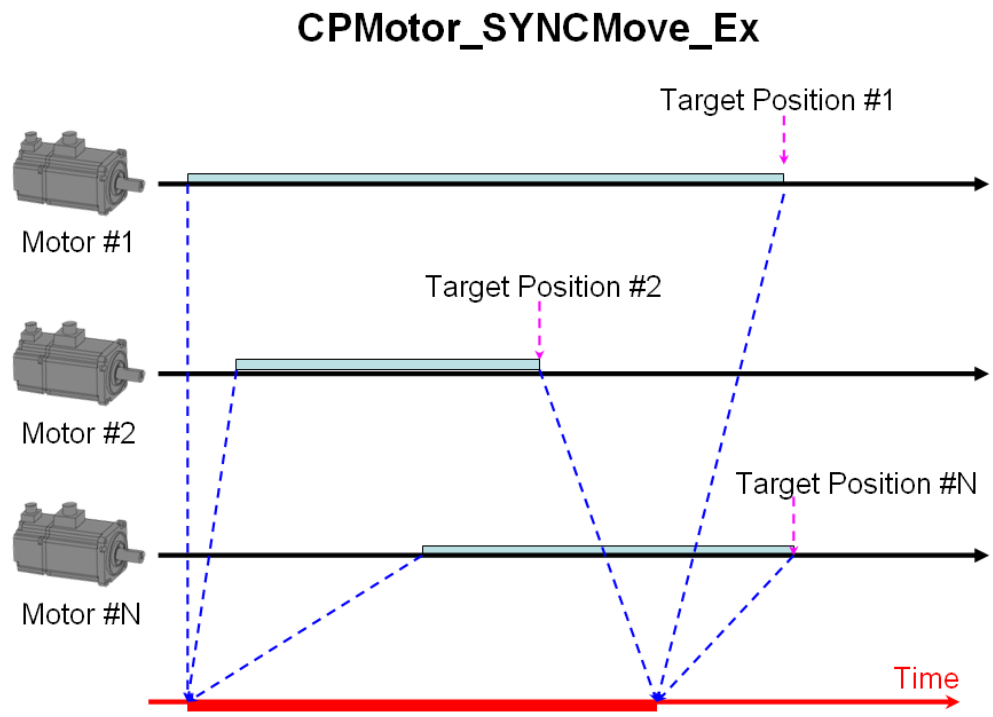
IsAbsolute: [input] The value is “0” means the position value is relative. The position value could be views as the moving distance. The value is “1” means the position is absolute.

- **Return:**

Please refer to the chapter 4 for the function return code.

- **Illustration:**

Here shows how the motor works with this function.



Every motor could start moving at the same time and reach the target at the same time.

3.29 CPMotor_SYNCMove

- **Description:**

  This function will make all servo motors move at the same time. All servo motors could move with their own velocity and reach their target positions individually at different time.

- **Syntax:**

  WORD CPMotor_SYNCMove (BYTE ModuleID, BYTE NodeNum,
  BYTE* Node_IDs, long* TargetPositions, DWORD* AccVals,
  DWORD* Velocities, DWORD* DecVals, BYTE IsAbsolute)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  NodeNum: [input] The amount of all servo motors in this synchronic motion.

  Node_IDs: [input array] The node IDs of all the CANopen servo motor. Provide the node ID array like below.
  Node_IDs[0] = the ID of motor #1.
  Node_IDs[1] = the ID of motor #2.
  …
  Node_IDs[NodeNum-1] = ID of motor #n.

  TargetPositions: [input array] The target positions of all the CANopen motor. Provide the target position array like below.
  TargetPositions[0] = the target position of motor #1
  TargetPositions[1] = the target position of motor #2
  …
  TargetPositions[NodeNum-1] = the target position of motor #n

AccVals: [input array] The acceleration values of all the CANopen servo motor. Provide the acceleration array like below.
AccVals[0] = the acceleration of motor #1.
AccVals[1] = the acceleration of motor #2.
…
AccVals[NodeNum-1] = the acceleration of motor #n.

Velocities: [input array] The velocity values of all the CANopen servo motor. Provide the velocity array like below.
Velocities[0] = the velocity of motor #1.
Velocities[1] = the velocity of motor #2.
…
Velocities[NodeNum-1] = the velocity of motor #n.

DecVals: [input array] The deceleration values of all the CANopen servo motor. Provide the acceleration array like below.
DecVals[0] = the deceleration of motor #1.
DecVals[1] = the deceleration of motor #2.
…
DecVals[NodeNum-1] = the deceleration of motor #n.

IsAbsolute: [input] The value is “0” means the position value is relative. The position value could be views as the moving distance. The value is “1” means the position is absolute.

