

MAT-LC-CXX Series C06/C07/C0X PLC Based Motion Controller User Manual



Contents

	1 System Overview.....	5
1.1	Product Description.....	5
1.1.1	Introduction to CPU.....	5
1.1.2	C06/C07 Series Expansion Modules	6
1.2	System Structure.....	8
1.3	Electrical Specifications and Environmental Conditions	9
1.4	Functional Application	10
1.4.1	Motion Control Function	10
1.4.2	Logic Control Function	11
	2 Getting started.....	12
2.1	Connect the Main Control Module	12
2.2	Start the CODESYS Software.....	13
2.3	Set Up Communication.....	18
2.4	Programming	19
2.5	Compile and Run the Program	20
2.6	Monitoring and Debugging.....	21
	3 Install.....	23
3.1	Installation Precautions.....	23
3.2	Installation Size	25
3.3	Use Rack	25
3.4	Installation Method.....	26
3.5	Grounding and Wiring	28
3.6	Suppression Circuit.....	29
	4 Power Module.....	31
4.1	Specifications	31
4.2.1	Interface diagram	32
4.2.2	Interface definition.....	32
	5 Main Control Module.....	34
5.1	Basic Performance Parameters.....	34
5.2	CPU Input Function.....	35
5.2.1	Digital Input	35
5.2.2	High-speed count input	37
5.3	Communication Function	41
5.3.1	Communication Port Specification	41
5.3.2	Schematic diagram and definition of external interface	44
5.3.3	Make standard network cables	46
5.4	Data Memory Specifications	46

5.5	Features of C06/C07 Series Motion Controllers	48
	6 Structure of CODESYS.....	52
6.1	Composition of The Project.....	52
6.2	Language	54
6.2.1	Instruction List (IL).....	54
6.2.2	Structured Text (ST)	56
6.2.3	Sequential Function Chart (SFC)	61
6.2.4	Functional block diagram (FBD)	65
6.2.5	Continuous Function Chart (CFC)	65
6.2.6	Ladder Diagram (LD)	66
6.3	Online debugging function	67
6.4	Standardization	68
	7 User's Guidance.....	69
7.1.1	Bus Communication	69
7.1.2	Modbus RTU Communicaiton	71
7.1.3	Modbus TCP Communication	74
7.1.4	CANopen Communication	77
7.1.5	Electronic cam based on EtherCAT communication.....	88
7.1.6	EtherNET/IP communication.....	94
7.2.1	Application example of high speed counting module.....	95
7.2.2	Application example of CPU high speed counter	98
7.3	HSC Interrupt example	105
7.4	CNC routine	110
	Appendix.....	111
B	Power budget	111
C	Specifications of Extension Modules	112
C.1	Power module.....	112
C.2	Digital module	115
C.3	Analog module.....	119
C.5	High speed counting module	121
C.6	Pulse output module.....	123
C.7	EtherCAT slave.....	124
D	IO Module channel configuration.....	127
D.1	Digital input module channel configuration	127
D.2	Digital output module channel configuration	128
D.3	AI module channel configuration	128
D.4	Analog output module channel configuration	129
E	SMB_MODBUS library	130
E.1	Install SMB_Modbus library file	130
E.2	Modbus_RTU library.....	131
E.3	Modbus_TCP library	134
F	Operation instruc tions in CODESYS	135
F.1	Add device.....	135
F.2	Configurate CANopen slave device	139

F.3	Configure EtherCAT slave device	142
G	ExtBus library	147
G.1	Install ExtBus library files.....	147
G.2	ExtBus library instruction description.....	148
H	Install the device description file in CODESYS	155
H.1	Install EtherCAT servo slave device description file	155
H.2	Install PLC/402 axis device description file	155
H.3	Install CANopenslave EDS file	156
I	Instruction Quick Check.....	158
I.1	Operator summary in CODESYS.....	158
I.2	Standard.lib library	160
I.3	Util.lib library.....	161
I.4	SoftMotion library	162
J	Ordering Information.....	174

1 System Overview

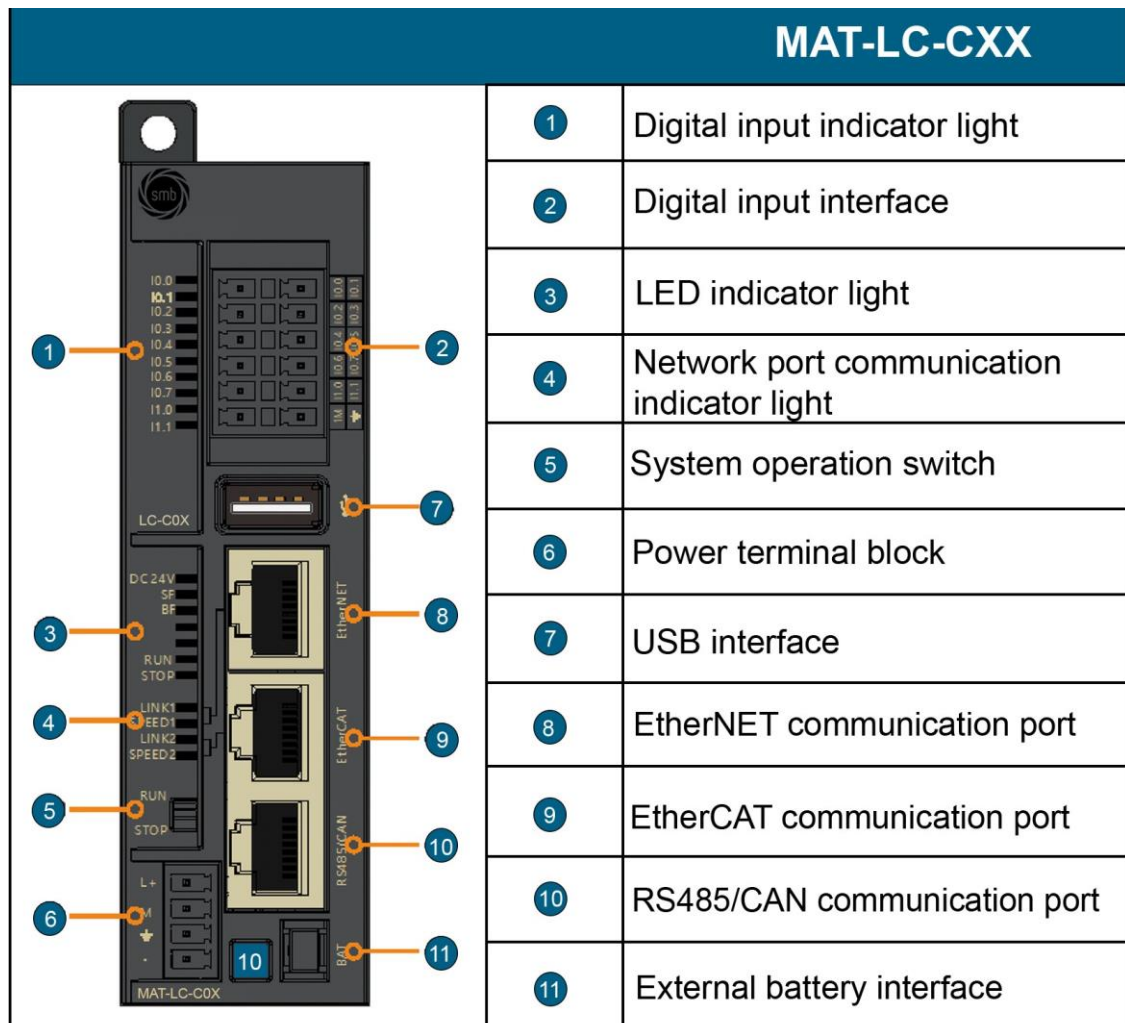
1.1 Product Description

MAT-LC-CXX series C06/C07 motion controllers can be combined with a variety of expansion modules or units to form a powerful programmable logic controller system. It has the characteristics of high speed, high precision, intelligence and ease of use, strong communication and interconnection capabilities, and medium I/O scale.

1.1.1 Introduction to CPU

The MAT-LC-CXX series C06/C07 motion controller is the central processing unit of the PLC application system, which is used to process various operations and the operation of user programs, as the core control device of the system.

The following is the appearance diagram of MAT-LC-CXX series C06/C07:



The characteristics of MAT-LC-CXX series C06/C07 are described as follows:

◆ **Fast Operation**

The bit instruction execution speed of the CPU is 0.015μs/step.

Floating point instruction execution speed is 1μs/step

◆ **High-speed Transmission**

The high-speed bus between CPU and expansion module adopts M-LVDS technology, and the data transmission rate reaches 55Mbps. The CPU integrates 10 digital inputs and supports 6 high-speed counters.

◆ **Powerful System Expansion Capability**

Through ETC-00 expansion, the maximum allowable expansion of 1032 I/O modules (128 ETCs, each ETC Expandable up to 8 modules).

◆ **Supports multiple communication protocols**

The CPU natively supports communication protocols such as EtherNET, EtherNET/IP, EtherCAT, CANopen, and Modbus.

◆ **Easy and Fast Programming**

Use CODESYS V3.5 SP18 PATCH6 for programming, and the CPU has its own USB port host device interface.

◆ **Perfect Operation Control Function**

Supports single-axis motion function, electronic cam and electronic gear function SoftMotion commands and CNC functions that conform to the PLCopen standard.

◆ **Support External Battery**

The C06/C07 comes with an external battery port. After the external battery is connected, the power-down retention time is a typical value of more than 2 years.

Note: The external battery needs to be purchased separately from Hexin (Order No.: MAT-LC-BAT-00S1). For the installation of the external battery, see [3.4 Installation Method](#)

1.1.2 C06/C07 Series Expansion Modules

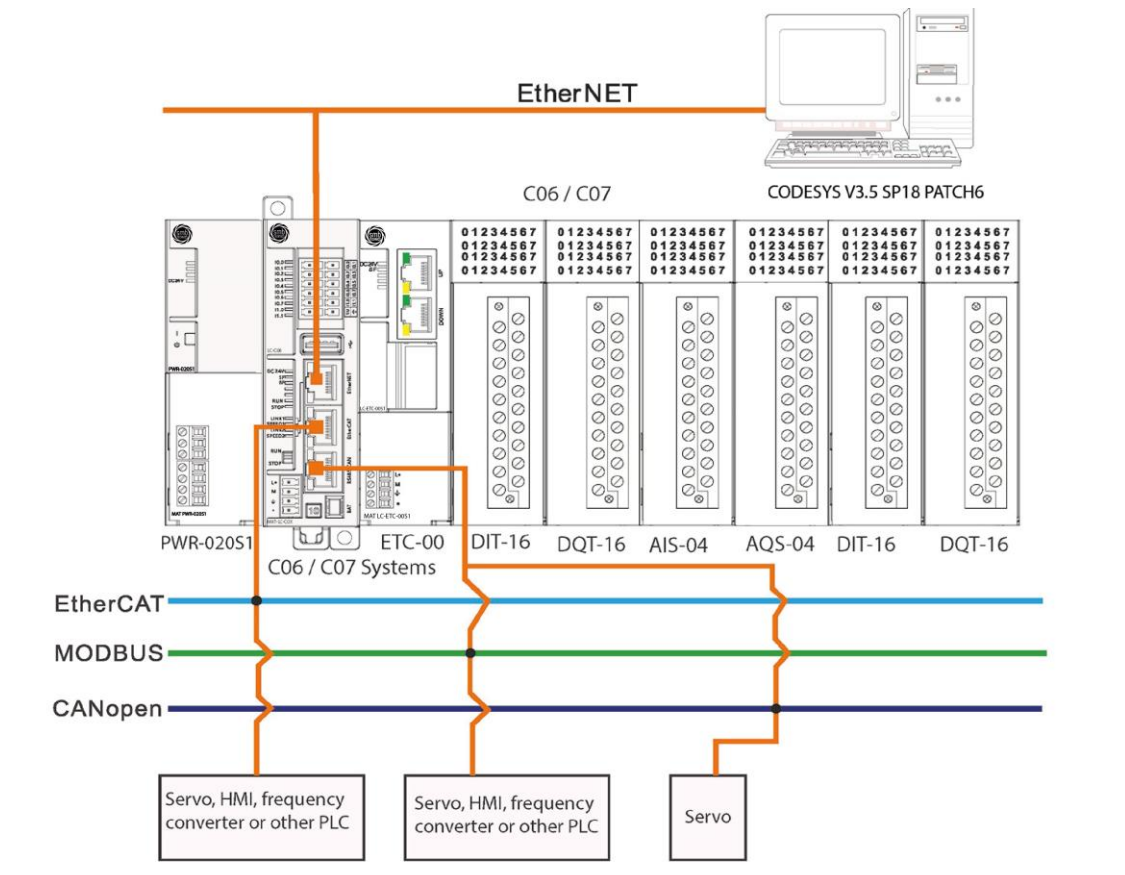
C06/C07 motion controller belongs to MAT-LC-CXX series medium-sized PLC family. It can support MAT-LC-CXX series expansion modules. Its module types include but are not limited to: digital input and output, analog input and output, high-speed counting, etc. Please refer to the table below for model numbers:

Table 1-1 MAT-LC-CXX Series Expansion Modules:

	Digital Input "Module MAT-LC-DIT-08 MAT-LC-DIT-16	Digital Output Module MAT-LC-DQT-08 MAT-LC-DQT-16 MAT-LC-DQR-08 MAT-LC-DQR-16			
	Analog Input Module MAT-LC-AIS-04	Analog Output Module MAT-LC-AQS-04			
	High-speed Counting Module MAT-LC-HSC-02		Pulse Output Module MAT-LC-HSP-04		Power Module MAT-PWR-020S1
	EtherCAT Slave Module MAT-LC-ETC-00S1				

1.2 System Structure

The typical system architecture of C06/C07 application system is shown in the following figure:



- C06/C07 can communicate with the host computer through EtherNET communication port (using standard network cable) or RS485 communication port.
- The application system can be composed of one central rack and three expansion racks at most, and up to 32 expansion units can be installed.
- C06/C07 supports RS485, EtherCAT, EtherNET, EtherNET/IP, CANopen and other communication methods, and CANopen communication can be realized through the CAN communication port of the CPU body (as shown in the figure above).

In the application system, various I/O expansion modules are connected to the MAT-LC-CXX series C06/C07 main control module. The I/O expansion module sends the collected data and diagnostic information to the C06/C07 main control module through the bus, and the C06/C07 motion controller processes it according to the user program and related information, that is, in the application system, the C06/C07 10 The main control module acts as a "brain".

1.3 Electrical Specifications and Environmental Conditions

Electromagnetic compatibility (EMC) refers to the ability of electrical equipment to function properly in its electromagnetic environment without interfering with the environment. Table 1-2 and Table 1-3 describe the electrical specification environmental condition standards that the C06/C07 should comply with. Programmable logic controller standard: IEC61131-2, GB15969.

Table 1-2 Electrical Specifications

Electromagnetic Compatibility - Immunity		
Electrostatic Discharge IEC61000-4-2	Contact discharge: ±4KV Air discharge: ±8KV	
Electrical Fast Transient Burst IEC61000-4-4	Power cord: 2KV, 5KHz Signal line: 2KV, 5KHz (I/O coupling clip) 1KV, 5KHz (communication coupling clip)	
Surge IEC61000-4-5	Power cord: 2KV (asymmetric), 1KV (symmetric)	
Radio Frequency Electromagnetic Field Radiation IEC61000-4-3	80MHz~1GHz, 10V/m, 80%AM (1KHz)	
RF Field Induced Conducted Disturbance IEC61000-4-6	0.15MHz~80MHz, 10V/m, 80%AM (1KHz)	
DC power input port for short interruptions and voltage changes IEC61000-4-29	Short interruption: 10ms Voltage change: 80%~120%, 100ms	
Environmental Testing		
High temperature operation IEC60068-2 Low temperature operation IEC60068-2	60℃ for 16 hours -10℃ 16 hours	
High temperature startup IEC60068-2 Low temperature start IEC60068-2	60℃ for 2 hours -10℃ for 2 hours	
High and low temperature cycle operation IEC60068-2	-10℃ ~60℃ dwell time 3 hours, temperature rise rate 1℃/min, 2 cycles	
High temperature storage IEC60068-2 Low temperature storage IEC60068-2	70℃ for 72 hours -40℃ 72 hours	
Thermal shock IEC60068-2	-40℃~70℃ dwell time 3 hours, temperature change time <1min, 5 cycles	
High temperature and high humidity IEC60068-2	40℃ for 48 hours	
Alternating Damp Heat IEC60068-2	25℃~55℃ 95% 2 cycles	

Sinusoidal vibration (bare metal) IEC60068-2	5~150Hz, 0.05G2/Hz 150Hz~500Hz -3dB/oct, 1 hour/axis, X, Y, Z total 3 axes
Shock (bare metal) IEC60068-2	15G, 11ms pulse, 3 times/direction
High Voltage Insulation Test	
Between 24V/5V nominal circuits	500 VAC
110V/220V circuit to ground	1500 VAC
110V/220V circuit to 110V/220V circuit	1500 VAC
110V/220V circuit connected to 24V/5V circuit	1500 VAC

Table 1-3 Environmental conditions that the C06/C07 motion controller should follow

Environmental Conditions -- Transportation and Storage		
Temperature		-40℃~+70℃
Atmospheric Pressure		1080 hPa~660 hPa (corresponding height is -1000m~+3500m)
Relative Humidity		10%~95%, non-condensing
Fall		1m, 10 times, shipping package
Environmental Conditions -- Work		
Temperature	Horizontal installation position	0℃~60℃ 0℃~40℃
	Vertical installation position	
Atmospheric		1080 hPa~795 hPa (corresponding height is
Relative		10%~95%,non-condensing
Harsh environment Pollutant concentration		Low salt fog, humidity, dust fog and other environments SO2<0.5ppm, relative humidity<60%, non-condensing H2S<0.1ppm, relative humidity<60%, non-condensing

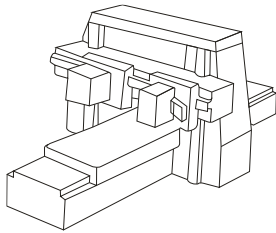
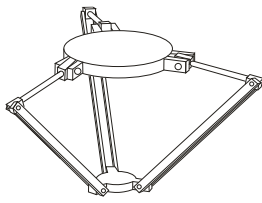
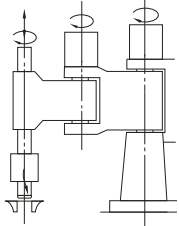
1.4 Functional Application

C06/C07 series motion controller has two control functions: motion control function and logic control function.

1.4.1 Motion Control Function

Table 1-4 Motion Control Functions

PLCopen-based single-axis control/master-slave control	
Single	Homing, relative/absolute positioning, position/velocity/acceleration trajectory planning, continuous motion, JOG
Master-slave	Electronic gear, electronic cam, phase shift
CNC function (LC-C07only)	
Standard CNC functions	3-5 axis linkage, GCODE import, coordinate transformation, etc.

	Gantry (gantry table)	Parallel (parallel robot)	Scara (select compliance equipped robotic arm)
Application-oriented CNC functions			

1.4.2 Logic Control Function

Three Program Organization Units

- ☐ Program (Program)
- ☐ Function Block (Function Block)
- ☐ Function (Function)

Six programming languages

- ☐ Instruction list (IL)
- ☐ Structured text (ST)
- ☐ Sequential function chart (SFC)
- ☐ Function block diagram (FBD)
- ☐ Ladder Diagram (LD)
- ☐ Continuous function graph (CFG)

< **Remark** > Please refer to [Section 6.2 Languages](#) for a description of the programming languages.

Standardized PLC functions

Type Conversion, Numeric Function, Arithmetic Function, Shift Function, Boolean Operation Function, Select Function, Compare Function, String Function, Timer, Counter, Edge Detection, Bistable Element

2 Getting started

Based on a specific example, this chapter introduces how to create a simple project including a PLC program, download the program to the target device (LC-C06 motion controller), run and monitor the program.

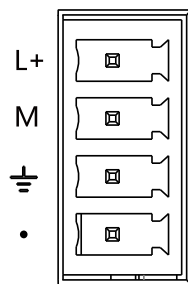
The sample program is written in LD language and includes a program: PLC_PRG. PLC_PRG contains two timers, in which timer 1 starts timing when the program starts, and is set after 5 seconds; timer 2 starts timing after timer 1 is set, and timer 2 starts to reset after 5 seconds; Timer 1 and Timer 2 repeat set and reset operations in 5 second intervals.

2.1 Connect the Main Control Module

Connect the programming device and the PLC with a communication cable (such as a standard network cable), and then supply power to the PLC.

□ Power supply to PLC

The following figure is a schematic diagram of the power supply terminals of the PLC:



WARN

Before installing or removing the LC-C06 motion controller, you must observe the corresponding safety protection regulations, and be sure to disconnect the power supply.

□ Connecting cable

Use a standard network cable to connect the programming device PG/PC and the EtherNET communication port of the LC-C06, or use a programming cable to connect the programming device to the RS485 communication port of the LC-C06.

<Remark> Please refer to [Section 5.3.3 Making Standard Network Cables](#) to make standard network cables.

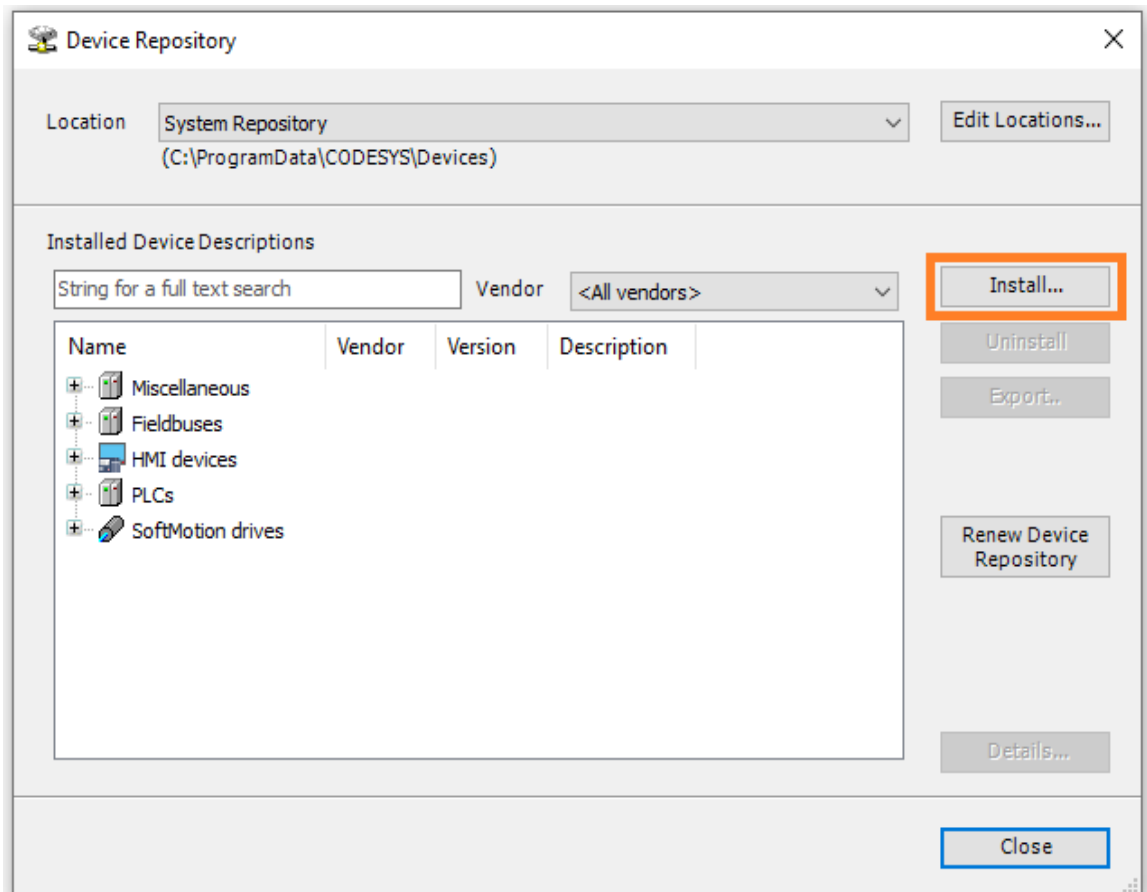
2.2 Start the CODESYS Software

Double click the desktop icon  Or start the CODESYS software in the start menu and perform the following steps:

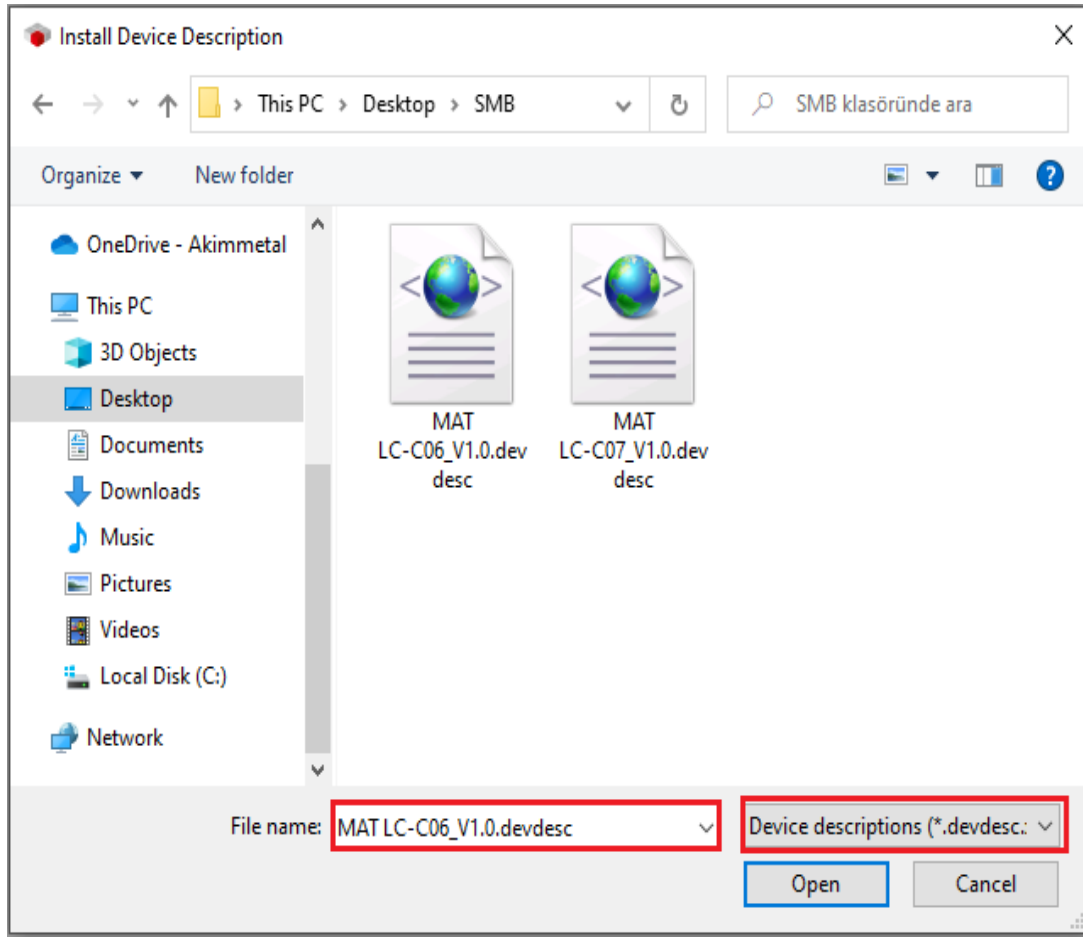
1. Install the SMB device description file (MAT LC-C06_V1.0.devdesc.xml)

First, you need to install the SMB device description configuration file (MAT LC-C06_V1.0.devdesc.xml). After the installation is successful, you can use the SMB device in the program system. The specific operations are as follows:

❑ Select the menu item "Tools" → "Device Repository" to open the following dialog box, which lists the device descriptions in the system, click "Install" in the dialog box.

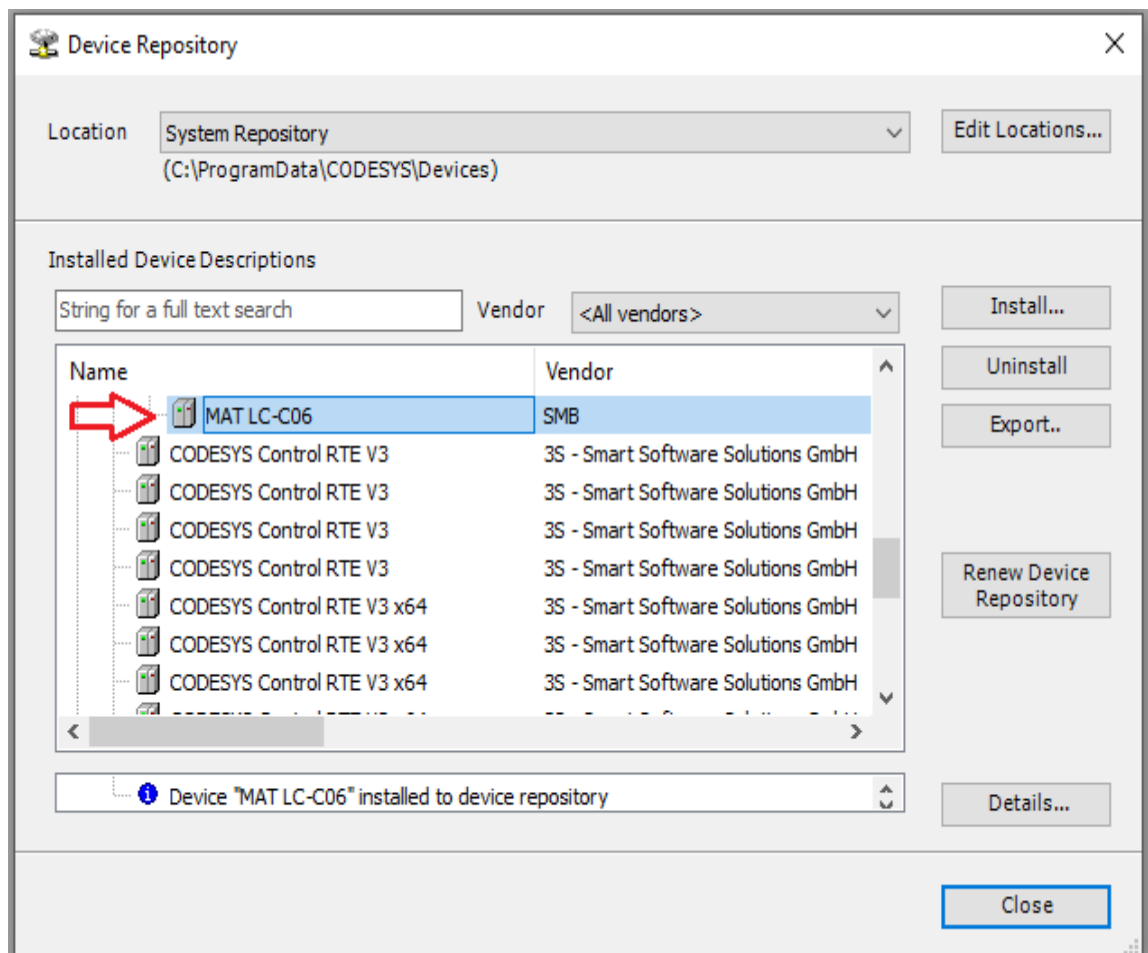


❑ In the opened dialog box, first select the file format, the format is "device description file (*.devdesc.xml)" and then select the desired MAT LC-C06_V1.0.devdesc.xml, click the "Open" button to confirm the options, the new The device is added to the device catalog in the Device Repository. The profiles provided by SMB can be selected by setting the corresponding filters.



After the above operations are completed, you can refer to the following steps to enter the device directory to check whether the files are installed correctly:

Select the menu item "Tools" → "Device Repository" to open the Device Repository dialog box, as shown below:



Expand "PLC" → "SoftMotion PLCs" in the Device Repository dialog box, you can see the SMB device "MAT-LC-C06_V1.0" that has been successfully installed. After checking, if you need to continue to install the device file, please uncheck "Enable Device Repository Dialog" in "Tools" → "Options".

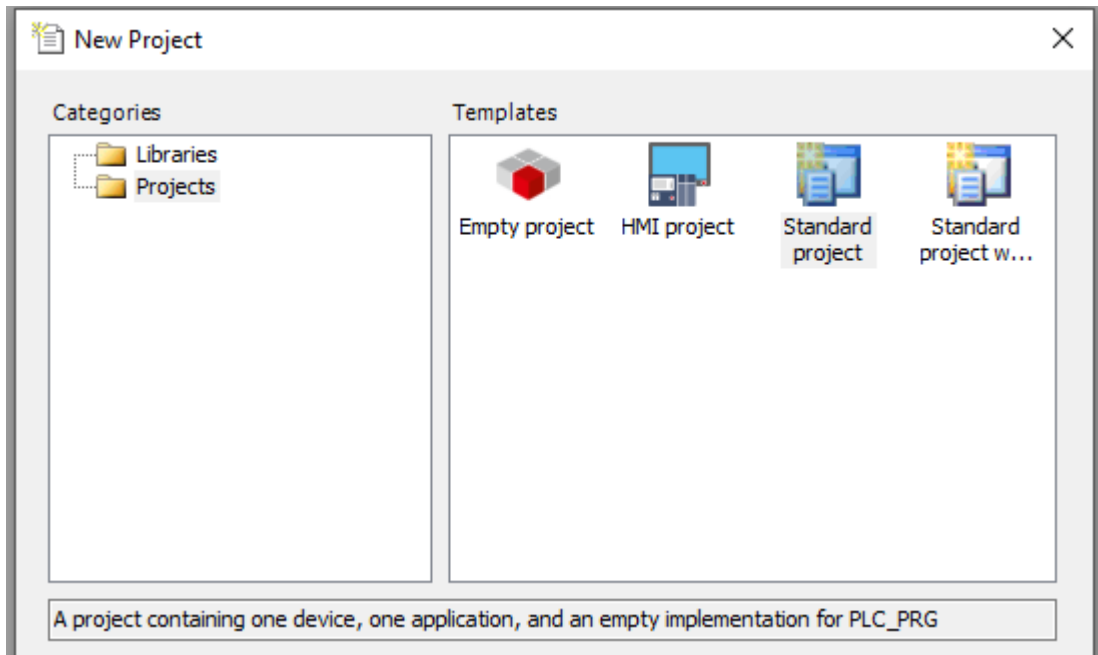


HINT

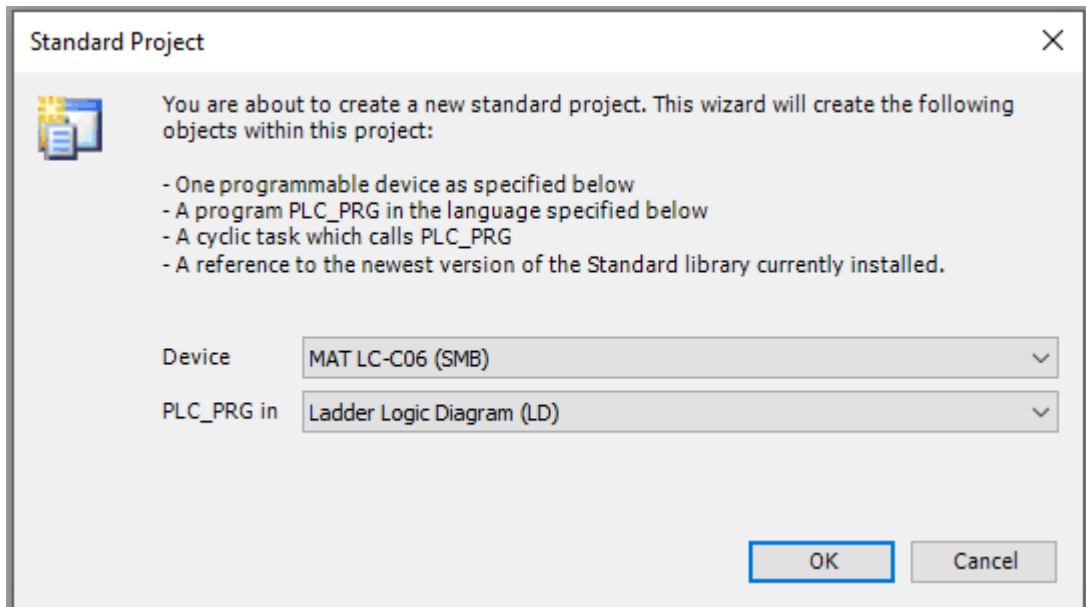
- ♦ 1) If you also need other devices that use SMB (CANopen or EtherCAT slave devices), then you also need to install the following device description files:
 - ♦ CANopen slave device configuration file (*.eds): SMB_SD_LP_C_VX_XX.eds.
 - ♦ EtherCAT slave device configuration file (*.xml): SMB_SD_LP_E_VX_XX.xml;
- 2) Please download the required device configuration file from our website: <https://www.smbtechnics.com>.

2、 Create a New Project

(1) Select the menu item "File" → "New Project" on the main page of CODESYS, and then select "Standard project" in the pop-up dialog box and set the file name and storage directory).



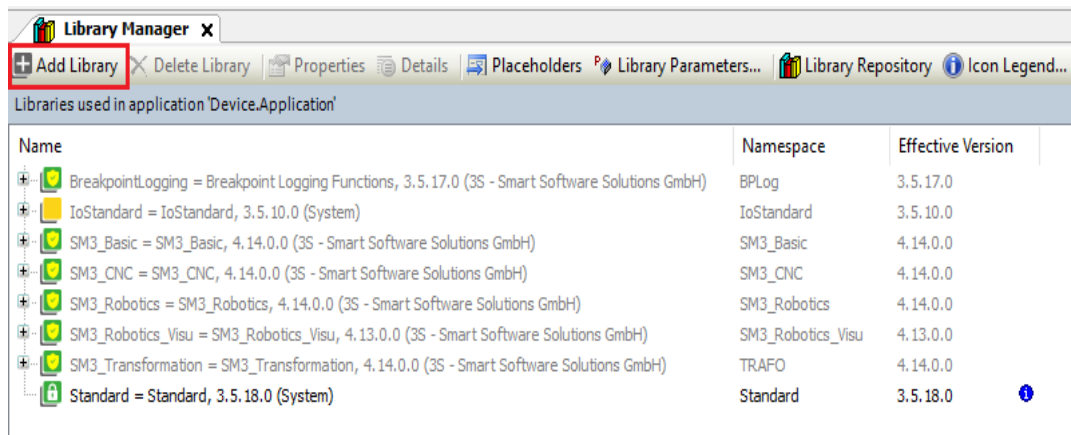
(2) After performing the above operations and confirming, another dialog box will pop up. Select the required file device () and programming language in the dialog box, and then click "OK" to complete the creation of the project.



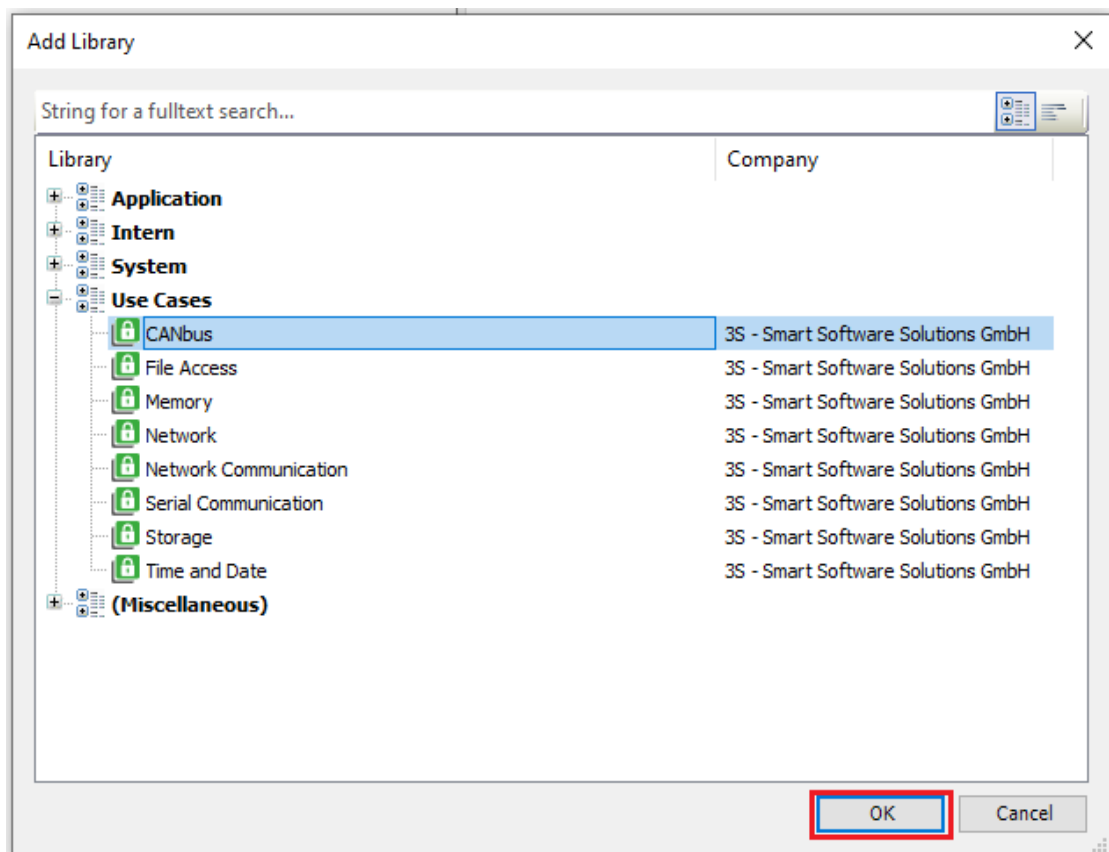
3、Add Library

Users can add instruction library in CODESYS according to their specific needs. The specific operations are as follows:

- ◆ Open the Library Manager in the project tree and select Add Library.



- ◆ In the displayed interface, select the library you want to add and click "OK" to add the required library to the library manager.



2.3 Set Up Communication

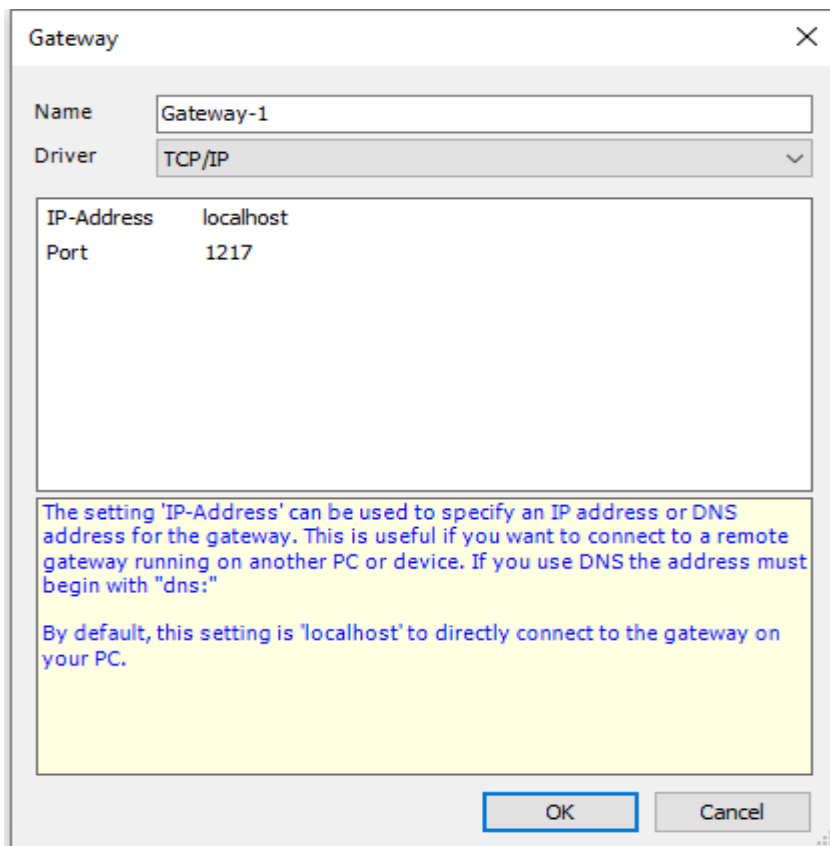
Please refer to the following steps to set up the communication between LC-C06 and the host computer:

1、Set the IP of the programming device to the same network segment as the LC-C06

Before setting the communication, it is necessary to set the IP of the programming device PG/PC to the same network segment as the LC-C06 (IP: 192.168.0.x). Setting method: open the local connection properties of the PC, double-click the TCP/IP protocol, Change "Obtain an IP address automatically" to "Use the following IP address", and then fill in "192.168.0.X" in the IP address.

