WEINTEK IIOT LTD.

Velocity Control - JOG

Function Block

MC_MoveVelocity

Demo Project

Contents

1. Overview and Operation 1
2. Installing Weintek Library 4
3. Adding iR-PU01-P to CODESYS Project 5
4. iR-PU01-P Parameter Settings 10
5. Function Blocks 13
6. Demo Project Settings 16
7. Login and Operate 19

1. Overview and Operation

To open the project, it is recommended to use the CODESYS IDE version (or a later version) provided on the official Weintek website.

Overview

This demo project introduces how to use Weintek Library Function Block and iR-PU01-P to perform velocity control by outputting pulse signals to servo/stepper motors.

CODESYS can be used to control iR-PU01-P to output pulse signals to a servo/stepper motor, which determines the distance and speed in which the motor rotates.

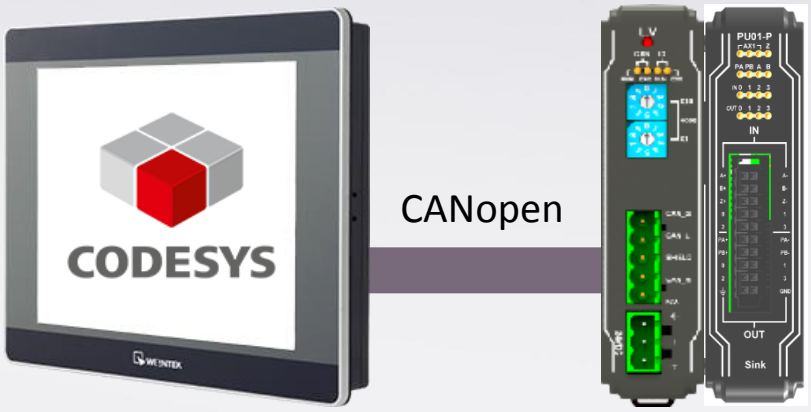
Please see the related demo projects according to the device used:

Using iR-PU01-P (V1001): See iR_Application_JOG_Demo_20190906

Using cMT-CTRL01: See iR_Application_JOG_Demo_CTRL_20230523

Using cMT Series HMI: See iR_Application_JOG_Demo_HMI_20230523

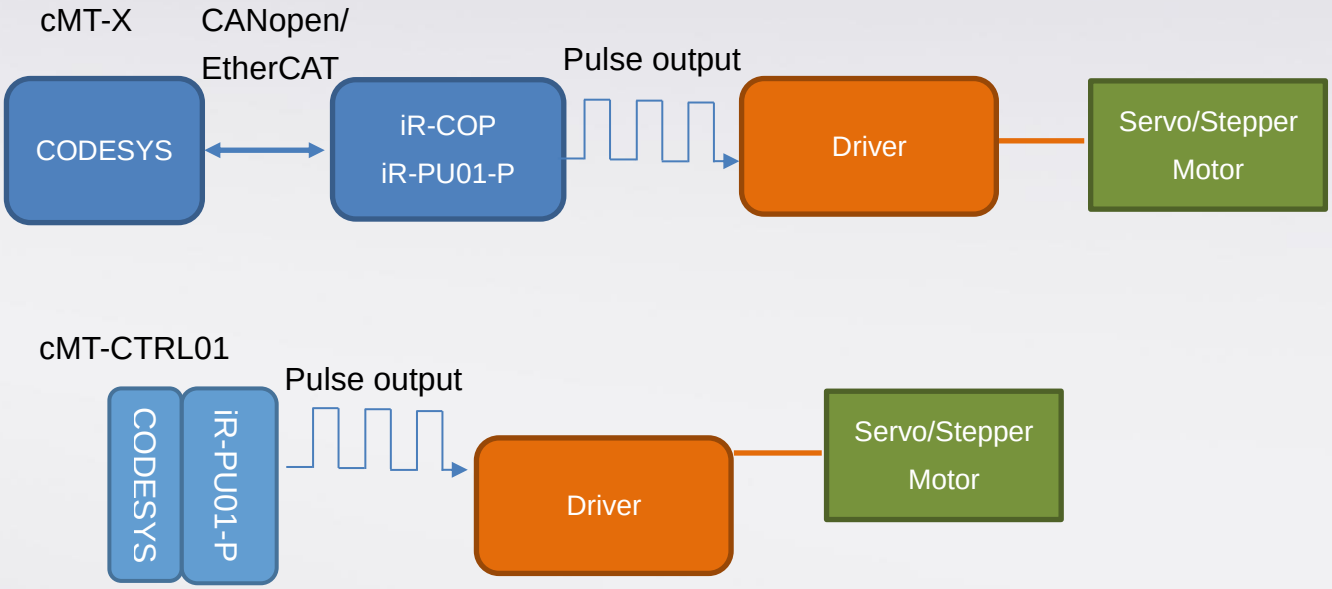
System



iR-COP iR-PU01-P

cMT-CTRL01 iR-PU01-P

Operation


cMT-X
CANopen/
EtherCAT
Pulse output
CODESYS
iR-COP
iR-PU01-P
Driver
Servo/Stepper
Motor
cMT-CTRL01
Pulse output
CODESYS
iR-PU01-P
Driver
Servo/Stepper
Motor