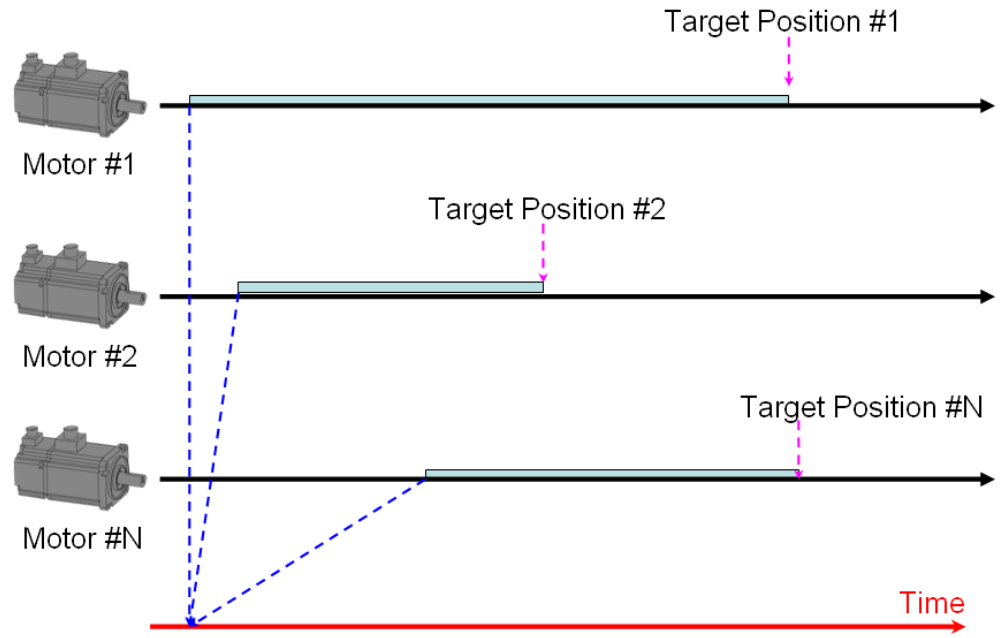
- ## Return:

Please refer to the chapter 4 for the function return code.

- ## Illustration:

Here shows how the motor works with this function.

CPMotor_SYNCMove



Every motor could start moving at the same time and reach the target at different time.

3.30 CPMotor_SYNCStop

- **Description:**

  This function will make all servo motors which are synchronic moving stop at the same time.

- **Syntax:**

  WORD CPMotor_SYNCStop (BYTE ModuleID, BYTE NodeNum,
  BYTE* Node_IDs)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  NodeNum: [input] The amount of all servo motors in this synchronic

  motion.

  Node_IDs: [input array] The node IDs of all the CANopen servo motor.
  Provide the node ID array like below.
  Node_IDs[0] = the ID of motor #1.
  Node_IDs[1] = the ID of motor #2.
  …
  Node_IDs[NodeNum-1] = the ID of motor #n.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.31 CPMotor_IPInitial

- **Description:**

  Before using the interpolation function, the initial function needs to be called at least one time. This will make the servo motor into the interpolation mode. **This function could not be used with the I-7565-CPM module.**

- **Syntax:**

  WORD CPMotor_IPInitial (BYTE ModuleID, BYTE Node)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.32 CPMotor_IP_LineMove

- **Description:**

Before using the line interpolation function, the initial function needs to be called at least one time for every motor which participates in the interpolation. This function will make every servo motor to move synchronously with the line interpolation. This function is very similar with the CPMotor_SYNCMove_Ex API. All motors will start moving at the same time and go to their targets. Finally, all motors will stop at the same time. During all motors moving with the line or pat interpolation, the user could call this function to assign another line interpolation. This function will queue max. 1000 line commands when all motors are moving. **This function could not be used with the I-7565-CPM module.**