2、Execute "Communication settings" in the CODESYS device view

Double-click "Device (MAT-LC-C06_V1.0)" in the device view to open the device dialog box, then click the "Gateway" button in the "Communication Settings" tab, then click "Add Gateway", in the pop-up "Gateway" dialog box, enter "Name", "Driver", select "TCP/IP", "IP Address" and select "localhost", and finally click "OK" to close the dialog box, LC-C06 is added to the communication dialog box.



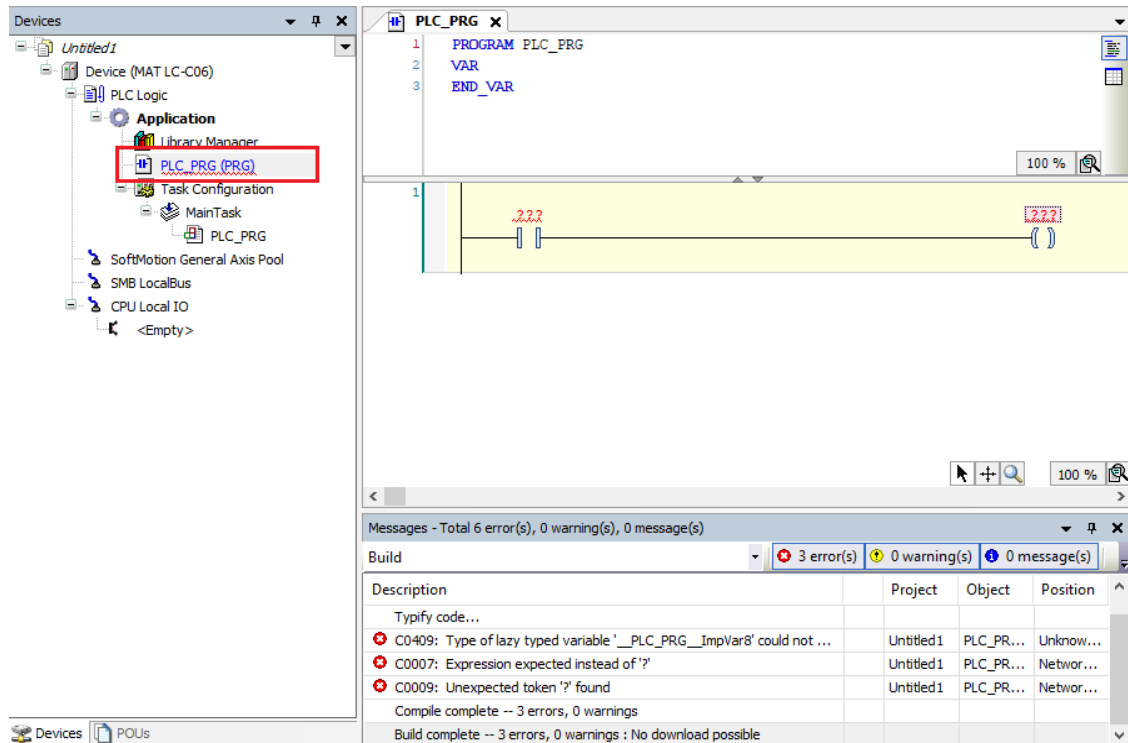
Select the successfully added LC-C06, and then click the "Scan Network" button to search for available devices on the local network. If the search is successful, select the searched device and click the "Set Active Path" button, this operation will activate the communication channel setting, that is, all communication-related operations will be associated with this channel.

<Remarks> When the system starts, CODESYS related service programs (such as  ,  etc) , will appear in the system tray. If there are no special requirements, there is no need to operate the service programs in the tray.

2.4 Programming

This example program achieves the purpose: Timer 1 and Timer 2 repeat the set and reset operations in the 5S interval.

In the device view, the default POU is "PLC_PRG". Double-click "PLC_PRG" in the device view to automatically open it in the LD language editor in the middle of the CODESYS user interface. The LD language editor includes a declaration part and an implementation part.



The declaration part includes: the line number, the POU type and name (eg "PROGRAM PLC_PRG") displayed in the left border, and the variable declaration between the keywords "VAR" and "END_VAR".

The implementation section shows a contact and an output coil.

1、Declare variables in PLC_PRG

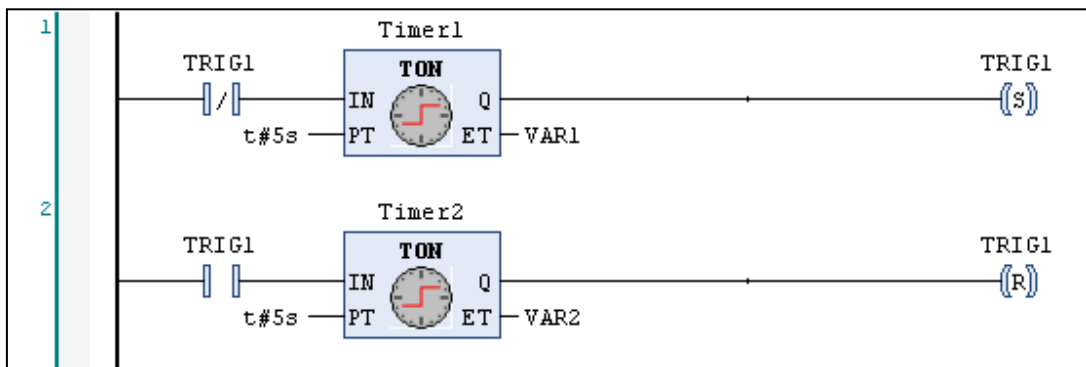
In the Declarations section of the editor, move the cursor to VAR, hit Enter, insert a new blank line, and declare the variables you need to use. Or in the implementation part of the program, use the automatic declaration function: enter the command in the implementation part of the program, click Enter, if there are undeclared variables in the new line, the system will open the automatic declaration dialog box, where you can make declaration settings. The declaration part of this example is as follows:

PROGRAM PLC_PRG							
	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	Timer1		TON			
2	VAR	TRIG1		BOOL			
3	VAR	VAR1		TIME			
4	VAR	Timer2		TON			
5	VAR	VAR2		TIME			

2、Enter the command in the implementation part of PLC_PRG

Expand "Ladder Diagram Elements" in the toolbox on the right side of the LD language editor, use the mouse to drag "Power-on Delay Timer TON" to the implementation part of the language editor, and then drag from the "Ladder Diagram Elements" of the toolbox A reset coil is placed behind the output point Q of the TON command. Using the same operation, create a TON instruction in network 2.

The final LD program completed in this example is as follows:



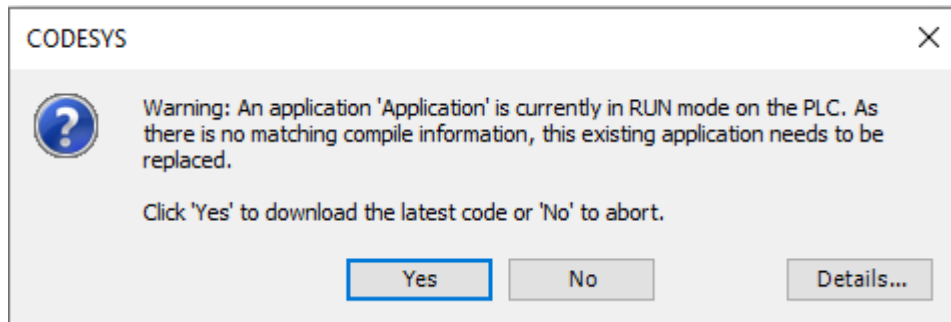
2.5 Compile and Run the Program

1、Save the current project and compile

After the program is written, save the current project, and then select the menu item "Compile" to check the syntax of the current object. When the syntax check is completed, any error messages and warnings will be displayed in the message window of the "Compile" class. If there are no errors, the compilation is successful.

2、 Login, download and start the program

Select the menu item "Online" → "Login..." to establish a connection between the application and the LC-C06, and enter the online state. If the communication settings have been made, the following dialog box will pop up:



Click "Yes" to start the compilation and download of the application.

Finally, select the menu item "Debug" → "Start" to make the application program in LC-C06 start to run, and then you can monitor and debug the current project.

2.6 Monitoring and Debugging

Using the following three methods, you can monitor variables in your application:

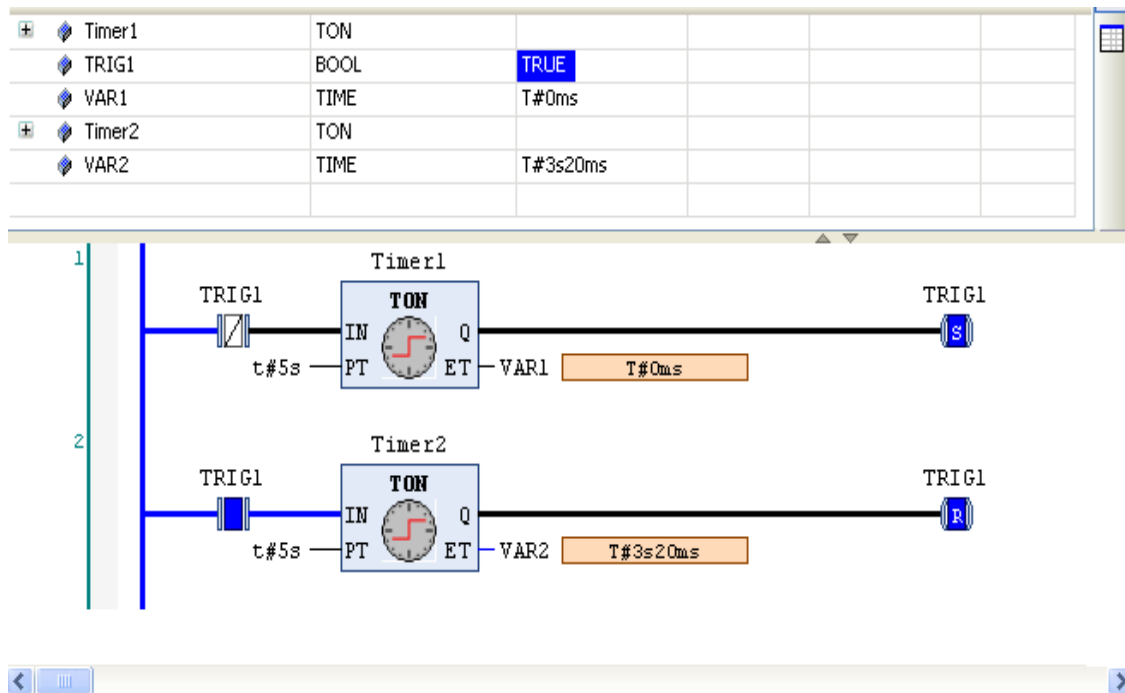
- Watch window with defined watchlists
- Write and coerce variables
- Online View of Special POU's

1、 Open the sample window of the program

Double-click to open PLC_PRG, and the following online view will appear: the upper part displays the program body corresponding to the partial view of PLC_PRG, and the internal monitoring window behind each variable displays the actual value.

2、 Write and coerce variables

A "ready value" is assigned to the variable TRIG1 by writing or forcing, and at the beginning of the next cycle, the variable will display this value. Enter the desired integer value in the input box of "Prepare Value", click Enter or outside this area, and then execute the command "Write Value" or "Force Value" to write or force the value to the PLC.



3、Use watch window

Select the menu item "View" → "Monitor" → "Monitor 1" to open the monitor window. Then, click the first row of the expression column to open the edit box, enter the full path of the variable TRIG1 to be monitored: "Device.Application.PLC_PRG.TRIG1", and then the variable can be written and forced.

Watch 1			
Expression	Application	Type	Value
PLC_PRG.TRIG1	Device.Application	BOOL	FALSE

3 Install

3.1 Installation Precautions

The shape of the C06/C07 motion controller makes it extremely easy to install. In the field, the module can be fixed on the back panel of the control cabinet by using the mounting holes, or the module can be fixed on a standard (DIN) rail by using the DIN clip on the device.

The installation of C06/C07 motion controller should pay attention to the following matters:

❑ Isolate the PLC from heating, high voltage and electrical noise

As a general practice, when installing equipment components, always separate equipment that generates high voltage and high electrical noise from low voltage electronic equipment such as the C06/C07 motion controller.

When arranging the C06/C07 motion controller on the backplane of the control cabinet, consider arranging the electronic devices in the lower temperature area of the control cabinet, because the electronic devices work in a high temperature environment for a long time will shorten their trouble-free time.

Consider the backplane wiring of the control cabinet, and try to avoid designing the AC power supply line, the DC signal line with high energy and high switching frequency, the low-voltage signal line and the communication cable in the same trunking.

❑ Allow proper space for heat dissipation and wiring

The design of the C06/C07 motion controller adopts natural convection heat dissipation, and there must be at least 60mm of space above and below the module for normal heat dissipation.



Notice

The maximum ambient temperature allowed for vertical installation is 10°C lower than that for horizontal installation, and the CPU should be installed below all expansion modules.

When installing the C06/C07 motion controller, leave enough space for wiring and connecting communication cables.

Figure 3-1 shows a CPU mounted on multiple racks, showing the spacing between each rack and adjacent components, cable trays, and cabinets. When wiring the module through the cable duct, the minimum distance between the bottom of the shield connection element and the cable duct is 60mm.

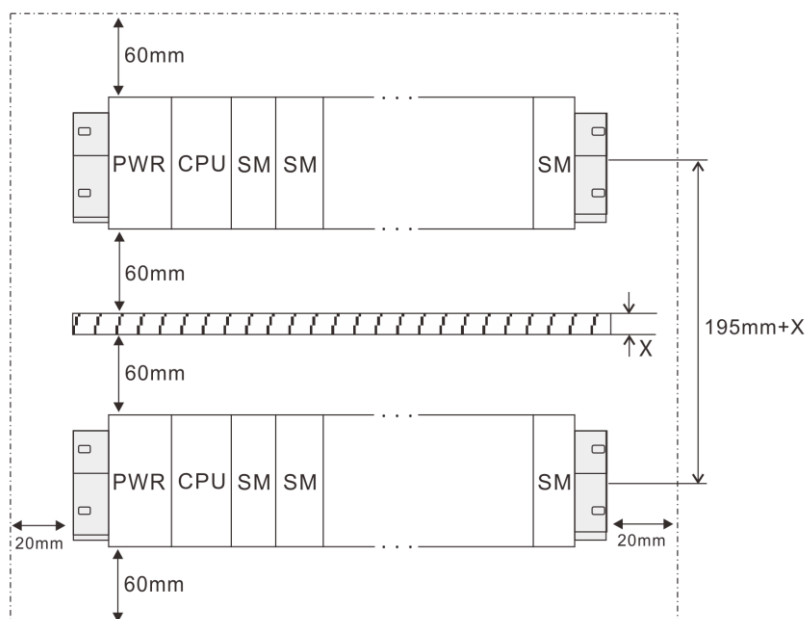


Figure 3-1 Installation Diagram

❑ Power Budget

After selecting the CPU, power supply module, relay module and expansion module of each rack, it is also necessary to confirm whether the current consumption and power consumption of the system bus meet the following conditions:

Condition 1: Confirmation of bus current consumption

The internal bus voltage is 5VDC, and the current is provided by the CPU . The sum of the bus current consumption of the expansion modules on each rack cannot exceed the maximum bus current allowed by the CPU .

Condition 2: Power consumption confirmation

When using a power supply module, the sum of the power consumption of the other modules in each rack cannot exceed the maximum power consumption allowed by the power supply module.

When using an external power supply, select a model with an appropriate power size according to the sum of the connected power.

<Remark> Please refer to Appendix B Power Budget for the budget of the bus power supply current of the C06/C07 motion controller.

❑ Equipment is Powered Off

When installing and removing the C06/C07 motion controller and its related equipment, you must take appropriate safety measures in advance and confirm that the power supply of the C06/C07 motion controller is cut off.

WARN



Installing or removing the C06/C07 motion controller and its related equipment under live power may result in electric shock or malfunction of the equipment, resulting in further serious personal injury or even death and equipment damage!

When replacing or installing a C06/C07 motion controller, be sure to use the correct or equivalent module. When replacing the C06/C07 motion controller, in addition to using the same module, make sure that the installation direction and position are correct.



NOTIFICATION

- If you install an incorrect module, the program of the C06/C07 motion controller may produce incorrect functions.
- Failure to replace the C06/C07 motion controller with the same module in the same direction and sequence may result in serious personal injury and equipment damage.

3.2 Installation Size

The C06/C07 motion controller has mounting holes, which can be easily installed on the backplane. The installation dimensions are shown in Figure 3-2.

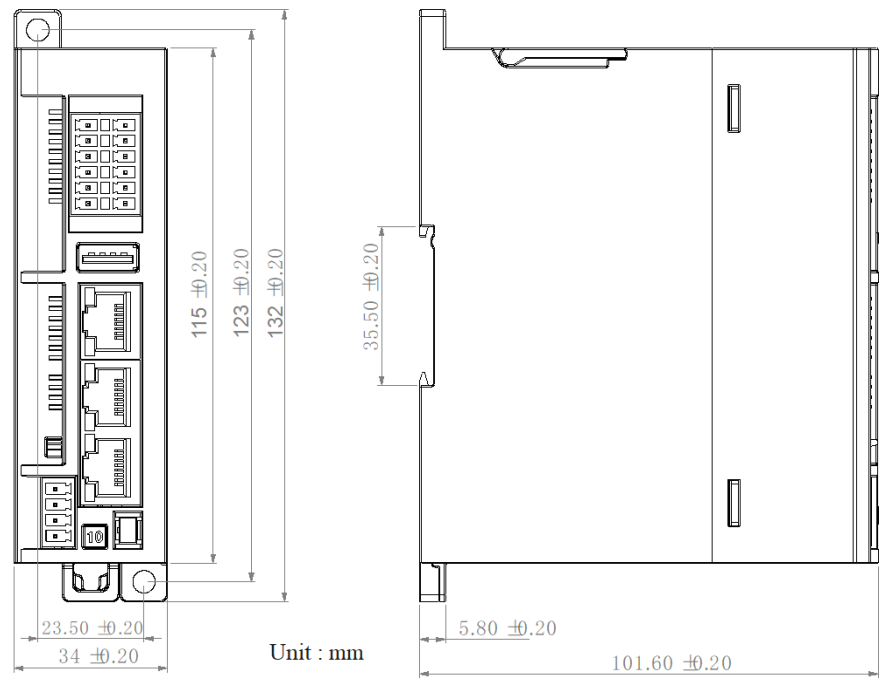


Figure 3-2 C06/C07 motion controller and installation dimension drawing

3.3 Use Rack

Central Rack the C06/C07 application system consists of a central unit and one or more expansion modules. The rack containing the CPU is the central unit. Equipped with modules

A-Use of Expansion Racks

In your application, if all the slots of the central rack are used, then you can use the expansion racks.

B-Module Layout on the Rack

The rack of the C06/C07 is a mounting rail. You can use this guide rail to install the modules to which the C06/C07 application system belongs.

If you are installing modules in a rack, note the following:

- 1-The number of modules installed on the right side of the CPU does not exceed eight (SM, FM, CP).
- 2-The cumulative power consumption of the modules on the rack on the C06/C07 backplane bus must not exceed 1,6A.

The figure below shows the layout for assembling eight signal modules in a C06/C07 system.

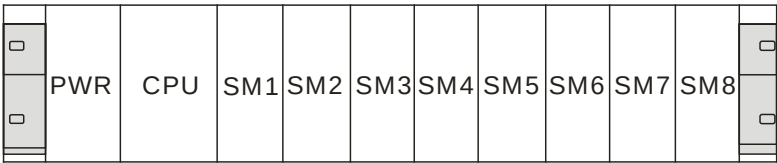


Figure 3-3 C06/C07 application system layout

3.4 Installation Method

Installation method

The C06/C07 motion controller can be installed on the back panel of the control cabinet or on the standard DIN rail; it can be installed horizontally or vertically.

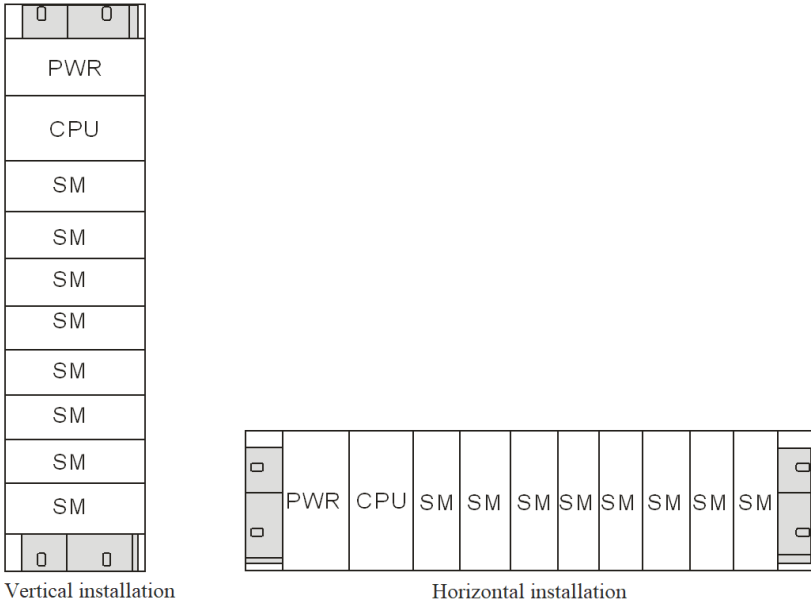


Figure 3-5 CPU and module installation method

Installation and removal operations

Follow the steps below to install or remove the C06/C07 motion controller.

♦ Installation panel

- 1) Position and punch holes according to size requirements;

- 2) Fix the module on the backplane with suitable screws;
- 3) If an expansion module is used, connect the flat cable of the expansion module to the expansion port under the bus cover.

- ♦ **DIN rail mounting**

- 1) Fix the guide rail on the back panel;
- 2) Open the DIN clip at the bottom of the module and clamp the back of the module on the DIN rail;
- 3) If an expansion module is used, connect the flat cable of the expansion module to the expansion port under the bus cover;
- 4) The rotating module is close to the DIN rail, and the DIN clip is closed;
- 5) Carefully check whether the DIN clip on the module and the DIN rail are tightly fixed;
- 6) To avoid damage to the module, do not press the front of the module directly, but press the part of the mounting hole.



NOTIFICATION

When the C06/C07 motion controller is used in a vibration environment or is installed vertically, the DIN rail stopper should be used. If the system is in a high vibration environment, a higher vibration protection level can be obtained by using the backplane mounting method.

- ♦ **Removing the CPU or expansion module**

- 1) Remove the power supply of the C06/C07 motion controller;
- 2) Remove all connections and cables from the module;
- 3) If there are other expansion modules connected to the module you disassembled, please open the bus cover and unplug the expansion flat cables of the adjacent modules;
- 4) Remove the mounting screws or open the DIN clips;
- 5) Remove the module, remove and install the terminal block.

- ♦ **Terminal block installation**

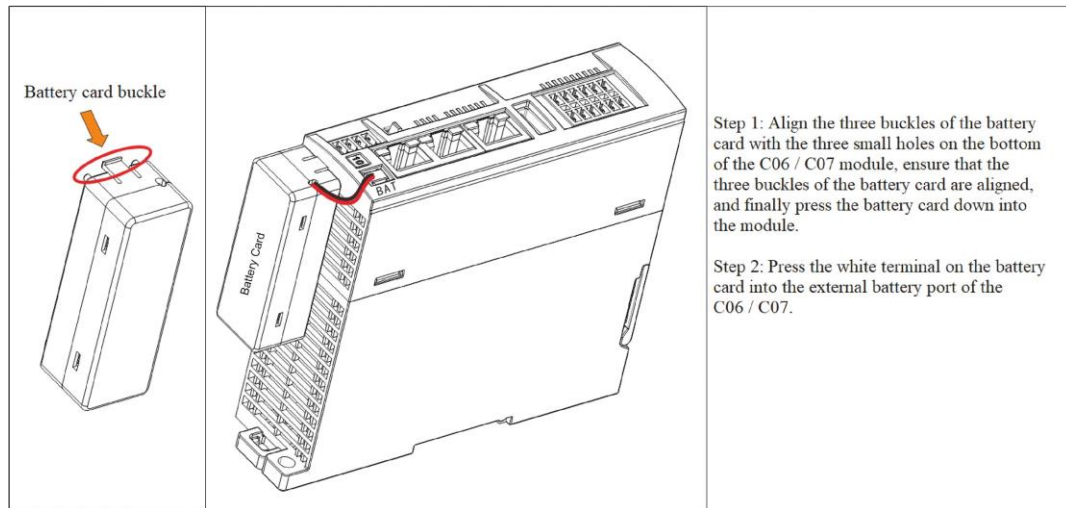
- 1) Make sure the pins on the module are aligned with the holes on the edge of the terminal block;
- 2) Press the terminal block down into the module, make sure the terminal block is aligned and locked.

- ♦ **Removal of terminal block**

Hold the terminal block and pull it upwards with even force.

- ♦ **Battery card installation**

Align the three snaps of the battery card with the three holes on the bottom of the C06/C07 module, make sure that the three snaps of the battery card are aligned, and then press the battery card down into the module. Finally, press the white terminal on the battery card into the external battery port of the C06/C07.



Wiring

Wiring Precautions:

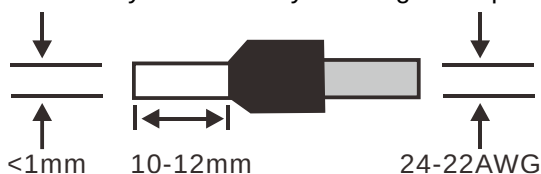
NOTIFICATION



- When installing or wiring, be sure to turn off all external power sources. Failure to turn off all power may result in electric shock to the user or damage to the product.
- After completing the installation or wiring, when starting the power supply or operating the PLC, it should be confirmed whether the terminal block is correctly inserted into the corresponding terminal base of the PLC. Otherwise, electric shock or malfunction may result.
- When wiring in the PLC, check the rated voltage and terminal configuration defined in the product specification to ensure correct and safe wiring. Connecting to a power source that is not rated or incorrectly wired for product safety may result in hazardous conditions such as fire or damage.
- External wiring configurations should be crimped, crimped and properly soldered using special tools. Improper wiring configuration can result in short circuits, fires, or incorrect work operation.
- It must be ensured that there are no foreign objects such as iron filings or wiring residues in each module. These foreign objects may cause fire, damage, or incorrect work operation.

Wiring Instructions:

- 1) Terminal blocks with press-fit insulating sleeves cannot be used. It is recommended to use a sleeve with a label or insulating material to cover the crimp terminal lugs.
- 2) Please use 24-22AWG single-core wire or multi-core wire for wiring connected to the terminal block, and use pin-type terminals with a diameter of less than 1mm for wiring, so as to improve the stability and reliability of wiring. The specifications are shown in the figure below:



3.5 Grounding and Wiring

❑ C06/C07 Motion Controller Grounding and Wiring Guidelines

Proper grounding and wiring are essential for all electrical equipment to ensure optimal operating characteristics for your system, as well as provide better protection from electrical noise for your system.

The wiring of the C06/C07 motion controller and its related equipment follows all valid electrical coding rules. Install and operate all equipment in compliance with all valid national or regional standards. Liaise with regional authorities to determine criteria that meet special needs.

Safety factors must be considered when designing the grounding and wiring of the C06/C07 system, otherwise the equipment may malfunction. Therefore, you should implement all safety regulations to avoid personal injury and equipment damage.

WARN



1. Attempting to ground or wire under live conditions may result in death or serious personal injury and equipment damage.
2. The control equipment may cause misoperation of the equipment it controls, resulting in death or serious personal injury and equipment damage. Therefore, the C06/C07 system must have the emergency stop function, electromechanical interlock or other redundant safety facilities independent of the C06/C07 motion controller.

3.6 Suppression Circuit

When using an inductive load, add a suppression circuit to limit the voltage rise when the output is turned off. Suppression circuitry protects the output from premature failure due to high inductive switching currents. In addition, the suppression circuit can limit the electronic noise generated when the inductive load is switched.

NOTIFICATION



The effectiveness of the suppression circuit depends on the application, and you should adjust its parameters to suit your particular application. Make sure that all device parameters match the actual application.

❑ Transistor outputs and relay outputs to control DC loads

The transistor outputs are internally protected to accommodate a variety of applications. Since the relay type output can be connected to both DC and AC loads, there is no internal protection.

Figure 3-6 shows an example of a DC load suppression circuit. In most applications, an additional diode A will suffice, but if your application requires a faster turn-off speed, it is recommended that you add a Zener diode B. Make sure that the Zener diode can meet the current requirements of the output circuit.

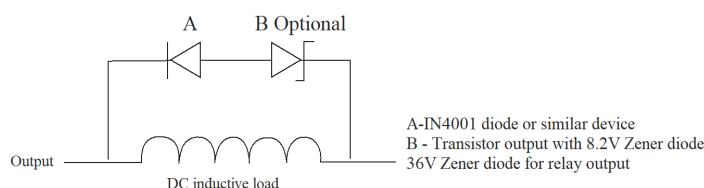


Figure 3-6 Suppression circuit for DC load

❑ AC output and relay output to control AC load

The AC output has internal protection for most applications, as the relay can be used for DC or AC loads so no internal protection is provided.

Figure 3-7 shows an example of an AC load suppression circuit. In most applications, an additional metal oxide variable resistor (MOV) can limit the peak voltage, thereby protecting the internal circuitry of the C06/C07 motion controller. Make sure the MOV is operating at least 20% higher than the normal line voltage.

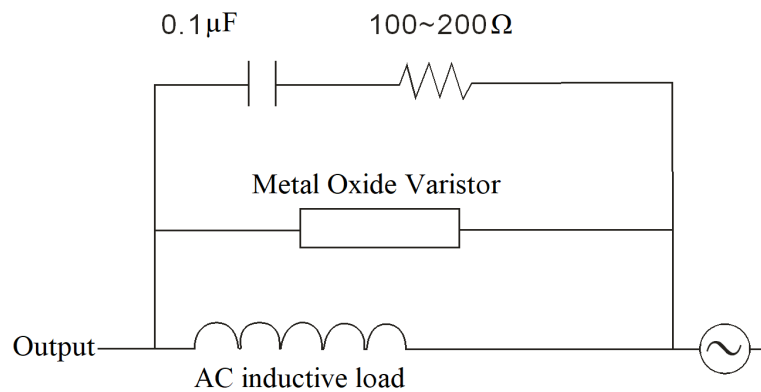


Figure 3-7 Suppression circuit for AC load

4 Power Module

4.1 Specifications

PWR-020S1 power supply module, the best match for MAT-LC-CXX series CPU, can provide 24V DC power supply for CPU and expansion modules (except digital modules). Please select a power supply module for each rack, and select other power supply for digital input and output power supply and sensor power supply.

Table 4-1 Basic attributes of the power module PWR-020S1

Name	Specifications	Order NO.
PWR-020S1 power module	Input: 85~264V AC Output: 24V DC/2A	MAT-PWR-020S1

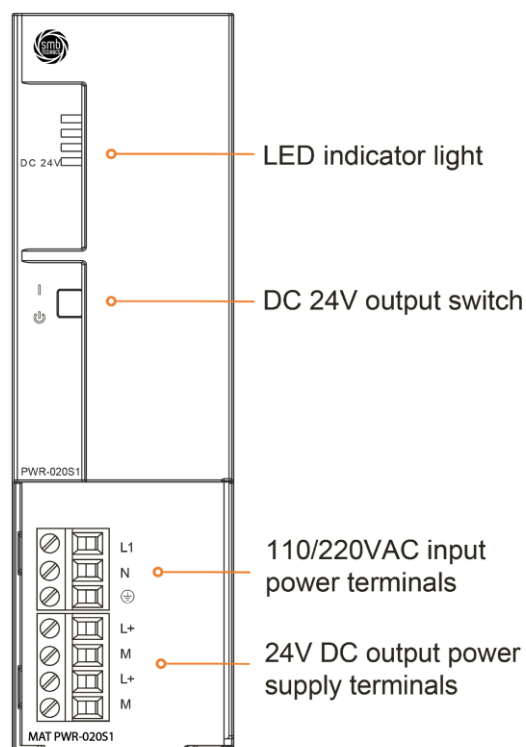
Table 4-2 Specifications and Characteristics of Power Module PWR-020S1

Physical Properties	
Size (W×H×D)	34×115×101.6 mm
LED Indicator Characteristics	
24V power indicator (green)	On: With 24V DC output, Off: Without 24V DC output
Switching Characteristics	
24V DC power control switch	On: With 24V DC output, Off: Without 24V DC output
Input Voltage Characteristics	
Voltage range	85~264VAC, Wide voltage input
Rated frequency	50Hz/60Hz
Frequency Range	47Hz~63Hz
Efficiency	75%
Alternating current	0.9A/110V、0.5A/220V
Inrush current (25°C max)	≤20A/110V、≤35A/220V
Leakage current	≤5mA/220VAC
Output Voltage Characteristics	
DC voltage/rated current	24VDC/2A
Rated power	48W
Ripple and Noise (Maximum)	150mVp-p
Voltage output range	±5%
Start/Rise/Hold Time	≤2.5s/≤50ms/≥20ms
Isolation (power input and output)	Isolation between 110V/220V AC and 24V DC
Protective function	
Overload protection	105%~130% of the rated output power, cut off the output, automatic recovery after troubleshooting
Overvoltage protection	115%~135%Ue; protection mode: hiccup mode, automatic recovery after troubleshooting
Surge protection	The power supply side provides surge absorption function

Overcurrent Protection	Power output provides overcurrent protection
Safe Electromagnetic Compatibility	
Withstand voltage	Input~Output: 1.5KVDC, Input-PE: 1.5KVDC, Output-PE: 500VDC
Isolation resistance	Input~Output, Input-PE, Output-PE: 100MΩ/500VDC
According to the standard	Safety refer to UL60950 and UL1950, electromagnetic compatibility refer to EN55022

4.2 Wiring Specifications

4.2.1 Interface diagram



4.2.2 Interface definition

Table 4-4 Definition of 220V AC input power interface of PWR-020S1

3-Position Removable Terminal	Signal	Signal definition
	L	Fire Wire
	N	Zero Line
		Ground

Table 4-5 Definition of 24V DC output power interface of PWR-020S1

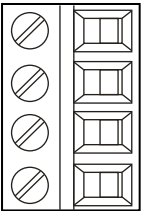
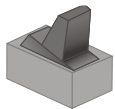
4-Position Removable Terminal		Signal	Signal definition
		L+	24Vpower supply positive
		M	24Vpower supply negative
		L+	24Vpower supply positive
		M	24Vpower supply negative

Table 4-6 DIP switch definition of PWR-020S1

Two-state switch	Tag number	Dial direction	Signal definition
	ON	Up	With 24V DC output
	OFF	Down	No 24V DC output

5 Main Control Module

The C06/C07 main control module is the central processing unit of the application system. As the core control device of the system, it is used to process various operations and the operation of user programs.

Table 5-1 Basic characteristics of the main control module

Name	Specifications	Order NO.
LC-C06 main control module	32MB program data space, 64KB power-down retention data, 55M expansion bus, 24V DC power supply, 10 digital inputs, 6*500KHz high-speed counter, 1 RS485 communication port (MODBUS free port protocol), 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, supports SoftMotion instruction set functions such as single-axis motion control, interpolation, electronic gear and electronic cam, and supports the SP18 Patch6 version of the CODESYS programming platform.	MAT-LC-C06
LC-C07 main control module	32MB program data space, 64KB power-down retention data, 55M expansion bus, 24V DC power supply, 10 digital inputs, 6*500KHz high-speed counter, 1 RS485 communication port (MODBUS free port protocol), 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, supports SoftMotion instruction set functions and Robotics, CNC functions such as multi-axis motion control, interpolation, electronic gear and electronic cam, and supports CODESYS programming platform SP18 Patch6 version.	MAT-LC-C07

5.1 Basic Performance Parameters

Table 5-2 General Features

Physical properties	
Dimensions (W×H×D)	34×115×101.6 mm
Power loss	19.2W
Power characteristics	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input Current	0.8A
Reverse polarity protection	With
Bus supply voltage	+5V DC
Bus supply current	1.6A
LED Indicator Features	
24V power indicator (green)	On: 24VDC power supply is normal; off: no 24VDC power

	supply
SF indicator (red)	On: System failure; Off: No error
BF indicator (red)	On: bus fault; off: no error
RUN indicator (green)	On: the system is running; off: the system is stopped
STOP indicator (yellow)	On: the system is stopped; off: the system is running
Network port communication indicator	
Link1 indicator (green)	On: Connected; Off: Not connected
SPEED1 indicator (yellow)	On: 100Mbps; Off: 10Mbps
Link2 indicator (green)	On: Connected; Off: Not connected
SPEED2 indicator (yellow)	On: 100Mbps; Off: 10Mbps
I0.0~I1.1 (green)	On: signal input; off: no signal input
Expanded I/O capability	
1 CPU supports the number of ETC-00 slaves	Up to 128
Number of modules supported by each ETC-00	Supports up to 8 modules
Power-down retention	
Clock power-down hold time	After the C06/C07 is connected to an external battery, the power-off retention time is typically more than 2 years (for the ordering information of the external battery, see I Ordering Information)
Programming software	
Programming package	CODESYS V3 (SP18 Patch6 version)
Programming language	Programming languages according to IEC61131-3: CFC/FBD/LD/IL/ST/SFC
Motion Controller Features	
Supported Motion Controller Features	LC-C06: SoftMotion LC-C07: SoftMotion and CNC

5.2 CPU Input Function

5.2.1 Digital Input

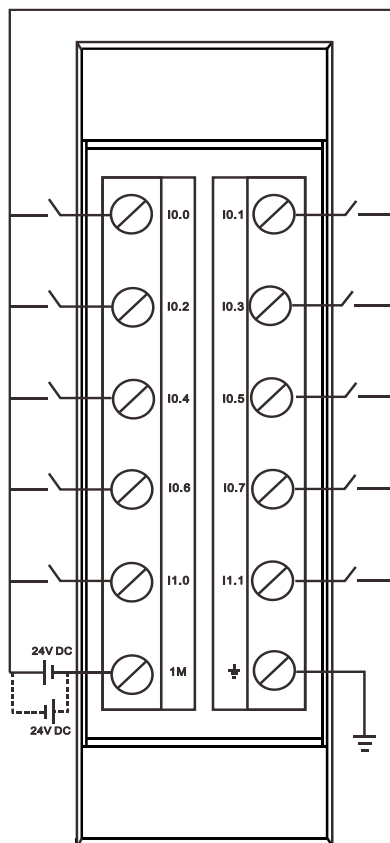
The MAT-LC-CXX series C06/C07 motion controller integrates 10 digital inputs, and its input characteristics are shown in the table below.