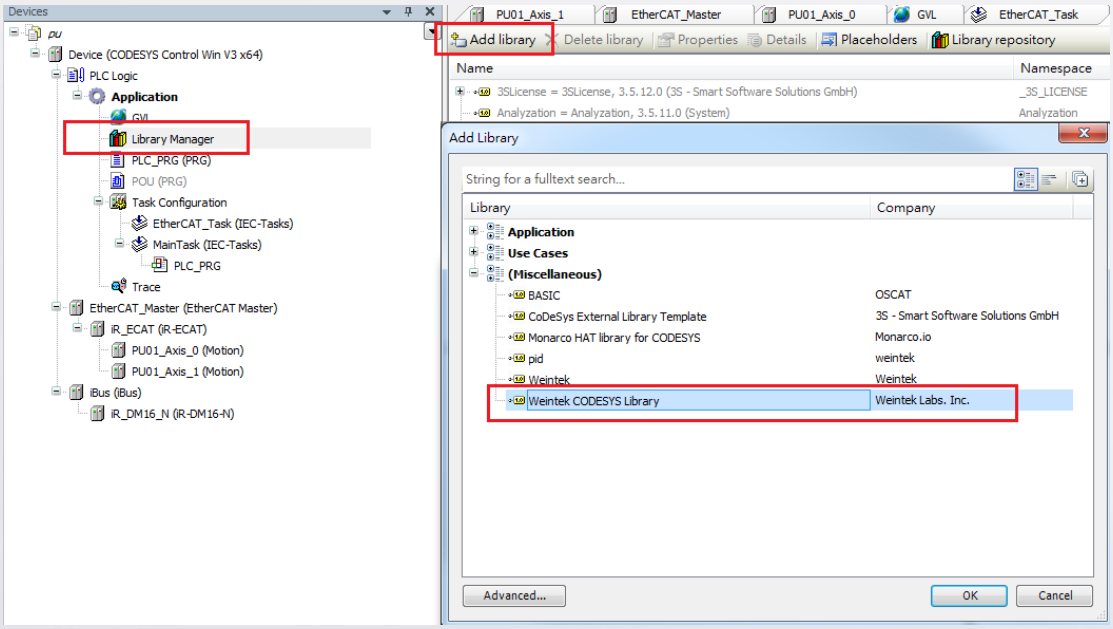
2. Installing Weintek Library

Step 1. In the download page in Weintek official website, search for [cMT+CODESYS Package], and then download and install the package.

https://www.weintek.com/globalw/Download/Download.aspx

(The description file of iR-PU01-P is included in the package)

Step 2. In CODESYS interface add Weintek CODESYS Library.



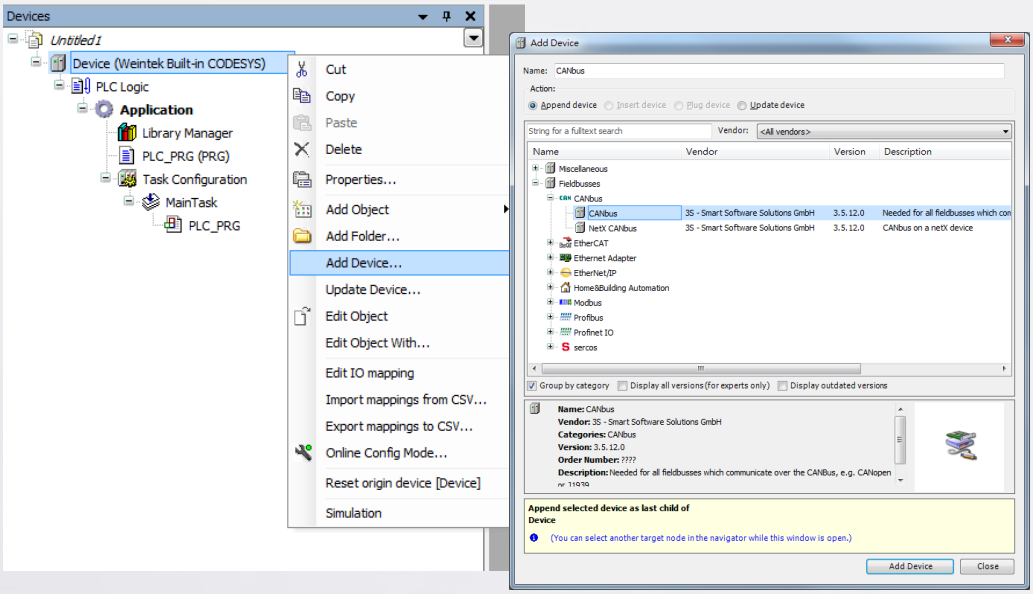
Step 3. Motion Function Block is ready for use after installation.

3. Adding iR-PU01-P to CODESYS Project

Adding iR-PU01-P by using Weintek Built-in CODESYS:

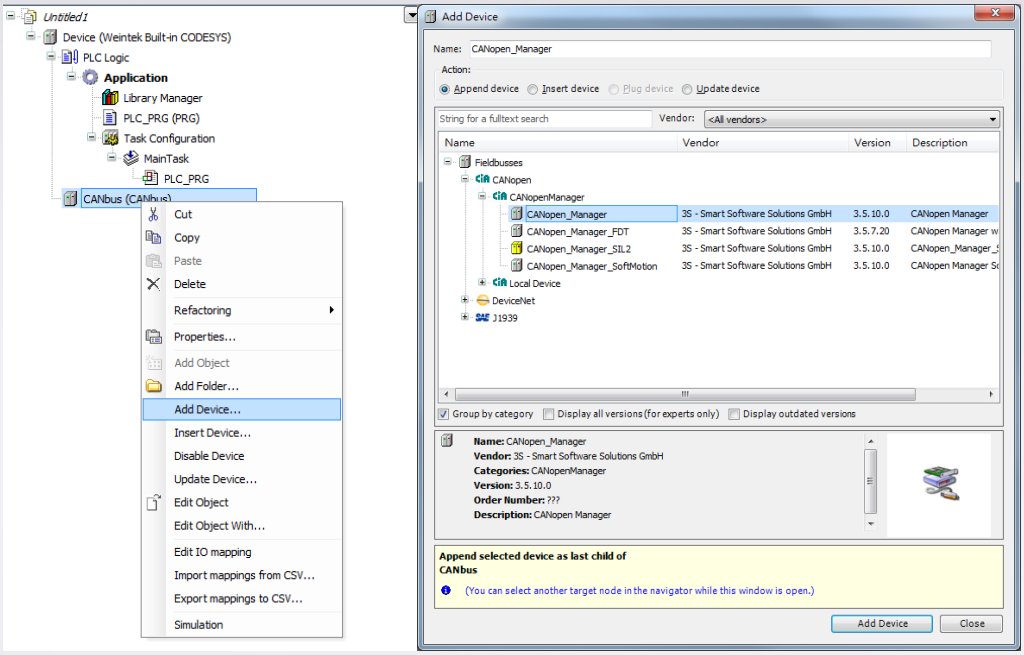
- Add CANbus device:

  [Device]->[Add Device]->[Fieldbusses]->[CANbus]



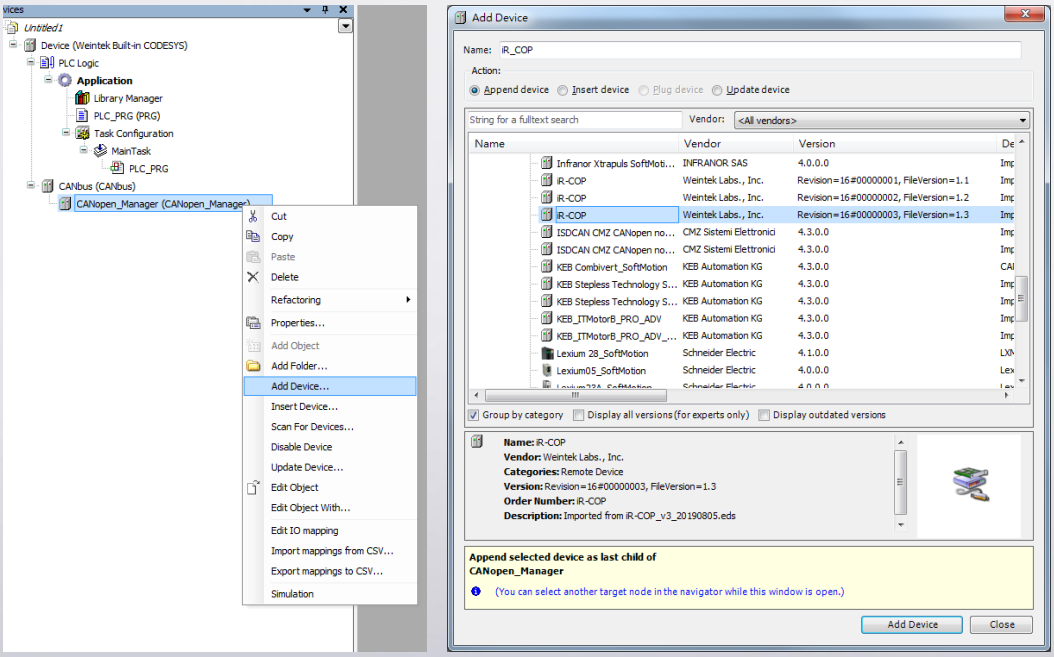
- Add CANopen_Manager device:

  [CANbus]->[Add Device]->[CANopen_Manager]



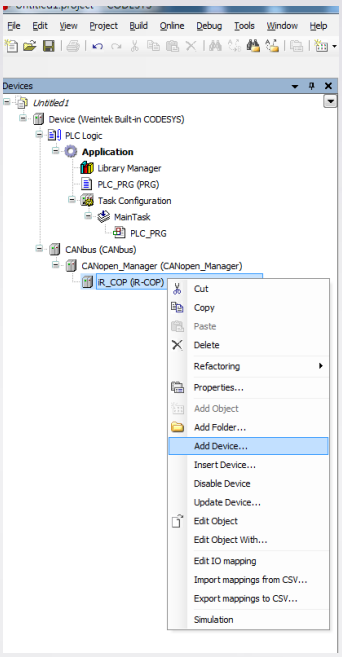
- Add iR-COP module:

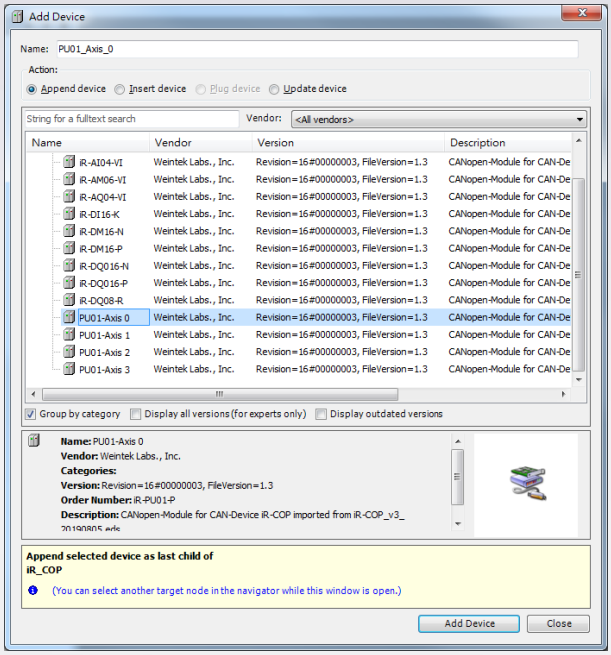
  [CANopen_Manager] ->[Add Device]->[iR-COP] (V1.3)