- **Syntax:**

```
WORD CPMotor_IP_LineMove(BYTE ModuleID, BYTE NodeNum,
        BYTE Node_IDs[], long Node_TargetPos[], BYTE Rel_Abs[],
        DWORD Velocity, DWORD AccTime, DWORD DecTime,
        BYTE IsWaitForPreviousMoving)
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

NodeNum: [input] The amount of all servo motors in interpolation mode.

Node_IDs: [input array] The node IDs of all the CANopen servo motor. Provide the node ID array like below.
Node_IDs[0] = the ID of motor #1.
Node_IDs[1] = the ID of motor #2.
…
Node_IDs[NodeNum-1] = ID of motor #n.

Node_TargetPos: [input array] The target positions of all the CANopen motor. Provide the target position array like below.
Node_TargetPos[0] = the target position of motor #1
Node_TargetPos[1] = the target position of motor #2

…
Node_TargetPos[NodeNum-1] = the target position of motor #n

Rel_Abs: [input array] The value is “0” means the position value is relative. The position value could be views as the moving distance. The value is “1” means the position is absolute. Provide the optional position value array like below.
Rel_Abs[0] = the optional position of motor #1
Rel_Abs[1] = the optional position of motor #2
…
Rel_Abs[NodeNum-1] = the optional position of motor #n

Velocity: [input] The max. speed of this line interpolation motion.

AccTime: [input] The duration(ms) of the acceleration.

DecTime: [input] The duration(ms) of the deceleration.

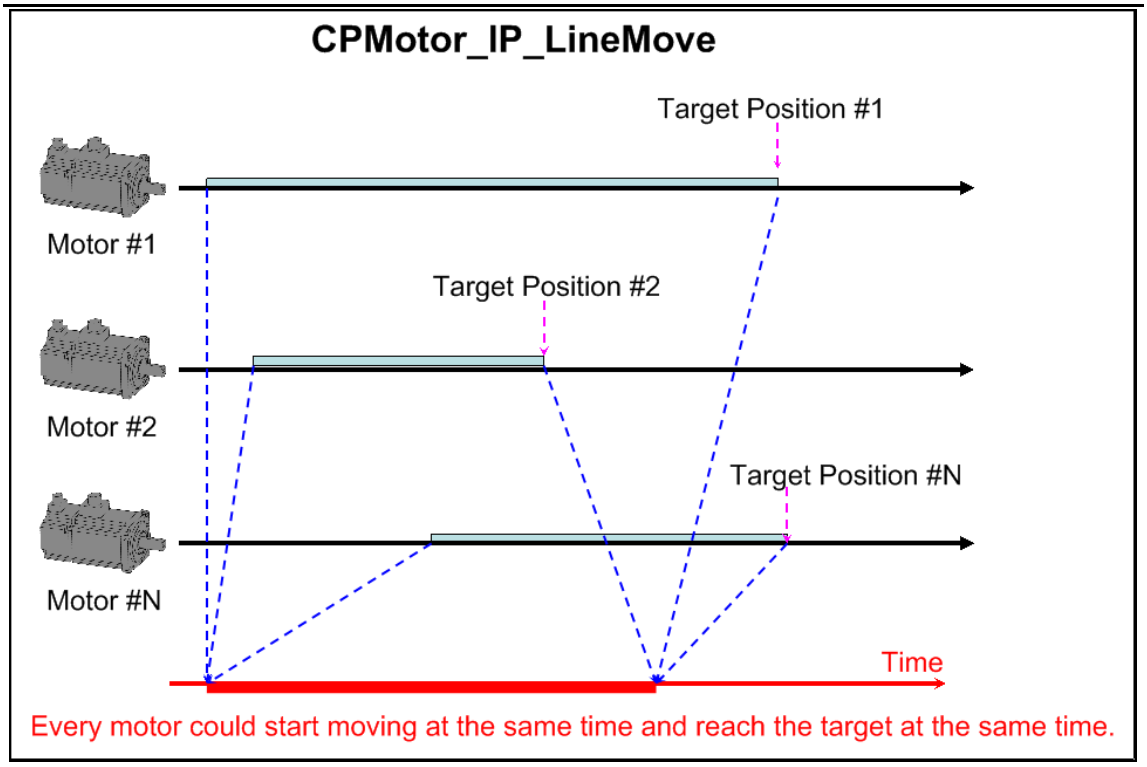
IsWaitForPreviousMoving: [input] The optional waiting flag which means that this line interpolation will start at the end of the previous moving or start right now. The value is “1” means it will wait for the previous moving. If the motor is static, the motor will run immediately. The value is “0” means it will go to the target immediately and regardless of the previous moving.

- ## Return:

Please refer to the chapter 4 for the function return code.

- ## Illustration:

Here shows how the motor works with this function.


CPMotor_IP_LineMove
Target Position #1
Motor #1
Target Position #2
Motor #2
Target Position #N
Motor #N
Time
Every motor could start moving at the same time and reach the target at the same time.

3.33 CPMotor_IP_PathMove

- **Description:**

Before using the path interpolation function, the initial function needs to be called at least one time for every motor which participates in the interpolation. This function will make every servo motor to move synchronously with the path interpolation. The users could provide the path which includes a bulk sequential target positions. This function will make all motors go to their target positions sequentially in every 5 ms. During all motors moving with the line or pat interpolation, the user could call this function to assign another path interpolation. This function will queue max. 1000 path commands when all motors are moving. **This function could not be used with the I-7565-CPM module.**

- **Syntax:**

```
WORD CPMotor_IP_PathMove(BYTE ModuleID, BYTE NodeNum,
                         BYTE Node_IDs[], BYTE Rel_Abs[], long StepNum,
                         long Node_PathSteps[], BYTE IsWaitForPreviousMoving)
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

NodeNum: [input] The amount of all servo motors in interpolation mode.

Node_IDs: [input array] The node IDs of all the CANopen servo motor. Provide the node ID array like below.
Node_IDs[0] = the ID of motor #1.
Node_IDs[1] = the ID of motor #2.
…
Node_IDs[NodeNum-1] = ID of motor #n.

Rel_Abs: [input array] The value is “0” means the position value is relative. The position value could be views as the moving distance. The value is “1” means the position is absolute. Provide the optional position value array like below.
Rel_Abs[0] = the optional position of motor #1

Rel_Abs[1] = the optional position of motor #2

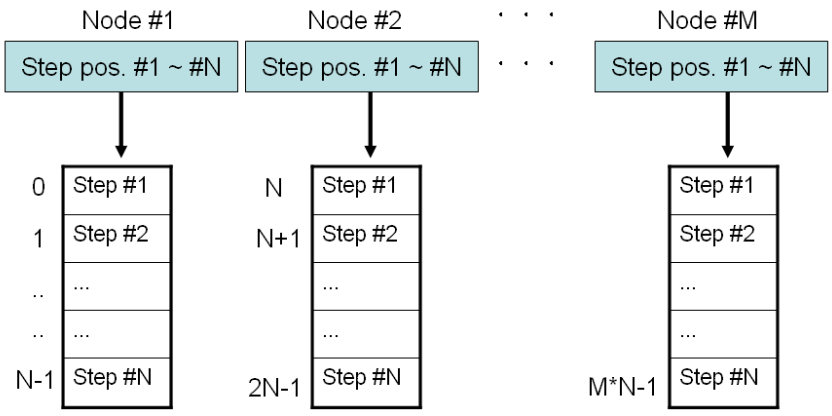
…

Rel_Abs[NodeNum-1] = the optional position of motor #n

StepNum: [input] The amount of the steps.