Table 5-3 Digital input characteristics of C06/C07 motion controller

Digital input feature	
Native integrated IO points	10
Input type	Sink/Source
Rated voltage	24V DC
Input voltage range	20.4~28.8V DC
Surge voltage	35V DC for 0.5s

Logic 1 signal (minimum)	15 VDC, 2.5mA	
Logic 0 signal (max)	5 VDC, 1mA	
Connecting a 2-wire proximity switch sensor (BERO)	1mA (maximum leakage current allowed)	
input filtering	Configurable, support 0.2us, 0.4us, 0.8us, 1.6us, 3.2us, 6.4us, 12.8us, 0.2ms, 0.4ms, 0.8ms, 1.6ms, 3.2ms, 6.4ms, 12.8ms, the default is 6.4ms	
Isolation (field vs logic)	500V AC, 1 minute	
Isolation group	see wiring diagram	
Inputs that are switched on at the same time	10	
Maximum cable length	500 meters (standard input)	
Shield	50 meters (high-speed counter input)	
Unshielded	300 meters (standard input)	
Digital Input Characteristics		
Pulse catch input	10	
High counter	Total	6
	Simplex	6×500KHz
	Two-phase	4×250KHz

CPU comes with IO wiring diagram



5.2.2 High-speed count input

Table 5-4 C06/C07 motion controller high-speed counter input points

Model	Description	Input			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.3	I0.4	I0.5	
	HSC2	I0.6	I0.7		
	HSC3	I1.0	I1.1		
	HSC4	I0.2			
	HSC5	I0.5			
0	Single-phase counter with internal direction control	Clock			
1		Clock		Reset	
2					
3	Single-phase counter with external direction control	Clock	Direction		
4		Clock	Direction	Reset	
5					
6	Two-phase counter with up/down count clock	Increase the clock	Minus the clock		
7		Increase the clock	Minus the clock	Reset	
8					
9	A/B Phase Quadrature Counter	Clock A	Clock B		
10		Clock A	Clock B	Reset	
11					

Table 5-5 High-speed counter parameters

Parameter	Type	Value	Default	Description
Module Id	DWORD	16#0a000605	16#0a000605	Module Id
channel0-3filter	BYTE	6	6	filter time, 1: 0.2ms 2: 0.4ms 3: 0.8ms 4: 1.6ms 5: 3.2ms 6: 6.4ms 7: 12.8ms 8: 0.2us 9: 0.4us 10: 0.8us 11: 1.6us 12: 3.2us 13: 6.4us 14: 12.8us
channel4-7filter	BYTE	6	6	filter time, 1: 0.2ms 2: 0.4ms 3: 0.8ms 4: 1.6ms 5: 3.2ms 6: 6.4ms 7: 12.8ms 8: 0.2us 9: 0.4us 10: 0.8us 11: 1.6us 12: 3.2us 13: 6.4us 14: 12.8us
channel8-9filter	BYTE	6	6	filter time, 1: 0.2ms 2: 0.4ms 3: 0.8ms 4: 1.6ms 5: 3.2ms 6: 6.4ms 7: 12.8ms 8: 0.2us 9: 0.4us 10: 0.8us 11: 1.6us 12: 3.2us

				13: 6.4us 14: 12.8us
HSC0 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC0 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(HZ) 1: 750K 2: 500K 3: 375K 4: 250K 5: 125K 6: 100K 7: 75K
HSC0 Current Value	DINT	0	0	
HSC0 Preset value	DINT	0	0	
HSC0 Speed Test Time(ms)	BYTE	5	5	
HSC0 Ctrl	BYTE	16#F9	16#F9	bit0 : Reset Level 0 : Low 1 : High bit1~bit2: quad Rate 0: 4x 1: 2x 2: 1x bit3: Direction 0:Decrease 1: Increase bit4: Direction Update 0: Not Update 1: Update bit5: Preset Value Update 0: Not update 1: Update bit6: Current Value Update 0: Not update 1: Update bit7: HSC Enable 0: Disable 1: Enable
HSC1 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC1 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(HZ) 1: 750K 2: 500K 3: 375K 4: 250K 5: 125K 6: 100K 7: 75K
HSC1 Current Value	DINT	0	0	
HSC1 Preset value	DINT	0	0	
HSC1 Speed Test Time(ms)	BYTE	5	5	
HSC1 Ctrl	BYTE	16#F9	16#F9	bit0 : Reset Level 0 : Low 1 : High bit1~bit2: Quad Rate 0: 4x 1:2x 2:1x bit3: direction 0:decrease 1: increase bit4: direction update 0: not update 1: update bit5: preset value update 0: not update 1: update

				bit6: Current Value update 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC2 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC2 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(HZ) 1: 750K 2: 500K 3: 375K 4: 250K 5: 125K 6: 100K 7: 75K
HSC2 Current Value	DINT	0	0	
HSC2 Preset value	DINT	0	0	
HSC2 Speed Test Time(ms)	BYTE	5	5	
HSC2 Ctrl	BYTE	16#F9	16#F9	bit0 : Reset Level 0 : Low 1 : High bit1~bit2: Quad Rate 0: 4x 1:2x 2:1x bit3: direction 0:decrease 1: increase bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update, 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC3 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC3 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(HZ) 1: 750K 2: 500K 3: 375K 4: 250K 5: 125K 6: 100K 7: 75K
HSC3 Current Value	DINT	0	0	
HSC3 Preset value	DINT	0	0	
HSC3 Speed Test Time(ms)	BYTE	5	5	
HSC3 Ctrl	BYTE	16#F9	16#F9	bit0 : Reset Level 0 : Low 1 : High bit1~bit2: Quad Rate 0: 4x 1:2x 2:1x bit3: direction 0:decrease 1: increase bit4: direction update, 0: not update 1: update

				bit5: preset value update, 0: not update 1: update bit6: Current Value update' 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC4 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC4 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(HZ) 1: 750K 2: 500K 3: 375K 4: 250K 5: 125K 6: 100K 7: 75K
HSC4 Current Value	DINT	0	0	
HSC4 Preset value	DINT	0	0	
HSC4 Speed Test Time(ms)	BYTE	5	5	
HSC4 Ctrl	BYTE	16#F9	16#F9	bit0 : Reset Level 0 : Low 1 : High bit1~bit2: Quad Rate 0: 4x 1:2x 2:1x bit3: direction 0:decrease 1: increase bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update, 0: not update 1: update bit7: HSC Enable 0: disable 1: enable
HSC5 MODE	BYTE	0	0	bit0~bit3: HSC Mode(0,1,3,4,6,7,9,10) bit4: Z lock disable 0: enable 1: disable bit5: z dear disable 0: enable 1: disable bit6: reserve bit7: clear lock data 0:clear 1:not clear
HSC5 Filter	BYTE	16#2	16#2	bit0~bit3: HSC Filter(HZ) 1: 750K 2: 500K 3: 375K 4: 250K 5: 125K 6: 100K 7: 75K
HSC5 Current Value	DINT	0	0	
HSC5 Preset value	DINT	0	0	
HSC5 Speed Test Time(ms)	BYTE	5	5	
HS8 Ctrl	BYTE	16#F9	16#F9	bit0 : Reset Level 0 : Low 1 : High bit1~bit2: Quad Rate 0: 4x 1:2x 2:1x bit3: direction 0:decrease 1: increase

				bit4: direction update, 0: not update 1: update bit5: preset value update, 0: not update 1: update bit6: Current Value update, 0: not update 1: update
--	--	--	--	---

5.3 Communication Function

5.3.1 Communication Port Specification

Table 5-6 Integrated Port Specifications

Integrated communication functions	
Bus interface	Provide expansion module interface
USB port	Provides USB host interface
EtherNET port	Provide 1 EtherNET interface
RS485 port	Provide 1 RS485 interface
EtherCAT port	Provides 1 EtherCAT interface
CAN port	Provide 1 CAN interface

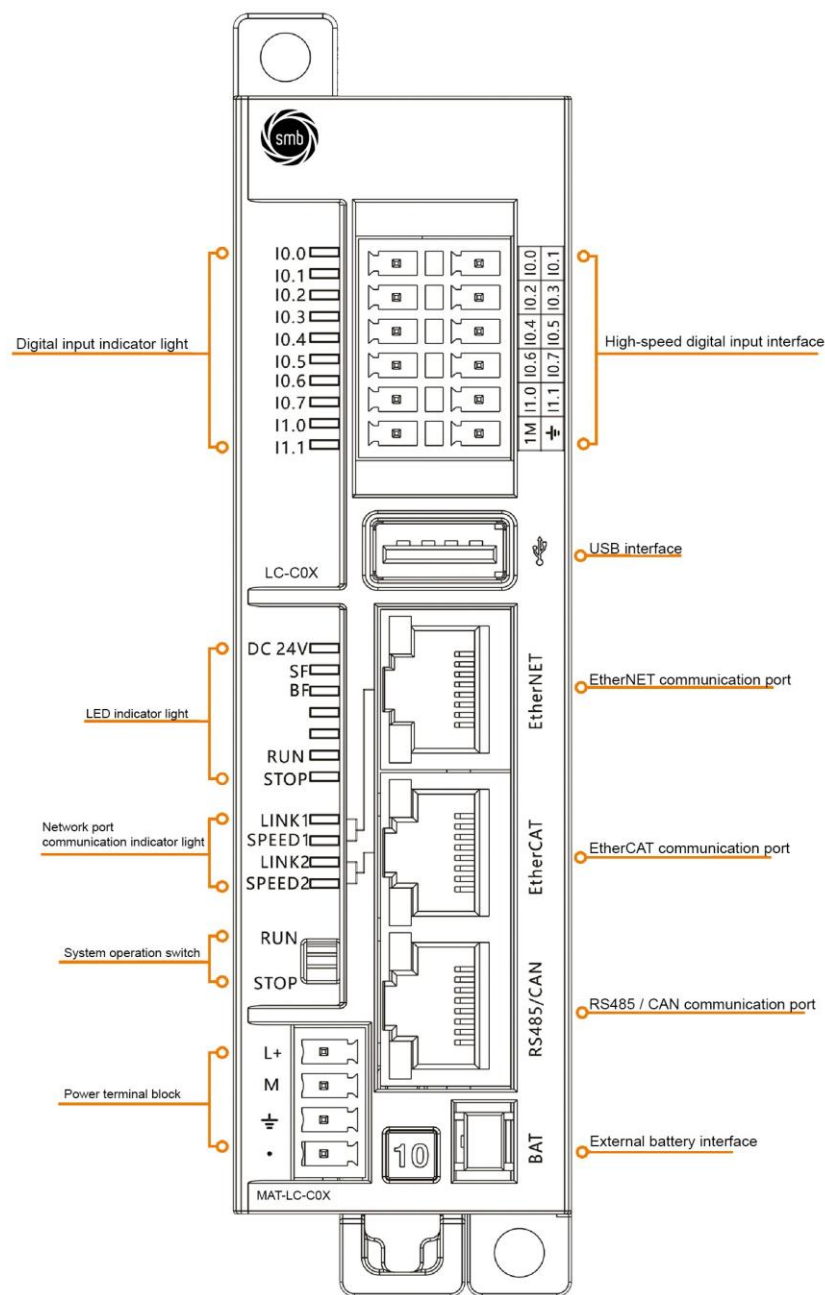
Table 5-7 Communication port specifications

RS485 communication port	
Number of communication interfaces and protocols	1 RS485 communication port (MODBUS free port mode)
Free port baud rate	1.2Kbps~115.2Kbps, built-in Modbus master/slave function
Maximum cable length per segment	Using an isolated repeater: 1000m (115.2Kbps)/1200m (38.4Kbps) Without isolation repeater: 50m
Maximum number of sites	32 stations per segment, 126 stations per network
Isolation	Communication port isolation
EtherNET Communication Port	
Communication Interface	1 EtherNET port
Baud rate	10/100Mbps adaptive
Agreement Type	UDP protocol, Modbus TCP protocol, EtherNET/IP protocol
Maximum cable length per segment	100m
The maximum number of connections to a site	UDP supports up to 16 connections and Modbus TCP supports up to 32 connections.
Isolation	Communication port isolation

Ethernet network read and write commands	Parameter	Type	Meaning
TCP_MBUS_MSG	EN		TCP_MBUS_MSG enable bit
	First	BOOL	Read and write request bits, each new read and write request must be triggered by a pulse
	IP	STRING	Destination IP address, MODBUS over TCP fixed port 502
	RW	BYTE	Operation command: 0--read, 1--write
	Addr	DWORD	Select the data type for reading and writing, 0000 to 0xxxx--switch output 10000 to 1xxxx--switch input 30000 to 3xxxx--analog input 40000 to 4xxxx--holding registers
	Count	INT	The number of communication data (number of bits or words), the maximum amount of data that can be read/written by each MBUS_MSG instruction of the Modbus master is 120 words
	DataPtr	INT	Data pointer, if it is a read command, the data read back is placed in this data area; if it is a write command, the data to be written is placed in this data area
	Done	BOOL	Done bit, read and write function done bit
	Error	BYTE	Error code, only when the Done bit is 1, the error code is valid, the error code is as follows: 0 = no error 1 = response checksum error 2 = not used 3 = Receive timeout (no response from slave) 4=Request parameter error (slave address, Modbus address, count, RW) 5=Modbus/Freeport not enabled 6=Modbus is busy with other requests 7=response error (response is not the requested operation) 8=esponse CRC checksum error 101=Slave does not support the requested function 102=Slave does not support data address 103=Slave does not support this data type 104=Slave device failure 105=Slave accepted the message, but the response was delayed 106=Slave is busy and rejected the message

			107 = Slave rejected the message 108 = Slave memory parity error				
EtherCAT Communication Port							
Communication Interface	1 EtherCAT communication master interface						
Baud rate	10/100Mbps adaptive						
Agreement type	EtherCAT interface protocol						
Maximum cable length per segment	100m						
Maximum number of sites	Each master supports up to 128 EtherCAT slaves						
Supported Features	Support distributed clock configuration, redundant settings Support startup parameter configuration Support for configuring PDO parameters and mapping Supports configuration of bus cycle period, configuration startup checks vendor ID and product ID						
Power protection	The power supply end provides reverse polarity protection and surge absorption						
Interface protection	Lightning protection of communication port						
Isolation	Communication port isolation						
CANopen Communication Port							
Communication Interface	1 CAN communication master interface						
Maximum number of slave sites	Up to 32 slaves can be connected behind 1 master						
Agreement type	CANopen DS301 standard protocol						
Supported Features	Auto start CANopen manager, optional slave polling, start slave, NMT, synchronous production, synchronous consumption, heartbeat generation, create activation time						
Transmission rate (kbit/s)	1000	800	500	250	125	50	20
Maximum length (m)	25	50	100	250	500	1000	2500
Isolation	Communication port isolation						

5.3.2 Schematic diagram and definition of external interface



Schematic diagram of external interface

Table 5-8 Definition of power interface

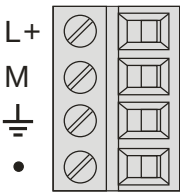
4-position removable terminals (3.50mm pitch)	Symbol	Signal definition
	L+	24Vpower supply positive
	M	24Vpower supply negative
	⏏	24Vpower ground
	--	--

Table 5-9 Definition of USB interface

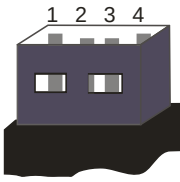
USB interface	Tag number	Symbol	Signal definition
	1	V_BUS	+5V power supply
	2	Data-	negative data
	3	Data+	data is positive
	4	GND	ground

Table 5-10 Definition of EtherNET/EtherCAT interface

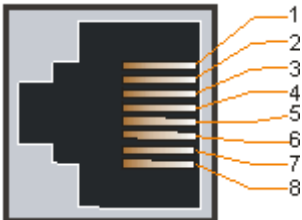
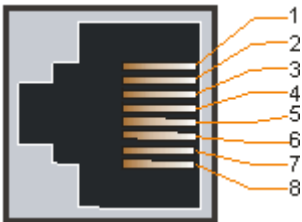
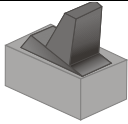
Dual network port interface	Tag number	Symbol	Signal definition
	1	TX+	Data sending positive end
	2	TX-	Data sending negative terminal
	3	RX+	Data receiving positive end
	4	TERM	--
	5	TERM	--
	6	RX-	Data receiving negative terminal
	7	TERM	--
	8	TERM	--
	Shell	PE	Ground

Table 5-11 Definition of RS485/CAN communication interface

Single Network Port	Tag number	Symbol	Signal definition
	1	CAN_H	Data sending positive end
	2	CAN_L	Data sending negative terminal
	3	--	--
	4	A0	RS485 Signal A/—
	5	B0	RS485 signal B/+

	6	--	--
	7	CAN_GND	CAN/RS485 signal ground
	8	--	--
	Shell	PE	Chassis ground

Table 5-12 Definition of system running DIP switch

System Run Switch	Symbol	Dial direction	Signal definition
	RUN	Up	System Operation
	STOP	Down	System Stop

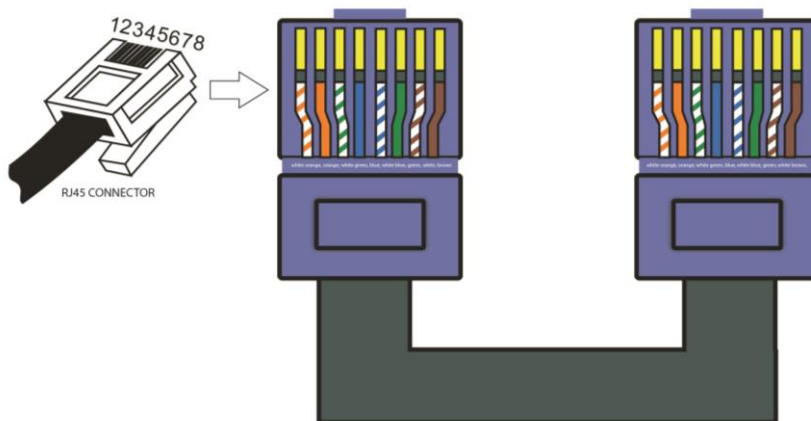
5.3.3 Make standard network cables

When the C06/C07 motion controller performs the following communications, it is recommended to use a standard network cable:

- ♦ The main control module communicates with the host computer via EtherNET
- ♦ Master control module for EtherCAT communication
- ♦ Main control module for CANopen communication
- ♦ The main control module performs CANopen communication with CAN slaves such as SMB

Refer to the following diagram to make a standard network cable:

RJ45 type network cable plug 568B wire sequence connection method



5.4 Data Memory Specifications

C06/C07 series motion controllers store information in different memory units, each unit has a unique address, you can specify the memory address to be accessed, which allows the user program to directly access this information.

Memory location and size are represented by special character sequences, syntax: %<range prefix><length prefix><number|.number|.number....>

Table 5-13 Available memory areas:

Storage Area	Name	Size (Bytes)
I	Input fields (physical inputs driven by inputs, "sensors")	1280
Q	Output area (physical outputs driven by outputs, "actuators")	1280
M	Memory area	32767

Table 5-14 The following length prefixes can be used:

X	Single binary bit
None	Single binary bit
B	Byte (8 bits)
W	Word (16 bits)
D	Double word (32 bits)

Table 5-15 Examples

%QX7.5	Address 7 of the output area, bit 5
%IW215	Address 215 of the input area, 1 word
%QB7	Address 7 of the output area, 1 byte
%MD48	Address 48 of the memory area, double word.
%IW2.5.7.1	Depends on PLC settings
ivar AT %IW0 : WORD;	t Example of variable declaration with specified address

Make sure to use a valid address:

To specify an effective address in your application, you must first know the appropriate location in the process image, which is the memory area used: Input(I), Output(Q) or Memory(M) area, as described in the table above. Then about specifying the desired size:

The current device configuration and settings (hardware structure, device description, IO settings) play a decisive role. In particular, take into account the difference in address interpretation between devices using "byte addressing mode" and devices using "IEC addressing mode" word-oriented mode.

So depending on the size and addressing mode, different memory locations can be addressed by the same address definition.

See the table below for a comparison of byte addressing and word-oriented IEC addressing for bits, bytes, words, and dwords. This table plots the overlapping memory regions in byte addressing mode (see example after table).

Regarding the writing method, please note that for bit addresses, the IEC addressing mode is always word-oriented, that is, the area before the dot corresponds to the number of words, and the area after the name corresponds to the number of bits.

Table 5-16 Comparison of byte- and word-oriented addressing for address sizes D, W, and X:

Double word/Word				Byte	X (bit)					
Byte addressing		Word-Oriented IEC Addressing			Byte addressing			Word-Oriented IEC Addressing		
D0	W0	D0	W0	B0	X0.7	...	X0.0	X0.7	...	X0.0

D1	W1			B1	X1.7	...	X1.0	X0.15	...	X0.8
...	W2		W1	B2				X1.7	...	X1.0
	W3			B3				X1.15	...	X1.8
	...	D1	W2	B4						
				B5						
			W3	B6						
				B7						
		D2	W4	B8						
		...		...						
		...								
		...								
D(n-3)		D(n/4)								
	W(n-1)		W(n/2)							
				Bn	Xn.7	...	Xn.0	X(n/2). 15	...	X(n/2). 8

n = number of bytes

In byte addressing mode, the memory overlap varies, for example:

D0 includes B0-B3, W0 includes B0 and B1, W1 includes B1 and B2, and W2 includes B2 and B3. To avoid overlapping, neither W1 nor D1, D2, D3 can be used for addressing!

<Note> If no explicit single-bit address is specified, the Boolean value will be one byte in length. For example: changing the value of varbool1 AT %QW0 will affect the content from QX0.0 to QX0.7.

<Note> Be careful when using pointers to addresses, online modifications may change the contents of the address!

5.5 Features of C06/C07 Series Motion Controllers

1、Communication function

Select the communication interface for the network

♦ C06/C07 series motion controllers support various types of communication networks, and the interfaces that can access these networks include:

- ♦ Bus interface
- ♦ RS485 interface (RJ45 standard network port)
- ♦ EtherCAT interface (RJ45 standard network port)
- ♦ CANOpen interface (RJ45 standard network port)
- ♦ EtherNET interface (RJ45 standard network port)

Choose a communication protocol for your network

Protocols supported by C06/C07 series motion controllers include:

- ♦ Bus protocol
- ♦ UDP protocol

- ♦ Modbus TCP protocol
- ♦ Modbus RTU protocol
- ♦ CANopen DS301 standard protocol
- ♦ EtherCAT protocol
- ♦ EtherNET/IP protocol

Building a network based on communication protocols

Table 5-17 Introduction to the communication network

Way of communication	Communication Interfaces and Protocols	Implement communication on CODESYS
Bus communication	Connect the C06/C07 series expansion modules through the bus interface of the C06/C07 motion controller (supporting bus protocol) to realize bus communication.	Add a device that matches the actual expansion module through "SMB LocalBus" in the device catalog, and then read and write memory through the IO map of the corresponding module.
EtherNet communication	Through the EtherNET communication port of the C06/C07 motion controller (supporting UDP, EtherNET/IP industrial Ethernet protocols), the C06/C07 series motion controller and the communication device are connected via a communication cable, and EtherNET/IP communication can be performed.	Add EtherNET/IP communication devices through the device catalog: EtherNet master, EtherNET/IP slave, etc.
EtherCAT communication	Through the EtherCAT interface of the C06/C07 motion controller (supporting the EtherCAT protocol), the C06/C07 motion controller and the EtherCAT slave station are connected via a standard Ethernet cable to construct a linear network.	Add EtherCAT communication devices through the device catalog: EtherCAT Master, EtherCAT Slave, etc.
Modbus communication	Modbus RTU communication can be carried out through the RS485/CAN communication port of the C06/C07 motion controller; Modbus TCP communication can be carried out using its EtherNET communication port.	The SMB_Modbus_V1.1.library file provided by SMB can be used for MODBUS RTU and MODBUS TCP communication. For the use of the SMB_Modbus_V1.1.library, please refer to the appendix "E Introduction to the Use of the SMB_Modbus_V1.1.library".
CANopen	The C06/C07 motion controller is	Add CANopen communication

communication	connected to the CAN slave station through the RS485/CAN communication port and the standard network cable, and CANopen communication can be realized.	devices through the device catalog: CANBUS, CAN master, CAN slave, CAN slave module, etc.
---------------	--	---

<Remarks> Please refer to chapter "7 User Guide" for the usage of various communication methods.

2、Motion control function

Using a high-performance motion control software platform, embedded single-axis or multi-axis motion control algorithms and instructions.

Electronic cam function

- ◆ Support synchronization functions such as electronic gear/electronic cam, virtual axis and encoder axis.
- ◆ Graphical cam editor enables fast execution of complex motion control trajectories. The cam table can be corrected in real time and dynamically.
- ◆ Motion control of general functions can be realized through PLCopen function blocks in the motion control library.

CNC function (only supported by LC-C07)

- ◆ Support DIN66025 G code and DXF file import.
- ◆ Support 5-axis linkage, support parallel robot control.
- ◆ CNC Path View, all changes made in the CNC text editor are automatically updated in the Path View.

3、Logic control function program organization unit

The C06/C07 motion controller consists of the following program organization units:

Program (Program): The program is the location where the control application program instructions are stored. The instructions in the program are executed in sequence and in different scanning methods.

Function Block: A function block is a block written by the user with its own storage area. Each time a function block is called, various types of data need to be provided to the function block, and the function block also returns variables to the calling block.

Function (Function): Function (FC) belongs to the block of personal programming. A function is a logical block without memory. Temporary variables belonging to FCs are kept in the local data stack. Since the FC itself has no memory, it must always be assigned actual parameters. An initial value cannot be assigned to the local data of the FC.

Programming language

All programming languages specified in the IEC_61131 standard are supported:

- ◆ Languages in text form: Instruction List (IL), Structured Text (ST)

- ♦ Graphical languages: Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD), Continuous Function Chart (CFC) based on Function Block Diagram

<Remark> Please refer to chapter "6.2 Language" for the description of programming language.

Standardized PLC functions

Standardized PLC functions can be realized by using related instructions, such as: type conversion, numerical function, arithmetic function, shift function, Boolean operation function, selection function, comparison function, character string function, timer, counter, edge detection, bistable state elements, etc.

5.6 Real Time Clock Function

- ♦ Power-down retention time is about 112 hours (typical value)
- ♦ The C06/C07 comes with an external battery port. After the external battery is connected, the power-down retention time is a typical value of more than 2 years.
- ♦ Deviation < 60 seconds per month
- ♦ Read/set the clock with the command setrtc

Set or read the real-time clock of the C06/C07 series motion controller in CODESYS: double-click to open "Device (MAT-LC-C06_V1.0)" in the device view, and then select the tab "PLC Shell":

1) Enter a command in the PLC shell, such as "setrtc 2023-09-09 09:00:00", and then press the ENTER key to set the current RTC;

2) Enter the command "setrtc" in the PLC shell, and then press the ENTER key to read the current RTC;

<Remarks> The C06/C07 motion controller has a random time when it leaves the factory, so you need to manually set the current time first, and then read the RTC again to display the correct time.

6 Structure of CODESYS

CODESYS is a device-independent PLC programming system. CODESYS supports not only all programming languages compliant with the IEC 61131-3 standard, but also the C language. Combined with the CODESYS real-time system, multiple controller devices can be configured in one project. The CODESYS programming user interface consists of an arrangement of "components".

<Remarks> This manual only introduces the application of CODESYS. For the detailed use of CODESYS, please refer to the online help of CODESYS software.

6.1 Composition of The Project

A project contains all the objects in the PLC program, and is stored in the name of the project file. The project contains the following objects: POU, data types, visualization interfaces, resources and libraries.

POU

POUs refer to "Program Organization Units", which can be programs, function blocks, or function types, and they can be added at any time.

Each POU contains a local and subject declaration definition, the main part can be written in IEC languages, these languages include Instruction List (IL), Structured Text (ST), Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD) or Continuous Function Chart (CFC).

CODESYS supports all IEC standard POU. If these POU are used in the project file, the standard library file standard.lib must be included in the project file. POU can call other POU, but recursive calls are not allowed.

Function

A function is a POU that correctly produces a data element (which can contain several elements, such as fields or structures), which can be called as an operand in a textual language expression during processing. When declaring a function, be sure to give it a type, that is, put a colon after the function name followed by a data type.

A correct function declaration can refer to the following example:

```
FUNCTION Fct: INT
```

In addition, a result must be assigned to the function, that is, the function name as an output variable.

The declaration of a function begins with the keyword FUNCTION.

Function calls in IL are arranged within a single step or a single transition. A call to a function in ST can be used as an operand in an expression. Below is a function in IL that takes three input

variables and returns the result of dividing the product of the first two variables by the third variable.

Function block

A function block is a POU that provides one or more values in a program. In contrast to functions, a function block has no return value. The declaration of a function block begins with the keyword `FUNCTION_BLOCK`, and a copy or instance of the function block can be created.

Program

A program is a POU that returns several values during operation, and the program is global in the project file. All final values of the program are retained until the next program run.

PLC_PRG

`PLC_PRG` is a special predefined POU, and each project file must contain such a special program. Actually this POU is called only once per control loop.

After a new project file is created, the command “Project” → “Object Add” will be used for the first time. The default item in the dialog box of the POU is a POU of program type named `PLC_PRG`. These default settings cannot be changed.

If tasks are defined, `PLC_PRG` may not be included in the project, because in this case, the timing of the program depends on the assignment of tasks.

<Note> Do not delete or rename the POU `PLC_PRG` (if no task configuration is used), `PLC_PRG` is the main program in a single task program.

Action

Actions can be defined and assigned to function blocks and programs. Actions represent a further execution. It can be created in other languages. Each action has a name.

Each action works with data in a function block or program, and the action is executed using the same input/output variables and local variables.

Resource

You need resources to configure and organize project files and track variable values.

- Global variables are used in all project files or in the network.
- The library manager is used to add library files to the project.
- Log files record actions while online.

PLC Configuration configures the hardware of the programmable controller.

- Task configuration guides the work of the program through task division.
- Watch and Receive Manager to display variable values and set default variable values.
- Target system settings are used to select and, if necessary, make final configuration of the target system.
- Workspace as an image of project options.

The target system and target system settings required to build the project in CODESYS may also use the following resources:

- Sample trace for graphical display of variable values.
- Parameter manager for exchanging data with other controllers in the same network.

- PLC browser for controller monitoring.
- Tools, related to the target system, call external tool programs inside and outside CODESYS.

Library file

A series of library files can be included in the project file, and the POU's, data types, and global variables of the library files can be used like custom variables. The standard.lib and util.lib in the library file are program units that can be called freely.

Type of data

With reference to standard data types, users can define their own data types, and can create structure enumeration types and reference types.

Visual interface

CODESYS provides a visual interface, so the variables of the project can be displayed. With the help of the visualization, the geometry can be drawn offline. In the online mode, their shape, color and text output can be changed according to the value of specific variables.

The visual interface can be used as a PLC utility operator interface for HMI with CODESYS, or as a web page or target system display, visualized directly via the Internet or PLC.

6.2 Language

CODESYS provides corresponding editors that support all programming languages specified in the IEC_61131 standard:

- Languages in text form: Instruction List (IL), Structured Text (ST)
- Graphical languages: Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD), Continuous Function Chart (CFC) based on Function Block Diagram

6.2.1 Instruction List (IL)

The instruction list contains a series of instructions, depending on the type of operation, each instruction starts on a new line and contains an operation symbol and one or more operands separated by commas. In front of an instruction, there can also be a label followed by a colon. The comment section is at the end of a line, and blank lines can be inserted between instructions.

The following operators and qualifiers will be used in the instruction list:

Qualifier:

C is used in conjunction with operators JMP, CAL, RET: This instruction is only executed when the result of processing the preceding expression is TRUE.

N is used in conjunction with operators JMPC, CALC, RETC: this instruction is only executed when the result of processing the preceding expression is FALSE.

N is used in other cases: negates the operand (excluding the accumulator).

Here are the operators and their possible qualifiers and their associated meanings:

Operator and Qualifier Meaning	
LD N	make the current value equal to the operand
ST N	saves the current value at the operand's place
S	Set the Boolean operand to TRUE when the current value is TRUE
R	Set the boolean operand to FALSE when the current value is TRUE
AND N, (	Bitwise logical operator "and"
OR N, (	Bitwise logical operator "or"
XOR N, (	Bitwise logical operator "exclusive or"
ADD (	Addition
SUB (	Subtraction
MUL (	Multiplication
DIV (	Division
GT (	>
GE (	>=
EQ (	=
NE (	<>
LE (	<=
LT (	<
JMP CN	Jump to label
CAL CN	Caller function block
RET CN	Leave the POU and return to where it was called
) To perform a delayed operation	

[Click here for a list of all IEC operators.](#)

Example: Program written with qualifiers

LD TRUE (* load TRUE into accumulator*)

ANDN BOOL1 (*"AND" after performing AND and inversion of BOOL1 variable *)

JMPC mark (*When the above result is TRUE, jump to the label "mark" *)

LDN BOOL2 (*save the reverse of BOOL2 *)

ST ERG (*save BOOL2 in ERG *)

Lable:

LD BOOL2 (*Save the value of BOOL2 *)

ST ERG (*save BOOL2 in ERG *)

It is also possible to put a parenthesis after an operation in IL. The value in parentheses is considered an operand. E.g:

LD 2

MUL 2

ADD 3

Erg

Here Erg is 7, but if you add a parenthesis:

LD 2

MUL (2

ADD 3

)

ST Erg

The result of Erg is 10, and the operation MUL starts to calculate when ")" is reached; at this time, the operand 5 calculates the MUL.

6.2.2 Structured Text (ST)

Structured text contains a series of instructions written in a high-level language that can be executed (eg IF...THEN...ELSE) or in a loop (WHILE...DO).

Expression

An expression is a structure that returns a value after an operation. Expressions consist of operators and operands, which can be constants, variables, function calls, or other expressions.

Evaluating expressions

The value of the expression is calculated by processing the operands according to certain rules, with the most binding operators participating first, followed by the slightly more binding operators, until all operators have been processed. Operation symbols of the same binding force are processed in left-to-right order.

The following is a hierarchical arrangement of operator symbol constraints in structured text:

Action	Symbol	Binding
Put in parentheses	(expression)	The strongest binding force
Function call	Function name (parameter list)	
Exponentiation	EXPT	
Negate number	-	
Multiplication	*	
Division	/	
Modulo	MOD	
Addition	+	
Subtraction	-	
Compare	<,>,<=,>=	
Equal	=	
Not equal	<>	
Boolean operation "and"	AND	
Boolean operation "XOR"	XOR	
Boolean operation "XOR"	OR the weakest binding	

The following are other instructions in structured text, arranged in a table with the example.

Instruction type	Example
Assign	A: =B; CV : = CV + 1 ; C: =SIN(X);
Call a function block and use the function block output	CMD_TMR(IN : = %IX5, PT : = 300) ; A: =CMD_TMR.Q
RETURN	RETURN ;
IF	D: =B*B ; IF D<0.0 THEN C: =A; ELSIF D=0.0 THEN C: =B; ELSE C: =D; END_IF;
CASE	CASE INT1 OF 1: BOOL1: = TRUE; 2: BOOL2: = TRUE; ELSE BOOL1: = FALSE; BOOL2: = FALSE; END_CASE;
FOR	J: =101 ; FOR I: =1 TO 100 BY 2 DO IF ARR[I] = 70 THEN J: =I; EXIT; END_IF; END_FOR;
WHILE	J: =1 ; WHILE J<= 100 AND ARR[J] <> 70 DO J: =J+2; END_WHILE;
REPEAT	J: =-1 ; REPEAT J: =J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT ;
Empty instruction	;

Assignment operator

The left side of the assignment symbol is an operand (variable, address), and the right side of the ":= " is the value of the expression assigned to it, for example:

Var1:=Var2*10

After the operation is over, the variable Var1 gets 10 times the value of Var2.

Calling function blocks in structured text

A function block is called by writing the instance name of the function block and then assigning values to the parameters in parentheses. In the following example, a timer is called by assigning values to two parameters IN and PT, and then the value of the resulting variable Q is assigned to variable A.

RETURN instruction

The return instruction can be used to conditionally leave a POU (POU).

IF instruction

The IF instruction can check a condition and, according to this condition, execute the instruction.

Grammar:

```
IF <Boolean_expression1> THEN
<IF_instructions>
{ELSIF <Boolean_expression2> THEN
<ELSIF_instructions1>
.
.
ELSIF <Boolean_expression n> THEN
<ELSIF_instructions n-1>
ELSE <ELSE_instructions>}
END_IF;
```

Parts within {} are optional.

If the Boolean operation expression <Boolean expression> returns TRUE, only the part of the if instruction is executed, and the other parts are not executed. Otherwise, the Boolean expression starts with <Boolean expression 2> and is evaluated one by one until a Boolean expression returns TRUE. Then, after this Boolean operation expression 2, the part before ELSE or ELSE IF is calculated.

If none of the boolean expressions return TRUE, then only the instructions under ELSE are evaluated. E.g:

```
IF temp<17
THEN heating_on : = TRUE;
ELSE heating_on : = FALSE;
END_IF;
```

Here the heating starts when the temperature drops below 17 degrees, otherwise it remains off.

CASE Instruction

Using the CASE instruction, you can combine multiple conditional judgment instructions with the same condition variable in one structure. Sentence:

```
CASE <Var1> OF
<Value1>: <Instruction 1>
Languages...
```

```
<Value2>: <Instruction 2>  
<Value3, Value4, Value5>: <Instruction 3>  
<Value6 .. Value10>: <Instruction 4>  
...  
<Value n>: <Instruction n>  
ELSE <ELSE instruction>  
END_CASE;
```

CASE instructions are processed according to the following patterns:

If the variable Var1 has the value Value1, then the instruction Instruction1 is executed.

If the variable Var1 is not the specified value, then execute the ELSE Instruction.

If there are multiple variable values to execute the same instruction, then these conditions execute a common instruction.

If the same instruction is executed within a range of values for a variable, the initial and final values are separated by two periods, so common conditions can be specified.

Example:

```
CASE INT1 OF  
1, 5: BOOL1 : = TRUE;  
      BOOL3 : = FALSE;  
2: BOOL2 : = FALSE;  
      BOOL3 : = TRUE;  
10..20: BOOL1 : = TRUE;  
      BOOL3: = TRUE;  
ELSE  
      BOOL1 : = NOT BOOL1;  
      BOOL2 : = BOOL1 OR BOOL2;  
END_CASE;
```

FOR Loop

Through the FOR loop program, it is possible to write processing procedures that are executed repeatedly.

Sentence:

```
INT_Var :INT;  
FOR <INT_Var> : = <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO  
<Instructions>  
END_FOR;
```

Parts within {} are optional.

As long as the counter INT_Var is not greater than END_VALUE, the instruction Instructions will always be executed. Before executing the Instructions, check the value of the counter first. If the INIT_VALUE is greater than the END_VALUE, the Instructions will not be executed.

When Instructions are executed, INT_Var is usually increased by a Step size. Step size can be any integer value. If there is no Step size, it will be set to 1. When INT_Var reaches a certain value, the loop ends.

Example:

```
FOR Counter: =1 TO 5 BY 1 DO
```

```
Var1: =Var1*2;
```

```
END_FOR;
```

```
Erg: =Var1;
```

Let's assume that the default value of Var1 is 1, then it will get the value 32 after the loop ends.

<Note> END_VALUE must not be greater than or equal to the limit value of the counter INT_VAR, for example: if the variable counter is a SINT type and END_VALUE is 127, then this will be an infinite loop.

WHILE Loop

A WHILE loop can be used like a FOR loop, except that the exit condition of the WHILE loop can be any boolean expression, and when the condition is met, the loop will be executed. Sentence:

```
WHILE <Boolean expression>
```

```
<Instructions>
```

```
END_WHILE;
```

As long as the Boolean_expression returns TRUE, the Instructions are repeated. If the Boolean_expression evaluates to FALSE for the first time, the instruction will not be executed. If the Boolean_expression never occurs FALSE, the Instructions will be repeated endlessly.

<Note> The programmer must ensure that an infinite loop does not occur. This can be achieved by changing the condition of the instruction part of the loop, eg, the counter can be incremented or decremented.

Example:

```
WHILE counter<>0 DO
```

```
Var1 : = Var1*2;
```

```
Counter : = Counter-1;
```

```
END_WHILE
```

For the WHILE and REPEAT loops, it is not necessary to know the number of loops before the loop, and in this sense, these two kinds of loops are more powerful than FOR.

So in such cases, both loops can be used. If the number of loops is more specific, then it is better to use FOR because there is no infinite loop.

REPEAT Loop

The difference between a REPEAT loop and a WHILE loop is that its break condition is checked after the loop executes, which means that the loop executes at least once, regardless of the break condition.

Sentence:

REPEAT

<Instructions>

UNTIL <Boolean expression>

END_REPEAT;

Instructions are executed until the Boolean expression returns TRUE. If the Boolean expression is assigned a true value for the first time, the Instructions are executed only once, otherwise the Instructions will be executed repeatedly and will cause a time delay.

<Note> The programmer can ensure that no infinite loop occurs by changing the condition of the instruction part of the loop, for example, by incrementing or decrementing the counter.

Example:

REPEAT

Var1 : = Var1*2

Counter : = Counter-1;

UNTIL

Counter=0

END_REPEAT;

EXIT Instruction

If there is an EXIT instruction in a FOR, WHILE or REPEAT loop, then the inner loop ends, regardless of the interrupt condition.

6.2.3 Sequential Function Chart (SFC)

Sequential function chart is a graphical language that can be used to describe the sequence of different actions in a program. Because these actions are assigned to single-step elements, and transition variables are used to control the order of processing.

Step

A POU written in a sequential function diagram contains a series of steps, which are realized through directed connections (transition conditions).

There are two types of steps:

- Simple type: Each step consists of an action and a flag to indicate whether the step is active or not. If a single step action is in progress, a small triangle will appear in the upper right corner of the step.
- IEC Types: Actions or Boolean variables with each step consisting of a flag and one or more assignments. The associated action appears to the right of the step.

Action

An action can consist of a series of instruction lists or structured text instructions, a network of function block diagrams or ladder diagrams, or another sequential function diagram.

In simple steps, actions are often linked to steps. To edit an action, double-click on the step or select the step, and then choose the menu command "Extras" "Zoom Action / Transition". In addition, only one input or output action is allowed in each step.

The action of the IEC step is attached in the Object Manager within the SFC - POU, which can be loaded by double-clicking or pressing the Enter key in its editor. You can also create a new action through "Project" "Add Action". Up to nine actions can be assigned to an IEC step.

Entry and exit actions

An additional entry and exit action can be added to a step. After a step is activated, an entry action can only be executed once. The exit action is executed only once before the step expires.

A step with an entry action is represented by an "E" in the lower left corner, and an exit action is represented by an "X" in the lower right corner.

Transition / Transition Condition

There are so-called transitions between steps.

The value of the transition condition must be TRUE or FALSE, so it can be a Boolean variable, a Boolean address, or a Boolean constant. It can also include a sequence of commands with boolean results in structured text patterns (eg $I \leq 100$) AND b) or in any desired language (see 'Extras' 'Zoom Action/Transition') . A transformation cannot include programs, function blocks, or assignments.

<Note> In addition to transitions, you can also use progressive mode to skip to the next step, see SFCTip and SFCTipmode.

Activation step

After calling the POU of the SFC, the actions of the initialization step (surrounded by a double-sided line) will be executed first. The step action being executed, the status is active, in the online mode, the active step is displayed in blue. All actions of the active step in a control loop will be executed. So, when the transition condition after the active step is TRUE, the step after it is active. The currently active step will be executed again in the next cycle.

<Note> If the active step contains an output action, such as the transition condition below it is TRUE, then it can only be executed during the next cycle.

IEC step

Standard IEC steps can be used in SFC.

In order to use the IEC step, the lesfc.lib library file must be linked in the project file.

An IEC step cannot be assigned more than nine actions. The IEC actions are not fixed as input or output actions to a certain step like simple steps, but are stored separately from the steps and

can be reused many times in a POU. Therefore, they must be associated with a single step with the command "Extras Associate action".

In addition to actions, Boolean variables can also be assigned to steps.

Active and inactive actions and boolean variables can be controlled using so-called qualifiers. There may be a time delay, and if an action is still active and the next step has already started processing, a concurrent process can be achieved with the qualifier S (set).

With each invocation of the sequential function block, the associated boolean variable is set or reset, that is, with each invocation, the value will change back and forth between TRUE and FALSE.

The associated actions of an IEC step are represented in two rectangles to the right of the step, the left area contains qualifiers, possibly with time constants, and the right area contains the action name and the respective Boolean variable name.

Qualifier

To associate actions with IEC steps, use the following qualifiers

N	non-storage	Actions and steps are activated together
R	reset	Action is inactive
S	set up	Action is activated and remains activated until reset
L	time limit	The action is activated for a period of time, the maximum value is the same as the step activation time
D	time delay	If the step is still active, the action is active after a certain amount of time, then it remains active as long as the step is active
P	pulse	If the step is active, the action is executed only once.
SD	storage and time delay	The action is activated after a certain time and remains activated until the next reset begins.
DS	delay and hold	The action is activated after a certain time as long as the step is still active and held until the next reset begins
SL	Hold and time limit	The action activates and remains for a period of time

The qualifiers L, D, SD, DS, and SL require a time value in TIME constant format.

<Note> When an action is deactivated, it will be executed again. This means that each action is executed at least twice.

Implicit Variables in SFC

Use some implicitly declared variables in SFC.

Each step has a marker that stores the state of the step. The step tag (active or inactive) is called <StepName>.x for IEC steps or <StepName> for a simple step. This Boolean variable has the value TRUE when the associated step is active, and FALSE otherwise. It can be used in every action and transition in the SFC module.

Whether an IEC step is active can be queried by querying <ActionName>.x.

The implicit variable <StepName>.t can be used to query the time of step activation.

Implicit variables can also be accessed by other programs, for example, `boolvar1:=sfc1.step1.x`; where `step1.x` is an implicit boolean variable, which represents the state of IEC step `step1` in `POUsfc1`.

SFC designator

Identifiers are used to control the operation of SFC POU's and are implicitly created during project operation. In order to be able to read these identifiers, appropriate global or local variables must be defined. For example, if a step in an SFC POU is active longer than its defined properties, then a flag is set, which can be accessed by using a "SFCError" variable (in which case SFCError gets true).

The following identifier variables can be defined:

SFCEnableLimit: The type of this variable is Boolean. When its value is TRUE, the timeout of this step will be registered in SFCError, and other timeouts will be ignored.

SFCInit: When the value of this Boolean variable is TRUE, the SFC is reset to its initial state, and other SFC identifiers are also reset. The initial step remains active until the variable value is TRUE, and execution begins. The module works correctly only when SFCInit is reset to FALSE.

SFCReset: This boolean variable is very similar to SFCInit, except that further processing takes place after the initialization of the initialization step, thus, for example, the SFCReset flag can be reset to FALSE during the initialization step.

SFCQuitError: When this boolean variable gets TRUE, the execution of the SFC will be stopped, so a possible timeout in the variable SFCError will be reset, when this variable is FALSE, all times in the active step will be reset, the precondition is that in the SFC SFCError is pre-defined in , and the flags for any timeout settings are registered.

SFCPause: When this boolean variable is TRUE, the SFC chart stops executing.

SFCError: This variable gets TRUE when there is a timeout in the SFC chart. If other timeouts occur after this one, these states will not be recorded unless this variable has been reset. If you want to use other time control flags (SFCErrorStep, SFCErrorPOU, SFCQuitError, SFCErrorAnalyzation) the precondition is to define SFCError.

SFCTrans: This boolean variable gets true when a transition is activated.

SFCErrorStep: This variable is a string variable. If a timeout is registered in SFCError, this variable will store the name of the timeout step. The precondition is that the flag SFCError to register any timeout has been defined in the SFC.

SFCErrorPOU: This string variable contains the name of the module where the timeout occurred. The precondition is that the flag SFCError to register any timeout has been defined in the SFC.

SFCCurrentStep: This string variable stores the name of the active step, it is not related to time monitoring. In the case of simulation, this step is stored in the branch of external authority. If a timeout occurs, the timeout will no longer be registered and SFCError will not be reset.

SFCErrorAnalyzationTable: This is an array variable `ARRAY[0..n]` that provides the analysis result of a conversion expression. For each element that has an effect, the conversion is registered as FALSE, and the previous step time-out record, so write the following information into the ExpressionResult structure, write: name, address, comment, current value.

A maximum of 16 elements can be held, so the range of the array is from (0~15).

The ExpressionResult structure and implicitly used analysis modules are provided by the AnalysisNew.lib library file, and the analysis modules can also be used explicitly by other POUs not written in SFC.

The timeout registration in the previous step is a prerequisite for parsing a conversion expression, so there must be a time watch applied here, and SFCErrors must be defined in the declaration window.

SFCTip, SFCTipMode: This boolean variable allows the progressive mode of the SFC. When used to toggle it with SFCTipMode=TRUE, it may only skip to the next step if SFCTip is set to TRUE, and it may skip the transition as long as SFCTipMode is set to FALSE.

Optional branch

Two or more optional branches can be defined in SFC. Each branch must start and end with a transition. Alternative branches can contain parallel branches and other alternative branches, an alternative branch starts at a horizontal line and ends at a horizontal line (choice end), or a jump.

If the step before the start line of the alternative branch is active, the first transition of each alternative branch will be calculated from left to right, the first transition will be from the left transition condition is TRUE, then the following steps will be active .