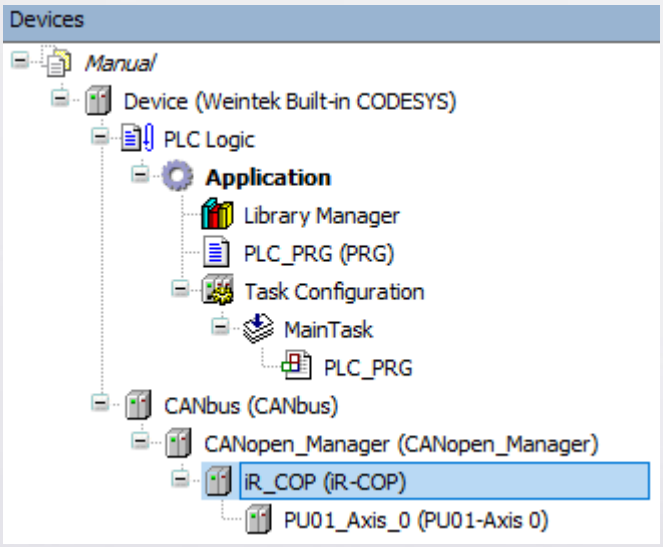


- Add iR-PU01-P module:

  [iR-COP]->[Add Device]->[PU01-Axis 0]



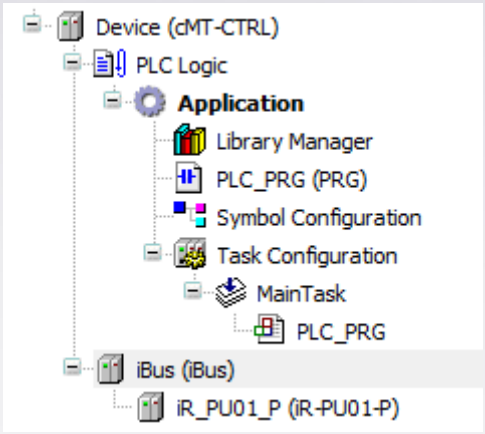




Adding iR-PU01-P by using cMT-CTRL:

- Add iR-PU01-P device:

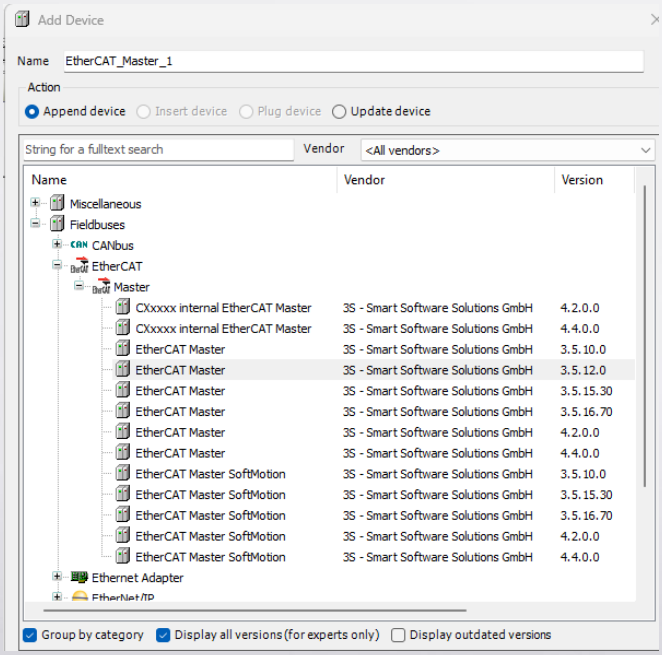
  [iBus]->[Add Device]->[Miscellaneous]->[iR-PU01-P]



Adding iR-PU01-P by using Weintek Built-in CODESYS:

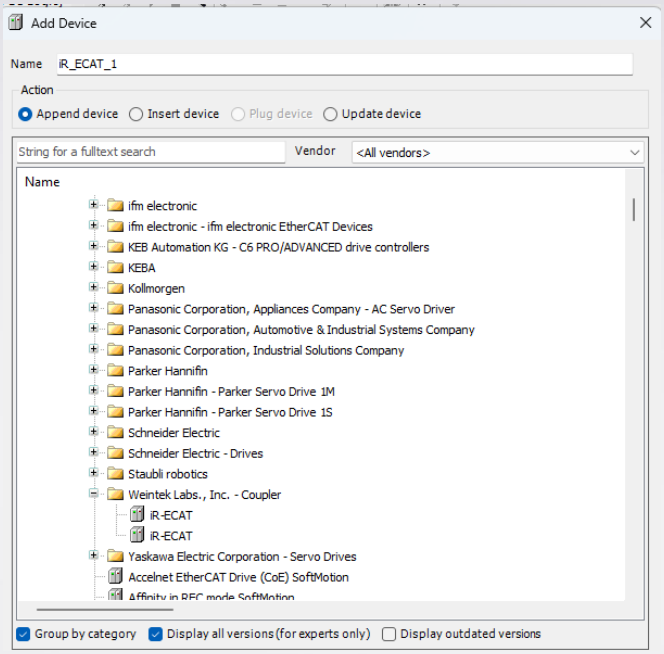
- Add EtherCAT_Master device:

  [Device]->[Add Device]->[Fieldbuses]->[EtherCAT]->[Master]