Node_PathSteps: [input array] The all steps of all motors which participate this path interpolation. Here is the illustration of the array.

array size = NodeNum x StepNum (Motor amount x Step amount)



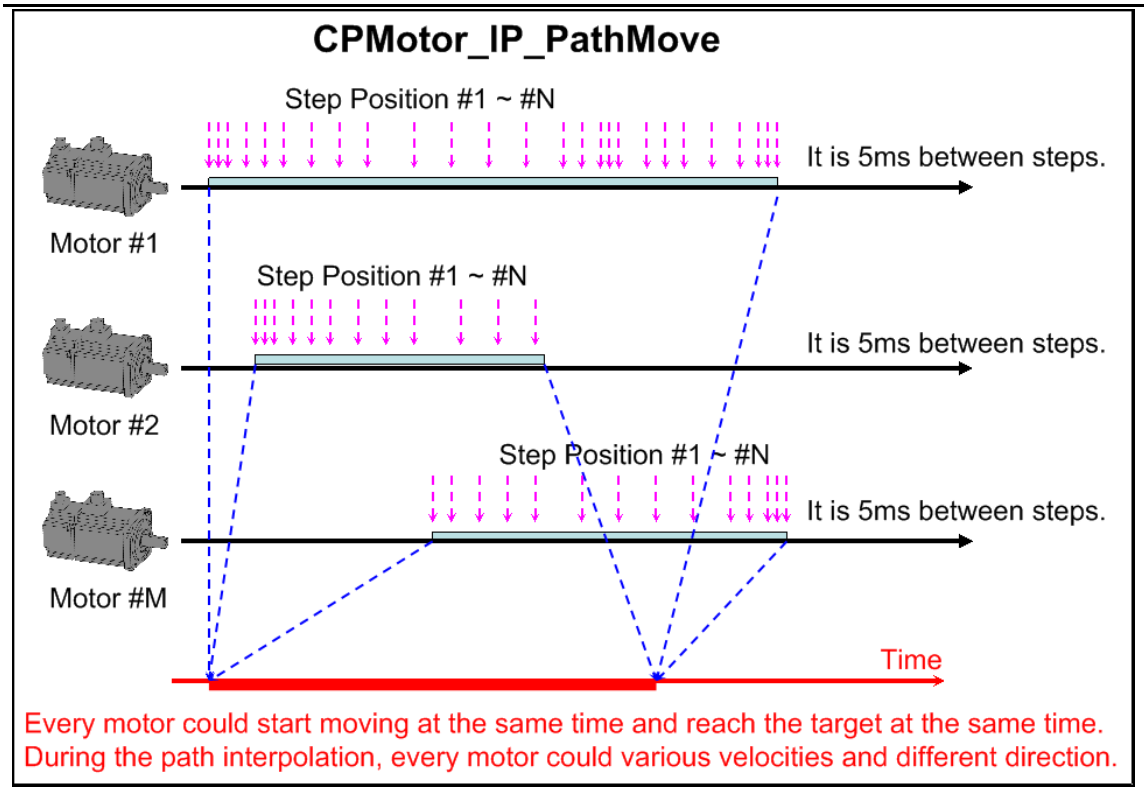
IsWaitForPreviousMoving: [input] The optional waiting flag which means that this line interpolation will start at the end of the previous moving or start right now. The value is “1” means it will wait for the previous moving. If the motor is static, the motor will run immediately. The value is “0” means it will go to the target immediately and regardless of the previous moving.

- ## Return:

Please refer to the chapter 4 for the function return code.

- ## Illustration:

Here shows how the motor works with this function.


CPMotor_IP_PathMove
Step Position #1 ~ #N
It is 5ms between steps.
Motor #1
Step Position #1 ~ #N
It is 5ms between steps.
Motor #2
Step Position #1 ~ #N
It is 5ms between steps.
Motor #M
Time
Every motor could start moving at the same time and reach the target at the same time.
During the path interpolation, every motor could various velocities and different direction.

3.34 CPMotor_IP_PositionCompare

- **Description:**

  This function can compare current position of a motor to specific position. When current position is arrived the specific position, a set DO bit will be triggered. **This function could not be used with the I-7565-CPM module**.

  *** The DO bit be triggered is RxPDO1 data of a slave node.**

- **Syntax:**

  WORD CPMotor_IP_PositionCompare(BYTE ModuleID, BYTE Node,
  BYTE Position_Cnt, long Position_Array[],
  BYTE Target_Node, BYTE Trigger_Bit[])

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  Position_Cnt: [input] Total number of position point need be compared.

  Position_Array: [input] Save the all positions to be compared to the Position_Array array parameter.

  Target_Node:[input] The triggered device node ID.

  Trigger_Bit: [input] The triggered data bit array of RxPDO1. Every data bit is mapped to relative position value in Psition_Array.

- **Return:**

  Please refer to the chapter 4 for the function return code.

- **Illustration:**

  Here is a example for how to use the function and its executes result.