Parallel branch

Two or more branches can be defined as parallel branches in SFC. Each parallel branch must have one step at the beginning and end. Parallel branches can contain optional branches or other parallel branches. A parallel branch begins with a double dash and ends with a double dash or a jump, which provides a jump indicator.

The first step of the parallel branch is active if the previous step of the parallel branch is active and the transition condition value after this step is TRUE. These branches are processed in parallel with each other. When all parallel steps are activated and the transition condition after these steps is TRUE, then the step after the end line of the parallel split is activated.

A jump is a link to the step name indicated under the jump symbol. Jumps must be used when creating upward or cross-connections is not allowed.

6.2.4 Functional block diagram (FBD)

A function block diagram is a graphics-based programming language that works with a series of networks, each network containing a structure that provides arithmetic or logical expressions, call, jump, or return instructions for function blocks.

6.2.5 Continuous Function Chart (CFC)

The continuous function diagram editor does not operate like a function block diagram, but elements can be placed freely, which allows the use of feedback.

6.2.6 Ladder Diagram (LD)

Ladder diagram is also a graphics-based programming language, which is close to the structure of electronic circuits. On the one hand, ladder diagram is very suitable for building logic switches. On the other hand, it can also create network diagrams like FBD. Useful when controlling calls to other POU's.

The ladder diagram contains a series of networks. There is a vertical current line on the left and right sides. The network diagram is only limited to the range between the left and right bus bars. In the middle is a circuit diagram composed of coil contacts and connecting lines.

Each net contains a series of contacts on the left that communicate the on and off states from left to right according to the TRUE and FALSE values of Boolean variables. Each contact is a Boolean variable. If the variable value is TRUE, the circuit is connected from left to right through the connecting line. Otherwise the right side receives the value "off".

Contact

There are contacts on the left side of each network diagram in the ladder diagram (contacts are represented by two parallel lines | |), which are used to indicate the "on" and "off" states of the circuit.

These states correspond to Boolean variables TRUE and FALSE. Boolean variables belong to each contact. If the value of the variable is TRUE, then the state can be passed from left to right by connecting lines. Otherwise, "disconnect" is received on the right.

Contacts can be used in parallel, and one of the parallel branches must pass the "on" state before the parallel branch can pass "on". Or the contacts are connected in series. At this time, when the contacts must transmit the "on" state, the last contact transmits "on", which is consistent with the circuit connected in series and parallel.

Coil

On the right side of the ladder network diagram there are some so-called coils, which are represented by () and can only be connected by horizontal lines. The coil transfers the connection state from left to right, and copies the state into a boolean variable, which can describe the state of the entry wire as "on" (corresponding to TRUE for a boolean variable) or "off" state (corresponding to FALSE for a boolean variable).

Contacts and coils can also be negated (contact SWITCH1 and coil %QX3.0 are negated in the example above). If the contact is negated (represented by "/" in the contact symbol), then it copies the negated value to the corresponding boolean variable. If a contact is negated, the circuit can be connected only when the corresponding Boolean variable is set to FALSE.

Function Blocks in Ladder Diagram

Function blocks and programs can be added to the network diagram, but they must have Boolean-valued inputs and outputs and can be used on the left side of the ladder diagram like contacts.

Contact setting/resetting

Contacts can be defined as set/reset contacts. A set contact is denoted by an "S" in the contact symbol, it never overrides the TRUE value in the corresponding boolean variable, that is, if the variable is set to TRUE, it will remain in the same state.

The reset contact is denoted by "R", it never overwrites the FALSE value in the corresponding boolean variable, and if the variable has been set to FALSE, it will remain in that state.

When using the ladder diagram, the result of the contact switch can be used to control other POUs. On the one hand, the coil can be used to output the result to a global variable, which can be used elsewhere. It is also possible to insert possible calls directly in the Ladder diagram by introducing a POU with EN input.

These POUs are complete normal operands, functions, programs or function blocks. They all have an additional input designator EN. The EN input is a boolean variable, and a POU with an EN input will only be calculated if the value of EN is TRUE.

6.3 Online debugging function

Sampling Tracking

Sampling tracing allows tracing the continuously changing value of a variable, relying on a so-called trigger event, which is a rising or falling edge of a previously defined Boolean variable (trigger variable). CODESYS allows tracking of 20 variables, each of which can track 500 values.

Debug

The debugging capabilities of CODESYS make errors easy to find. To debug, run the command "Project" "Options" and select the activated option in the automatically popped up dialog for creating options.

Breakpoint

A breakpoint is a place where program processing stops so that it can observe changes in the value of a variable at a specific point in the program.

The breakpoint can be set in the editor, in the text editor the breakpoint is set at the line number, in the continuous function diagram and the ladder diagram it is set at the network number, in the CFC it is set at the POU, in the SFC It is set at the step, and the breakpoint cannot be set in the instance of the function block diagram.

Single step

A single step is:

In the instruction list: Execute the program until the CAL LD and JMP commands are run.

In structured text: execute the next instruction.

In the function block diagram ladder diagram: Execute the next step network.

In SFC: Continue the current action until the next step begins.

Logic errors in the program can be checked by running it step by step.

Single cycle

If single loop is selected, execution ends when each loop ends.

Change value in online mode

During operation, variables can be set to a specific value (write value), or redefined to a specific value (force value) after each loop. In online mode, the value of a variable can be changed by double-clicking on its value. Boolean variables can be changed from TRUE to FALSE, or from FALSE to TRUE. The Write Variable xy dialog box opens for each type of variable, where the actual value of the variable can be edited.

Monitor

In online mode, all variables can be read from the controller and displayed in real time. These displays can be found in the Definition and Program editors. The current values of variables can also be read out in watchers and receivers, and they can be seen. If you want to monitor variables in an instance of a function block, the corresponding instance block must already be open.

Simulation

During the simulation process, the created PLC program does not run in the actual PLC, but runs in the calculator in the CODESYS system. All online functions are available. It allows checking the correctness of logic without PLC hardware.

<Note> POU's of external library files cannot be run in emulation mode.

Log

The log records user operations, internal processes, state transitions, and unexpected situations that occur during online mode processing. It is used to monitor and track errors.

6.4 Standardization

IEC61131-3 is an internationally standardized PLC programming language. The programmable languages offered in CODESYS conform to standardization requirements. According to this standard, a program contains the following elements:

- Structure
- POU
- Global variables

Elements of the common language are described in the sections Identifiers, Addresses, Types, Comments, and Constants. The processing of the CODESYS program starts with a special POU (POU) PLC_PRG, which can call other POU's.

7 User's Guidance

7.1 Application example of communication method

C06/C07 motion controller supports bus, CANopen, EtherCAT, EtherNET and other communication methods. This section will introduce the communication methods supported by LC-C06 through specific examples.

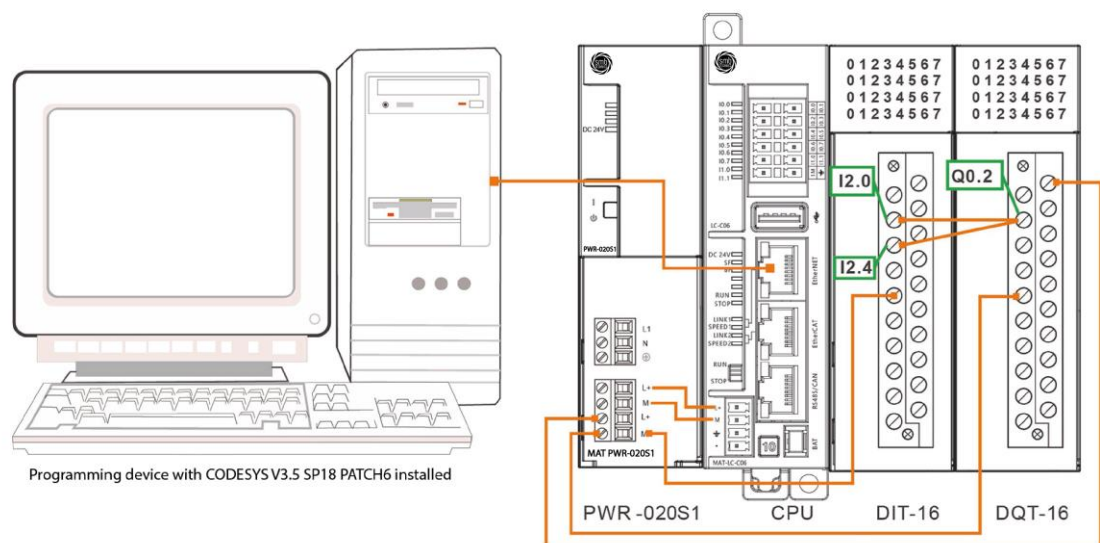
7.1.1 Bus Communication

1、Preparation before communication

Table 7-1 Example Components for Bus Communication

Components	Function
Programming device PG\PC	Programming device with CODESYS V3.5 SP18 PATCH6 installed to configure, program and debug the LC-C06
Assembly rail	LC-C06 rack, used to hold the modules in the system
Power Module PWR-020S1	Powers the LC-C06 motion controller and its 24 VDC load circuit
LC-C06 main control module	The CPU executes the user program, provides 5V voltage to the LC-C06 backplane bus, and communicates with other modules through the Ethernet interface
Expansion module SM	The SM converts the different process signal levels to match the LC-C06. In this example, the digital input module DIT-16 and the digital output module DQT-16 are used.
Standard network cable	Connecting a programming device to the LC-C06

2、Internet connection



Implement function

The output point Q0.2 is lit, and the input points I2.0 and I2.4 are lit at the same time.

3、Steps

Step 1: Wiring

Open the front panels of the LC-C06, power supply module, digital input (DIT-16), and output module (DQT-16) and wire them by referring to the communication connections in this section. The specific operations are as follows:

- 1) Use the programming cable to connect the PC and LC-C06
- 2) DIT-16 is connected to LC-C06 by bus
- 3) DQT-16 is connected to DIQ-16 by bus

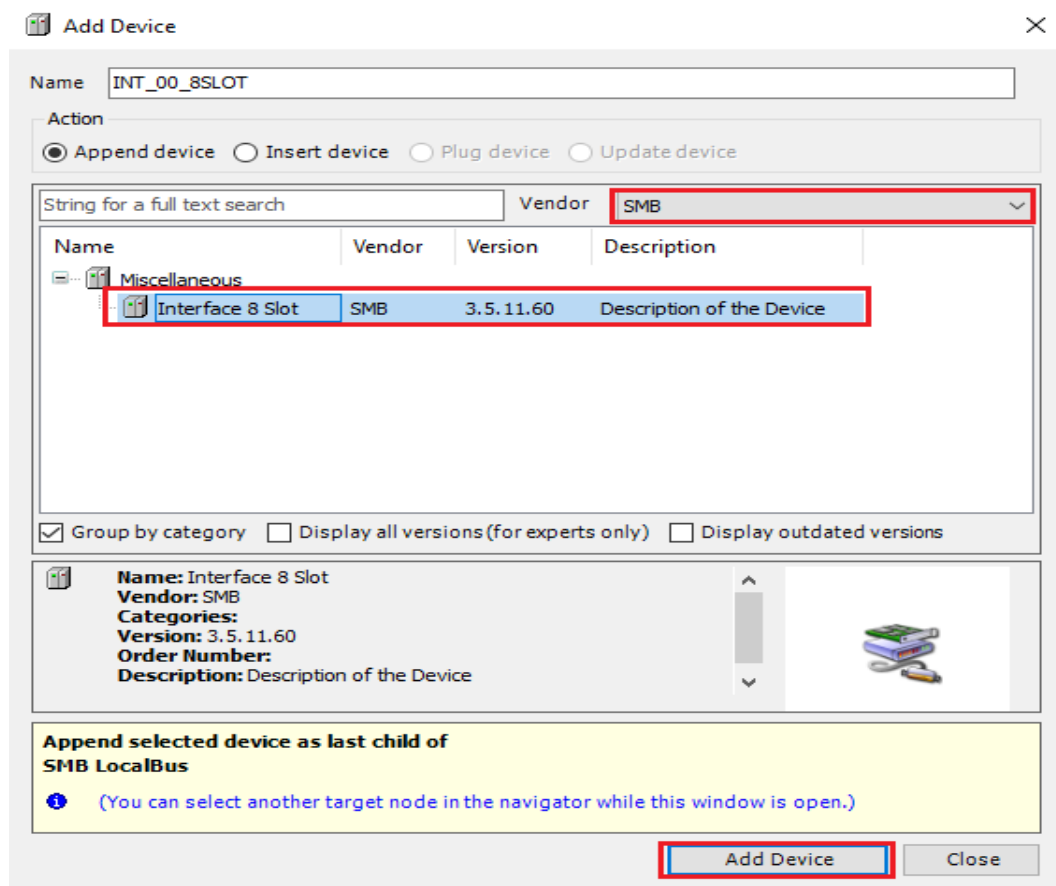


WARN

Do not touch the power cable after the power module is powered on or after connecting the power cable to the main power supply, and any wiring work must be carried out under the condition of power off!

Step 2: Configuration in CODESYS

- 1) Select "SMB LocalBus" in the device view and right-click to select "Add Device", you can select and add a relay module in the pop-up dialog box, refer to the following diagram:



- 2) Referring to the above operations, right-click the relay module in the device view and select "Add Device", then in the "Add Device Dialog Box" you can select to add extension modules

DIT-16 and DQT-16. Or right-click the relay module and select "Scan Device", and the expansion modules connected to the LC-C06 will be displayed under the relay module.

Step 3. Refer to the chapter "[2.3 Set up Communication](#)" to set the communication between LC-C06 and the host computer

Step 4. Running, debugging and monitoring

1) Select the menu item "Online" → "Login to..." to establish a connection between the application and the LC-C06 and enter the online state. Then, select the menu item "Debug" → "Start" to make the application program in LC-C06 start to run, at this moment, the current project can be monitored and debugged.

2) Write 1 to %QX0.2 in the Internal I/O mapping table of the CODESYS software, then the output point Q0.2 of the DQT-16 module will be lit. At the same time, the input points I2.0 and I2.4 of the DIT-16 module are lit, and the values of %IX2.0 and %IX2.4 are changed to 1 in the Internal I/O mapping table of the DIT-16 module.

7.1.2 Modbus RTU Communicaiton

In this instruction, you will be guided to creat an example application to get familiar with Modbus RTU function of C06/C07.

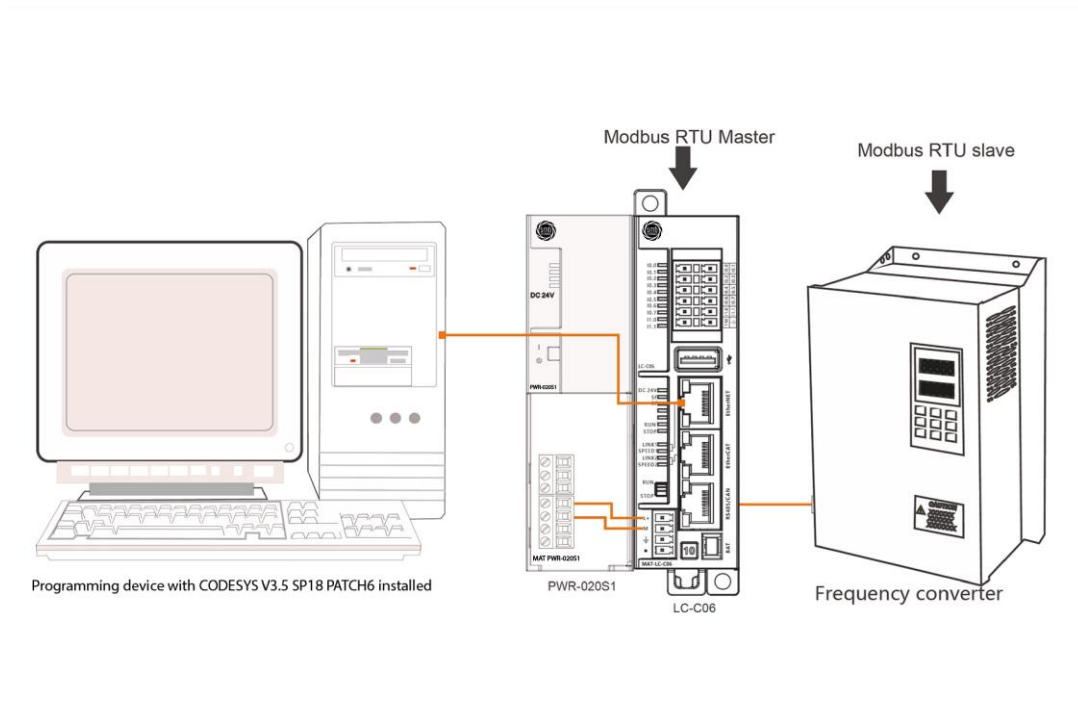
1. Preparation

Table 7-2 Example components of Modbus RTU communicaiton

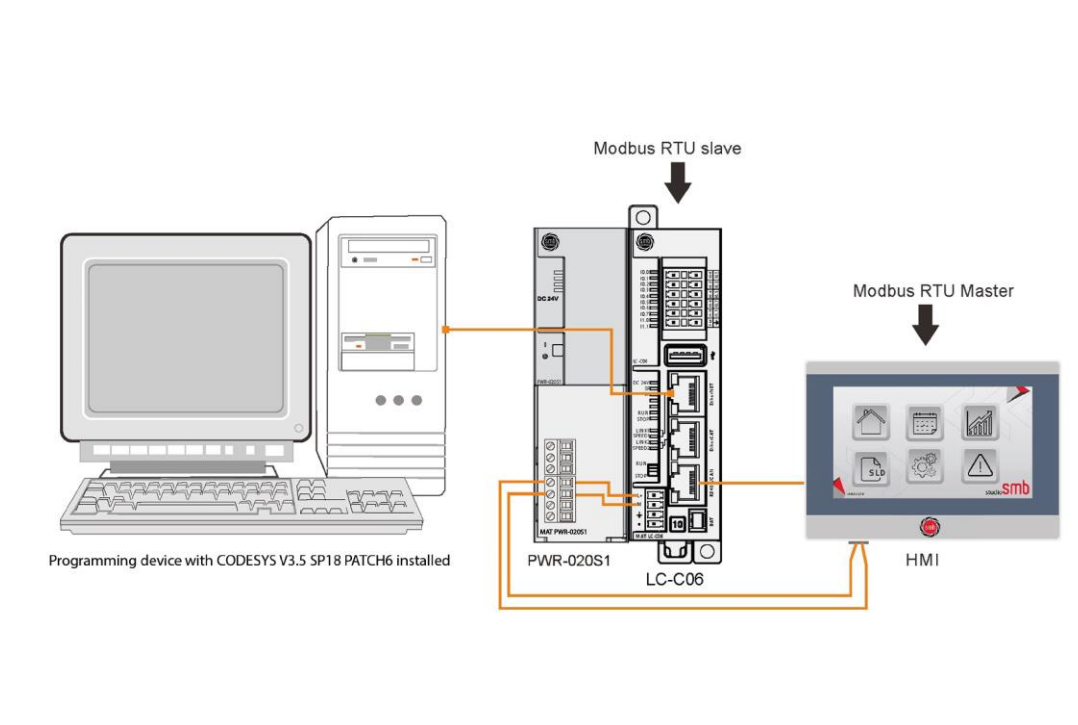
Component	Function
Programming device PG\PC	Install programming device with CODESYS V3.5SP18 Patch6, to configurate, program and debug LC-C06 motion controller.
Power module: PWR-020S1	Supply 24VDC power for LC-C06 and HMI.
Master module: LC-C06	CPU executes user program, supplies 5V voltage to bus of LC-C06 system and communicates with other modules via Ethernet interface.
SMB-MAT-HMI	SMB-MAT-HMI serves as Modbus master and communicates with LC-C06.
Inverter	Inverter, which supports Modbus protocol, serves as slave of Modbus RTU and communicates with LC-C06.
Standard network cable	Connects LC-C06 to programming device.

2. Network connection

◆ LC-C06 serves as master of Modbus RTU and communicates with inverter.



◆ LC-C06 serves as slave of Modbus RTU and communicates with SMB-MAT-HMI.



3. Operating steps

Step 1: Wiring for LC-C06, power module, HMI and inverter.

Refer to above diagram for wiring of LC-C06, power module PWR-020S1, SMB-MAT-HMI and inverter.

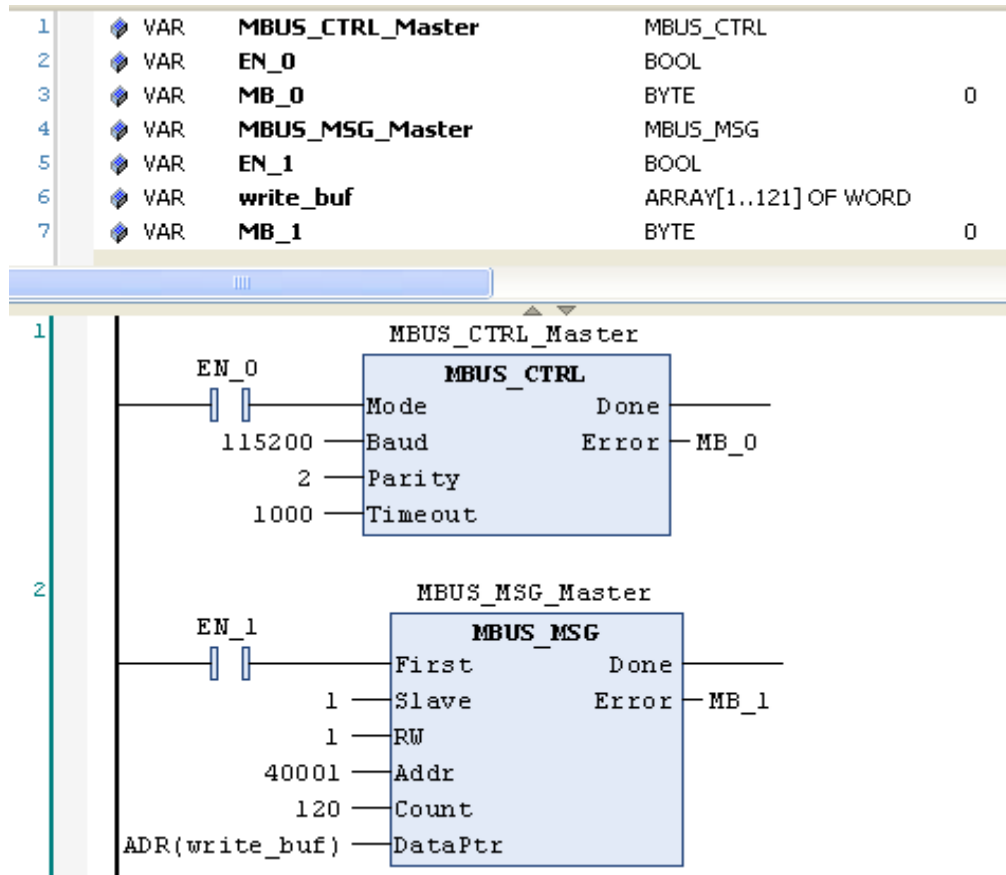
Step 2: Cable connecting

- 1) Connect PC and LC-C06 with standard network cable.

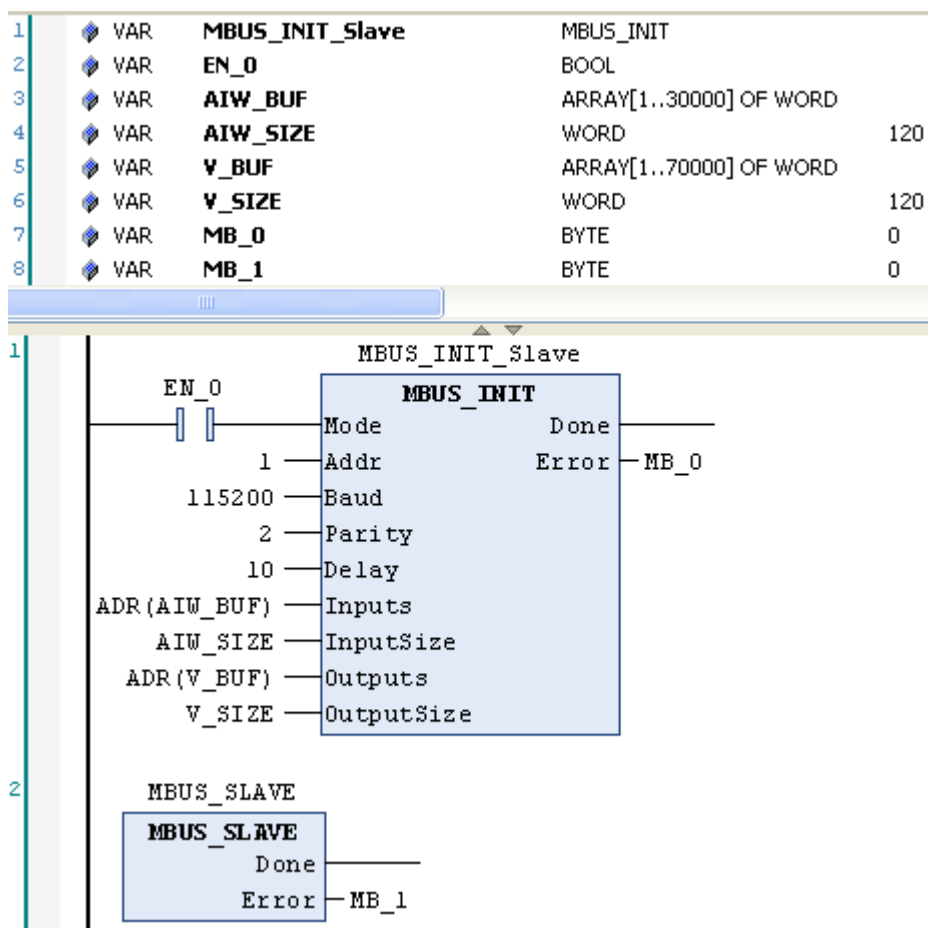
2) When LC-C06 serves as master of Modbus RTU and communicates with inverter, connect RS485 port of LC-C06 to Modbus port of inverter with communication cable.

3) When LC-C06 serves as slave of Modbus RTU and communicates with SMB-MAT-HMI, connect RS485 port of LC-C06 to Modbus port of SMB-MAT-HMI with communication cable.

Step 3: When LC-C06 serves as master of Modbus RTU and communicates with inverter, program LC-C06 as below:



Step 4: When LC-C06 serves as slave of Modbus RTU and communicates with SMB-MAT-HMI, program the slave (LC-C06) as below:



7.1.3 Modbus TCP Communication

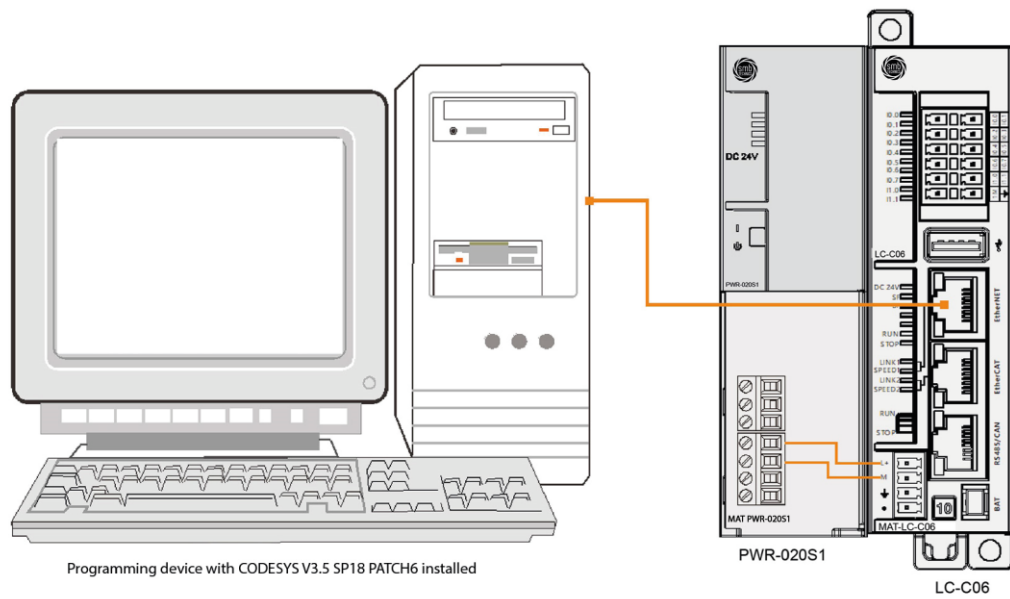
In this chapter, one example will be used to display Modbus TCP communication function of LC-C06 motion controller.

1. Preparation

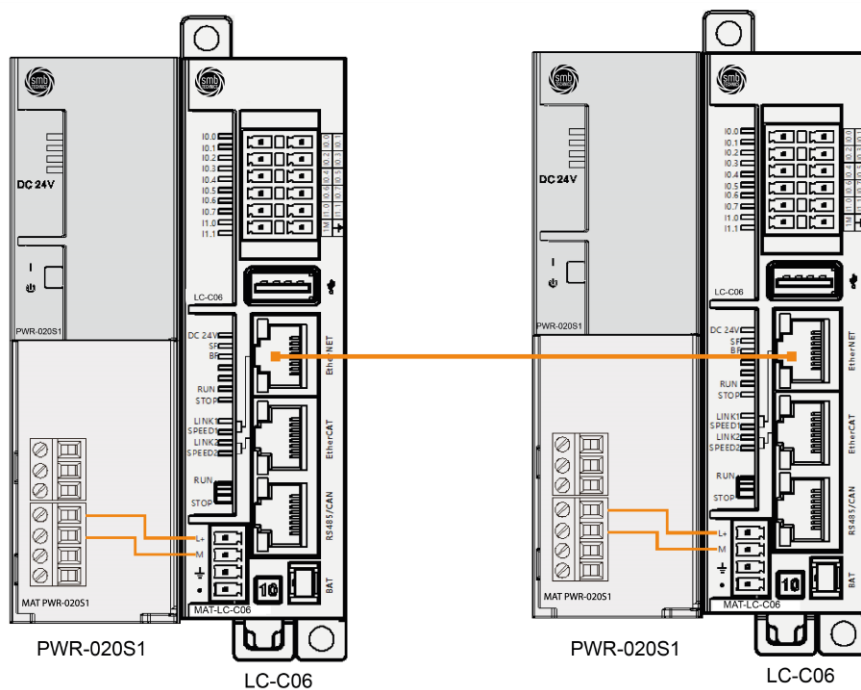
Table 7-3 Example components of Modbus TCP

Component	Function
Programming device PG\PC	Install programming device with CODSYS V3.5SP18 Patch6, to configurate, program and debug LC-C06 PLC.
Power module PWR-020S1	Supply power for LC-C06 motion controller and 24 VDC load circuit.
Master module LC-C06	CPU executes user program, supplies 5V voltage to bus of LC-C06 system and communicates with other modules via Ethernet interface.
Standard network cable	Connects LC-C06 to programming device.

2. Network connection



Connect two LC-C06, which serve as master and slave of Modbus TCP, with standard network cable and then start Modbus TCP communication:



3. Operating steps

Step 1: Wiring between LC-C06 and power module

Open front panels of LC-C06 and power module PWR-020S1, wiring according to above network connection diagram.

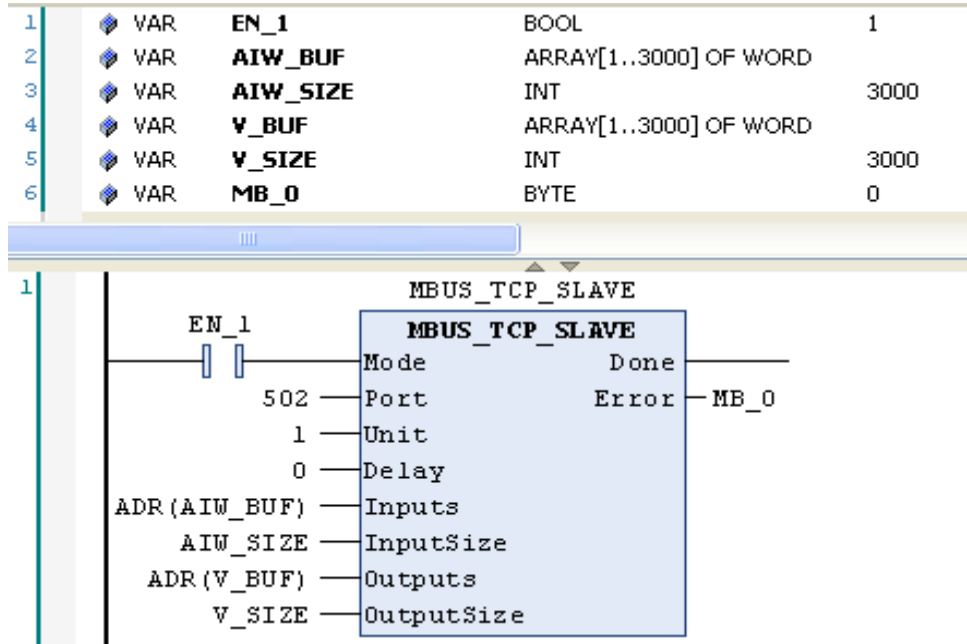
Step 2: Device connection

Connect program device and Modbus TCP with standard network cable.

Step 3: Program for LC-C06, which serves as slave of Modbus TCP.

- 1) Input order on shell of PLC, like "setip 192.168.0.1" and push ENTER to set IP address.
- 2) Program for slave of Modbus TCP in PLC_PRG(PRG).

Below are "Modbus_TCP_Slave" library instructions in "SMB_Modbus_V1.1.library" library file:

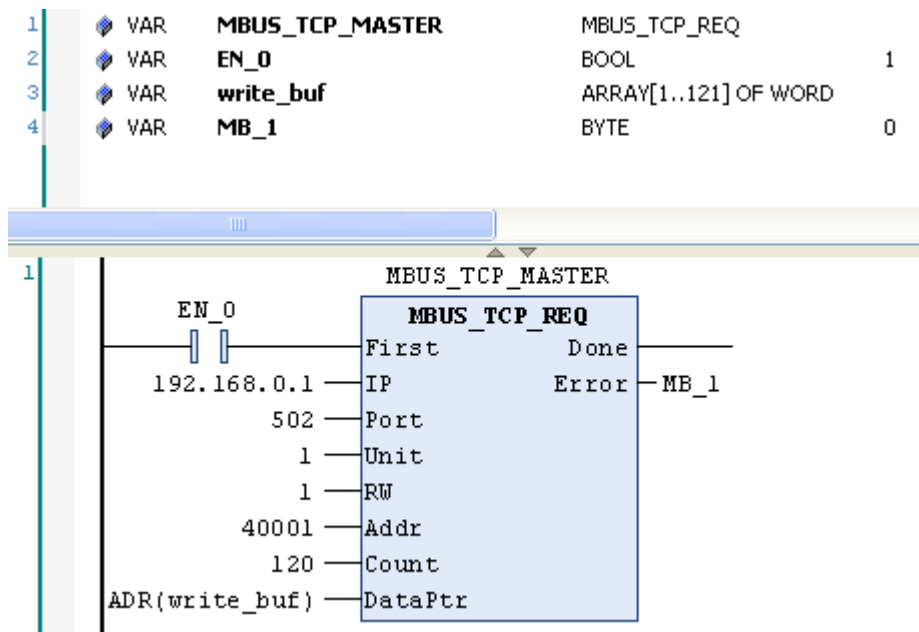


- 3) Compile and download program to Modbus TCP slave device.

Step 4: Program for LC-C06, which serves as master of Modbus TCP.

- 1) Program for Modbus TCP master in PLC_PRG(PRG).

Below are "Modbus_TCP_REQ" library instructions in "SMB_Modbus_V1.1.library" library file:



Step 5: Interconnection communication between master and slave of Modbus TCP

- 1) Connect power to master and slave of Modbus TCP
- 2) Connect master and slave of Modbus TCP with standard network cable to start interconnection communication of Modbus TCP.

7.1.4 CANopen Communication

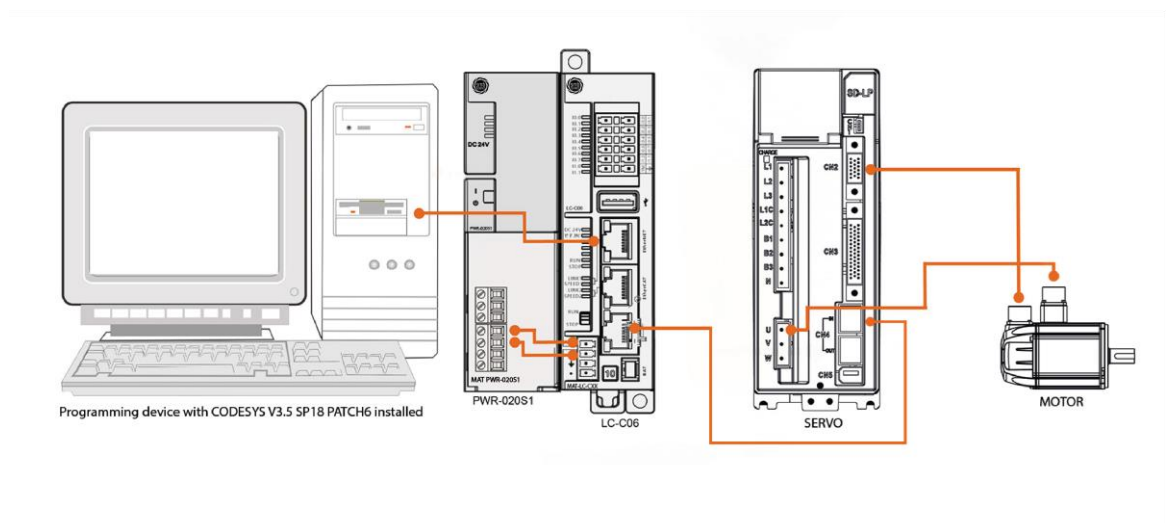
Single-axis motion control based on CANopen communication

1. Preparation

Table 7-5 Example components of CANopen

Component	Function
Install programming device with CODESYS V3.5 SP18 Patch6	To configure, program and debug C06/C07 series motion controller
Install guide rail	To fix modules of the system.
Power module PWR-02	To supply power for C06/C07 motion controller and its 24 VDC load circuit.
External power supply	To supply power for SMB_SD_LP_C servo driver.
LC-C06 motion controller	Executes user program, supplies 5V voltage to bus of LC-C06 system and communicates with other modules via Ethernet interface.
SMB servo driver and motor	In example: SMB servo driver and matching motor are used.
Standard network cable	Connect programming device with LC-C06.
Encoder cable	Connect servo driver and motor.

2. Network connection



3. Operating steps

Step 1. Connect to power supply for each device.

- 1) Open front panels of LC-C06, power module PWR-020S1 and refer to above diagram for connecting LC-C06 to PWR-020S1.
- 2) Connect SMB servo driver to main power and control power supply.

Step 2. Connect each device with cable.

Network connection details as below:

- 1) Connect PC and LC-C06 with standard network cable.
- 2) Connect LC-C06 and SMB servo driver with standard network cable.
- 3) Connect SMB servo driver and motor with encoder cable.

Step 3. Configure CANopen in CODESYS

- 1) Add CANbus

In devices view, right click "Device (MAT-LC-C06_V1.0)" and select "Add device", then select to add CANbus.

- 2) Add CAN master

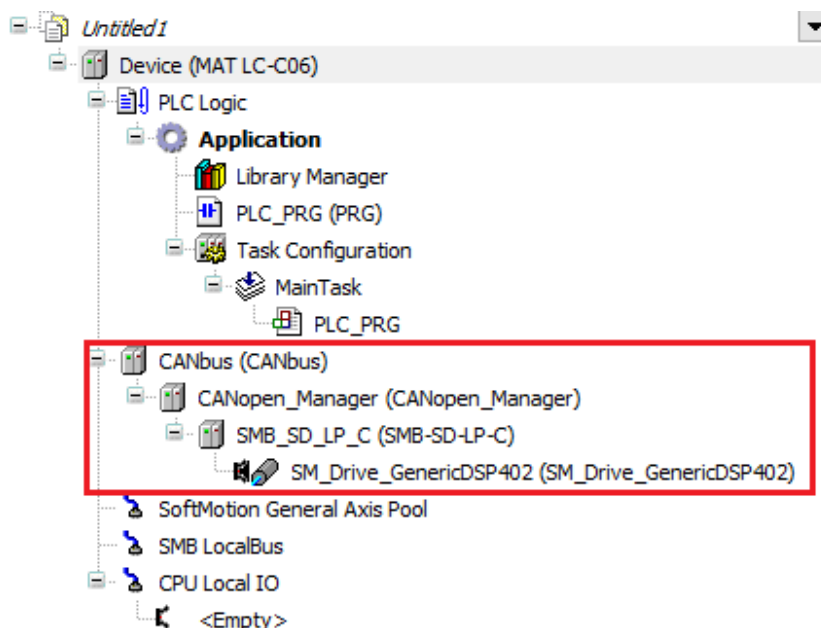
In devices view, right click "CANbus" to select "Add device" to add CAN master in dialogue of "Add device": field bus→CANopen→CANopen manager→CANopen_Manager.

- 3) Add or scan CAN slave

In devices view, right click "CANopen_Manager" and select "Add device" to add CAN slave in dialogue of "Add device": field bus→CANopen→remote device→SMB; or right click "CANopen_Manager" to select "Scan device" to display slave device (SMB) under CAN master.

- 4) Add drive of CAN slave (SMB)

If SMB is added as CAN slave in step 3, add its drive here: in devices view, right click CAN slave (SMB) and select "Add SoftMotion CiA 402 Axis" to add the drive.



5) CANbus configuration

General

Log

CANbus IEC Objects

Status

Information

General

Network 0

Baud rate (kbit/s) 1000000

6) CANOpen master configuration

Input node ID of CAN master in “Overview” of master: 127

7) CANOpen slave (SMB) configuration

First, in “Overview” tab, set node ID of slave (address of slave SMB) and then tick “Enable expert settings”.