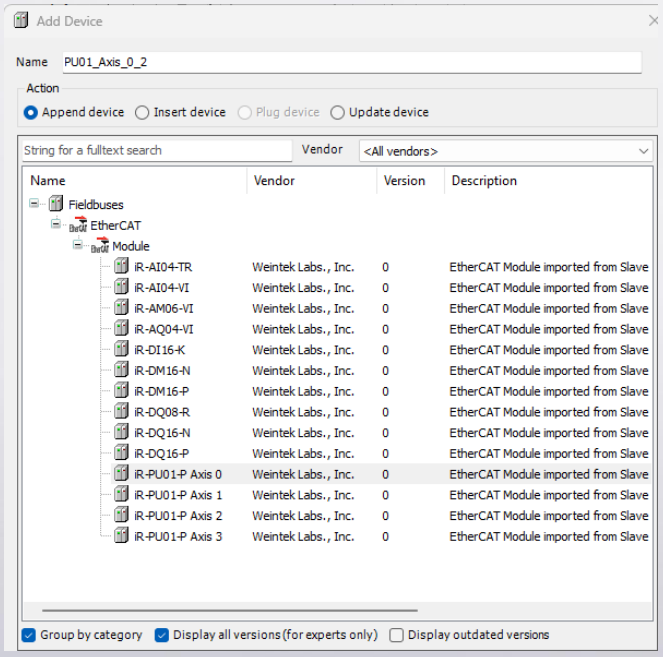
- Add iR-ECAT device:

  [EtherCAT_Master]->[Add Device]->[iR-ECAT]



- Add iR-PU01-P module:

  [iR-COP]->[Add Device]->[PU01-Axis 0]



4. iR-PU01-P Parameter Settings

Settings relating to velocity control:

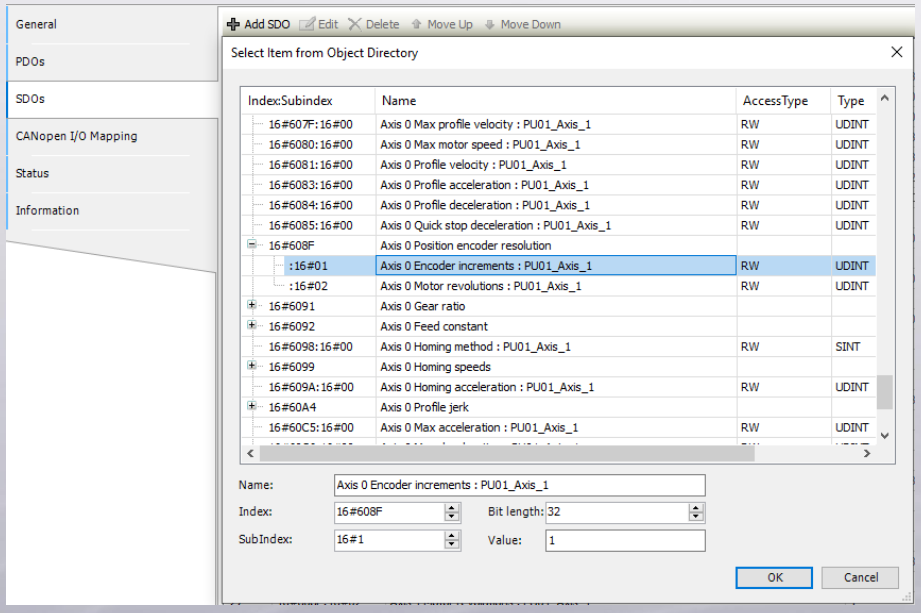
Add SDO | Edit | Delete | Move Up | Move Down

Line	Index:Subindex	Name	Value	Bit length
1	16#608F:16#01	Encoder increments : AX1_PU01	16#1	32
2	16#608F:16#02	Axis 1 Motor revolutions : PU01_Axis_1	1	32
3	16#5511:16#00	Axis 1 Pulse Output Method : PU01_Axis_1	4	8
4	16#6080:16#00	Axis 1 Max motor speed : PU01_Axis_1	200000	32
5	16#607F:16#00	Axis 1 Max profile velocity : PU01_Axis_1	200000	32
6	16#60C5:16#00	Axis 1 Max acceleration : PU01_Axis_1	100000	32
7	16#60C6:16#00	Axis 1 Max deceleration : PU01_Axis_1	100000	32
8	16#6085:16#00	Axis 1 Quick stop deceleration : PU01_Axis_1	100000	32

Before controlling a motor using a motion control module, please configure the settings relating to protection and unit carefully. iR-PU01-P's LED may show error state when skipping these settings and directly using function blocks.

Parameter settings:

- [iR-COP]->[SDOs]->[Add SDO]
- [iR-ECAT]->[Startup Parameters]->[Add]



The SDO settings will be written to iR-PU01-P after login.

- Motor Resolution Setting: 608Fh

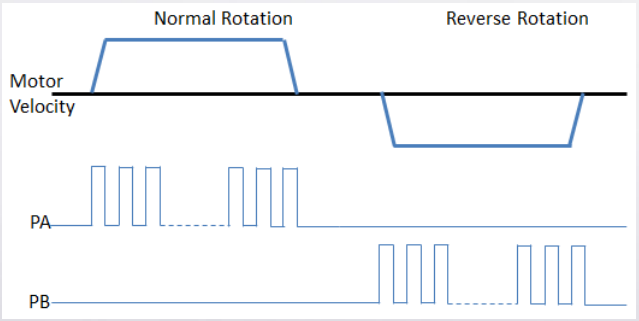
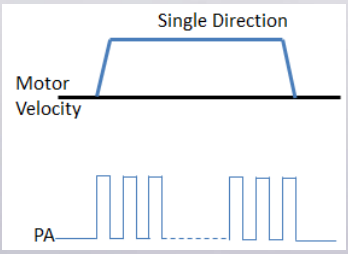
  Motor resolution: number of pulses per revolution. In the demonstration the values are set to 1.