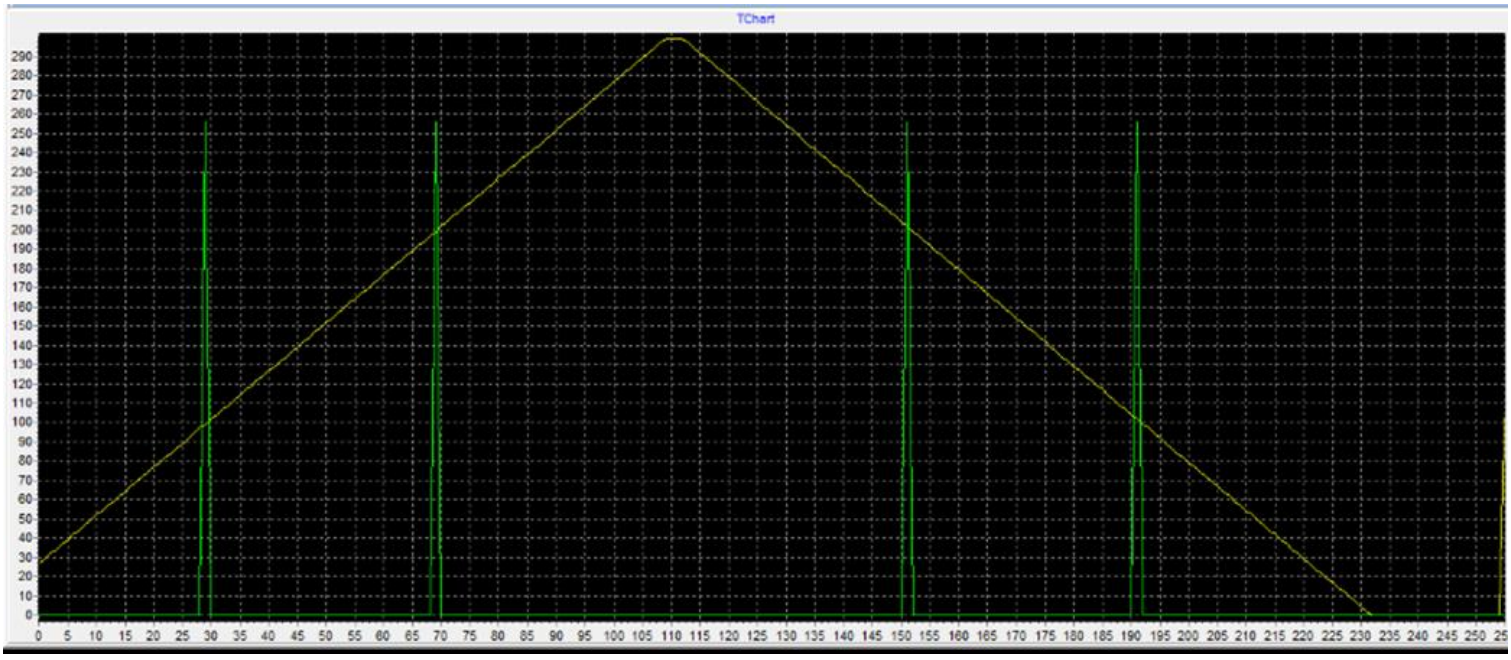
```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
position_array[0] = 100;
position_array[1] = 200;
// When the motor arrives position 100 and 200, a DO will be triggered.

trigger_bit[0] = 8;
trigger_bit[1] = 8;
// The be triggered DO is bit 8 (base 0) at RxPDO1.

CPMotor_IP_PositionCompare(Slot, Motor_Id, 2, position_array, target_id ,
                                                    trigger_bit);
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

The executes result as below (X-axis is time, Y-axis is data value).

The motor is moved from position 0 to 300 and then back to 0. When the motor moves to position 100, the target DO bit 8 (equals data 256) is on. And When the motor moves to position 200, the target DO bit 8 is on again. The same behavior of the DO after the motor back to position 200 and 100.



3.35 CPMotor_IP_Stop

- **Description:**

When user needs let all motors stop and leave interpolation mode, the CPMotor_IP_Stop function is useful. After user called the function, the all motors will start decelerate to a stop follow the DecTime parameter. And then the all motors will leave interpolation mode. If user want these motors re-into interpolation mode, he needs call CPMotor_IPInitial function again. **This function could not be used with the I-7565-CPM module**.

- **Syntax:**

WORD CPMotor_IP_Stop(BYTE ModuleID, DWORD DecTime)

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

DecTime: [input] The deceleration time of all servo motors in line or path interpolation motion.

- **Return:**

Please refer to the chapter 4 for the function return code.

3.36 CPMotor_HSIP_Initial

- **Description:**

The function is like CPMotor_IPInitial, but the CPMotor_HSIP_Initial can initial all motors to interpolation mode at once time. **Note that the CPMotor_HSIP series functions can't use with CPMotor_IP series functions. This function could not be used with the I-7565-CPM module.**

- **Syntax:**

```
WORD CPMotor_HSIP_Initial (BYTE ModuleID, BYTE NodeNum,
                           BYTE Node_IDs[])
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

NodeNum: [input] The amount of all servo motors in interpolation mode.

Node_IDs: [input array] The node IDs of all the CANopen servo motor.
Provide the node ID array like below.
Node_IDs[0] = the ID of motor #1.
Node_IDs[1] = the ID of motor #2.
…
Node_IDs[NodeNum-1] = ID of motor #n.