General

PDOs

SDOs

Log

General

Node-ID 1

SDO Channels (1/1 Active)

☒ Enable expert settings

☐ Optional device

☐ Enable SYNC producing

☐ No initialization

☐ Reset node

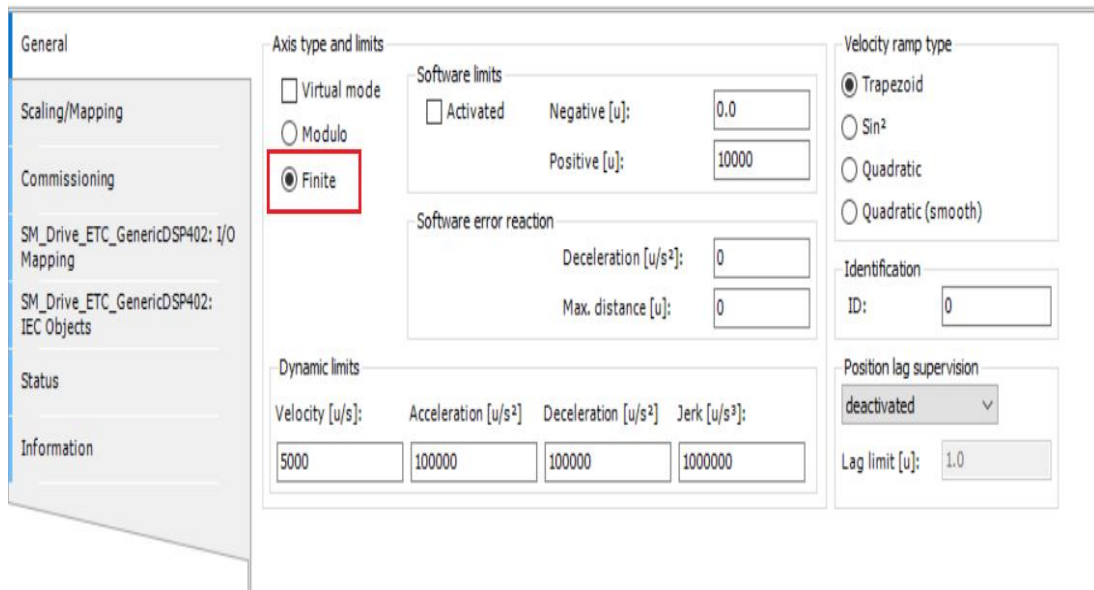
Then, select needed parameters in “PDOs” tab, setting of the example is as below:

General	Receive PDOs (Master => Slave)	Transmit PDOs (Slave => Master)																																																																														
PDOs	+ Add PDO + Add Mapping ✎ Edit ✕ Delete ↑ Move Up ↓ Move Down <table> <thead> <tr> <th>Name</th><th>Object</th><th>Bit length</th></tr> </thead> <tbody> <tr> <td>☒ 16#1400: Receive PDO Communicati</td><td>16#201 (\$NODEID+16</td><td>24</td></tr> <tr> <td>Controlword</td><td>16#6040sub00</td><td>16</td></tr> <tr> <td>Modes of operation</td><td>16#6060sub00</td><td>8</td></tr> <tr> <td>☒ 16#1401: Receive PDO Communicati</td><td>16#301 (\$NODEID+16</td><td>64</td></tr> <tr> <td>Target Position</td><td>16#607Asub00</td><td>32</td></tr> <tr> <td>Target velocity</td><td>16#60FFsub00</td><td>32</td></tr> <tr> <td>☒ 16#1402: Receive PDO Communicati</td><td>16#401 (\$NODEID+16</td><td>48</td></tr> <tr> <td>Target torque</td><td>16#6071sub00</td><td>16</td></tr> <tr> <td>Profile velocity</td><td>16#6081sub00</td><td>32</td></tr> <tr> <td>☒ 16#1403: Receive PDO Communicati</td><td>16#501 (\$NODEID+16</td><td>64</td></tr> <tr> <td>Profile acceleration</td><td>16#6083sub00</td><td>32</td></tr> <tr> <td>Profile deceleration</td><td>16#6084sub00</td><td>32</td></tr> </tbody> </table>	Name	Object	Bit length	☒ 16#1400: Receive PDO Communicati	16#201 (\$NODEID+16	24	Controlword	16#6040sub00	16	Modes of operation	16#6060sub00	8	☒ 16#1401: Receive PDO Communicati	16#301 (\$NODEID+16	64	Target Position	16#607Asub00	32	Target velocity	16#60FFsub00	32	☒ 16#1402: Receive PDO Communicati	16#401 (\$NODEID+16	48	Target torque	16#6071sub00	16	Profile velocity	16#6081sub00	32	☒ 16#1403: Receive PDO Communicati	16#501 (\$NODEID+16	64	Profile acceleration	16#6083sub00	32	Profile deceleration	16#6084sub00	32	+ Add PDO + Add Mapping ✎ Edit ✕ Delete ↑ Move Up ↓ Move Down <table> <thead> <tr> <th>Name</th><th>Object</th><th>Bit length</th></tr> </thead> <tbody> <tr> <td>☒ 16#1800: Transmit PDO Commun</td><td>16#181 (\$NODEID+</td><td>24</td></tr> <tr> <td>Statusword</td><td>16#6041sub00</td><td>16</td></tr> <tr> <td>Modes of operation display</td><td>16#6061sub00</td><td>8</td></tr> <tr> <td>☒ 16#1801: Transmit PDO Commun</td><td>16#281 (\$NODEID+</td><td>64</td></tr> <tr> <td>Position actual value</td><td>16#6064sub00</td><td>32</td></tr> <tr> <td>Velocity actual value</td><td>16#606Csub00</td><td>32</td></tr> <tr> <td>☒ 16#1802: Transmit PDO Commun</td><td>16#381 (\$NODEID+</td><td>64</td></tr> <tr> <td>Position demand value</td><td>16#6062sub00</td><td>32</td></tr> <tr> <td>Position actual increment value</td><td>16#6063sub00</td><td>32</td></tr> <tr> <td>☒ 16#1803: Transmit PDO Commun</td><td>16#481 (\$NODEID+</td><td>48</td></tr> <tr> <td>Following Error Actual Value</td><td>16#60F4sub00</td><td>32</td></tr> <tr> <td>Torque actual value</td><td>16#6077sub00</td><td>16</td></tr> </tbody> </table>	Name	Object	Bit length	☒ 16#1800: Transmit PDO Commun	16#181 (\$NODEID+	24	Statusword	16#6041sub00	16	Modes of operation display	16#6061sub00	8	☒ 16#1801: Transmit PDO Commun	16#281 (\$NODEID+	64	Position actual value	16#6064sub00	32	Velocity actual value	16#606Csub00	32	☒ 16#1802: Transmit PDO Commun	16#381 (\$NODEID+	64	Position demand value	16#6062sub00	32	Position actual increment value	16#6063sub00	32	☒ 16#1803: Transmit PDO Commun	16#481 (\$NODEID+	48	Following Error Actual Value	16#60F4sub00	32	Torque actual value	16#6077sub00	16
Name	Object	Bit length																																																																														
☒ 16#1400: Receive PDO Communicati	16#201 (\$NODEID+16	24																																																																														
Controlword	16#6040sub00	16																																																																														
Modes of operation	16#6060sub00	8																																																																														
☒ 16#1401: Receive PDO Communicati	16#301 (\$NODEID+16	64																																																																														
Target Position	16#607Asub00	32																																																																														
Target velocity	16#60FFsub00	32																																																																														
☒ 16#1402: Receive PDO Communicati	16#401 (\$NODEID+16	48																																																																														
Target torque	16#6071sub00	16																																																																														
Profile velocity	16#6081sub00	32																																																																														
☒ 16#1403: Receive PDO Communicati	16#501 (\$NODEID+16	64																																																																														
Profile acceleration	16#6083sub00	32																																																																														
Profile deceleration	16#6084sub00	32																																																																														
Name	Object	Bit length																																																																														
☒ 16#1800: Transmit PDO Commun	16#181 (\$NODEID+	24																																																																														
Statusword	16#6041sub00	16																																																																														
Modes of operation display	16#6061sub00	8																																																																														
☒ 16#1801: Transmit PDO Commun	16#281 (\$NODEID+	64																																																																														
Position actual value	16#6064sub00	32																																																																														
Velocity actual value	16#606Csub00	32																																																																														
☒ 16#1802: Transmit PDO Commun	16#381 (\$NODEID+	64																																																																														
Position demand value	16#6062sub00	32																																																																														
Position actual increment value	16#6063sub00	32																																																																														
☒ 16#1803: Transmit PDO Commun	16#481 (\$NODEID+	48																																																																														
Following Error Actual Value	16#60F4sub00	32																																																																														
Torque actual value	16#6077sub00	16																																																																														

In “CANOpen I/O mapping” tab, tick “Always in bus cycle task”.

8) SMB drive configuration (SM_Driver_GenericCiA402)

”Softmotion drive: basic” of SM_Driver_GenericCiA402 configuration as below:

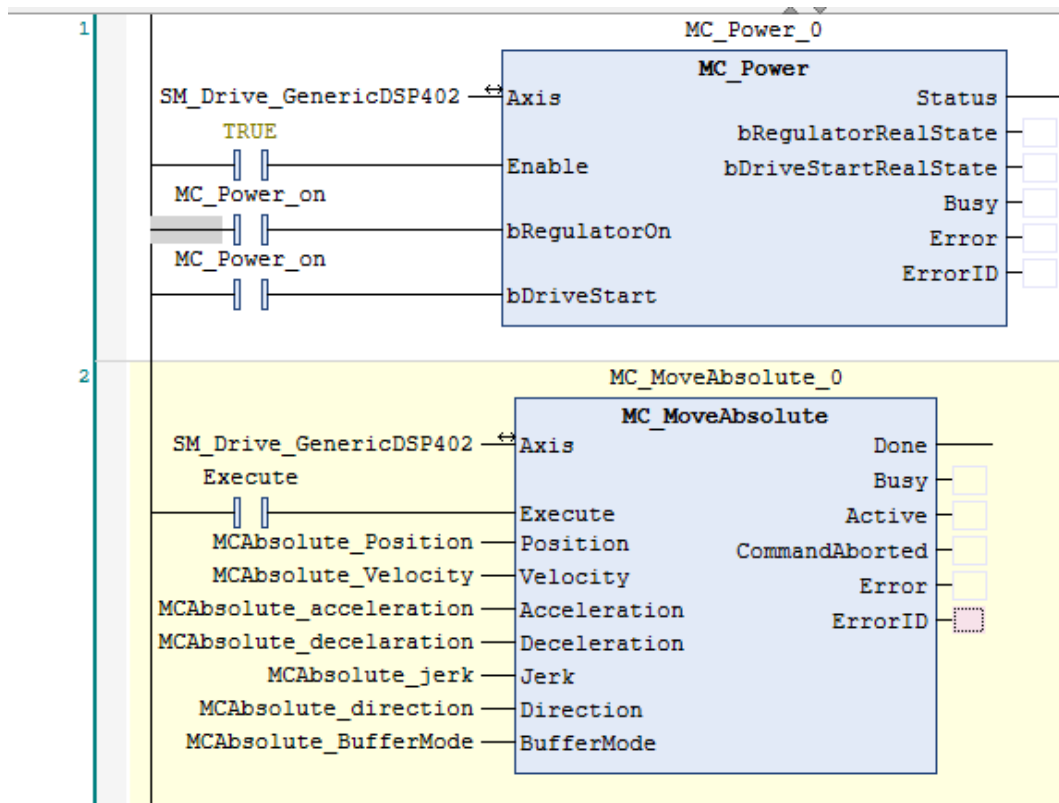


Set axis type and limit as “Finite”.

Step 4. Refer to chapter “2.3 Communication setup” for more information about communication setting between LC-C06 and IPC.

Step 5. Run and debug

- 1) Set the slave address on the front panel of the SMB servo driver: P4:00= 2 ,3 ,4.... etc., so that the slave addresses of all CanOpen communication drivers are different.
- 2) Then, control your motion with the CanOpen driver using Motion blocks on the Codesys platform via LC-C06.



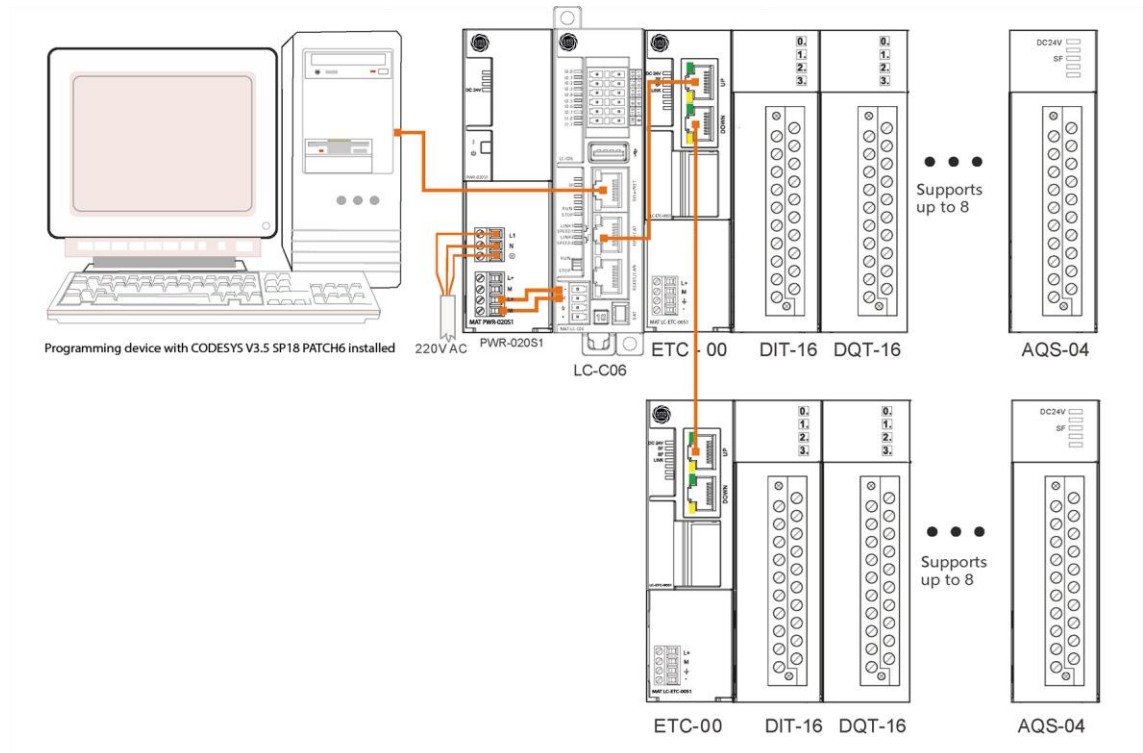
EtherCAT communication of LC-C06 will be displayed through the example in this chapter.

1. Preparation

Table 7-6 Example components of EtherCAT communication

Component	Function
Programming device PG\PC	Install programming device with CODSYS V3.5 SP18 Patch6, to configure, program and debug motion controller LC-C06.
Install guide rail	To fix modules of the system.
Power module PWR-020S1	Supply power for LC-C06 motion controller and 24 VDC load circuit.
LC-C06 master module	CPU executes user program, supplies 5V voltage to bus of LC-C06 system and communicates with other modules via Ethernet interface.
EtherCAT master module	1 pc of MAT-LC-CXX series CPU, LC-C0X
EtherCAT slave devices	2 ETC-00 modules
Extension I/O module	16 MAT-LC-CXX series digital and analog I/O modules, 8 modules should be attached to each EtherCAT slave module.
3 Standard network cables	●Connect LC-C06 and programming device. ●Connect EtherCAT communication port of LC-C06 to first IN port of first ETC-00. ●Connect OUT port of first ETC-00 to IN port of second ETC-00.

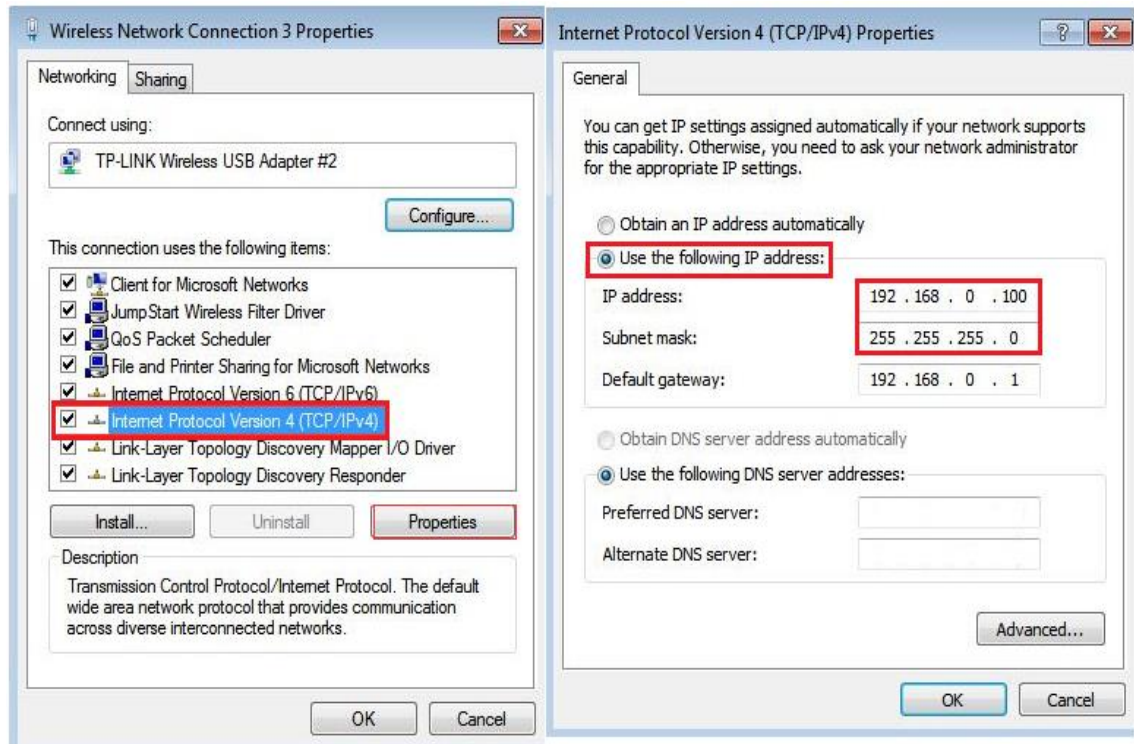
2. Network connection



Refer to above network connection diagram for system wiring.

3. Communication setup

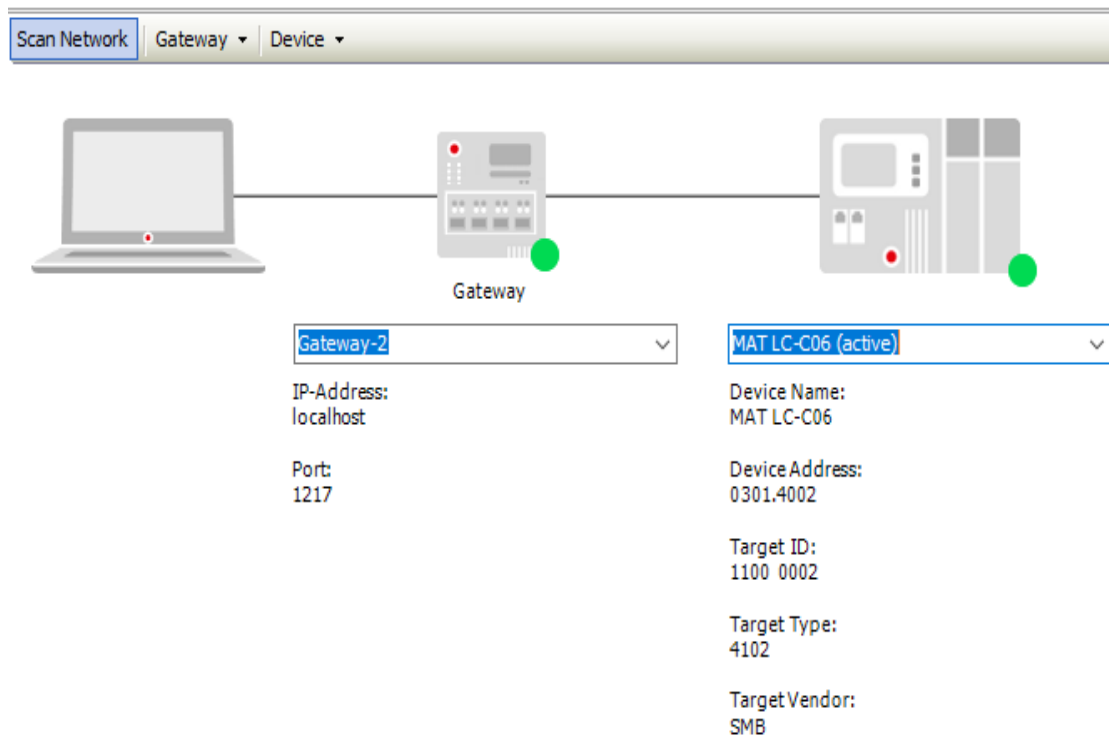
Before communication setting, set IP of programming device PG/PC and LC-C06 (IP: 192.168.0.X) in the same network segment: in local connection property of PC, double click TCP/IP protocol, change “Get IP address automatically” to “Use the following IP address”, then input the IP address “192.168.0.X”. Shown as below pic:



4. CPU setup

In devices view of Codesys, double click "device (MAT-LC-C06_V1.0)" to open the device dialog box, as shown in below figure. Click "add gateway" button in the "gateway" drop-down menu in the "communication setup" tab, enter "name", "drive" in the pop-up "gateway" dialog box, select "TCP / IP", "IP address" and "localhost", and finally click "OK" to close the dialog box. LC-C06 will be added to the communication dialog box.

Click “Scan network” after LC-C06 is added and find available devices in local network. If succeed, select the device and click “Set active path” button to activate setting of communication channel, which is related to all communication operations.



<Remark> When the system is started, CODESYS related service programs (such as ,  etc.) will appear at the system tray. If no special requirements, there is no need to operate the service programs in the tray.

5. Operating steps

Step 1. Connect to power supply for each device.

- 1) Open front panels of LC-C06, power module PWR-020S1 and refer to above diagram for connecting LC-C06 to PWR-020S1.
- 2) Connect the main system to main power and control power supply.

Step 2. Connect each device with cable.

Network connection details as below:

- 1) Connect PC and EtherNET port of LC-C06 with standard network cable;
- 2) Connect EtherCAT OUT port of LC-C06 to IN port of first EtherCAT slave with standard network cable;
- 3) Connect OUT port of first EtherCAT slave to IN port of second EtherCAT slave with standard network cable.

Step 2.1 Connect SMB device with cable.

Network connection details as below:

- 1) Connect PC and EtherNET port of LC-C06 with standard network cable;
- 2) Connect EtherCAT IN port of LC-C06 to OUT port of first EtherCAT slave with standard network cable;
- 3) Connect IN port of first EtherCAT slave to OUT port of second EtherCAT slave with standard network cable.

Step 3. Configure EtherCAT in CODESYS.

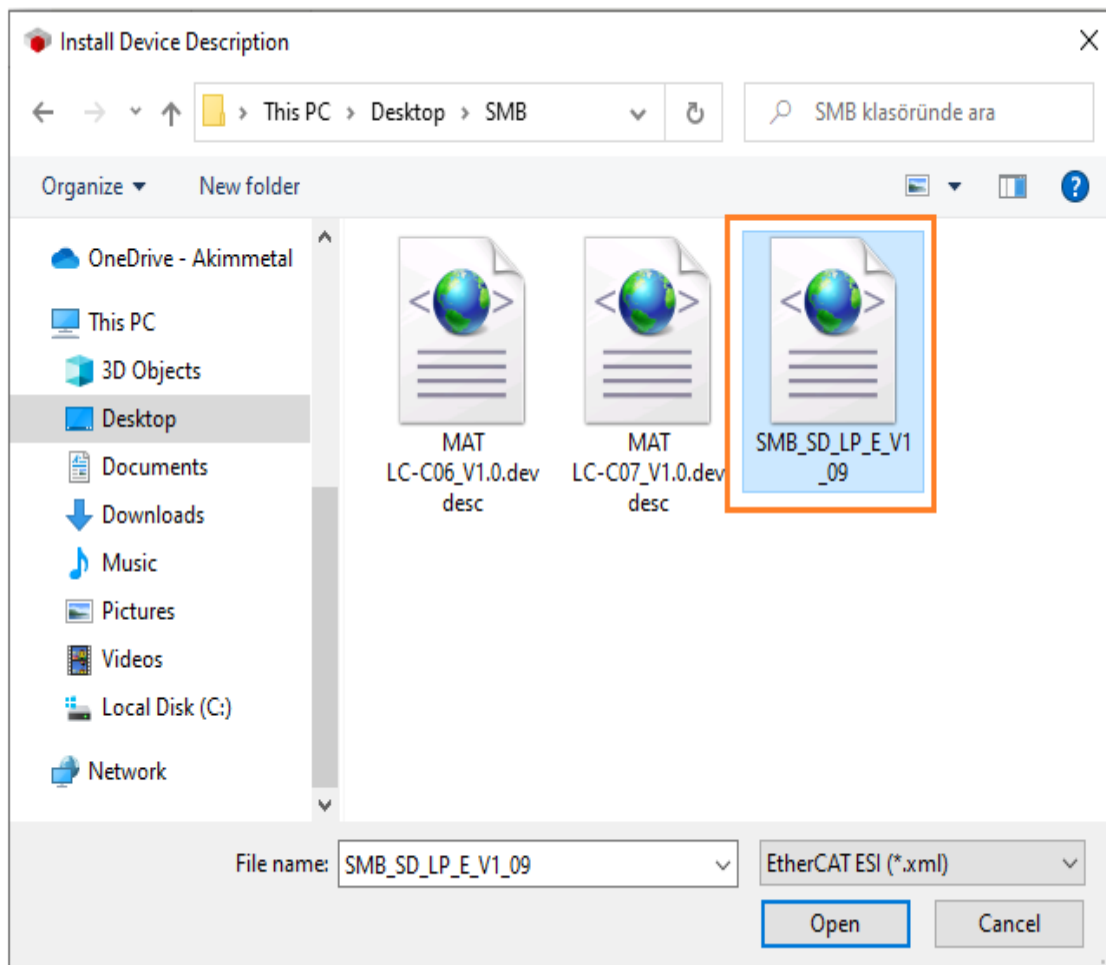
Double click desktop icon  or start CODESYS in menu of START, then operate as below:

1) Install device description file

Install device description files of CPU LC-C06 and EtherCAT slave in example componets.

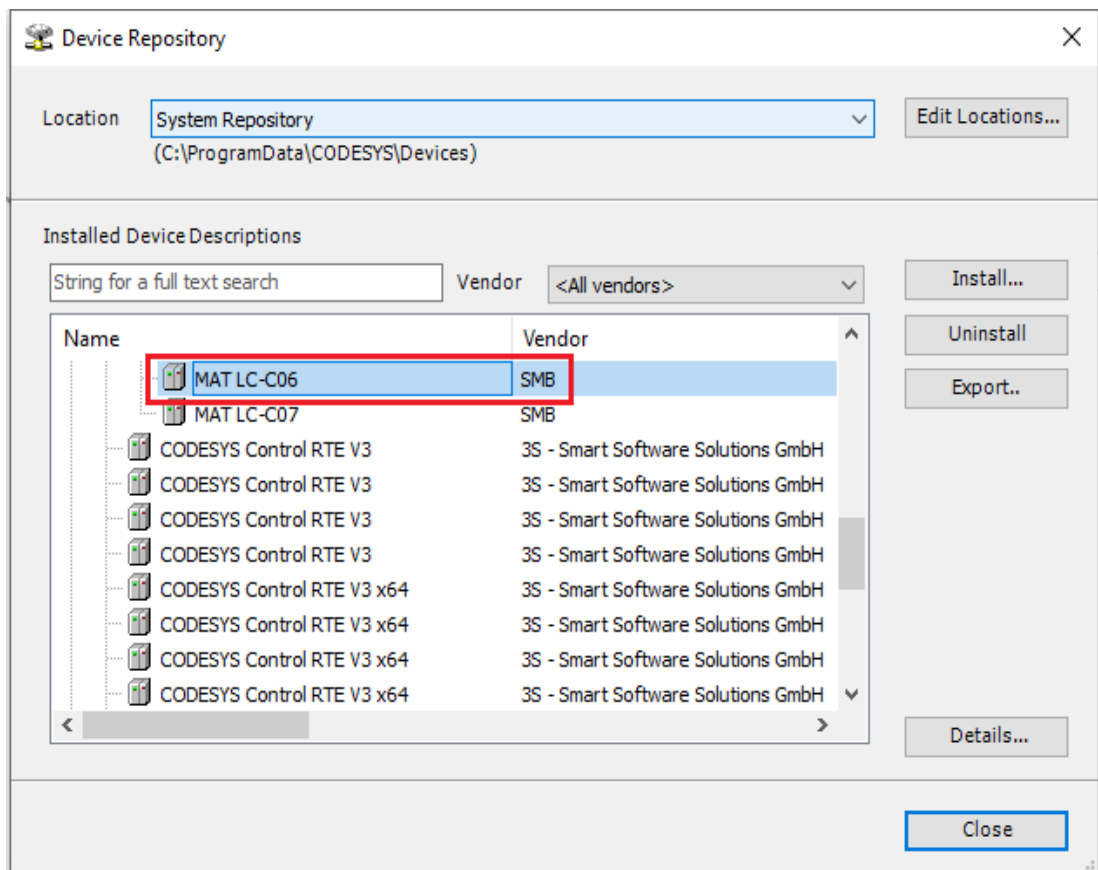
After (MAT LC-C06_V1.0.devdesc.xml) and (SMB_SD_LP_E_V1_XX.xml) are installed, SMB devices can be used in program system. Details as below:

In menu bar, select "Tools" → "Device Repository" → "Install" opens the following dialog box, where you can browse and install device description in the system. Description files provided by SMB can be selected by setting corresponding filters (**Note:** EtherCAT XML device description profile must be selected) . Click "open" button to confirm option, and then add the new device to device directory in the "Device Repository".



After above operations, refer to below steps and check if the files are installed correctly in device directory:

- In menu bar, select "Tools"→"Device Repository" to open Device Repository dialog box, shown as below:

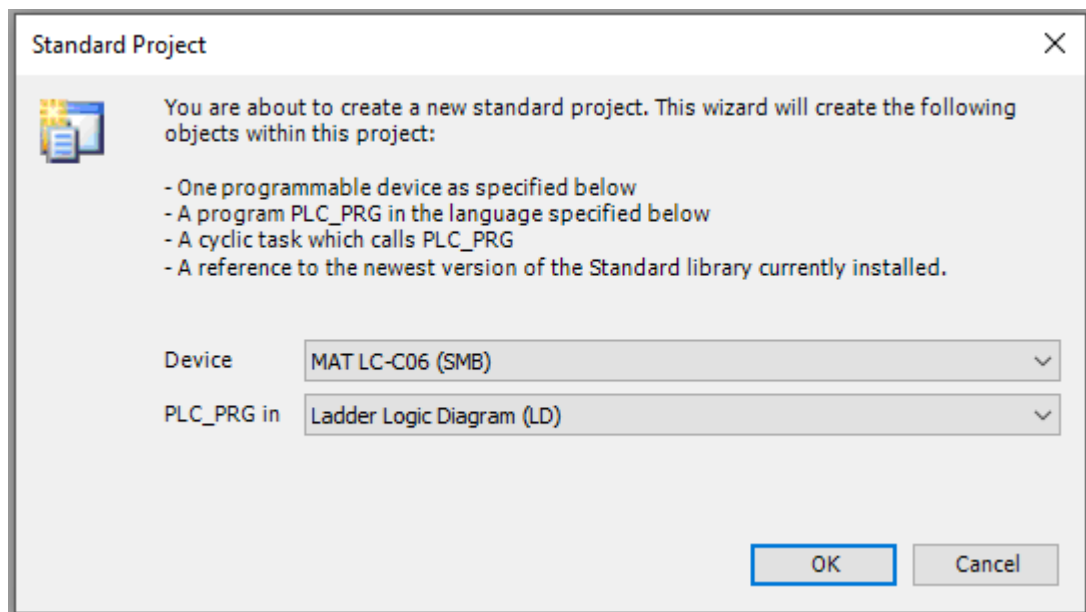


In dialog box of Device Repository, open "PLC"→"SoftMotion PLCs" and find installed "MAT-LC-C06_V1.0"; open "field bus"→"EtherCAT"→"Slave" to find installed "EtherCAT Slave". After checking, if more device files need to be installed, select "tools"→"Option", tick "enable Device Repository dialog box".

Hint
1) If you need to use other devices of SMB (CANopen or other EtherCAT slave device), installation of below device description files will be needed: CANopen slave device configuration files (*.eds) : SMB_SD_LP_C_V1_XX.eds ◆ EtherCAT slave device configuration files (*.xml) : SMB_SD_LP_E_V1_XX.xml; 2) Please download needed files in SMB website: https://www.smbtechnics.com.

2) Create a new project

- In menu bar of CODESYS main page, select "file"→"create new project", then select "standard project" in pop up dialog and set file name and storage directory.
- After above operations and confirmation, select added CPU file device (MAT-LC-C06_V1.0) and selected program language in pop up dialog, click "confirm" to finish the project creation.

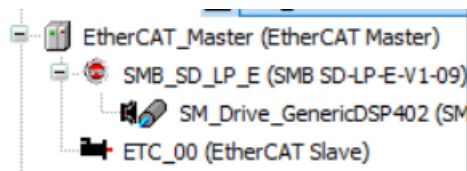


3) Add EtherCAT master

In overview of opened project, right click "Device (MAT-LC-C06_V1.0)", select "add device" to add EtherCAT master in dialog box: select supplier as <all suppliers>, field bus as "EtherCAT"→"master"→"EtherCAT Master".

4) Add EtherCAT slave

In device view, select EtherCAT master, right click and choose "add device" to add EtherCAT slave in pop up dialog: select supplier as <all suppliers>, field bus as "EtherCAT"→"slave"→"ETC-00". Then the EtherCAT slave (ETC-00), which is connected to LC-C06, will be displayed under EtherCAT master (LC-C06).



5) EtherCAT master configuration

Double click "EtherCAT-Master" and set source address (MAC) in "Overview" tab: click "browse" button on right side of "source address" (MAC) and select corresponding MAC address in pop up dialog.

General ☒ Autoconfig master/slaves

EtherCAT NIC Settings

Destination address (MAC) FF-FF-FF-FF-FF-FF ☒ Broadcast ☐ Redundancy

Source address (MAC) 00-00-00-00-00-00

Network name

☒ Select network by MAC ☐ Select network by name ☐ Compare exact name

Distributed Clock

Cycle time 4000 µs

Sync offset 20 %

☐ Sync window monitoring

Sync window 1 µs

6) EtherCAT slave (ETC-00) configuration

In “Overview” tab, tick “enable expert settings”, and set distributed clock as DC “SM-Synchron”.

EtherCAT_Master ECT_00

General **Address** **Additional**

AutoInc address -1

EtherCAT address 1002

☒ Expert settings ☐ Optional

Distributed Clock

Select DC SM-Synchron

☐ Enable 4000 Sync unit cycle (µs)

Sync0

☐ Enable Sync 0

☐ Sync unit cycle 0 Cycle time (µs)

☐ User-defined 0 Shift time (µs)

Sync1

☐ Enable Sync 1

☐ Sync unit cycle 0 Cycle time (µs)

☐ User-defined 0 Shift time (µs)

<Remark> Please refer to appendix “[F.3 EtherCAT slave device configuration](#)” for more information.

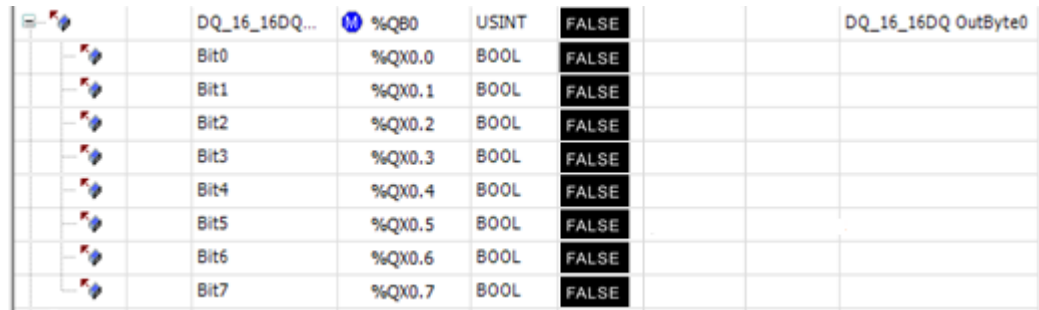
Step 5. Refer to “communication setup” about communication between LC-C06 and IPC.

Step 6. Run and debug

1) In menu, select “Online”→“Log on to...” to connect application program with LC-C06 and enter online status. Then select in menu “Debug”→“Activate” to run application program in LC-C06.

2) Monitor and debug present project

In “Module I/O mapping”(tick “always in bus cycle task”) of ETC_00 slave tab, debug attached modules. Taking Digital Output 16 Bits digital input module as example, click “ DQ_16_16DQ (Digital Output 16 Bits)” and light the output point Q0.5, operations as below:



Bit	Address	USINT	Value	DQ_16_16DQ OutByte0
Bit0	%QX0.0	BOOL	FALSE	
Bit1	%QX0.1	BOOL	FALSE	
Bit2	%QX0.2	BOOL	FALSE	
Bit3	%QX0.3	BOOL	FALSE	
Bit4	%QX0.4	BOOL	FALSE	
Bit5	%QX0.5	BOOL	FALSE	
Bit6	%QX0.6	BOOL	FALSE	
Bit7	%QX0.7	BOOL	FALSE	

7.1.5 Electronic cam based on EtherCAT communication

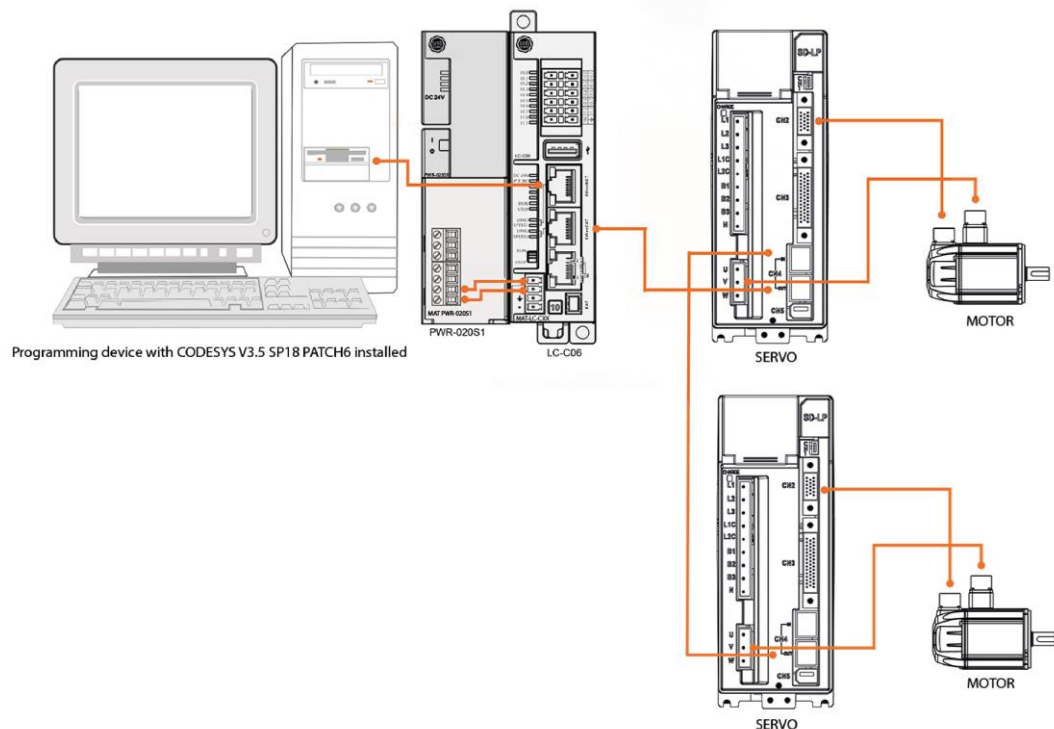
In this chapter, one example will be used to guide the creation of application program and display control function of electronic cam based on EtherCAT communication.

1. Preparation

Table 7-7 Example components of EtherCAT communication.

Component	Function
Install programming device with CODESYS V3.5 SP18 Patch6	To configure, program and debug C06/C07 series motion controller
Install guide rail	To fix modules of the system.
Power module PWR-020S1	To supply power for C06/C07 motion controller and its 24 VDC load circuit.
External power supply	To supply power for SMB servo driver.
LC-C06 motion controller	Executes user program, supplies 5V voltage to bus and communicates with other modules via Ethernet interface.
SMB servo driver and motor	In the example: SMB servo driver and matching motor are used
Standard network cable	♦ Connect LC-C06 and programming device. ♦ Connect LC-C06 and servo driver ♦ Connect SMB servo drivers
Encoder cable	Connect SMB servo driver and motor.

2. Network connection



3. Operating steps

Step 1. Connect to power supply for each device.

- 1) Open front panels of LC-C06, power module PWR-020S1 and refer to above diagram for connecting LC-C06 to PWR-020S1.
- 2) Connect SMB servo driver to main power and control power supply.

Step 2. Connect each device with cable.

Network connection details as below:

- 1) Connect PC and LC-C06 with standard network cable.
- 2) Connect LC-C06 and SMB servo driver with standard network cable.
- 3) Connect SMB servo driver and motor with encoder cable.

Step 3. Configure EtherCAT in CODESYS

- 1) Add EtherCAT master

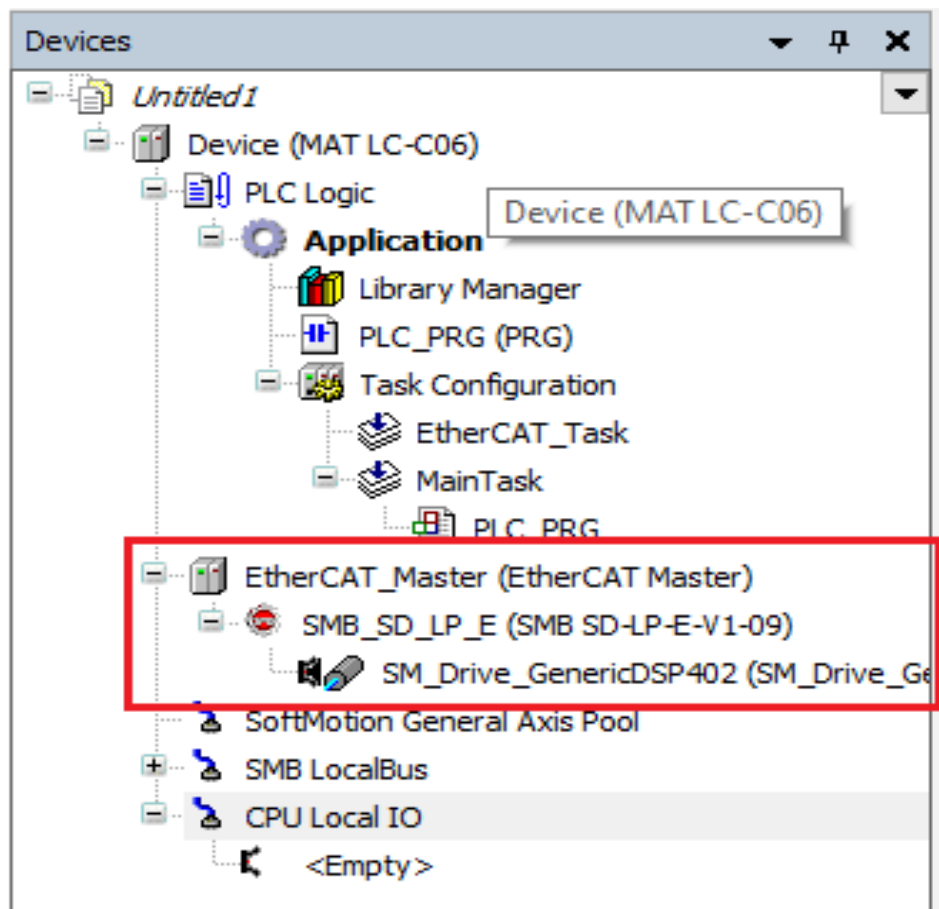
In devices view, right click "Device (MAT-LC-C06_V1.0)" and select "Add device", then select to add EtherCAT master: supplier selection "All suppliers", field bus "EtherCAT". → "master" → "EtherCAT Master".

- 2) Add or scan EtherCAT slave

In devices view, select EtherCAT master, right click to select “Add device” to add EtherCAT slave in dialogue of “Add device”: select supplier as <all suppliers>, select field bus as “EtherCAT” →”Slave”→”SMB-Servo”→”SMB”; or right click EtherCAT master (LC-C06) and select “Scan device” to display EtherCAT slave (SMB) of LC-C06 under EtherCAT master (LC-C06).

3) Add drive of EtherCAT slave

In device view, select EtherCAT slave (SMB) and right click to select “Add softmotion-CiA402-axis” in pop dialog to add the drive.



4) EtherCAT master configuration

Set source address (MAC) in “Overview” tab: click “browse” button on right side of “source address” (MAC) and select corresponding MAC address in pop up dialog.

5) EtherCAT slave (SMB) configuration

In “Overview” tab, tick “enable expert settings”, and set distributed clock as “SM-Synchron”

<Remark> Please refer to appendix “[F.3 EtherCAT slave device configuration](#)” for more information.