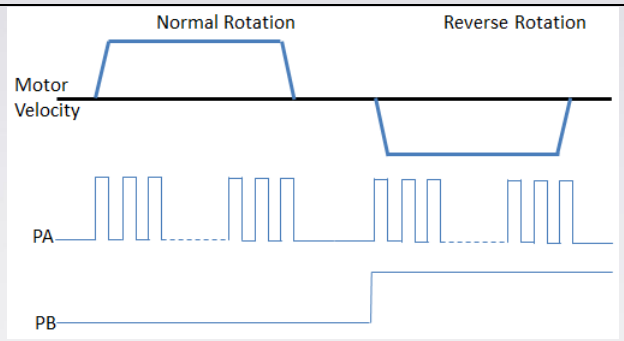
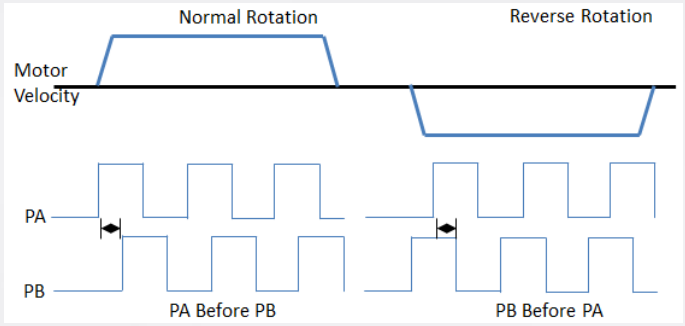
$$Position\ encoder\ resolution = \frac{encoder\ increments}{motor\ revolution}$$

- Pulse Output Method: 5511h

  Pulse output method is determined by the pulse type supported by the driver. The pulse type of both the driver and iR-PU01-P should be identical for the motor to rotate in the desired direction and distance.

  Sub Index 00h: Pulse Output Method

<table>
<tr><td>Bit7-
Bit 4</td><td colspan="4">Reserved</td></tr>
<tr><td rowspan="4">Bit3-
Bit 0</td><td>Value</td><td>PA</td><td>PB</td><td></td></tr>
<tr><td>0</td><td>Disable</td><td>Disable</td><td></td></tr>
<tr><td>1</td><td>CW</td><td>CCW</td><td> 
</td></tr>
<tr><td>2</td><td>Pulse</td><td>NC</td><td> 
</td></tr>
</table>

	3	Pulse	Direction	
	4	A	B	
	5	A(2x)	B(2x)	
	6	A(4x)	B(4x)	

- Max. Velocity: 6080h, 607Fh, 60C5h, 60C6h

  Max. Motor Speed: 6080h

  Enter the value according to the motor specification. Generally, the unit is RPM (Round Per Minute), but for this parameter, the unit is PPS (Pulse Per Second). Please convert the unit before entering the value.

  Max. Profile Velocity: 607Fh

  This is the maximum allowable velocity for the velocity profile. If 607Fh conflicts with 6080h, the lower value will be the maximum velocity.

  Max. Acceleration/Deceleration: 60C5h/60C6h

  When specifying a value greater than the value of 60C5h/60C6h, the value of 60C5h/60C6h will be the maximum acceleration/deceleration rate.

- Quick Stop Deceleration: 6085h

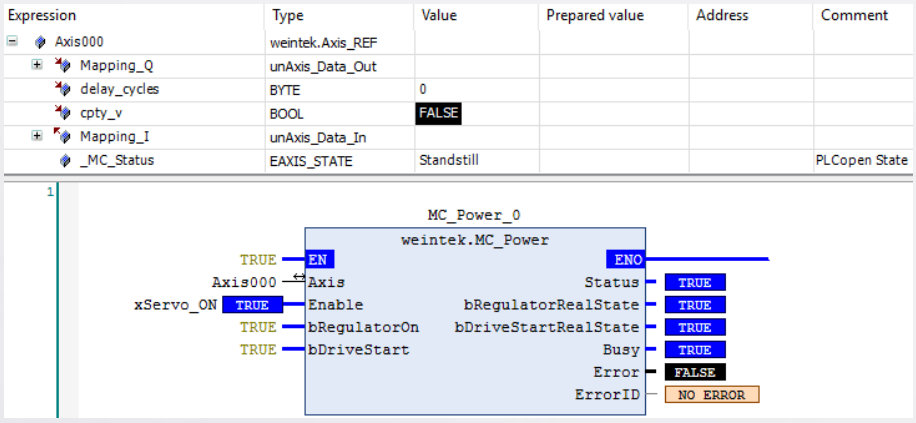
  When an error occurs during the runtime of iR-PU01-P, or when the limit sensor is encountered, this setting can decelerate the motor to stop at the specified deceleration rate.

5. Function Blocks

For more information on Weintek Library, please see chapter 2 of "CODESYS Weintek Library User Manual".

MC_Power

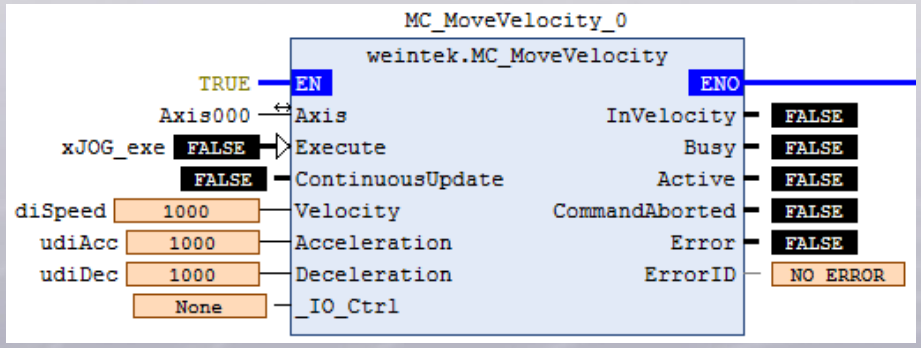
MC_Power must be executed before giving any motion instruction. When it is successfully executed and no error occurs, the axis enters Standstill state.