- **Return:**

Please refer to the chapter 4 for the function return code.

3.37 CPMotor_HSIP_PathMove

- **Description:**

The function is similar to function CPMotor_IP_PathMove, but the CPMotor_HSIP_PathMove has no path buffer and can only receive few position points once time. Instead, these position points can output to the motor faster and on time. **Note that the CPMotor_HSIP series functions can't use with CPMotor_IP series functions. This function could not be used with the I-7565-CPM module.**

- **Syntax:**

```
WORD CPMotor_HSIP_PathMove (BYTE ModuleID, BYTE NodeNum,
                            BYTE Node_IDs[], long StepNum,
                            long Node_PathSteps[])
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

NodeNum: [input] The amount of all servo motors in interpolation mode.

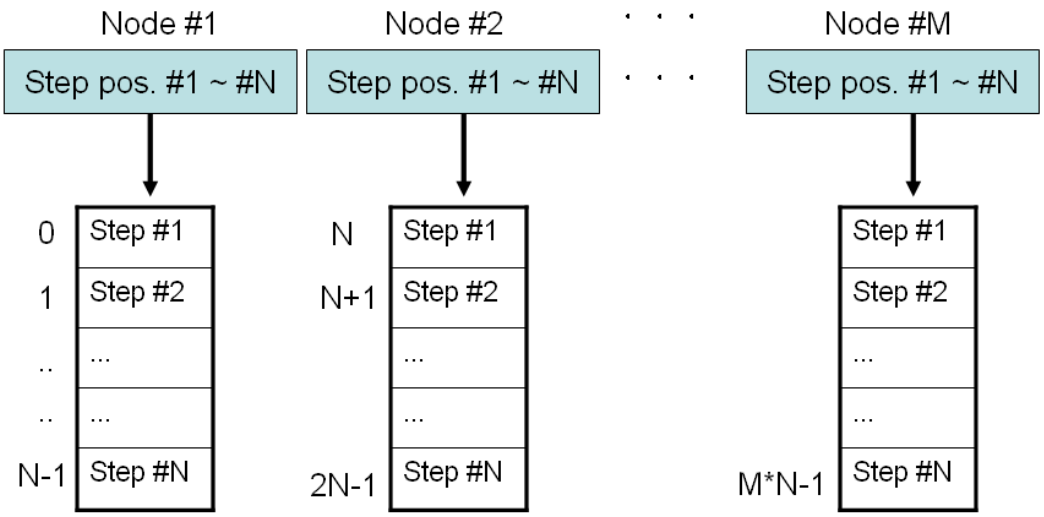
Node_IDs: [input array] The node IDs of all the CANopen servo motor. Provide the node ID array like below.
Node_IDs[0] = the ID of motor #1.
Node_IDs[1] = the ID of motor #2.
…
Node_IDs[NodeNum-1] = ID of motor #n.

StepNum: [input] The amount of the steps. Because the CPMotor_HSIP_PathMove has no path buffer, the amount of the steps at once input will be limited as below.

Motor Num	1 ~ 4	5	6	7	8
StepNum Max	15	12	10	8	7
Motor Num	9 ~ 10	11 ~ 12	13 ~ 15	16	--
StepNum Max	6	5	4	3	--

Node_PathSteps: [input array] The all steps of all motors which participate this path interpolation. **Note that the position of the function is absolute.** Here is the illustration of the array.

array size = NodeNum x StepNum (Motor amount x Step amount)



- **Return:**

  Please refer to the chapter 4 for the function return code.

3.38 CPMotor_HSIP_Stop

- **Description:**

  The function is similar to function CPMotor_IP_Stop, but the CPMotor_HSIP_Stop is use for the motors initialled by CPMotor_HSIP_Initial. **This function could not be used with the I-7565-CPM module**.

- **Syntax:**

  WORD CPMotor_HSIP_Stop(BYTE ModuleID, BYTE NodeNum,
  BYTE Node_IDs[])

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is board number for PISO-CPM100U and slot number for I-8123W.

  NodeNum: [input] The amount of all servo motors in interpolation mode.

  Node_IDs: [input array] The node IDs of all the CANopen servo motor.
  Provide the node ID array like below.
  Node_IDs[0] = the ID of motor #1.
  Node_IDs[1] = the ID of motor #2.
  …
  Node_IDs[NodeNum-1] = ID of motor #n.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.39 CPMotor_SetKeypad

- **Description:**

  This function is only for Delta A2 and Schneider LXM23A series servo. This function is to set the parameter in the ASDA-A2 and LXM23A servo motor. The parameter is like the format “Pa-bb”.

- **Syntax:**

  WORD CPMotor_ SetKeypad (BYTE ModuleID, BYTE Node,
  BYTE ParamIndex, BYTE ParamSubIndex,
  BYTE DataLen, DWORD SetParamData)

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  ParamIndex: [input] The index of the parameter. For example, the “P3-06” parameter, the [ParameterIndex] = 3.

  ParamSubIndex: [input] The sub-index of the parameter. For example, the “P3-06” parameter, the [ParameterSubIndex] = 6.

  DataLen: [input] The data length (in byte) of the parameter. The maximum data length is 4 bytes.

  SetParamData: [input] The data of the parameter.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.40 CPMotor_GetKeypad

- **Description:**

This function is only for Delta A2 and Schneider LXM23A series servo. This function is to retrieve the parameter in the ASDA-A2 and Schneider LXM23A servo motor. The parameter is like the format “Pa-bb”.

- **Syntax:**