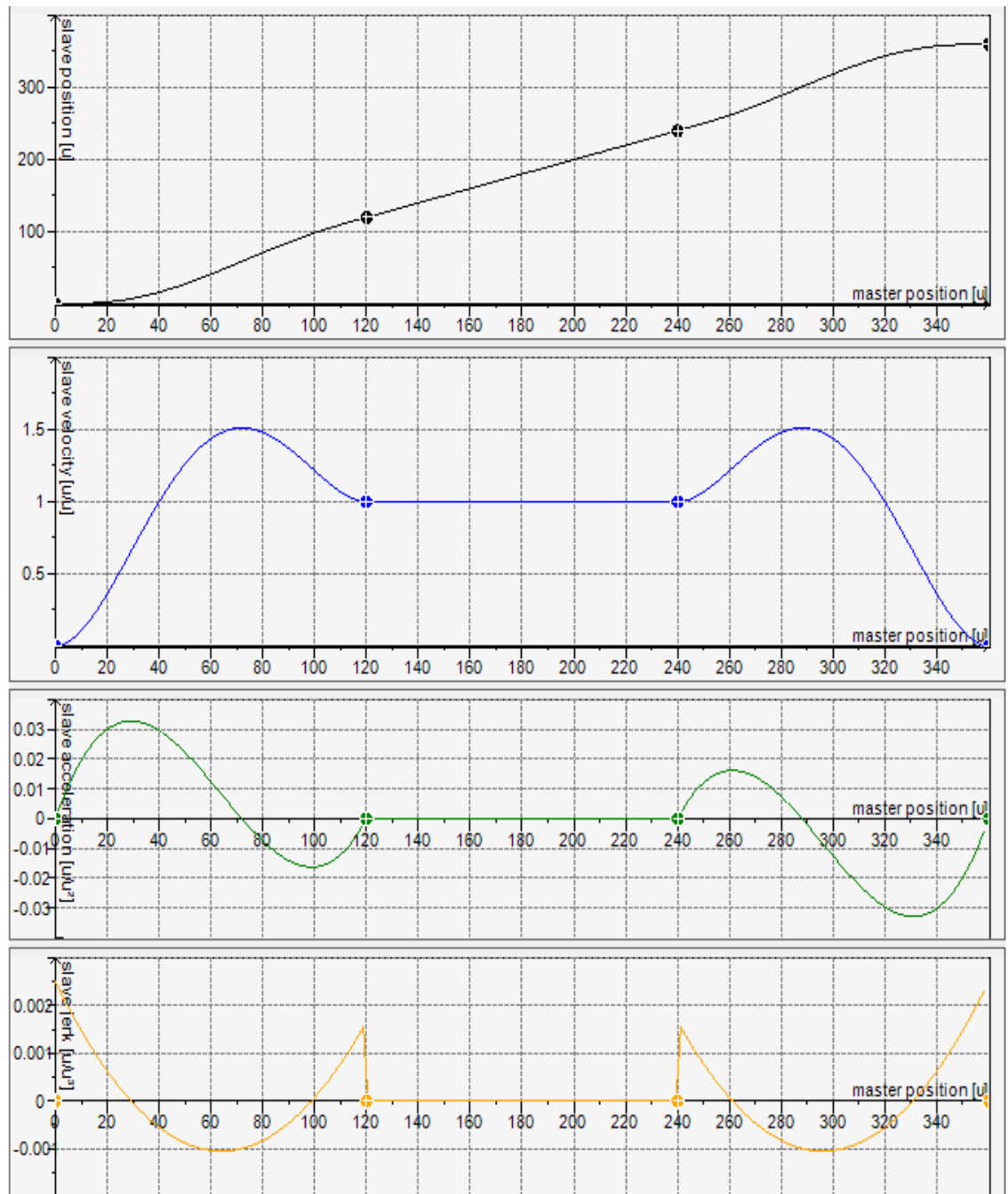
6) SMB drive (SM_Driver_GenericDSP402) configuration

“Softmotion drive: basic” tab of SM_Driver_GenericDSP402 is as below:

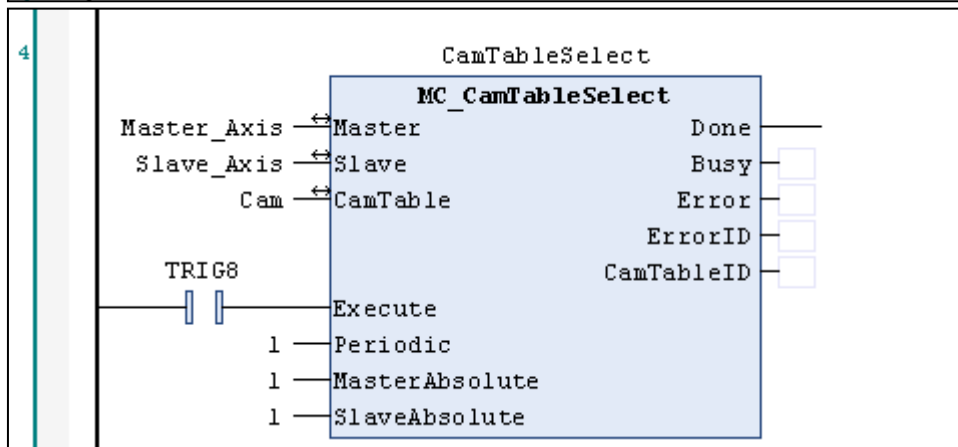
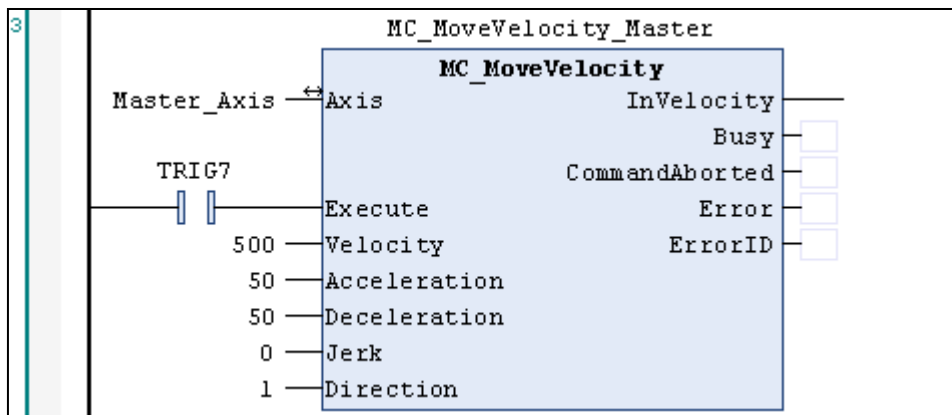
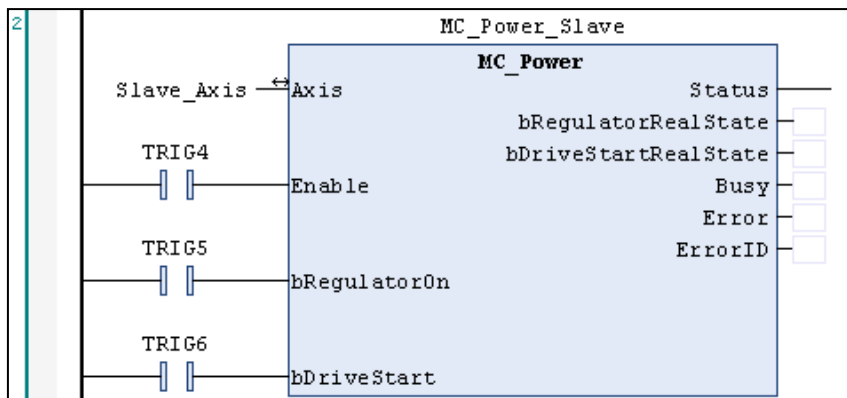
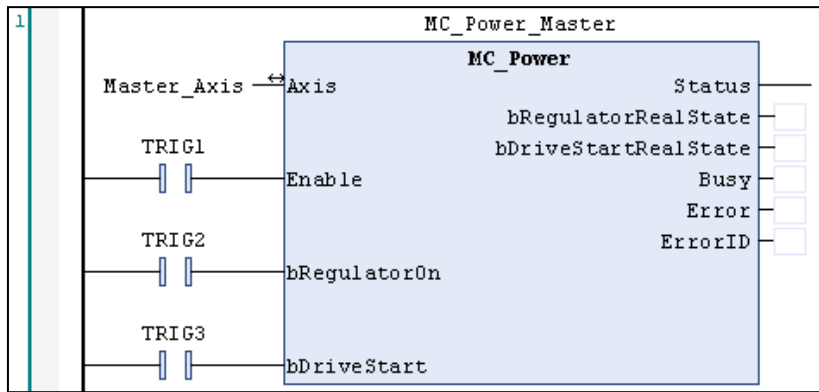
Set axis type and limit as “Finite”.

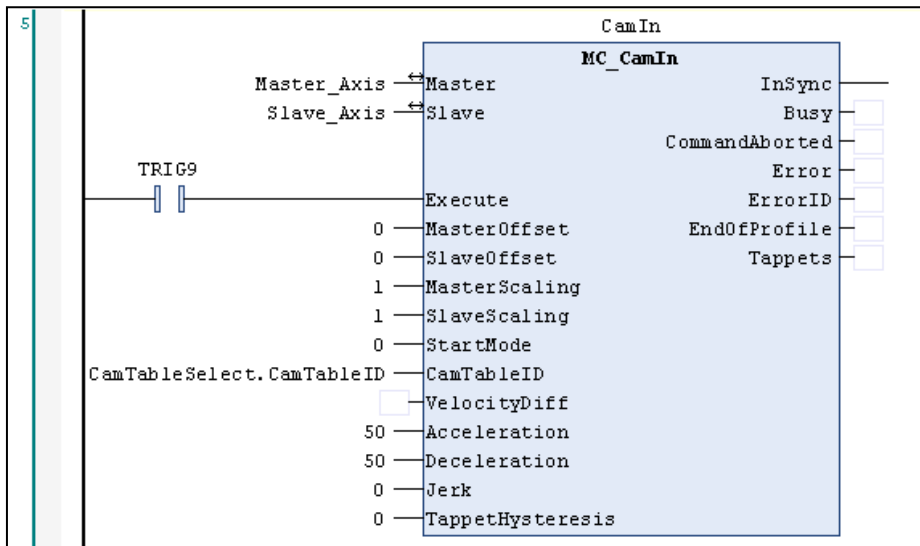
7) Create cam table

In device view, open "Device"→"PLC", right click "Application" and select "Add object"→"cam table", input table name in pop-up dialog, click "Open" and the cam table is created. Double click the created cam table and set it in ideal conditions as below:



Step 4. Programming





Step 5. Refer to chapter “2.3 Communication setup” for more information about communication setting between LC-C06 and IPC.

Step 6. Run and debug

1) In menu, select “Online”→“Log on to...” to connect application program with LC-C06 and enter online status. Then select in menu “Debug”→“Activate” to run application program in LC-C06.

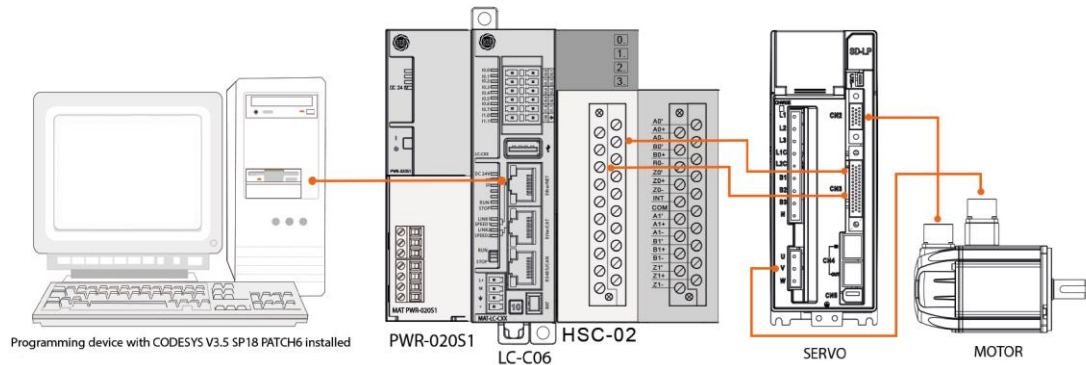
2) Start monitoring and debugging present project.

In device view, double click “PLC_PRG (PRG)” to open program diagram, trigger and download each instruction in sequence.

7.1.6 EtherNET/IP communication

You can add any X device that supports Ethernet/IP communication on Codesys and make the necessary adjustments to communicate with the SMB controller and perform the necessary operations.

2. Network connection



3. Operating steps

Step1: Wiring

Open front panels of LC-C06 and power module PWR-020S1, wiring according to below steps:

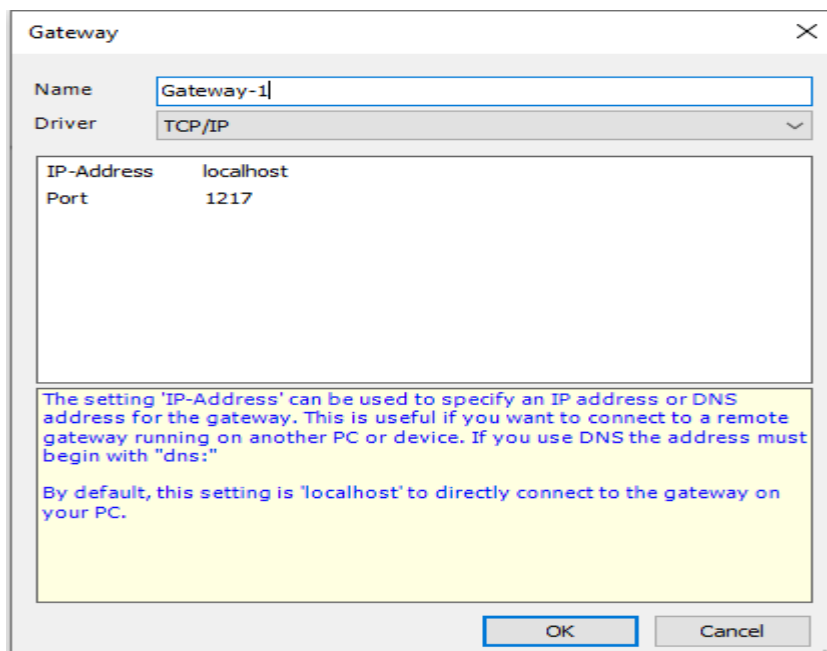
- 1) Connect programming device and LC-C06 (EtherNET communication port) with standard network cable.
- 2) Connect LC-C06 and high speed counting module via bus.
- 3) Connect programming device and SMB servo driver with standard network cable.
- 4) Connect SMB servo driver and motor with encoder cable.
- 5) Wiring for high speed counting module and SMB servo driver.

Step 2: Run the drive system

Set parameters of SMB servo driver to run the motor, refer to <SMB Series AC Servo Driver User Manual> for more details.

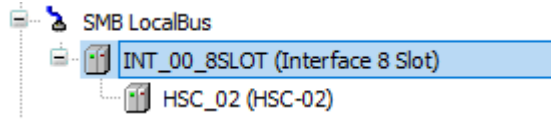
Step 3: PLC communication setup

Refer to Chapter 2 Introductory and set communication in CODESYS, to establish communication between LC-C06 and programming device.



Step 4: Configure in CODESYS

In device directory, right click "SMB LocalBus" and select "Add device", then in pop-up dialog, select supplier as "SMB", select "Specialized device Interface 8 Slot", then click "Add device" to add the device, which will be displayed under SMB LocalBus. At last, select added INT_00 and right click to select "Add device" and select HSC-02 device.



<Remark> Double click device HSC-02 to open its property dialog and configure mode, control word and other parameters of the module. If set here, then commands of HSC_300, HSC_SETMODE will not be needed.

Step 5: Run the drive system.

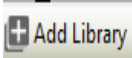
Set parameters of SMB servo driver to run the motor, refer to <SMB Series AC Servo Driver User Manual> for more details.

Step 6: Add library file Extbus Library

Install library file through "Tools"→"Library" in menu.

After library file is installed correctly, a dialog will pop up to remind. Add the new library to library manager according to below steps:

1) Open library manager in device directory: PLC→Application→Library manager

2) In library manager interface, click  to add library SMB_ExtBus_V1.3.library

Step 7: Program through Extbus library

After library file of HSC-02 is correctly added, information like module speed, position and status can be read through commands.

<Remark> HSC_300、HSC_SETMOD。 If parameters like control word, mode are configured in property dialog of HSC-02, HSC_300, HSC_SETMODE commands are not needed.

Step 8: Debug and monitor the program

1) In menu, select "Online"→"Log on to..." to connect application program with LC-C06 and enter online status. Then select in menu "Debug"→"Activate" to run application program in LC-C06.

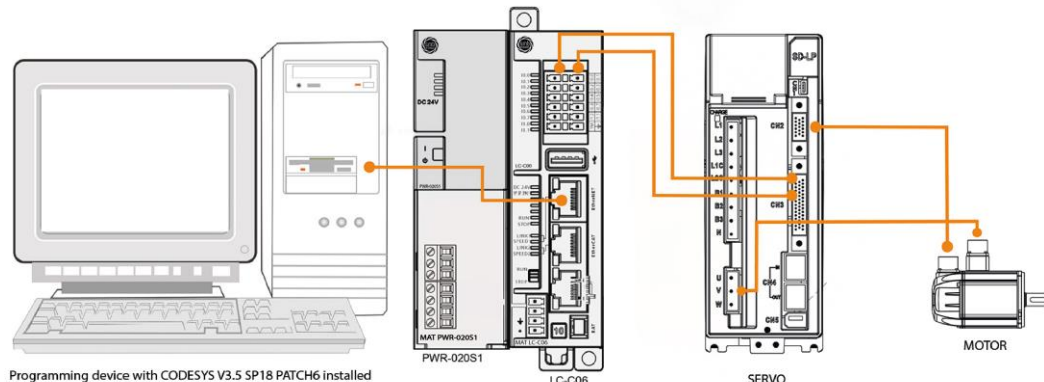
2) Program debugging

Read values like current position, speed of motor through commands like HSC_GETCV, HSC_GETSPEED of program.

<Remark> Refer to [G ExtBus Library Introduction](#) for details of Extbus library.

7.2.2 Application example of CPU high speed counter

1. Network connection:



2. Operating steps

Step1: Wiring

Open front panels of LC-C06 and power module PWR-020S1, wiring according to below steps:

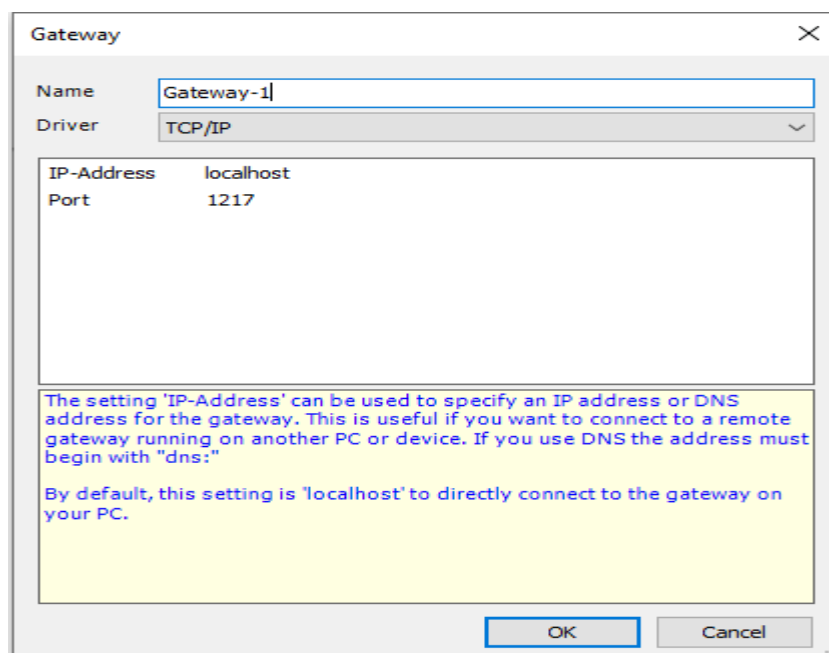
- 1) Connect programming device and LC-C06 (EtherNET communication port) with standard network cable.
- 2) Connect programming device and SMB servo driver with standard network cable.
- 3) Connect SMB servo driver and motor with encoder cable.

Step 2: Run the drive system

Set parameters of SMB servo driver to run the motor, refer to <SMB Series AC Servo Driver User Manual> for more details.

Step 3: PLC communication setup

Refer to Chapter 2 Introductory and set communication in CODESYS, to establish communication between LC-C06 and programming device.

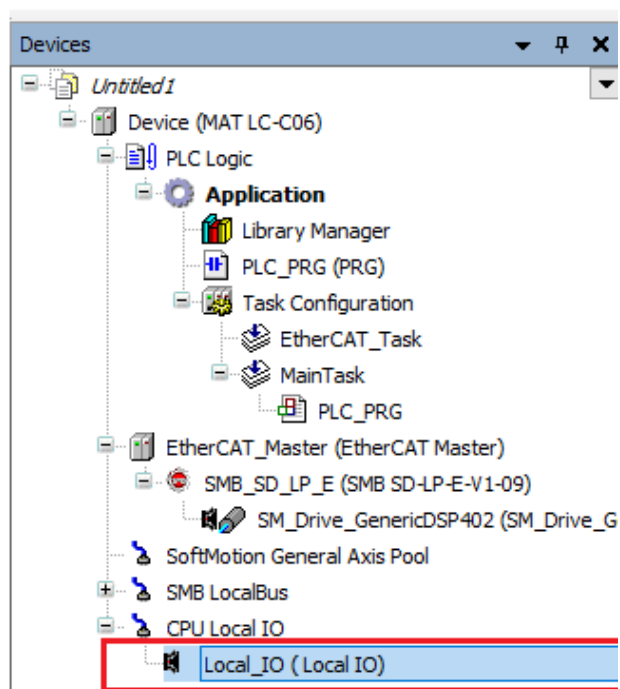


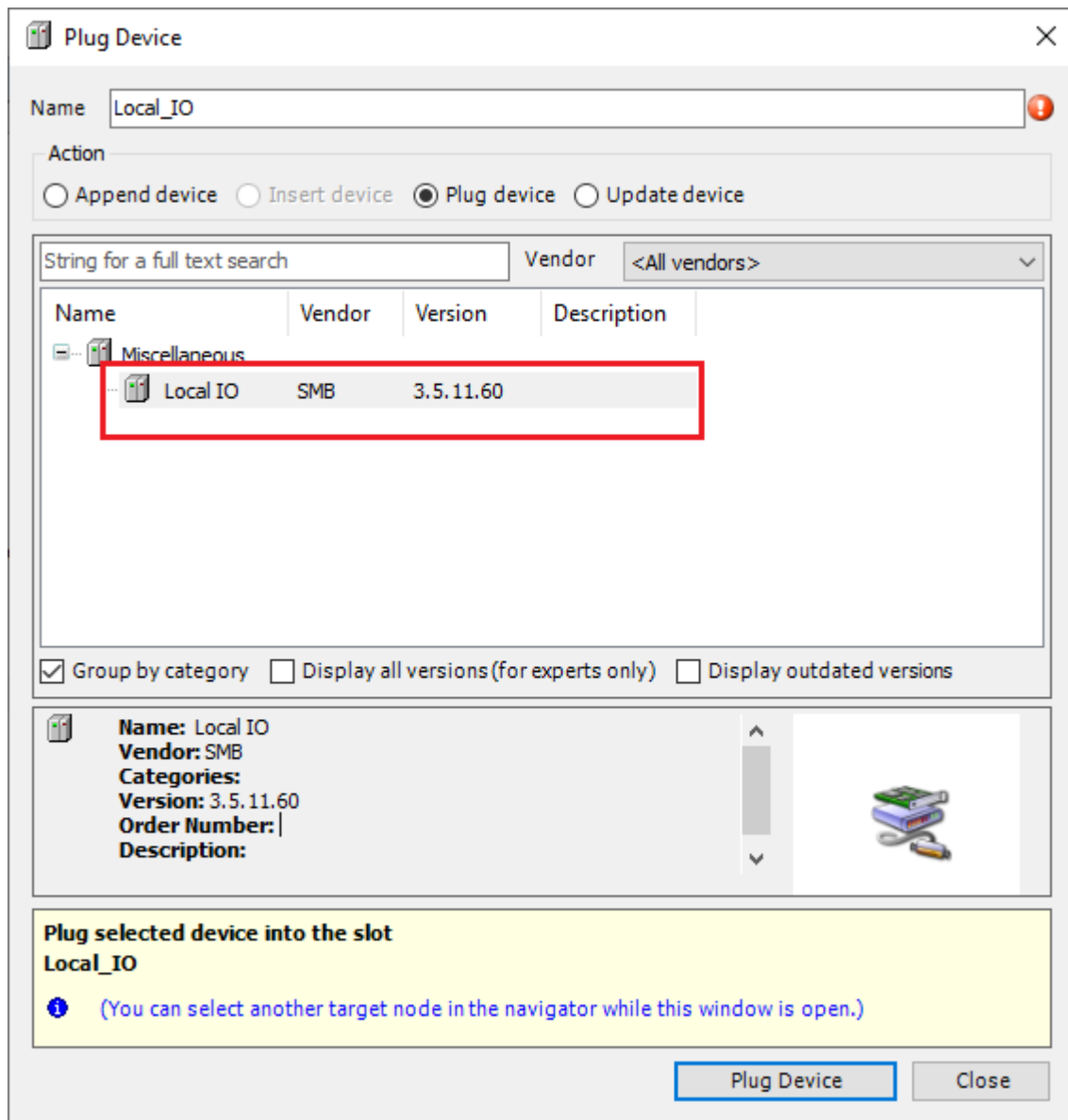
Step 4: Configure in CODESYS

Double click desktop icon  or start CODESYS in menu of START, then operate as below:

1) Create a new project

- In menu bar of CODESYS main page, select “file”→”create new project”, then select “standard project” in pop up dialog and set file name and storage directory.
- After above operations and confirmation, select added CPU file device (MAT-LC-C06_V1.0) and selected program language in pop up dialog, click “confirm” to finish the project creation.
- In device directory, right click “<blank><blank>” under “CPU Local IO”, select “Insert device” and in pop-up dialog, select supplier as “SMB”, select “Local IO” and click “Insert device” button to add the device, which will be finally displayed under CPU Local IO.

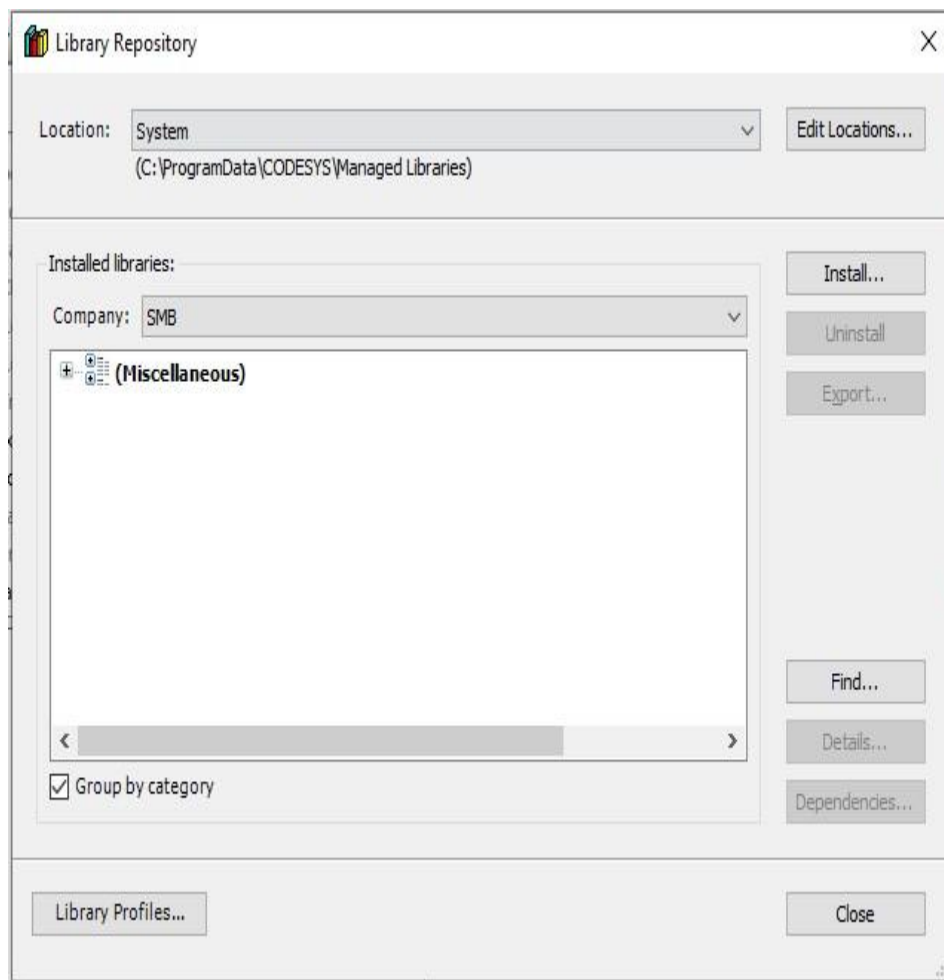




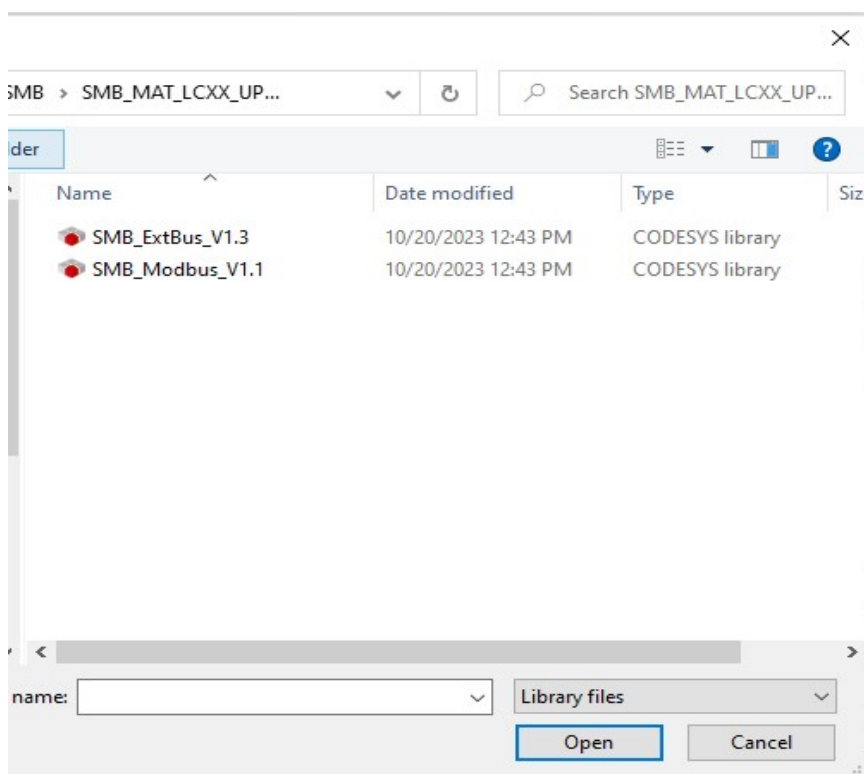
2) Install library file

After installation, high speed counter integrated in C06/C07 can be used in program system.
 Installation steps are as below:

In menu, select "Tools"→"Libray Repository" to open below dialog, click "Install".

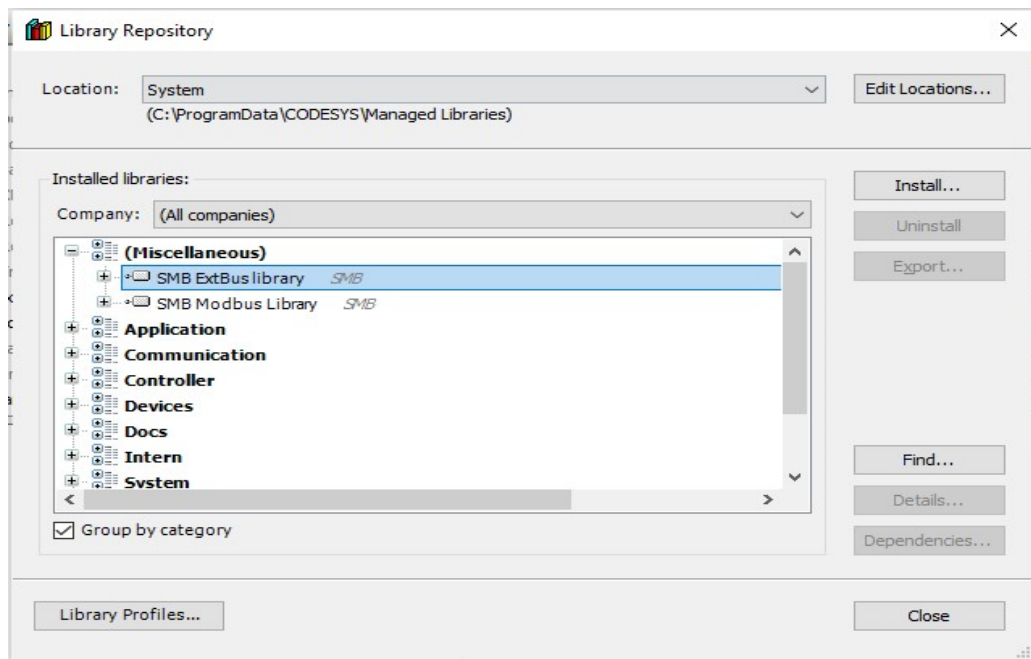


Select “SMB_ExtBus_V1.3.library” in dialog, click “Open” button to confirm selection, then in “Device Repository”, add the new device to device directory.



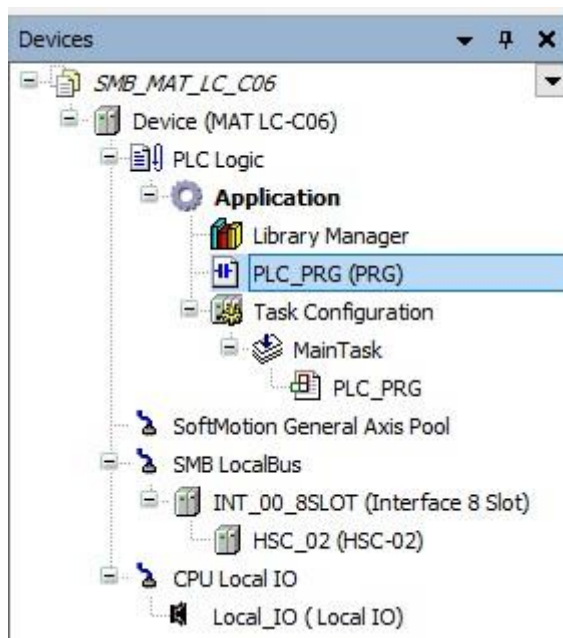
After above operations, refer to below steps to check if the file is correctly installed in device directory:

- In menu, select "Tools"→"Library Repository"→"Misc." to check installed SMB ExtBus library

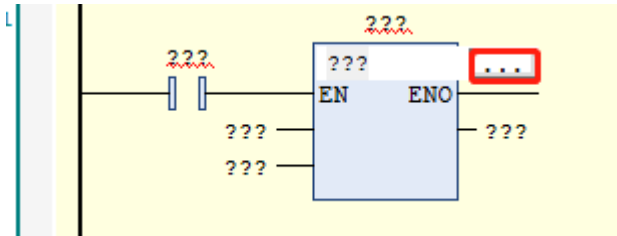
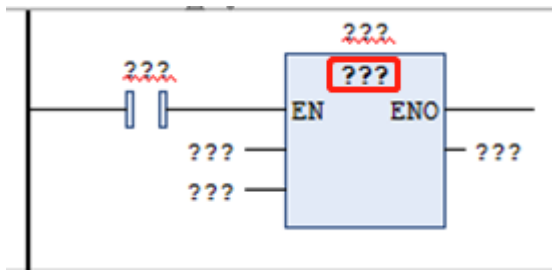


3) Program with library file

After library file is added correctly, the instructions of the high-speed counter can be called for programming, and HSC_GETCV instruction is the example. Firstly, program in PLC_RPG.



Drag a function block with EN / ENO from "Toolbox" on the right, and click the three question marks in the function block box to display option box.



Click the option box, select "SMB_ExtBus"→"HSC" to see instructions related to the high-speed counter, and click needed instruction to confirm and transfer instructions into program.

Name	Type	Origin
SM3_Basic	Library	SM3_Basic, 4.3.2.0 (3S - Smart Softwa...
SM3_CNC	Library	SM3_CNC, 4.3.2.0 (3S - Smart Softwa...
SM3_Math	Library	SM3_Math, 4.3.2.0 (3S - Smart Softwa...
SM3_Robotics	Library	SM3_Robotics, 4.3.2.0 (3S - Smart Sof...
SMB_ExtBus_library	Library	SMB ExtBus library, 1.3.0 (SMB)
SMB_Modbus_RTU_and_TCP	Library	SMB Modbus Library, 1.1 (SMB)
SML	Library	SML_Basic, 4.2.0.0 (3S - Smart Softwa...
SML_Drive_ETC	Library	SML_Drive_ETC, 4.3.0.0 (3S - Smart S...
Standard	Library	Standard, 3.5.9.0 (System)

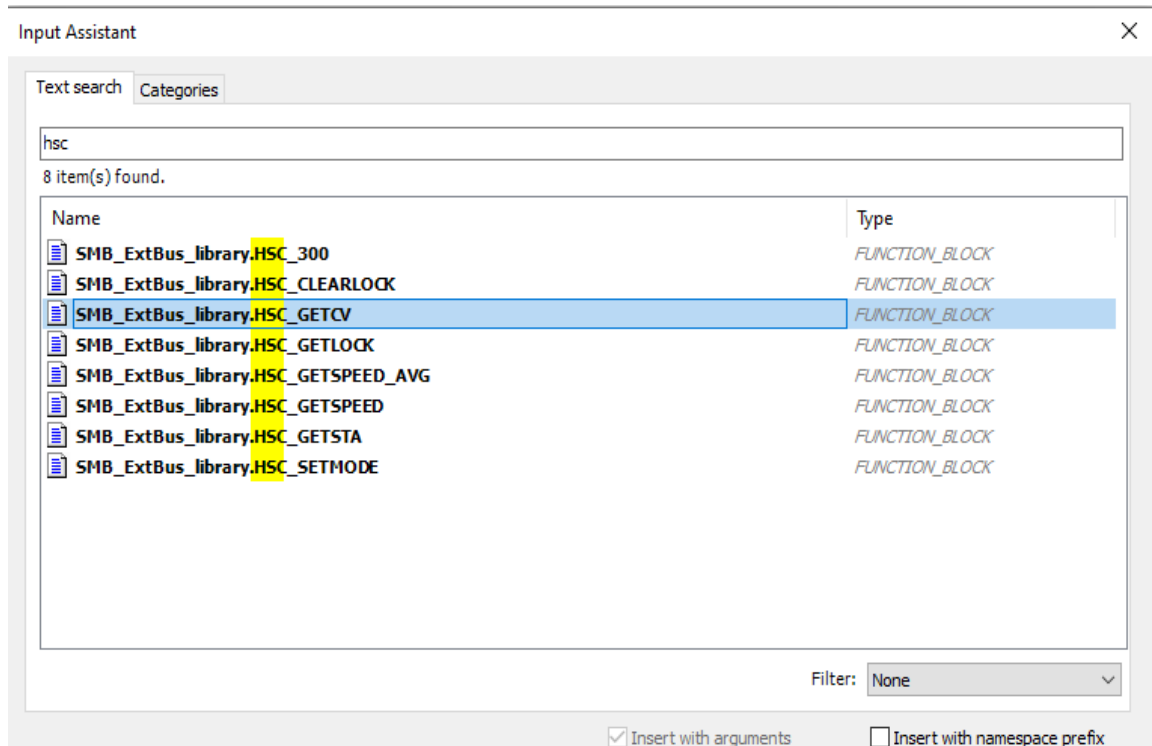
Text search
Categories

Function blocks
Module Calls
Keywords
Conversion Operators

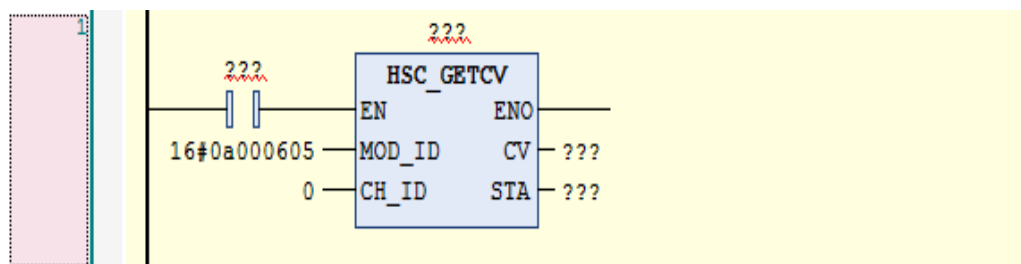
Name	Type	Origin
SM3_Basic	Library	SM3_Basic, 4.3.2.0 (3S - Smart Softwa...
SM3_CNC	Library	SM3_CNC, 4.3.2.0 (3S - Smart Softwa...
SM3_Math	Library	SM3_Math, 4.3.2.0 (3S - Smart Softwa...
SM3_Robotics	Library	SM3_Robotics, 4.3.2.0 (3S - Smart Sof...
SMB_ExtBus_library	Library	SMB ExtBus library, 1.3.0 (SMB)
EBUSBroadcast	FUNCTION_BLOCK	SMB ExtBus library, 1.3.0 (SMB)
EBUSRead	FUNCTION_BLOCK	SMB ExtBus library, 1.3.0 (SMB)
EBUSWrite	FUNCTION_BLOCK	SMB ExtBus library, 1.3.0 (SMB)
HSC		
HSP		
SMB_Modbus_RTU_and_TCP	Library	SMB Modbus Library, 1.1 (SMB)
SML	Library	SML_Basic, 4.2.0.0 (3S - Smart Softwa...
SML_Drive_ETC	Library	SML_Drive_ETC, 4.3.0.0 (3S - Smart S...
Standard	Library	Standard, 3.5.9.0 (System)

☒ Structured view

Select HSC_GETCV instruction.



After the instruction is called, the MOD_ID value it inputs is the default one in “Internal configuration”.



In device directory, click “Local_IO”, select “Internal configuration” to see default MDO_ID value.

Internal Parameters				
Internal I/O Mapping				
Status				
Information				
Parameter	Type	Value	Default Value	
Module Id	DWORD	16#0a000605	16#0a000605	
channel 0-3 filter	BYTE	6	6	
channel 4-7 filter	BYTE	6	6	
channel 8-9 filter	BYTE	6	6	
HSC0 MODE	BYTE	0	0	
HSC0 Filter	BYTE	16#2	16#2	

Remark: All MOD_ID value input by HSC related instructions, take default value in “Internal configuration” of “Local_ID”.

Step 5: Debug and monitor the program

1) In menu, select "Online" → "Log on to..." to connect application program with LC-C06 and enter online status. Then select in menu "Debug" → "Activate" to run application program in LC-C06.

2) Program debugging

Read values like current position, speed of motor through commands like HSC_GETCV, HSC_GETSPEED of program.