As shown above, MC_Status is in Standstill state, which means the axis is ready for any motion instruction given to it.

MC_MoveVelocity

MC_MoveVolocity function block performs velocity control for the specified axis. The following parameters are used when executing MC_MoveVelocity.

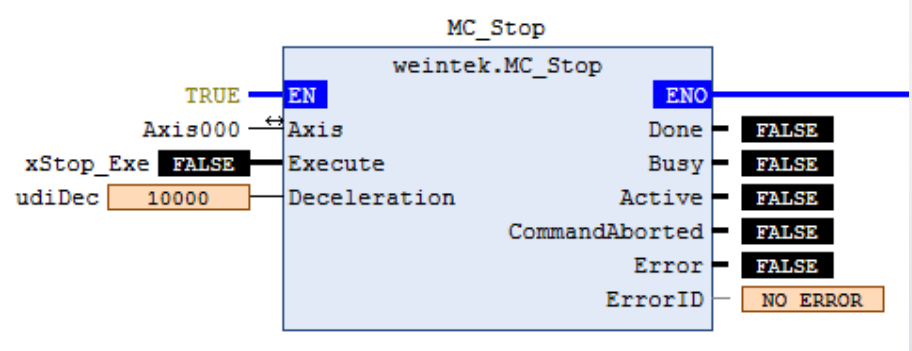


- Velocity: Specify the target velocity and the rotation direction. Positive velocity = normal rotation, negative velocity = reverse rotation.
- Acceleration: Specify the acceleration rate, the value cannot be 0.
- Deceleration: Specify the deceleration rate, the value cannot be 0.
- ContinuousUpdate: Continuously updates the velocity. TRUE= the target velocity, acceleration rate and deceleration rate can be changed when the axis is operating. An axis that is operating and is in Continuous Motion state can only be stopped using MC_Stop or MC_Halt.

MC_STOP

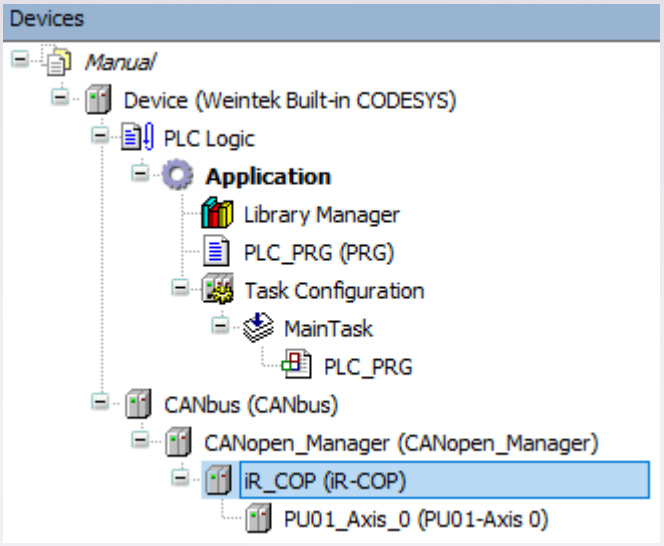
MC_STOP can stop axis operation. When using MC_STOP, it decelerates the axis to stop, and instructions can only be given after the axis stops.

The following parameters are used when executing MC_STOP.



- Deceleration: Specify the deceleration rate, the value cannot be 0.

The axis enters Standstill state after it stops.


Devices
Manual
Device (Weintek Built-in CODESYS)
PLC Logic
Application
Library Manager
PLC_PRG (PRG)
Task Configuration
MainTask
PLC_PRG
CANbus (CANbus)
CANopen_Manager (CANopen_Manager)
iR_COP (iR-COP)
PU01_Axis_0 (PU01-Axis 0)

6. Demo Project Settings

JOG

xEnable_Power: Start the system.

xServe_ON: Turn on server.

xJOG_exe: TRUE = Start JOG; FALSE = Stop JOG

xUpdate: TRUE = The speed can be changed during motion; FALSE = The speed cannot be changed during motion, please configure before executing this function block.

eIO_Control: Trigger motion control using iR-PU01-P's built-in digital input.

udiSpeed: Specify speed.

udiAcc: Specify acceleration rate.