```
WORD CPMotor_GetKeypad (BYTE ModuleID, BYTE Node,
                        BYTE ParamIndex, BYTE ParamSubIndex,
                        BYTE *DataLen, DWORD *GetParamData)
```

- **Parameter:**

ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

Node: [input] The node ID of the CANopen motor.

ParamIndex: [input] The index of the parameter. For example, the “P3-06” parameter, the [ParameterIndex] = 3.

ParamSubIndex: [input] The sub-index of the parameter. For example, the “P3-06” parameter, the [ParameterSubIndex] = 6.

DataLen: [output] The data length (in byte) of the parameter. The maximum data length is 4 bytes.

GetParamData: [output] The data of the parameter.

- **Return:**

Please refer to the chapter 4 for the function return code.

3.41 CPMotor_SetUserObject_1

- **Description:**

  This function is for user define setting PDO message. Sometimes some data is very important for user but not be mapped in PDO message. User just can read the data through SDO. But SDO is slower than PDO and may reduce performance of user's program. Through function CPMotor_SetUserObject_1 can map SDO object data into PDO message **TxPDO3** if the SDO object is allowed mapping.

- **Syntax:**

  WORD CPMotor_SetUserObject_1 (BYTE ModuleID, BYTE Node,
  BYTE Obj_Num, BYTE Obj_Len[],
  WORD Obj_Idx[], BYTE Obj_Subidx[])

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  Obj_Num: [input] The total object amount that want to be mapped.

  Obj_Len: [input] The byte length of every mapping object.

  Obj_Idx: [input] The object index of every mapping object.

  Obj_Subidx: [input] The object sub-index of every mapping object.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.42 CPMotor_SetUserObject_2

- **Description:**

  This function is similar CPMotor_SetUserObject_1 but the function CPMotor_SetUserObject_2 is mapping for **TxPDO4**.

- **Syntax:**

```
WORD CPMotor_SetUserObject_2 (BYTE ModuleID, BYTE Node,
                              BYTE Obj_Num, BYTE Obj_Len[],
                              WORD Obj_Idx[], BYTE Obj_Subidx[])
```

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  Obj_Num: [input] The total object amount that want to be mapped.

  Obj_Len: [input] The byte length of every mapping object.

  Obj_Idx: [input] The object index of every mapping object.

  Obj_Subidx: [input] The object sub-index of every mapping object.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.43 CPMotor_GetUserData_1

- **Description:**

  After use CPMotor_SetUserObject_1 to map PDO message, user can use the function CPMotor_GetUserData_1 to get the mapping data.

- **Syntax:**

  WORD CPMotor_GetUserData_1 (BYTE ModuleID, BYTE Node,
  BYTE* DLen, BYTE RData[])

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  DLen: [output] The parameter will response byte length of the mapping PDO message. It will be equal to the sum of the parameters Obj_Len of function CPMotor_SetUserObject_1.

  RData: [output] Response data setting by CPMotor_SetUserObject_1.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.44 CPMotor_GetUserData_2

- **Description:**

  After use CPMotor_SetUserObject_2 to map PDO message, user can use the function CPMotor_GetUserData_2 to get the mapping data.

- **Syntax:**