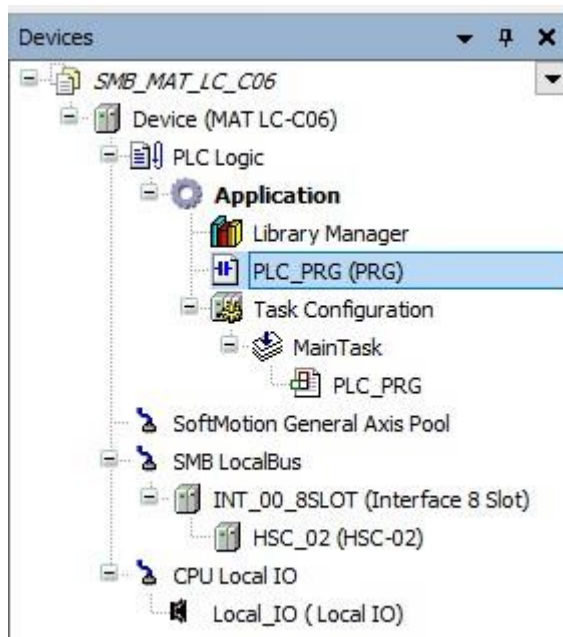
7.3 HSC Interrupt example

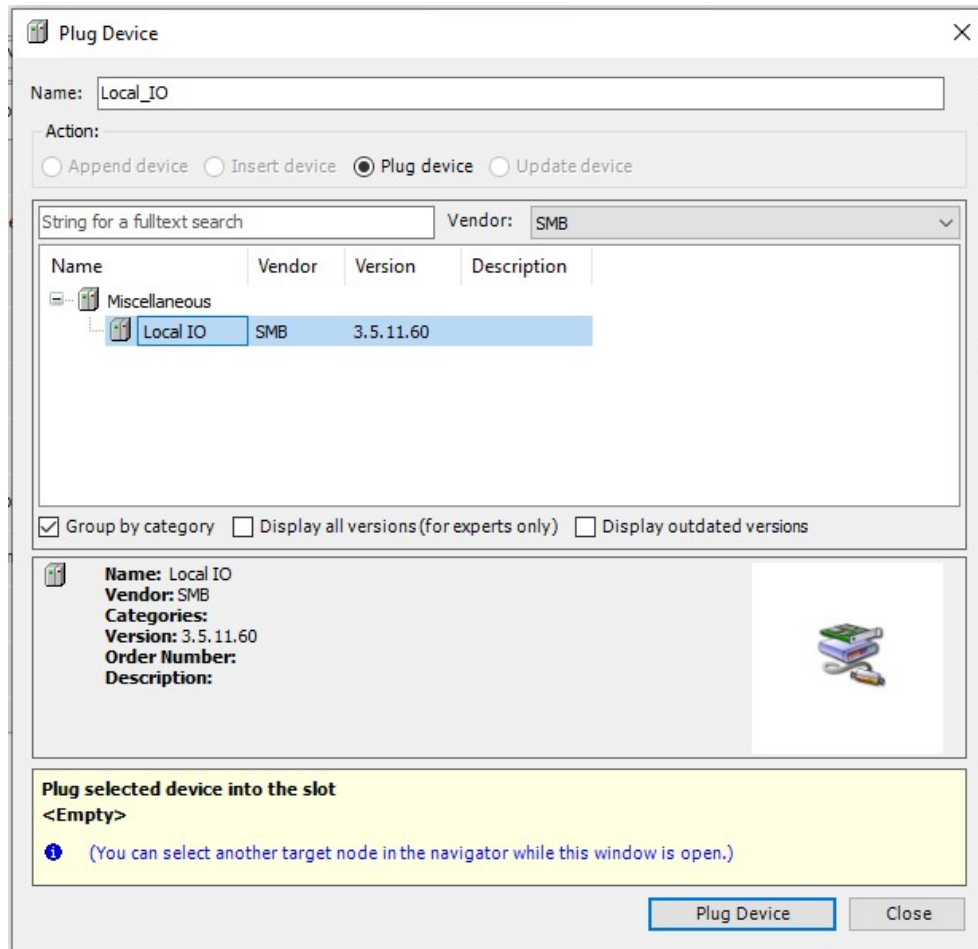
1. Configure in CODESYS

Double click desktop icon  or start CODESYS in menu of START, then operate as below:

1) Create a new project

- In menu bar of CODESYS main page, select "file" → "create new project", then select "standard project" in pop up dialog and set file name and storage directory.
- After above operations and confirmation, select added CPU file device (MAT-LC-C06_V1.0) and selected program language in pop up dialog, click "confirm" to finish the project creation.
- In device directory, right click "<blank><blank>" under "CPU Local IO", select "Insert device" and in pop-up dialog, select supplier as "SMB", select "Local IO" and click "Insert device" button to add the device, which will be finally displayed under CPU Local IO.

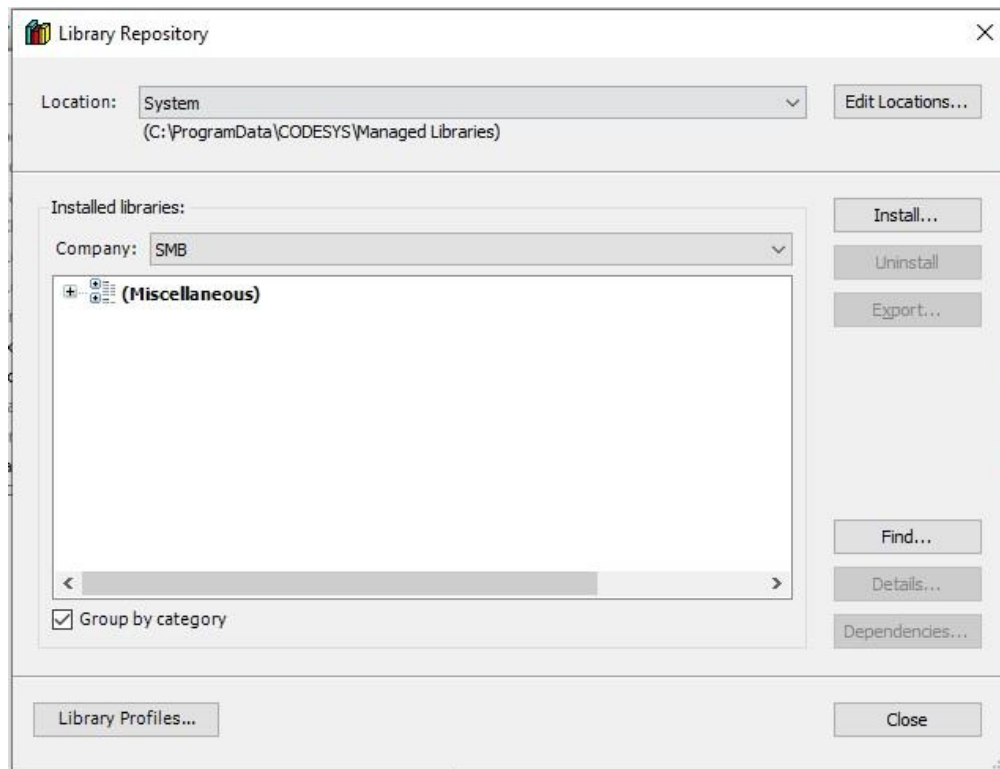




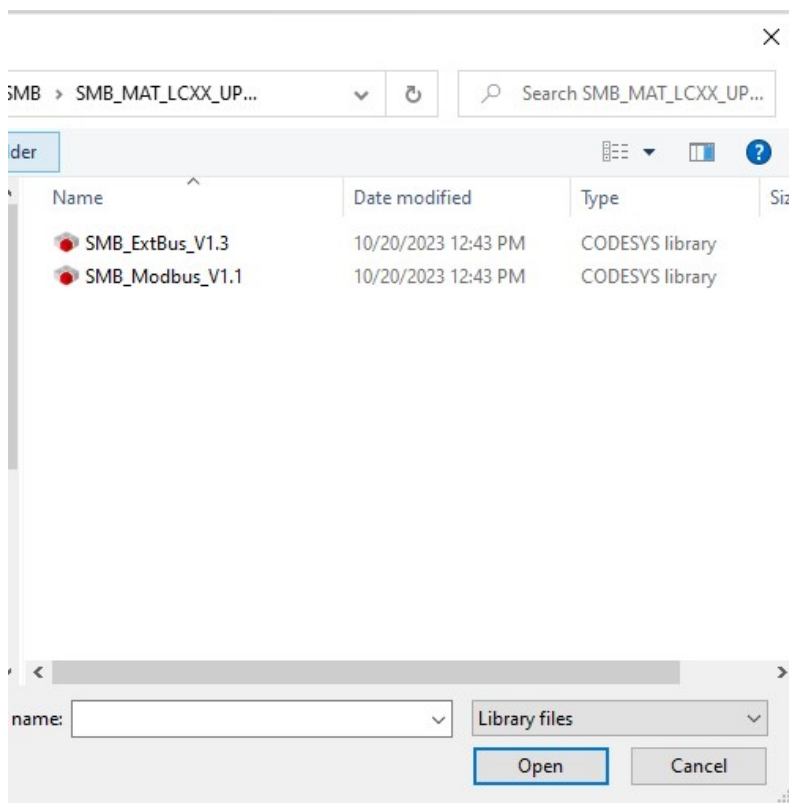
2) Install library file

After installation, high speed counter integrated in C06/C07 can be used in program system.
 Installation steps are as below:

In menu, select "Tools"→"Library Repository" to open below dialog, click "Install".

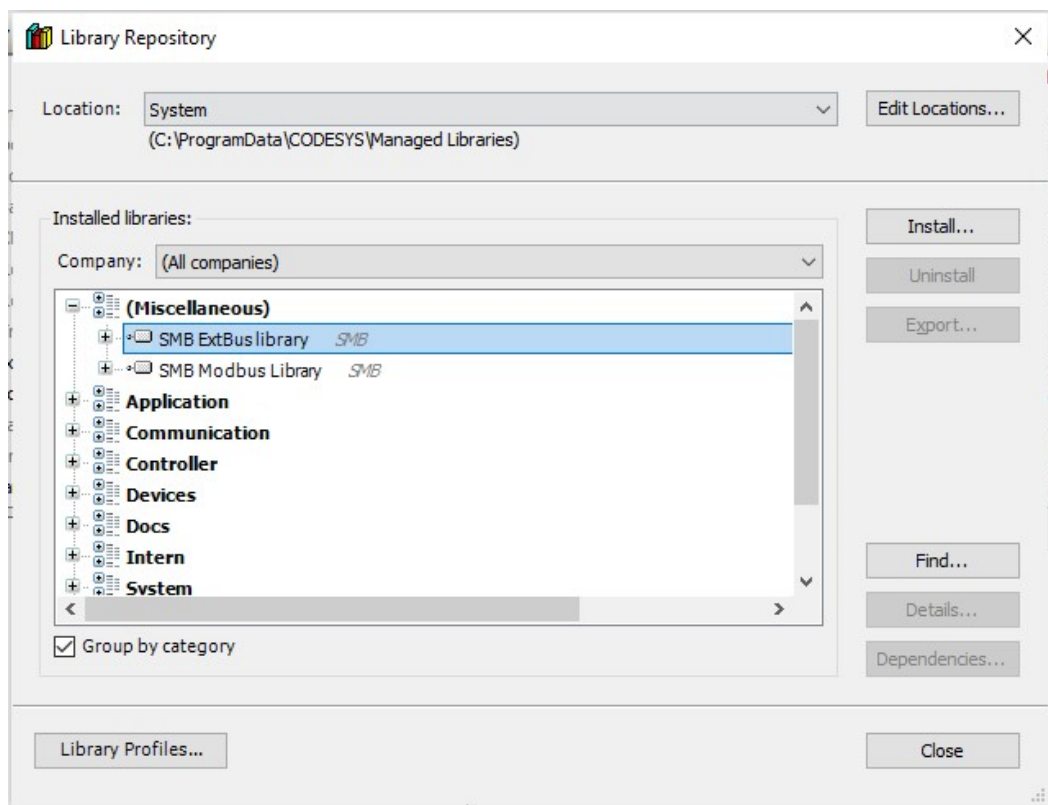


Select “SMB_ExtBus_V1.3.library” in dialog, click “Open” button to confirm selection, then in “Device Repository”, add the new device to device directory.



After above operations, refer to below steps to check if the file is correctly installed in device directory:

- In menu, select “Tools”→“Library Repository”→“Misc.” to check installed SMB ExtBus library.

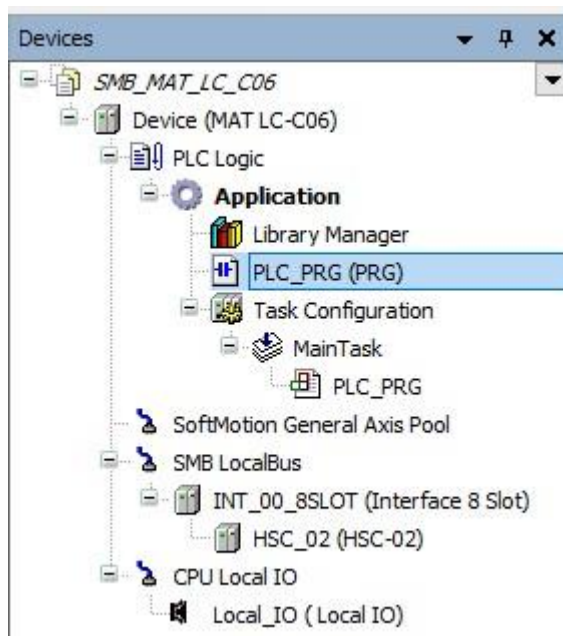


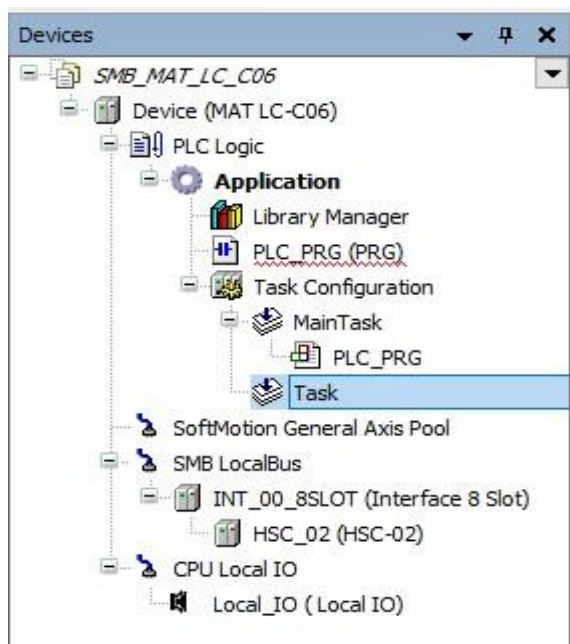
3) Program with library file

After library file is added correctly, the instructions of the high-speed counter can be called for programming.

2. Set interrupt

Right click “Task configuration”, select “Add object”, then “Task” to create a new task.





Click created “Task” to set interrupt. Firstly, set configure type as “External”.

Configuration

Priority (0..31):

Type
 External External event:

Watchdog
☐ Enable
 Time (e.g. t#200ms): ms
 Sensitivity:

Then select external event among the 30 options.

Configuration

Priority (0..31):

Type
 External External event:

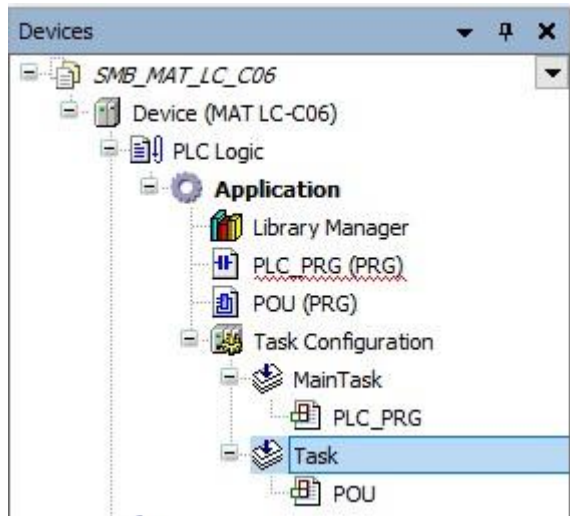
Watchdog
☐ Enable
 Time (e.g. t#200ms): ms
 Sensitivity:

+ Add Call X Remove Call Change Call Move Up Move Down Open POU

POU	Comment

3. Connection interruption

Right click “Application”→”Add object”→” Program organization unit”, click “Open” to create a new POU.



Drag created POU to place under configured “Task”(interruption task), and write sub-program of interruption. When interruption happens, corresponding sub-program will execute for once.

7.4 CNC routine

Please get CNC function related routine in installation directory of CODESYS:

C:\Program Files\SMB\CODESYS\Projects\SoftMotion\Examples\Tutorial

PC > Local Disk (C:) > Program Files > SMB > CODESYS > Projects > SoftMotion > Examples > Tutorial

- CNC_File
- CNC_File_3DPath
- CNCdirect
- CNConline
- CNCprepro
- CNCTable
- PLCopenMulti
- PLCopenMultiCAM

Appendix

Table A-1 Explanation of abbreviated terms

Abbreviation	Full name	Explanation
AC	Alternating Current	Alternating current
DC	Direct Current	Direct current
ADC	Analog to Digital Converter	Analog to digital converter
DAC	Digital-to-Analog Converter	Digital-to-analog converter
CODESYS	Controlled Development System	Development system under controll
PLC	Programmable Logic Controller	Programmable logic controller
CPU	Central Processing Unit	CPU of PLC
HMI	Human Machine Interface	Human machine interface
FB	Function Block	Function block
FC	Function	Function
HSC	High-speed counter	High-speed counter
IEC	International Electro Technical Commission	International electro Technical Commission
SDO	Service Data Object	Service data object
PDO	Process Data Object	Process data object
RTD	Resistance Temperature Detector	Resistance temperature detector
TC	Thermocouple	Thermocouple
UDP	User Datagram Protocol	User Datagram Protocol
IL	Instruction List	Instruction list
ST	Structure Text	Structure text
SFC	Sequential Function Chart	Sequential function chart
CFC	Continuous Function Chart	Continuous function chart editor
FBD	Function Block Diagram	Function block diagram
LD	Ladder Diagram	Ladder diagram

B Power budget

After selecting CPU, power module, relay module and extension modules of each frame, check if current and power consumption system bus meet below conditions:

Condition 1 : Check current consumption of bus

Internal voltage of bus is 5 VDC and current is provided by CPU(when there's no relay module) or relay module. Bus current consumption of all extension modules in total should not exceed max bus current allowed by CPU or relay module.

Condition 2: Check power consumption of bus

When using power module, power consumption of other modules of each frame in total should not exceed max power consumption allowed by power module.

When using external power supply, choose model of proper power according to total power consumption of attached modules.

Table B-1 Current supply and consumption of 5VDC bus

Model	Current supply	Current consumption
LC-C06	1600mA	--
LC-C07	1600mA	--
DIT-08	--	60mA
DIT-16	--	80mA
DQT-08	--	70mA
DQT-16	--	120mA
DQR-08	--	45mA
DQR-16	--	60mA
AIS-04	--	50mA
AQS-04	--	40mA
HSC-02	--	100mA
HSP-04	--	100mA

Table B-2 Current supply and consumption of 24VDC bus

Model	Current supply	Current consumption
PWR-020S1	2000mA	--
LC-C06	--	800mA
LC-C07	--	800mA
DIT-08	--	--
DIT-16	--	--
DQT-08	--	50mA
DQT-16	--	95mA
DQR-08	--	64mA
DQR-16	--	130mA
AIS-04	--	65mA
AQS-04	--	110mA
HSC-02	--	--
HSP-04	--	100mA

C Specifications of Extension Modules

C.1 Power module

PWR-020S1 power module, best match of MAT-LC-CXX system, supplies 24V DC power for CPU and extension modules (except for digital module). Select one power module for each frame, and other power supply for digital I/O and sensor.

Table C-1 Basic properties of PWR-020S1 power module

Name	Specification	Order No.
PWR-020S1 power module	Input: 85~264V AC Output: 24V DC/2A	MAT-PWR-020S1

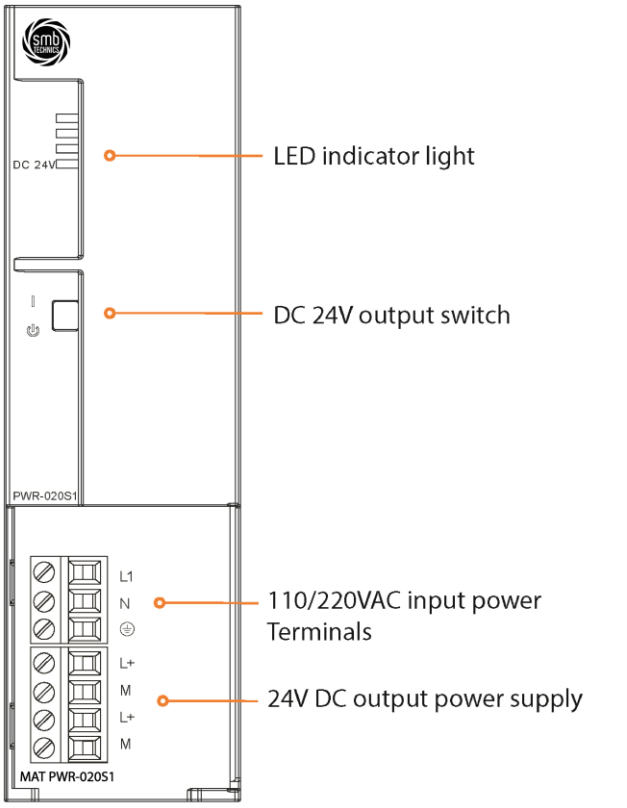
Table C-2 General properties of PWR-020S1 power module

Physical property	
Dimensions (W×H×D)	34×115×101.6 mm
LED Indicator characteristic	
24V power indicator (green)	On: with 24V DC output, OFF: without 24VDC output
Switch property	
Switch	Control 24V DC power output ON: with 24V DC output, OFF: without 24VDC output

Table C-3 Function characteristics of PWR-020S1 power module

Characteristics of input voltage	
Voltage range	85~264VAC, wide voltage input
Rated frequency	50Hz/60Hz
Frequency range	47Hz~63Hz
Efficiency	75%
AC current	0.9A/110V、0.5A/220V
Surge current (maximum at 25 °C)	≤20A/110V、≤35A/220V
Leakage current	≤5mA/220VAC
Characteristics of output voltage	
DC voltage / rated current	24VDC/2A
Rated power	48W
Ripple and noise (maximum)	150mVp-p
Voltage output range	±5%
Start / rise / hold time	≤2.5s/≤50ms/≥20ms
Isolation (power input and output)	Isolation between 110V/220V AC and 24V DC
Protection	
Overload protection	105%~130% of rated output power, cut off output and automatic recovery after troubleshooting
Overvoltage protection	115%~135%Ue; Protection mode: hiccup mode, automatic recovery after troubleshooting
Surge protection	The power supply terminal provides surge absorption function
Overcurrent protection	The power output terminal provides overcurrent protection
Safety electromagnetic compatibility	
Withstand voltage	Input~output:1.5KVDC,input-PE:1.5KVDC,output-PE: 500VDC
Isolation resistance	Input~output, input-PE, output-PE: 100MΩ/500VDC
Standards	Safety: UL60950 and UL1950, Electromagnetism: EN55022

Interface diagram



Interface definition

Table C-4 220V AC input power interface definition of PWR-020S1

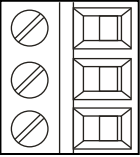
3-position removable terminal	Signal	Signal definition
	L	Live line
	N	Neutral wire
		Earth

Table C-5 24V DC output power interface definition of PWR-020S1

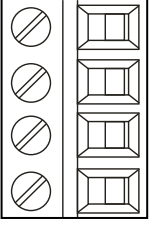
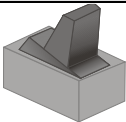
4-position removable terminal	Signal	Signal definition
	L+	24VPower supply (positive)
	M	24VPower supply (negative)
	L+	24VPower supply (positive)
	M	24VPower supply (negative)

Table C-6 Dial switch definition of PWR-020S1

Two-state switch	Position	Dial direction	Signal definition
	ON	Upward	With 24V DC output
	OFF	Downward	Without 24V DC output

C.2 Digital module

Table C-7 Basic properties of digital module

Name	Specification	Order No.
DIT-08 digital module	Digital input 8 x 24VDC	MAT-LC-DIT-08
DIT-16 digital module	Digital input 16 x 24VDC	MAT-LC-DIT-16
DQT-08 digital module	Digital output 8 x 24VDC	MAT-LC-DQT-08
DQT-16 digital module	Digital output 16 x 24VDC	MAT-LC-DQT-16
DQR-08 digital module	Digital output 8 x Relay	MAT-LC-DQR-08
DQR-16 digital module	Digital output 16 x Relay	MAT-LC-DQR-16

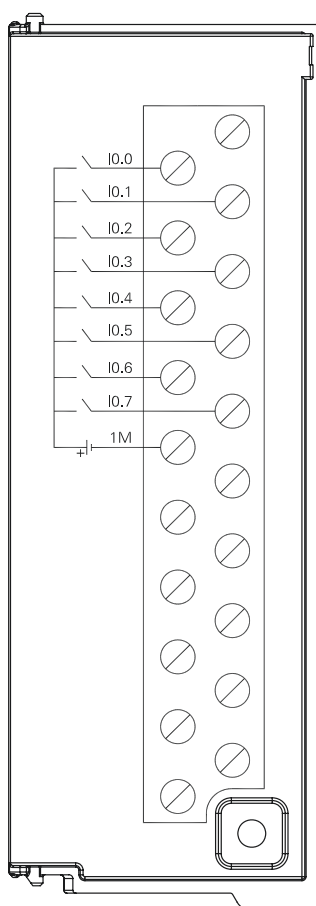
Digital input module

Table C-8 Specification of digital input module

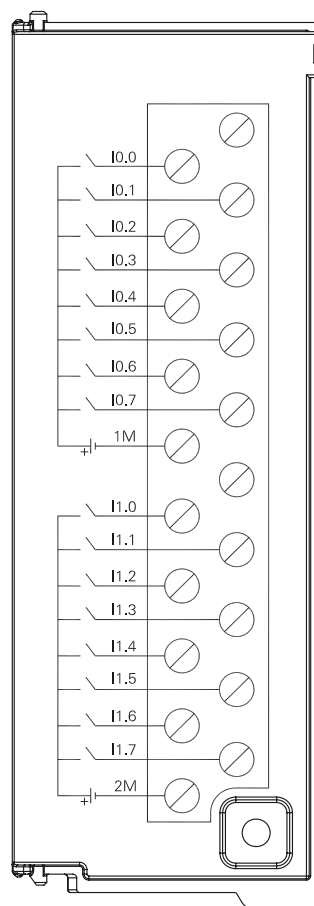
Item		DIT-08	DIT-16
Dimensions (W×H×D)		34×115×100 mm	
Input points		8	16
Current consumption	24V DC	4mA/channel	4mA/ channel
	+5V Bus	60mA	80mA
Input type		Drain type / source type (IEC class 1 drain type)	
Rated input voltage		24V DC, 6mA	
Input voltage range		20.4~28.8V DC	
Surge voltage		35V DC, duration: 0.5s	
Logic 1(Min)		15V DC, 2.5mA, Flip level: 10.5V DC±15%	
Logic 0(Max)		5V DC, 1mA	
Allowed leakage current of no.2 connecting wire proximity switch sensor(BERO) (Max)		1mA	
Input filtering		Configurable, support 0.2ms、0.4ms、0.8ms、1.6ms、3.2ms、6.4ms (default)、12.8ms	
Input frequency		1.5KHz, Duty cycle 50%	
Input impedance		6.6KΩ	
Isolation		500V AC, duration: 1min	
Isolation points of each group		8	8
Simultaneous ON points		8	16
Cable length	shield	500m	
	Unshielded	300m	

Wiring specification

◆ DIT-08 Wiring diagram



◆ DIT-16 Wiring diagram



Digital output module

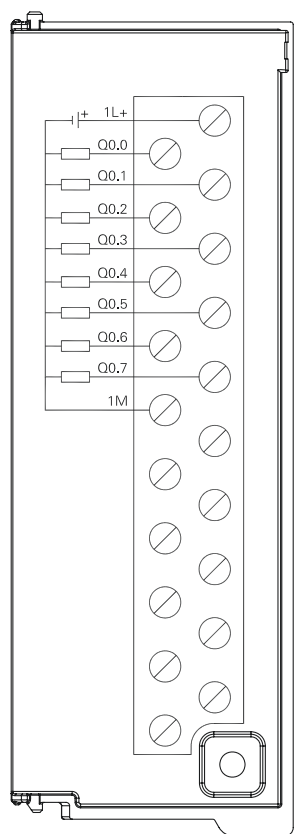
Table C-9 Specification of digital output module

Item	DQT-08	DQT-16	DQR-08	DQR-16
Dimensions (W×H×D)	34×115×100 mm			
Output points	8	16	8	16
Current consumption				
24V DC	50mA	95mA	64mA	130mA
+5V bus	70mA	120mA	45mA	60mA
Output type	Solid -MOSFET, source type			Relay-dry contact

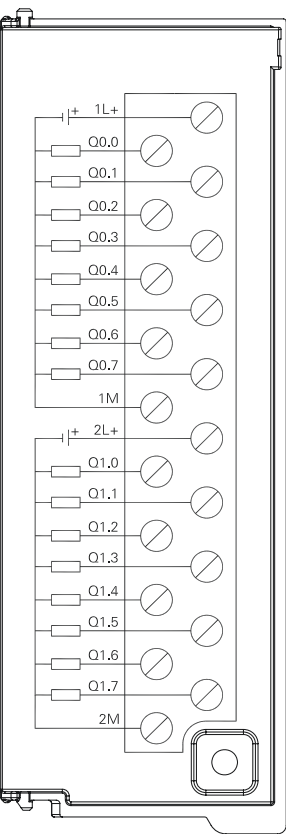
Rated output voltage	24V DC			DC: 24V, AC: 110V/220V
Output voltage range	20.4~28.8V DC			DC: 5~30V, AC: 5~250V
Logic 1(Min)	20V DC			-
Logic 0(Max)	0.1V DC, 10KΩ load			-
Output current (Max)	0.5A			2A
Current at each common terminal	Max 4A			Max 16A
Output leakage current (maximum)	15uA			-
Surge current	8A, 100ms			5A, 4s@10% duty cycle
Lamp load	5W			DC: 30W /AC: 200W
Contact resistance	0.3Ω, Max 0.6Ω			Max of new device 0.2Ω
Output delay	Off to on (Max): 50us On to off (Max): 200us			Max 10ms
Output frequency (Max)	1KHz			Resistive load: 10Hz Lamp load: 1Hz Inductive load: 0.5Hz
Mechanical life (without load)	-			10, 000, 000
Contact life (rated load)	-			100, 000
Isolation Field to logic Coil to contact	500V AC, duration: 1min -			500V AC, duration: 1min 1500V AC, duration: 1min
Isolation points of each group	8	8	8	8
Simultaneous ON points	8	16	8	16
Two outputs in parallel	Support parallel connection of two outputs in the same group			
Cable length	Shield	500m		
	Unshielded	150m		

Wiring specification

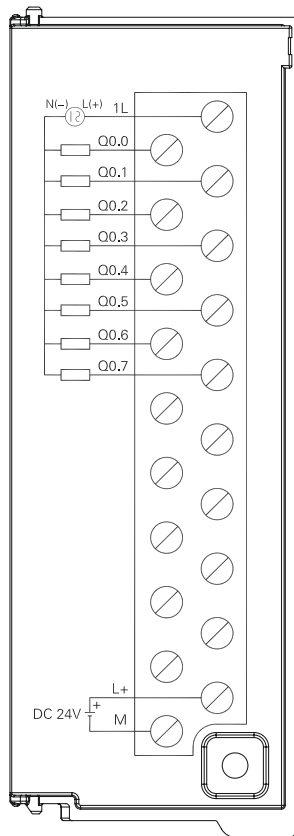
◆ DQT-08 Wiring diagram



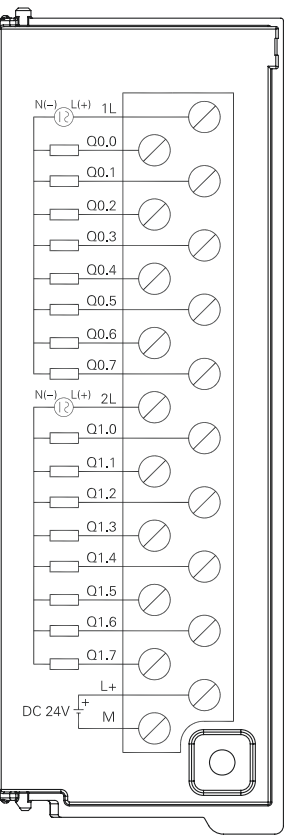
◆ DQT-16 Wiring diagram



◆ DQR-08 Wiring diagram



◆ DQR-16 Wiring diagram



C.3 Analog module

Table C-10 Basic properties of analog module

Name	Specification	Order No.
AIS-04 Analog module	Analog voltage and current input, 4AI x 12bit	MAT-LC-AIS-04
AQS-04 Analog module	Analog voltage and current output, 4AQ x 12bit	MAT-LC-AQS-04

Table C-11 General properties of analog module

Item	AIS-04		AQS-04	
Physical property				
Dimensions (W×H×D)	34×115×100 mm			
Power characteristics				
Rated input voltage	24V DC			
Input voltage range	20.4V~28.8V DC			
Input current	65mA	110mA		
Polarity reverse connection protection	Yes			
Bus supply voltage	+5V DC			
Bus supply current	50mA	40mA		
LED Indicator characteristics				
24VPower indicator	ON: With 24VDC power supply, OFF: without 24VDC power supply			
SF Indicator	ON: module fault, OFF: normal Flash: input current signal over limit alarm (only for: 4-20mA)			

C-12 Functional characteristics of analog module

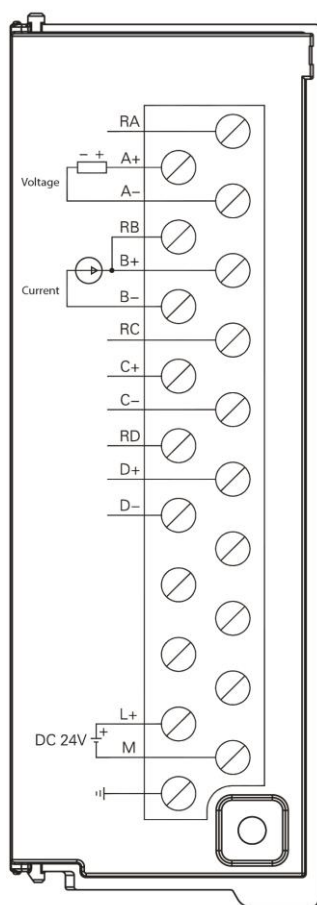
Function category	Function item	Description
Extension	Extension	Provides bus extension function.
Isolation	Signal isolation	Isolation between field and bus.
Protection	Power protection	The power supply terminal provides reverse connection protection function and surge absorption function
Filtering	Filtering	The combination of hardware filtering and software filtering is adopted
Power supply	Power supply	The module adopts 24V DC power supply.

Analog input module

Item	AIS-04
Input type	Voltage or current (differential input)
Input points	4
Input scale Voltage Current	Unipolar: 0~5V, 0~10V, bipolar: $\pm 2.5V$, $\pm 5V$ 0~20mA, 4~20mA
Max input voltage	30V DC
Max input current	40mA
Input impedance Voltage Current	$\geq 2M\Omega$ 250 Ω
Data format	Voltage: 0~+32000 (unipolar), -32000~+32000 (bipolar) Current: 0~+32000
Input step response	4-channel 5ms (max)
Module update frequency (all channels)	4-channel supports 200Hz、100Hz、50Hz、20Hz、10Hz configuration Default: 50Hz for all channels
Common mode suppression	>40dB
Channel crosstalk	>60dB
Common mode voltage	-12V $\leq$ Signal voltage + Common mode voltage $\leq$ +12V
Resolution	Unipolar: 12bit Bipolar: 11bit+ sign bit
Measuring principle	Successive approximation
Measurement error	0.5% (Max)
Disconnection detection(only for 4~20mA)	Disconnection calibration: -32768, 32767 Two values (optional)
Isolation Field to logic 24VDC to logic	500V AC 500V AC
Diagnosis Over negative range Over positive range Without power	Unipolar: 0 Bipolar: -32768 Unipolar: 32760 Bipolar: 32752 32736

Wiring Specification

◆ AIS-04 Wiring diagram



C.5 High speed counting module

Table C-20 Intruduction

Name	Specifications	Part No.
HSC-02	2-ch differential/single-ended signal input	MAT-LC-HSC-02

Table C-21 HSC-02 specifications

Dimensions		
W×H×D		34×115×100 mm
Power supply		
Bus supply voltage		+5V DC
Bus supply current		130mA
LED indicator light		
Indicator light		ON: Exists input signal OFF: No input signal
Connection of sensor		
Number of input channels		2
Signal type	Differential input	Signal voltage: 5VDC Max input frequency: 2MHz
	Single-ende	Signal voltage: 24VDC

	d input	Max input frequency: 500KHz Allowable range of signal duty cycle: 40%-60%
Max protection voltage of signal input		30VDC
Input filter		Configurable, 125KHz/250KHz/500KHz/1MHz/2MHz
Quadrature encoding		1、2、4 frequency multiplication
Counter format		32 Bit
Counter reset		Yes, Zsignal
Counter capture		Yes, Zsignal
Multi - counter synchronous counting		Yes, INTsignal
INTsignal voltage		24VDC
INT signal max input frequency		500KHz
INTsignal input filter		Configurable, 125KHz/250KHz/500KHz
Optoelectronic isolation		500VAC, 1min

Wiring diagram

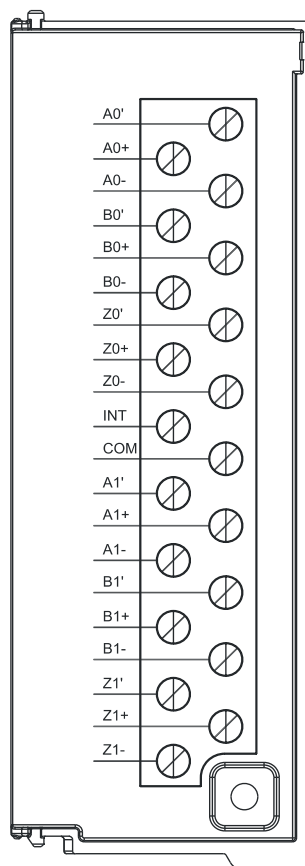


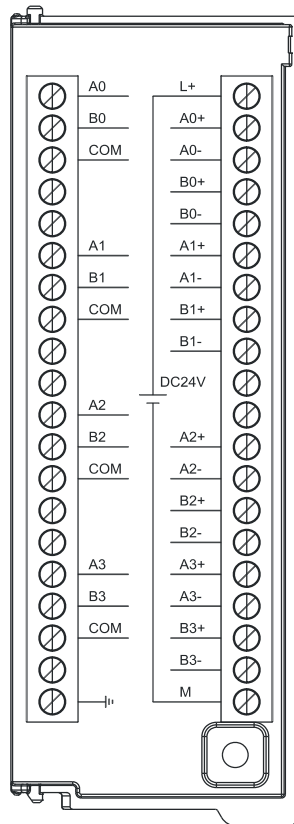
Table C-22 Introduction

Name	Specifications	Part No.
HSP-04	4-ch differential/single-ended signal output	MAT-LC-HSP-04

Table C-23 HSP-04 specifications

Dimensions		
W×H×D	34×115×101.6 mm	
Power supply		
Rated input voltage	24V DC	
Input voltage range	20.4V~28.8V DC	
Input current	100mA	
Reverse polarity protection	Yes	
Bus power voltage	+5V DC	
Bus power current	100mA	
LED indicator light		
Indicator light	ON: Exists input signal, OFF: No input signal	
Output		
Number of output channels	4	
Output type	Differential signal	Single-ended（NPN）signal
Max output frequency	4MHz	500KHz
Duty cycle of output signal	-	50%
Rated output voltage	5VDC	5~24VDC
Output voltage range	0~5.5VDC	5~28.8VDC
Output signal logic “0”	3.8V（min）	0.5V（max）
Output signal logic “1”	0.3V（max）	Vcc~0.5V（min）
Surge current	8A, sustain100ms	
Current per point（max）	20mA	20mA
Max current at each common terminal	No	160mA
Leak current（max）	10 μ A	
Isolation	500VAC, 1min	

Wiring diagram



C.7 EtherCAT slave

Table C-29 EtherCATslave introduction

Name	Specifications	Part No.
EtherCAT slave	Supports max 8 expansion modules, only digital module, analog module and temperature module can be connected	MAT-LC-ETC-00S1

Please visit the official website and download the " MAT-LC-CXX Series EtherCAT Slave Module User Manual" at <https://www.smbtechnics.com>.

Table C-30 Specifications

Dimensions	
W×H×D	34×115×100 mm
TDP	2.5W
Power supply	
Rated input voltage	24V DC
Input voltage range	20.4V~28.8V DC
Input current	0.8A
Reverse polarity protection	Yes
Bus power voltage	+5V DC

Bus power current	1.6A
LED indicator light	
24V power light (Green)	ON: 24VDC power supply is normal, OFF: No 24VDC power supply
SF light (Red)	ON: EtherCAT slave module failure, OFF: No error
BF light (Red)	ON: The extension bus is faulty. Off: No error
LINK light (Green) (Slave status indicator)	ON: Normal;Blinking: pre-operation (2), safe operation (4) (see Note 1);OFF: No connection (0), initialization (1)
RJ45 light(Green)	On: Connects to another EtherCAT interface Off: No EtherCAT interface is connected Blinking: The EtherCAT interface is communicating with the EtherCAT interface

Note 1: When the extension bus of the slave has no module of output type, the communication between the slave and the master is disconnected, and the slave will not switch to safe operation.

Table C-31 Functions

Type	Functions	Description
Hardware configuration	Codesys or other third party configuration	The added ETC-00 module supports 8 slot expansion
Extension	I/O Extension	Supports max 8 expansion modules, only digital module, analog module and temperature module can be connected
Communication	Bus interface	Provides expansion module interface, supports MAT-LC-CXX PLC custom 55MHZ bus protocol
	EtherCAT interface	Supports CANopen over EtherCAT (CoE)
Isolation	Power isolation	The external power supply is isolated from the system power supply
Protection	Power protection	The power supply provides reverse connection protection and surge absorption function

Table C-32 Communication port specifications

EtherCAT communication	
Interface	1double RJ45 port
Baud rate	100Mbps
Protocol type	CANopen over EtherCAT (CoE)
	Supports PDO
	Supports SDO
	Supports EtherCAT State machine instructions
	Supports the third-party EtherCAT master
The communication distance between	100m (100BASE-TX)

slave is the longest	
Isolation	Isolation of communication interface

The EtherCAT communication port uses shielded network cables as communication cables. The available network cables range from 22AWG to 25AWG, and their specifications are as follows. The resistance value is the DC resistance value of a single wire. 24AWG is recommended.

Table C-33 Communication cable specifications

AWG	Outside dia mm	Outside dia inch	Section area mm ²	Resistance value W/km
22	0.643	0.0253	0.3247	54.3
23	0.574	0.0226	0.2588	48.5
24	0.511	0.0201	0.2047	89.4
25	0.44	0.0179	0.1624	79.6

It is recommended to use a super shielded crystal connector, as shown in the following picture:



Interface diagram

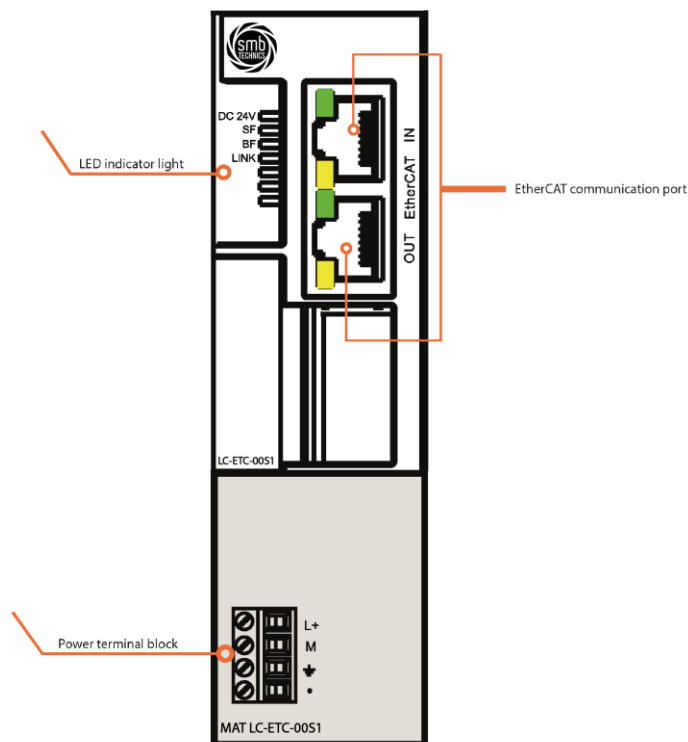
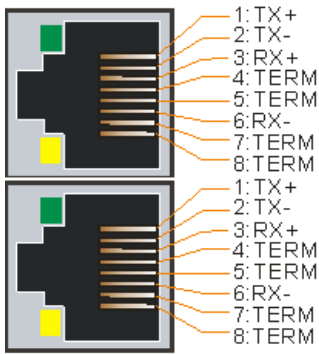


Table C-34 Power interface

4-bit detachable terminal	Number	Signal	Definition
L+	1	L+	24V power +
M	2	M	24V power -
$\perp$	3	$\perp$	GND
•	4	--	--

Table C-35 Double RJ45 interface

Double RJ45 interface	Number	Signal	Definition
	1	TX+	Positive end of data transmission
	2	TX-	Negative end of data transmission
	3	RX+	Positive end of data reception
	4	--	--
	5	--	--
	6	RX-	Negative end of data reception
	7	--	--
	8	--	--
	Connector housing	PE	Housing GND

D IO Module channel configuration

D.1 Digital input module channel configuration

The configuration parameters of each group (8 channels) are in the following format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Retian	Channel 4-7 Filtering time			Retian	Channel 0-3 Filtering time		

Filter time:

0: 0.20ms

1: 0.40ms

2: 0.80ms

- 3: 1.60ms
- 5: 3.20ms
- 6: 6.40ms (default)
- 7: 12.8ms

Default value for channel control bytes: 16#66

D.2 Digital output module channel configuration

The configuration parameters of each group (8 channels) are in the following format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Retain	Retain	Retain	Retain	Retain	Retain	Retain	Retain

Default value for channel control bytes: 16#00

D.3 AI module channel configuration

Analog input - Group Format of the parameter configuration

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sampling cycle			Input type and range				
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Retain	Retain	Retain	Retain	Retain	Retain	Retain	

Input type and range:

Bit4 is used to distinguish unipolarity from bipolarity. Bit4 is 1, indicating bipolarity. 0 indicates unipolarity. The specific code is as follows:

Module type	Input type	Input range	Range code (BIT4~0)
AI module	Voltage	0~5V	00000
		0~10V (default)	00001
		±2.5V	10000
		±5V	10001
	Current	0~20mA (default)	00010
		4~20mA	00011

Sampling cycle:

Module type	Update frequency (sampling cycle)	Sampling cycle code (BIT7~5)
AI module 4-ch	200 Hz	000
	100 Hz	001
	50 Hz (default)	010
	20 Hz	011
	10 Hz	100

Channel default: 16#0041 (Voltage 0~10V, 4 channel, 50Hz)

D.4 Analog output module channel configuration

Analog output - Group type configuration format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Retain		Retain	Voltage/C urrent	Range (including polarity)			
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Retain							

Type:

0: Voltage (default)

1: Current

Range:

0: $\pm 10V$ (Voltage)

1: 0~20mA (Current)

2: 4~20mA (Current)

default: $\pm 10V$ (Voltage) / 0~20mA (Current)

Channel default: 16#0000

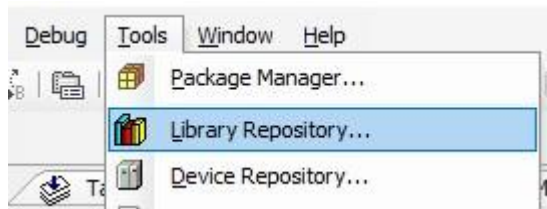
E SMB_MODBUS library

E.1 Install SMB_Modbus library file

To enable Modbus communication, install the SMB_MODBUS library file (download the library file from the SMB website: <https://www.smbtechnics.com>) and run the "Library" instruction to use the library instruction.