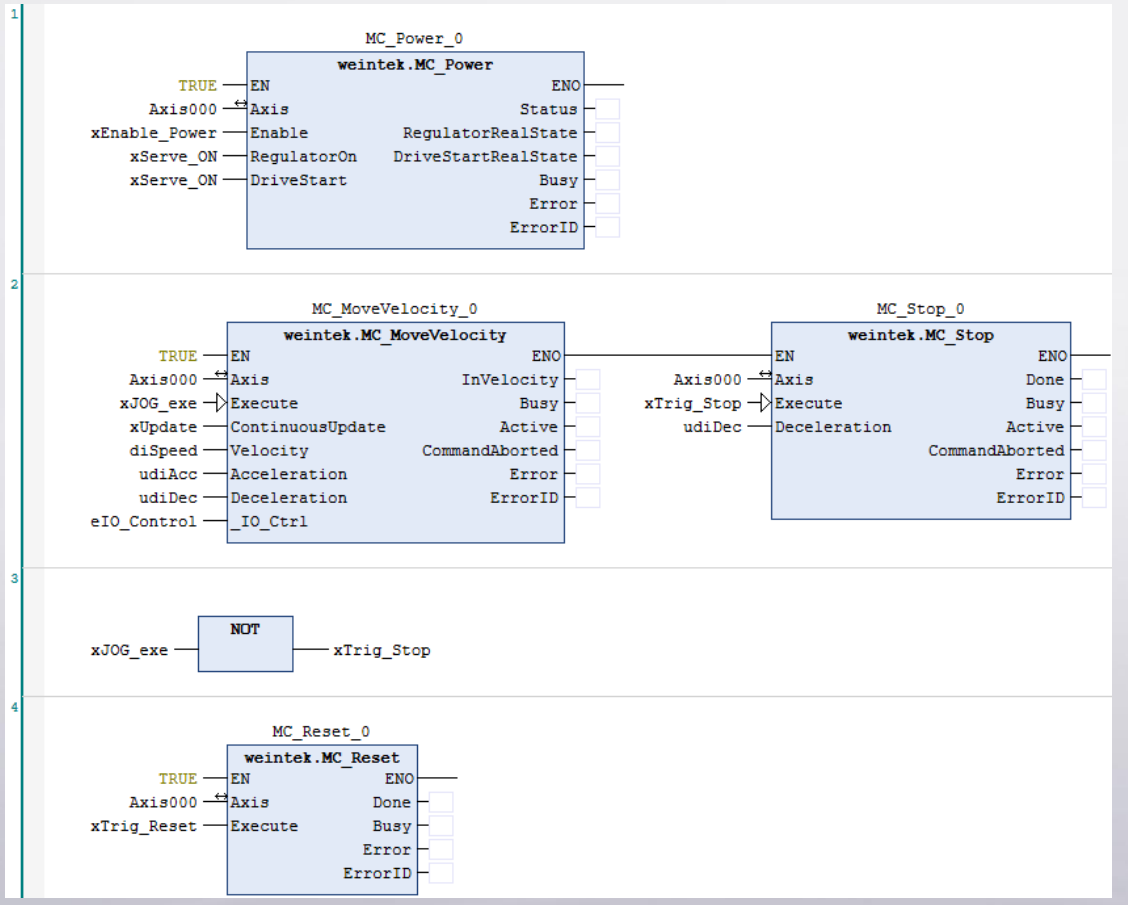
udiDec: Specify deceleration rate.

Declaration

```
// Axis reference
Axis000 : Weintek.Axis_REF_Lite ;
// Motion Control Function Block
MC_Power_0: weintek.MC_Power ;
MC_MoveVelocity_0: weintek.MC_MoveVelocity;
MC_Stop_0: weintek.MC_Stop;
MC_Reset_0: weintek.MC_Reset;
// JOG Button
xEnable_Power, xServe_ON, xJOG_exe, xTrig_Reset, xUpdate : BOOL ;
eIO_Control : weintek.eMC_IO_CTRL ;
// JOG parameter
diSpeed : DINT := 1000 ;
udiAcc : UDINT := 1000 ;
udiDec : UDINT := 1000 ;
```

Declare necessary variables and give initial value.

FBD



1: Starting motion control system: xEnable_Power & xServe_ON must be TRUE and no error occurs.

2: JOG function block.

3: Press xJOG_exe (TRUE) to start JOG, and release xJOG_exe (FALSE) to stop JOG.

4: When an error occurs during motion, triggering xTrig_Reset can reset iR-PU01-P.

7. Login and Operate

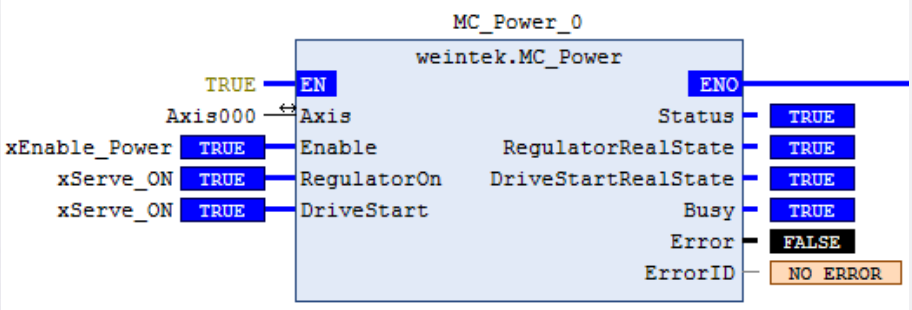
The following demonstrates how the project works.

Start Motion Control System

xEnable_Power & xServe_ON = TRUE, start controlling the axis.

Status, RegulatorRealState, DriveStartRealState, Busy = TRUE: No error.

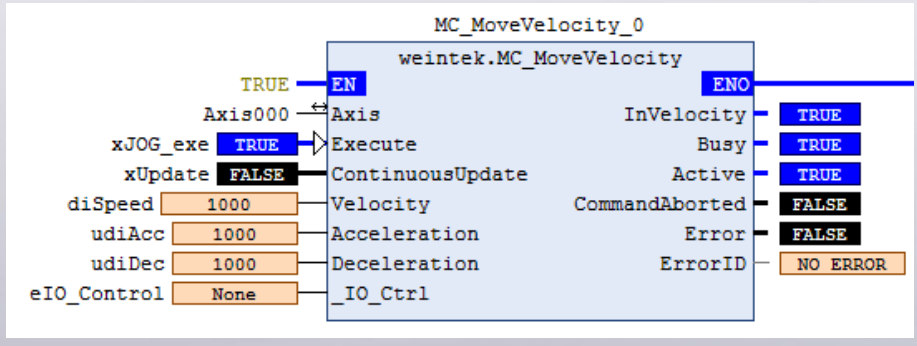
Error = TRUE: An error occurs.



Normal Rotation

The default setting in the project file is normal rotation, press xJOG_exe (TRUE) for iR-PU01-P to start outputting pulse signal.

Release xJOG_exe (FALSE) for iR-PU01-P to stop outputting pulse signal.



Reverse Rotation

Change diSpeed to a negative value and then press xJOG_exe for iR-PU01-P to start outputting pulse signal.

Release xJOG_exe (FALSE) for iR-PU01-P to stop outputting pulse signal.