```
WORD CPMotor_GetUserData_2 (BYTE ModuleID, BYTE Node,
                            BYTE* DLen, BYTE RData[])
```

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  DLen: [output] The parameter will response byte length of the mapping PDO message. It will be equal to the sum of the parameters Obj_Len of function CPMotor_SetUserObject_2.

  RData: [output] Response data setting by CPMotor_SetUserObject_2.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.45 CPMotor_SDOReadData

- **Description:**

  The function CPMotor_SDOReadData is useful to the SDO upload from a specified slave. When users use this function, pass the slave device node ID, object index and object subindex into this function. This function supports only expedition mode (less then 4-byte data length).

- **Syntax:**

  WORD CPMotor_SDOReadData (BYTE ModuleID, BYTE Node,
  WORD Index, BYTE SubIndex,
  DWORD *DataLen, BYTE bData[])

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  Index: [input] Object index of object dictionary of slave devices.

  SubIndex: [input] Object subindex of object dictionary of slave devices.

  DataLen: [output] Total read data length.

  bData: [output] SDO data respond from the specified slave device.

- **Return:**

  Please refer to the chapter 4 for the function return code.

3.46 CPMotor_SDOWriteData

- **Description:**

  The function CPMotor_SDOWriteData can send out a SDO message to the specified salve device. This procedure is also called download SDO protocol. The parameter node of the function is used to point which slave device will receive this SDO message. Because the data length of each object stored in object dictionary is different, users need to know the data length when writing the object of the object dictionary of the specified slave devices. This function supports only expedition mode (less then 4-byte data length).

- **Syntax:**

  WORD CPMotor_SDOWriteData (BYTE ModuleID, BYTE Node,
  WORD Index, BYTE SubIndex,
  DWORD DataLen, BYTE bData[])

- **Parameter:**

  ModuleID: [input] The CANopen master module id. It is COM port number for I-7565-CPM, board number for PISO-CPM100U and slot number for I-8123W.

  Node: [input] The node ID of the CANopen motor.

  Index: [input] Object index of object dictionary of slave devices.

  SubIndex: [input] Object subindex of object dictionary of slave devices.

  DataLen: [input] Total write data length.

  bData: [output] SDO data writes to the specified slave device.

- **Return:**

  Please refer to the chapter 4 for the function return code.

4. Function Return Code

CANopen Master Return Code Definition (1/2)

Return Code	Error ID	Comment
0	CPM_NoError	OK
1	CPM_DriverError	Kernel driver is not opened
3	CPM_BoardNumberErr	There is no CPM100 on the specific board No.
3	CPM_SlotNumberErr	There is no I-8123W on the specific slot No.
3	CPM_ComPortErr	There is no I-7565-CPM on the specific COM port.
5	CPM_ConfigErr	The Master hasn't been configured successfully
6	CPM_MasterInitErr	The Master initialization error.
7	CPM_MasterNotInit	The Master hasn't been initialized
8	CPM_ListenMode	The Master is in listen mode now
9	CPM_NodeErr	Set node number error
10	CPM_NodeExist	The node had been added to the Master
12	CPM_TxBusy	Tx buffer is busy, please wait a minute to send again
13	CPM_UnknowCmd	This version of firmware doesn't support the function
14	CPM_CmdReceErr	Master receive command of wrong length
15	CPM_DataEmpty	There is no data to receive
16	CPM_MemAllocErr	Master has not enough memory
17	CPM_SendCycMsgErr	Cyclic message send error
18	CPM_StatusErr	NMT state of CANopen slave is error
20	CPM_SetGuardErr	Set Guarding and LifeTime parameter error
21	CPM_SetHbeatErr	Set Heartbeat parameter error
22	CPM_SegLenErr	SDO Segment receive error length
23	CPM_SegToggleErr	SDO Segment receive error toggle
24	CPM_SegWriteErr	SDO write segment error
25	CPM_Abort	The return message is an Abort message
26	CPM_PDOLenErr	PDO output data error
27	CPM_COBIDErr	The COB-ID isn't exist or isn't correct one
28	CPM_PDOInstErr	Install the PDO object error
29	CPM_PDODynaErr	The PDO mapping data is setting error
30	CPM_PDONumErr	The PDO number and COB-ID is not match

CANopen Master Return Code Definition (2/2)

Return Code	Error ID	Comment
31	CPM_PDOSetErr	PDO parameter is setting error
32	CPM_PDOEntryErr	The PDO entry parameter is more then useful entry
33	CPM_SetCobIdErr	The EMCY or SYNC COB-ID is setting error
34	CPM_CycFullErr	There are already 5 cyclic message running
35	CPM_Timeout	Message response timeout
36	CPM_DataLenErr	Data length setting error
40	CPM_Wait	Command is uncompleted (only for non-block mode)
41	CPM_Processing	Command is running (only for non-block mode)
50	CPM_LoadEDSErr	Loading the EDS file fails
51	CPM_EDSFormatErr	The format of the EDS file is incorrect

CANopen Motion Return Code Definition

Return Code	Error ID	Comment
0	CPM_MT_NoError	OK
35	CPM_MT_Timeout	Message replies timeout
101	CPM_MT_CtrlNotSupport	The control code is not supported
102	CPM_MT_GetOperateModeErr	Get Operating Mode error (Position, Homing,...)
103	CPM_MT_SetOperateModeErr	Set Operating Mode error (Position, Homing,...)
104	CPM_MT_OperateModeChkErr	Operating Mode check error
105	CPM_MT_GetPositionErr	Get position error
106	CPM_MT_SetPositionErr	Set position error
107	CPM_MT_GetVelocityErr	Get velocity error
108	CPM_MT_SetVelocityErr	Set velocity error
109	CPM_MT_GetTorqueErr	Get Torque error
110	CPM_MT_SetTorqueErr	Set Torque error
111	CPM_MT_GetAccelerationErr	Get acceleration error
112	CPM_MT_SetAccelerationErr	Set acceleration error
113	CPM_MT_GetDecelerationErr	Get deceleration error

114	CPM_MT_SetDecelerationErr	Set deceleration error
115	CPM_MT_SetTorqueSlope	Set Torque slope error
116	CPM_MT_MethodSelectErr	Motion method select error or not support
117	CPM_MT_GetStatusErr	Get any status error
118	CPM_MT_SetStatusErr	Set any status error
119	CPM_MT_GetParameterErr	Read Parameter error
120	CPM_MT_SetParameterErr	Parameter set over range
121	CPM_MT_KeypadIndexErr	Parameter index out of range
122	CPM_MT_KeypadSubIndexErr	Parameter sub-index out of range
123	CPM_MT_KeypadDataLenErr	Data length error
124	CPM_MT_HeartbeatTimeout	Heartbeat is timeout
125	CPM_MT_MotorNodeError	The node id of the servo motor is invalid zero. Check the node ID is 1 ~ 127.
10000	CPMotor_MasterModuleError	The CANopen master hardware has some problem.
20000	CPMotor_LoadMasterDLLError	The DLL of the PISO-CPM100U or I-7565-CPM does not exist in the [C:\Windows\SYSTEM32] or in the [C:\Windows\SYSWow64].