1) Install the library

Start the CODESYS software, select the menu item "Tools" → "Library Repository" :

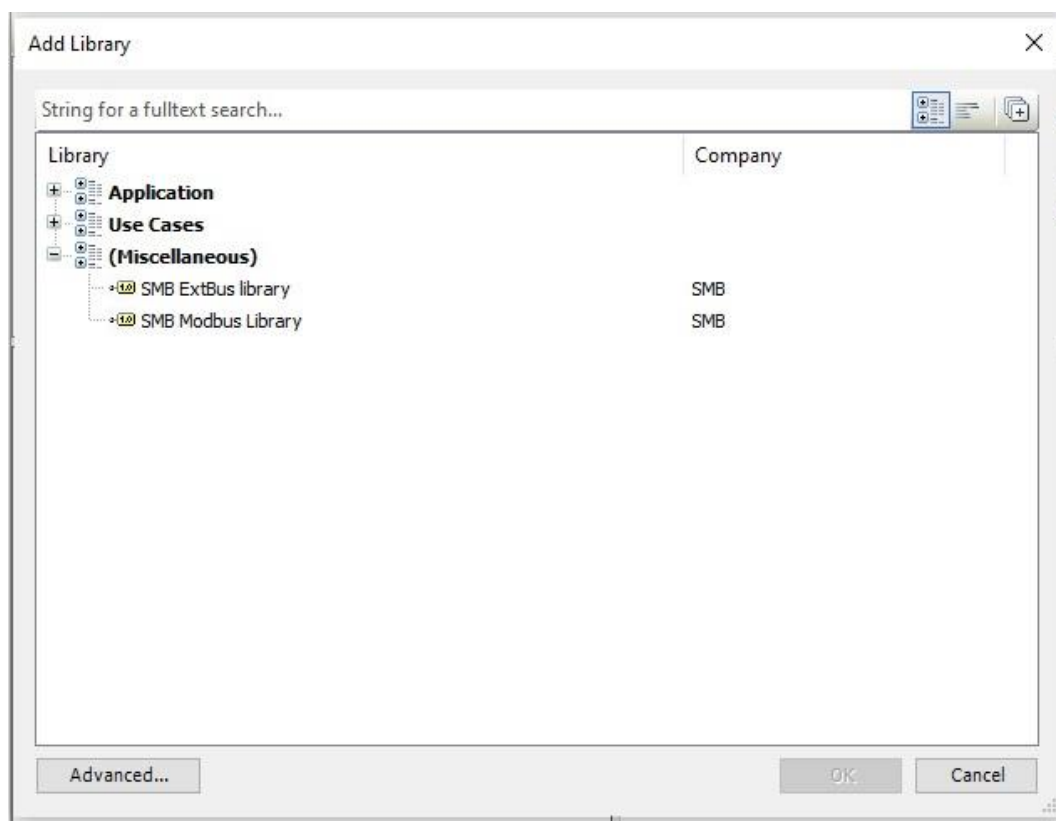


In the pop-up window, browse to the folder where SMB_modbus. library is located, select "SMB_Modbus_V1.1.library ", and click "Open" button. The SMB_MODBUS library is installed successfully.

2) Add Library

If you want to use the SMB_MODBUS library in the current project, you need to Add a library. The detailed operation steps are as follows:

Expand "Device" → "PLC" → "Application" in the Device view, double-click to open "Library Manager", then click "Add Library" button in the library manager dialog box, and the following "Add Library" dialog box will pop up:



Expand "Misc" in the above dialog box, select "SMB Modbus Library" in the sub-list and click **"Confirm"** button to add SMB_MODBUS Library to the current project.

E.2 Modbus_RTU library

1) Modbus_RTU master MBUS_CTRL

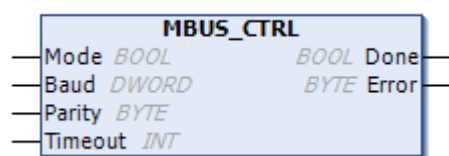


Table E-1 MBUS_CTRL instruction

Parameter	Description	Data type	Range
Mode	Enable switch (0:OFF 1:ON)	BOOL	
Baud	Baud rate	DWORD	1200 、 2400 、 4800 、 9600 19200、 38400、 57600、 115200
Parity	Set up calibration 0: no parity check 1: indicates odd check 2: parity check	BYTE	
Timeout	Timeout (read/write timeout if the slave does not reply data within the set time)	INT	
Done	Done bit 0: Instruction is being executed	BOOL	

	1: Instruction is completed		
Error	Error code	BYTE	

2) Master MBUS_MSG

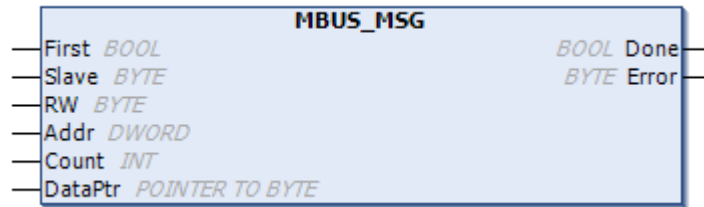


Table E-2 MBUS_MSG instruction

Parameter	Description	Data type	Initialization
First	Active bit (0: inactive 1: active). If not activated once, the instruction is executed once.	BOOL	0
Slave	Slave number	BYTE	0
RW	R/W read and write 0: Read 1: Write	BYTE	0
Addr	Read and write register address (eg. 40001,30001)	DWORD	0
Count	Number of read and write bytes (Value range: 0 to 120 characters)	INT	0
DataPtr	Read/write pointer (where data is read or written to a register)	POINTER TO BYTE	0
Done	Done bit 0: Instruction is being executed 1: Instruction is completed	BOOL	0
Error	Error code	BYTE	0

3) Slave MBUS_INIT

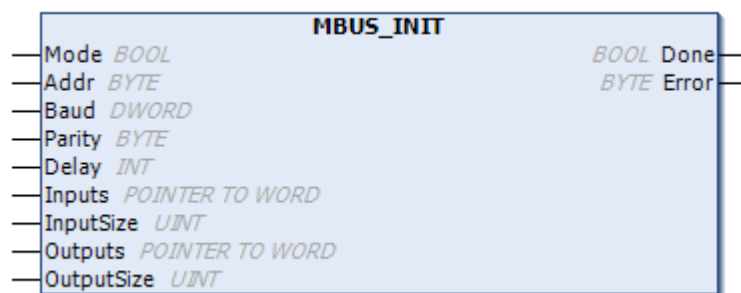


Table E-3 MBUS_INIT instruction

Parameter	Description	Data type	Initialization
Mode	Enable switch (0:OFF 1:ON)	BOOL	1
Addr	Address	BYTE	2
Baud	Baud rate (1200,2400,4800,9600,19200,38400,57600,115200)	DWORD	9600
Parity	Parity check 0: no parity check 1: indicates odd check 2: parity check	BYTE	0
Delay	Reply delay time (generally set to 0. If other communication parameters are correct but communication still fails, increase the value appropriately)	INT	0
Inputs	Pointer to Input Register	POINTER TO WORD	0
InputSize	Size of Input Register	UINT	0
Outputs	Pointer to Holding Register	POINTER TO WORD	0
OutputSize	Holding Register Size	UINT	0
Done	Done bit 0: Instruction is being executed 1: Instruction is completed	BOOL	0
Error	Error code	BYTE	0

4) Slave MBUS_SLAVE

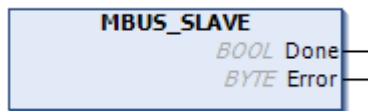


Table E-4 MBUS_SLAVE instruction

Parameter	Description	Data type	Range
Done	Done bit 0: Instruction is being executed 1: Instruction is completed	BOOL	
Error	Error code	BYTE	

1) Modbus_Tcp Master MBUS_TCP_REQ

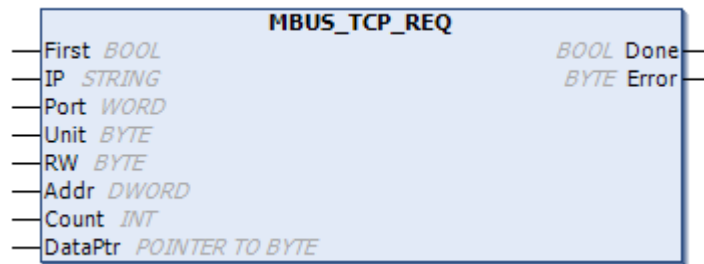


Table E-5 MBUS_TCO_REQ instruction

Parameter	Description	Data type	Initialization
First	The activation bit (0: inactive 1: active) activated once and the instruction executed once.	BOOL	0
IP	Salve ipaddress, eg. '192.168.0.X'	STRING	192.168.0.X
Port	Slave port number, eg. 502	WORD	502
Unit	Slave unit number, eg. 0	BYTE	0
RW	Read/Write 0: Read 1: Write	BYTE	0
Addr	Read and write register address (eg. 40001, 30001)	DWORD	0
Count	Read and write bytes (value range 0-120 characters)	INT	0
DataPtr	Read/write pointer (where data is read or written to a register)	POINTER TO BYTE	0
Done	Done bit 0: Instruction is being executed 1: Instruction is completed	BOOL	0
Error	Error code	BYTE	0

2) Modbus_Tcp Slave MBUS_TCP_SLAVE

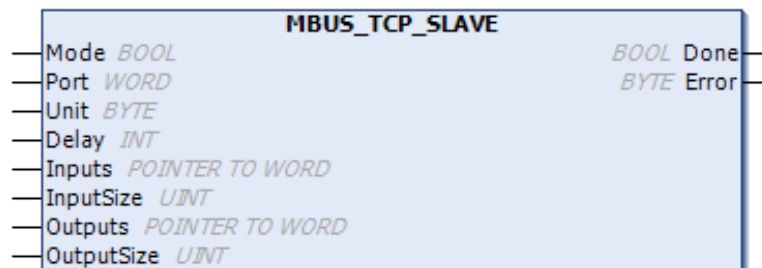


Table E-10 MBUS_TCP_SLAVE instruction

Parameter	Description	Data type	Initialization
Mode	Enable switch 0:OFF 1:ON	BOOL	1
Port	Salve port number, eg: 502	WORD	502
Unit	Slave unit number, eg: 0	BYTE	0
Delay	Reply delay time (generally set to 0. If other communication parameters are correct but communication still fails, increase the value appropriately)	INT	0
Inputs	Pointer to Input Register	POINTER TO WORD	0
InputSize	Input Register size	UINT	0
Outputs	Pointer to Holding Register	POINTER TO WORD	0
OutputSize	Holding Register size	UINT	0
Done	Done bit 0: Instruction is being executed 1: Instruction is completed	BOOL	0
Error	Error code	BYTE	0

F Operation instructions in CODESYS

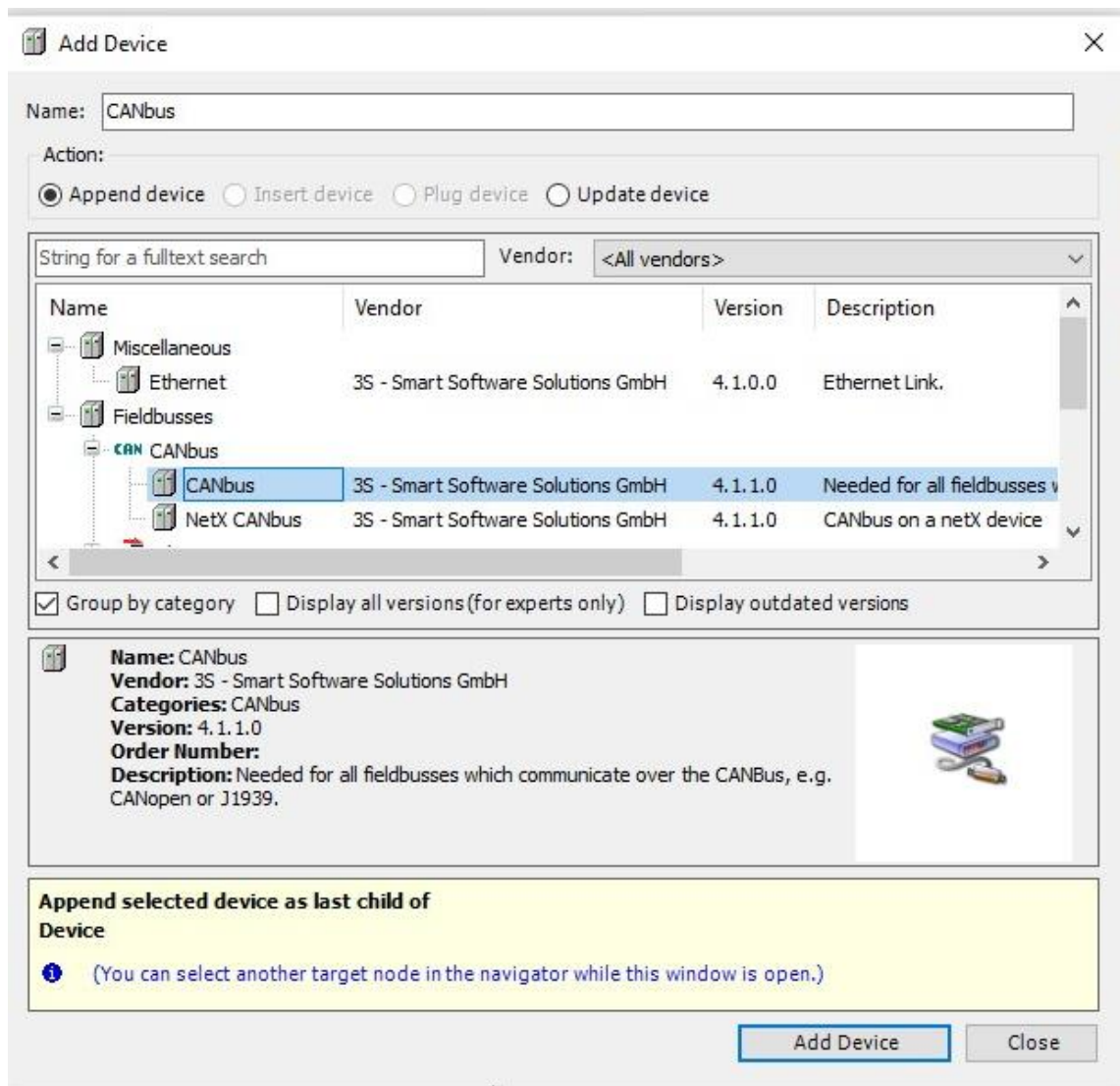
F.1 Add device

C06/C07 series controller can be used for CANopen, EtherCAT, Modbus_TCP/RTU communication on CODESYS. Modbus communication can be realized by programming Modbus instruction library (refer to "Appendix SMB_MODBUS Library Usage introduction"). For CANopen and EtherCAT communication, communication devices need to be added in the CODESYS device view to carry out specified communication. For details about how to add CANopen and EtherCAT communication devices, see the following description.

1、Add CANopen communication devices to CODESYS

1) Add CANbus

Right-click "Device(MAT LC-C06)" in the Device view and choose "Add Device" to add CANbus in the dialog box that pops up: select "< All Suppliers" for supplier and "CANbus" for field bus as CANbus.



2) Add CAN master

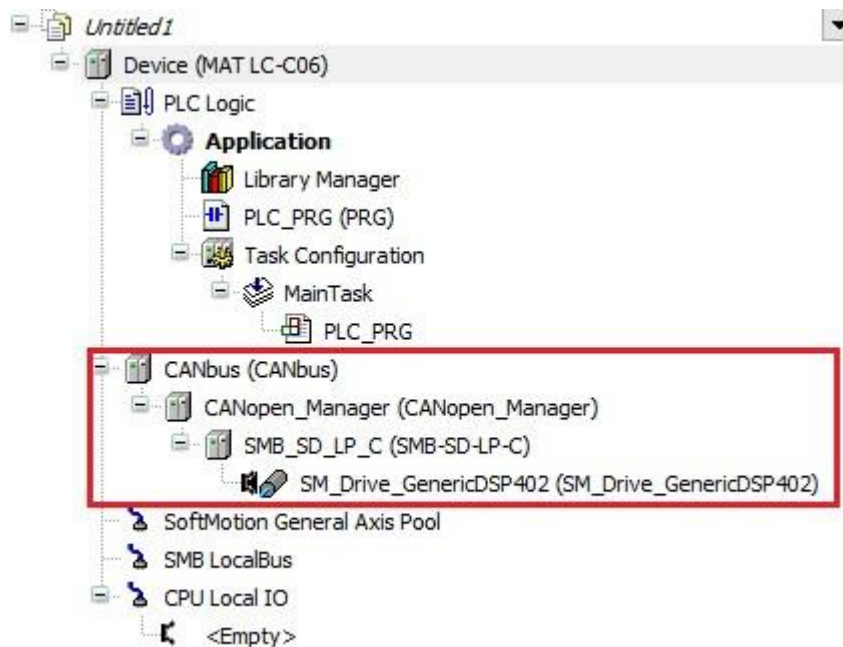
Select the successfully added CANBUS in the device view and right-click Add Device to add the CAN master station: Fieldbus → CANopen → CANopen Manager → CANopen_Manager in the Add Device dialog box.

3) Add or Scan CAN slave

Select the successfully added "CANopen_Manager" in the device view and right-click "Add Device" to add CAN slave station in the "Add Device" dialog box: Fieldbus → CANopen → Remote device → SMB_SD_LP_X; Or right-click "CANopen_Manager" and choose "Scan Device". The slave device connected with CANopen master can be displayed below the CAN master.

4) Add CANopen slave drive

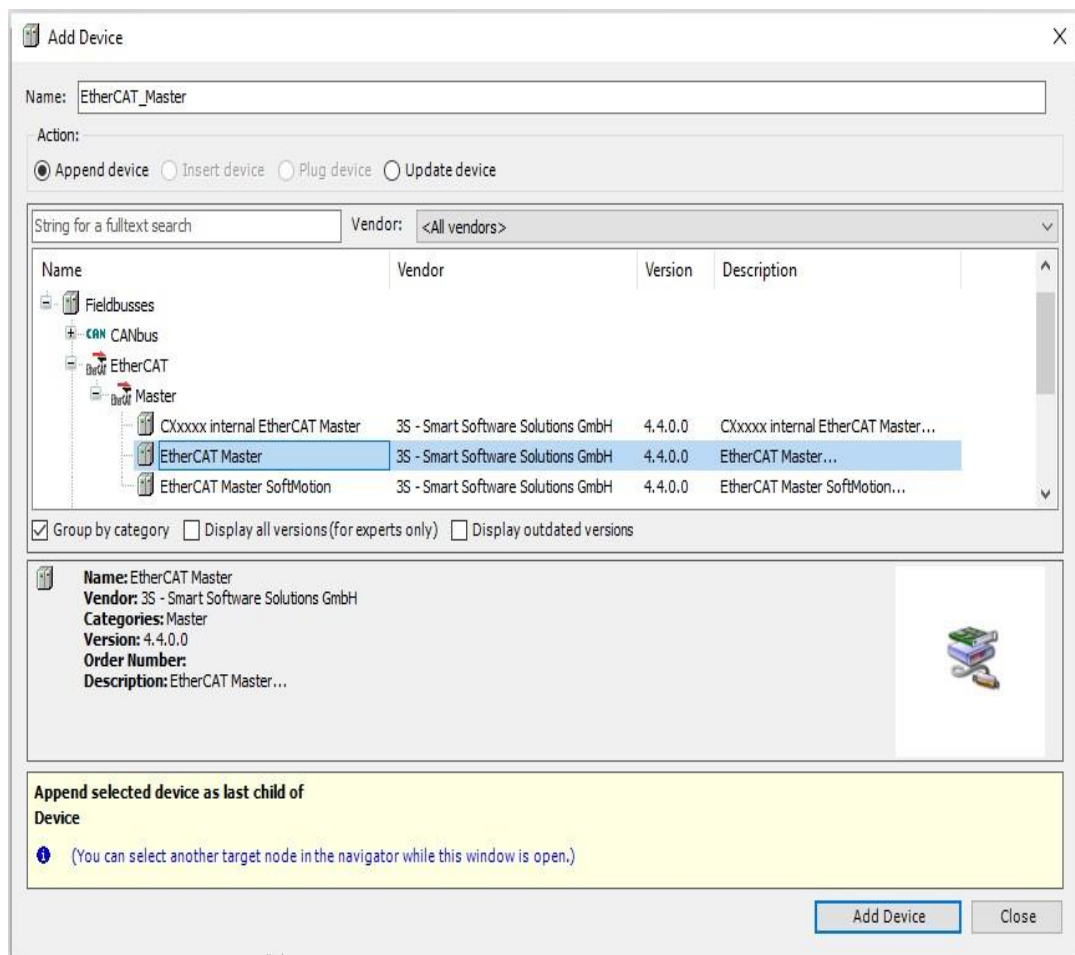
Select the CANopen slave station (SMB_SD_LP_X) in the device view and right click, and select "Add Softmotion - CIA402-Axis" in the pop-up menu, that is, the driver is added successfully.



2、Add EtherCAT communication devices to CODESYS

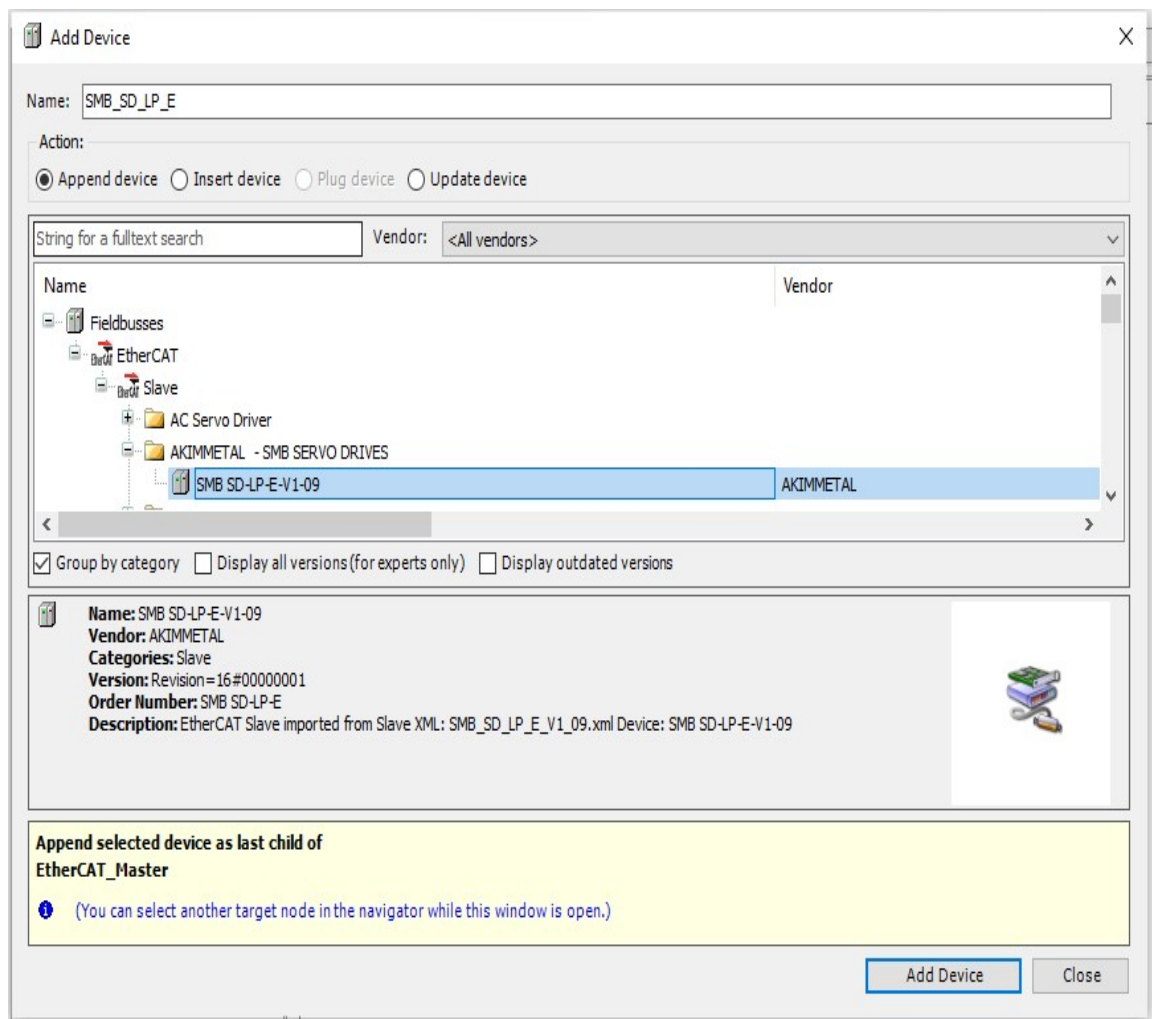
1) Add EtherCATmaster

Right-click "Device(MAT LC-C06)" in the Device view and choose "Add Device", then choose to add EtherCAT master station in the dialog box that pops up: Select "< All suppliers >" for suppliers and "EtherCAT" → "Master Station" → "EtherCAT Master" for fieldbus.



2) Add or Scan EtherCAT slave

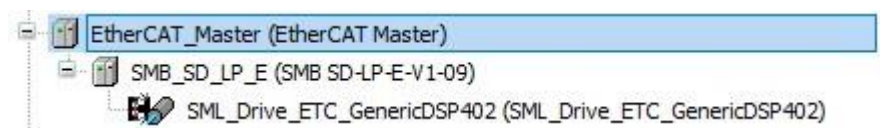
Select the EtherCAT master station that has been successfully added in the device view, right-click and choose "Add Device", then choose to add EtherCAT slave in the dialog box that pops up: Supplier select "< All suppliers >", fieldbus select "Ethercat" → "slave station" → "AKIMMETAL-SMB SERVO DRIVES" → "SMB_SD_LP_E_V1_XX "; Alternatively, right-click on the EtherCAT master (LC-C06) and select "Scan Device" to display the EtherCAT slave (SMB_SD_LP_X) connected to LC-C06 below the EtherCAT master (LC-C06).



Note: EtherCAT slave modules cannot be added by "scanning devices", they can only be added manually.

3) Add EtherCAT slave drive

Select the EtherCAT slave station (SMB_SD_LP_X) in the device view and right-click. In the pop-up menu, select "Add Softmotion - CIA402-Axis", that is, the driver has been added successfully.



F.2 Configure CANopen slave device

After CAN slave devices (SMB_SD_LP_X) are successfully added to CODESYS, you need to configure them. Double-click in the device view to open the CAN slave station, and then configure it. See below:

○, 1 Overview TAB: Set the node ID and select Enable Expert Settings.

General

Node ID: 1 SDO Channels (1/1 active)

☒ Enable Expert Settings ☐ Optional Device

☐ Enable Sync Producing ☐ No Initialisation ☐ Reset Node: [dropdown]

Nodeguarding

☐ Enable Nodeguarding ☒ Enable Heartbeat Producing

Guard Time (ms): 0 Producer Time (ms): 2000

Life Time Factor: 0 ✔ Heartbeat Consuming (1/1 active)

② “PDOs” tab

In the Receive PDOs (Master => Slave) and Transmit PDOs (Slave => Master) configuration boxes tick the parameter groups that need to be displayed in the Receive PDOs (Master => Slave) and Transmit PDOs (Slave => Master) tabs.

Receive PDOs (Master => Slave)

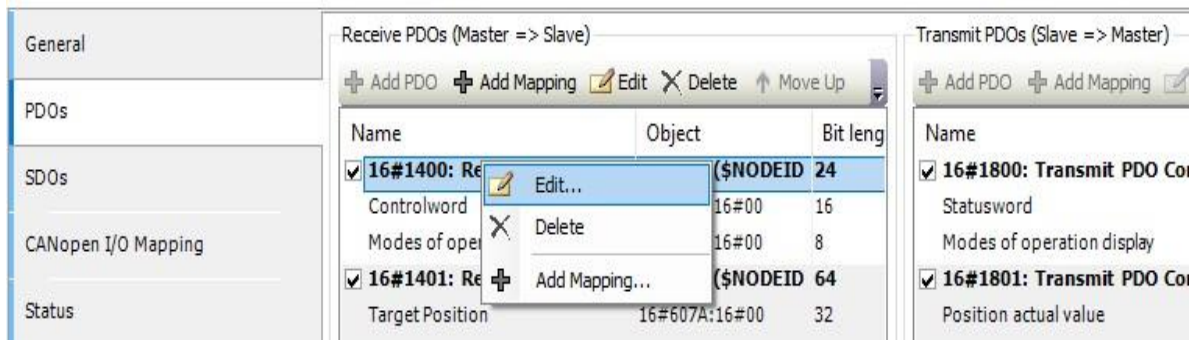
Name	Object	Bit leng
☒ 16#1400: Receive PDO Com	16#201 (\$NODEID 24)	
Controlword	16#6040:16#00	16
Modes of operation	16#6060:16#00	8
☒ 16#1401: Receive PDO Com	16#301 (\$NODEID 64)	
Target Position	16#607A:16#00	32
Target velocity	16#60FF:16#00	32
☒ 16#1402: Receive PDO Com	16#401 (\$NODEID 48)	
Target torque	16#6071:16#00	16
Profile velocity	16#6081:16#00	32
☒ 16#1403: Receive PDO Com	16#501 (\$NODEID 64)	
Profile acceleration	16#6083:16#00	32
Profile deceleration	16#6084:16#00	32

Transmit PDOs (Slave => Master)

Name	Object	Bit leng
☒ 16#1800: Transmit PDO Com	16#181 (\$NODEID 24)	
Statusword	16#6041:16#00	16
Modes of operation display	16#6061:16#00	8
☒ 16#1801: Transmit PDO Com	16#281 (\$NODEID 64)	
Position actual value	16#6064:16#00	32
Velocity actual value	16#606C:16#00	32
☒ 16#1802: Transmit PDO Com	16#381 (\$NODEID 64)	
Position demand value	16#6062:16#00	32
Position actual increment value	16#6063:16#00	32
☒ 16#1803: Transmit PDO Com	16#481 (\$NODEID 48)	
Following Error Actual Value	16#60F4:16#00	32
Torque actual value	16#6077:16#00	16

③ “Receive PDOs” tab

In this TAB, you can add parameter mapping for the selected parameter group. Right-click to select the PDO parameter group to be operated and select "Add Mapping". The specific operation is as shown in the figure below:



< Remarks > When SMB servo is used as CANopen slave station, please refer to the relevant servo operation manual for the description of each parameter. For example, if SMB_SD_LP_X servo is used as CANopen slave station, please refer to SMB_SD_LP_X Servo Operation Manual.④ “Transfer PDOs” tab

Same as the "Receive PDOs" operation, right-click and select the PDO parameter group to be operated, and then select "Add Mapping" to add send parameters for this group.

⑤ “Server data object” tab

The required SDO can be configured in the Service Data Object Configuration dialog, which defines the order in which objects need to be transferred, and the actions that will take place during an incomplete transfer.

⑥ “CANopen I/O map” tab

This dialog box is used to display the CANopen I/O map, where you can read/write the mapped data channel.

General	Find	Filter	Show all	
PDOs	Variable	Mapping	Channel	Address Type Unit Description
SDOs	+		Controlword	%QW8 UINT
CANopen I/O Mapping	+		Modes of operation	%QB18 SINT
Status	+		Target Position	%QD5 DINT
Information	+		Target velocity	%QD6 DINT
	+		Target torque	%QW14 INT
	+		Profile velocity	%QD8 UDINT
	+		Profile acceleration	%QD9 UDINT
	+		Profile deceleration	%QD10 UDINT
	+		Statusword	%IW8 UINT
	+		Modes of operation display	%IB18 SINT

⑦ “Status” tab

The dialog box displays the status information of the device (such as running and stopping) and the diagnosis information of the device.

⑧ “Information” tab

The dialog box displays information about the current device, such as the name, supplier, type, version, module number, and description.

F.3 Configure EtherCAT slave device

After the EtherCAT slave device (SMB_SD_LP_X) has been successfully added to CODESYS, it needs to be configured.

Configure (SMB_SD_LP_X)

- ① Overview TAB: Select Enable Expert Settings and DC for Synchronization for the distributed clock.

General Address Additional

AutoInc Address 0

EtherCAT Address 1001

☒ Enable Expert Settings

☐ Optional

▲ Distributed Clock

Select DC DC-Synchronous

☒ Enable 4000 Sync Unit Cycle (µs)

- ② “Expert Process Data” tab

"Expert process data". For details, see the following figure:

General Expert Process Data Process Data Startup Parameters EtherCAT I/O Mapping Status Information

Sync Manager:

SM	Size	Type
0	0	Mailbox Out
1	0	Mailbox In
2	12	Outputs
3	12	Inputs

PDO List:

Index	Size	Name	Flags	SM
16#1600	12.0	1st Receive PDO mapping		2
16#1601	6.0	2nd Receive PDO mapping		
16#1602	6.0	3rd Receive PDO mapping		
16#1603	12.0	4th Receive PDO mapping		
16#1A00	12.0	1st Transmit PDO mapping		3
16#1A01	6.0	2nd Transmit PDO mapping		
16#1A02	6.0	3rd Transmit PDO mapping		
16#1A03	8.0	4th Transmit PDO mapping		

PDO Assignment (16#1C12):

☒ 16#1600

☐ 16#1601

☐ 16#1602

☐ 16#1603

PDO Content (16#1600):

Index	Size	Offs	Name	Type
16#6040:00	2.0	0.0	Control word	UINT
16#607A:00	4.0	2.0	Target position	DINT
16#60FF:00	4.0	6.0	Target velocity	DINT
16#6060:00	1.0	10.0	Modes of operation	SINT
---	1.0	11.0	---	
		12.0		

For the description of each parameter, please refer to the relevant servo operation manual. For example, if the (SMB_SD_LP_X) servo is used as the EtherCAT slave station, please refer to the (SMB_SD_LP_X) Servo Operation Manual.

- ③ “Process data” tab

Displays the input/output process data of the slave, which is defined by the name, type, and index of the device description file.

General	Select the Outputs			Select the Inputs		
Expert Process Data	Name	Type	Index	Name	Type	Index
Process Data	☒ 16#1600 1st Receive PDO mapping			☒ 16#1A00 1st Transmit PDO map		
Startup Parameters	Control word	UINT	16#6040:00	Status word	UINT	16#6041:00
EtherCAT I/O Mapping	Target position	DINT	16#607A:00	Position actual value	DINT	16#6064:00
Status	Target velocity	DINT	16#60FF:00	Velocity actual value	DINT	16#606C:00
Information	Modes of operation	SINT	16#6060:00	Modes of operation display	SINT	16#6061:00
	---			---		
	☐ 16#1601 2nd Receive PDO mapping			☐ 16#1A01 2nd Transmit PDO map		
	Control word	UINT	16#6040:00	Status word	UINT	16#6041:00
	Target position	DINT	16#607A:00	Position actual value	DINT	16#6064:00
	☐ 16#1602 3rd Receive PDO mapping			☐ 16#1A02 3rd Transmit PDO map		
	Control word	UINT	16#6040:00	Status word	UINT	16#6041:00
	Target velocity	DINT	16#60FF:00	Velocity actual value	DINT	16#606C:00
	☐ 16#1603 4th Receive PDO mapping			☐ 16#1A03 4th Transmit PDO map		
	ProfileVelocity	DINT	16#6081:00	Status word	UINT	16#6041:00
	ProfileAcceleration	DINT	16#6083:00	Position actual value	DINT	16#6064:00
	ProfileDeceleration	DINT	16#6084:00	Torque actual value	INT	16#6077:00

The input (read) and output (write) parameter sets selected by the device can be used in the I/O mapping dialog as input and output to the PLC (project variables may be mapped).

To change the current selection, you must first deselect the current selection by clicking the box in front of it, and then you can set another group.

④ “Startup parameters” tab

Define specific parameters for the device that are passed by the SDO or IDN at system startup.

⑤ “EtherCAT I/O map” tab

This dialog is used to display the EtherCAT I/O mapping, and you can read/write the mapped parameters in this tab.

General	Find	Filter	Show all				
Expert Process Data	Variable	Mapping	Channel	Address	Type	Unit	Description
Process Data			Control word	%QW0	UINT		Control word
Startup Parameters			Target position	%QD1	DINT		Target position
EtherCAT I/O Mapping			Target velocity	%QD2	DINT		Target velocity
Status			Modes of operation	%QB12	SINT		Modes of operation
			Status word	%IW0	UINT		Status word
			Position actual value	%ID1	DINT		Position actual value
			Velocity actual value	%ID2	DINT		Velocity actual value
			Modes of operation display	%IB12	SINT		Modes of operation display

⑥ “Status” tab

The dialog box displays the status information of the device (such as running and stopping) and the diagnosis information of the device.

⑦ “Information” tab

The dialog box displays information about the current device, such as the name, supplier, type, version, module number, and description.

Configure ETC-00

① “DC for synchronization” .

General Address Additional

AutoInc Address 0

EtherCAT Address 1001

☒ Enable Expert Settings ☐ Optional

Distributed Clock

Select DC SM-Synchron

☐ Enable 4000 Sync Unit Cycle (µs)

Sync0:

☐ Enable Sync 0

☒ Sync Unit Cycle x 1 Cycle Time (µs)

☐ User Defined 0 Shift Time (µs)

Sync1:

☐ Enable Sync 1

☒ Sync Unit Cycle x 1 Cycle Time (µs)

☐ User Defined 0 Shift Time (µs)

EtherCAT

② “Expert Process Data” tab

"Expert Process Data" tab: You can add/delete PDO.

General Expert Process Data Process Data Startup Parameters EtherCAT I/O Mapping Status Information

Sync Manager:

SM	Size	Type
0	0	Mailbox Out
1	0	Mailbox In
2	12	Outputs
3	12	Inputs

PDO List:

Index	Size	Name	Flags	SM
16#1600	12.0	1st Receive PDO mapping		2
16#1601	6.0	2nd Receive PDO mapping		
16#1602	6.0	3rd Receive PDO mapping		
16#1603	12.0	4th Receive PDO mapping		
16#1A00	12.0	1st Transmit PDO mapping		3
16#1A01	6.0	2nd Transmit PDO mapping		
16#1A02	6.0	3rd Transmit PDO mapping		
16#1A03	8.0	4th Transmit PDO mapping		

PDO Assignment (16#1C12):

☒ 16#1600

☐ 16#1601

☐ 16#1602

☐ 16#1603

PDO Content (16#1600):

Index	Size	Offs	Name	Type
16#6040:00	2.0	0.0	Control word	UINT
16#607A:00	4.0	2.0	Target position	DINT
16#60FF:00	4.0	6.0	Target velocity	DINT
16#6060:00	1.0	10.0	Modes of operation	SINT
---	1.0	11.0	---	
		12.0		

③ “Process Data” tab

Displays the input/output process data of the slave, which is defined by the name, type, and index of the device description file.

Select the Outputs			Select the Inputs		
Name	Type	Index	Name	Type	Index
☒ 16#1601 DQ-16 Outputs			☒ 16#1A00 AI5-04 Input		
OutByte0	USINT	16#7100:01	AIW0	INT	16#6010:01
OutByte1	USINT	16#7100:02	AIW2	INT	16#6010:02
			AIW4	INT	16#6010:03
			AIW6	INT	16#6010:04
			Module State	USINT	16#8002:00
			Module Err Num	UDINT	16#8003:00
			☒ 16#1A01 DQ-16 Input		
			Module State	USINT	16#8102:00
			Module Err Num	UDINT	16#8103:00

The input (read) and output (write) parameter sets selected by the device can be used in the I/O mapping dialog as input and output to the PLC (project variables may be mapped).

Note: The process data of EtherCAT slave station is selected by default. Please do not change it, otherwise communication failure may occur.

④ “Startup parameters” tab

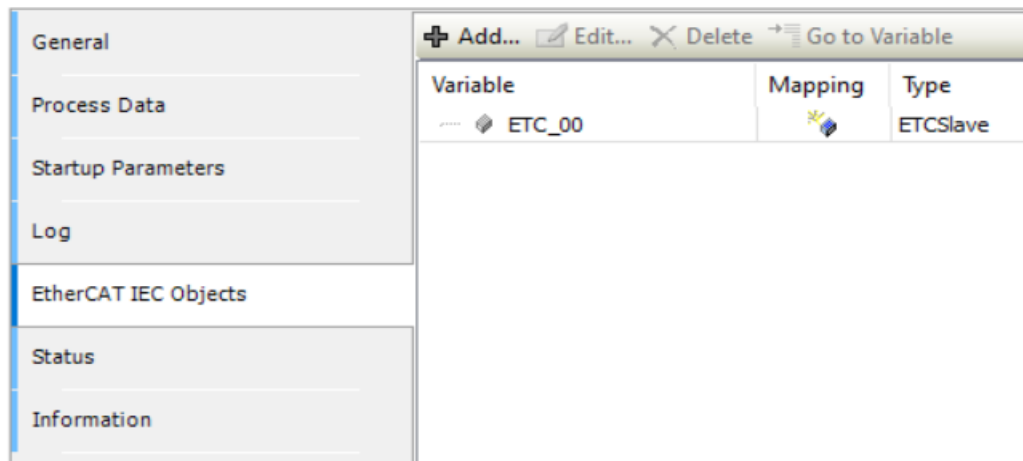
Define specific parameters for the device that are passed by the SDO or IDN at system startup.

Line	Index:Subindex	Name	Value	Bitlength	Abort if error	Jump to line if error	Next line	Comment
1	16#3000:16#00	Command_0	0	32	☐	☐	0	
2	16#8001:16#00	Command_0	1052672	32	☐	☐	0	Command_0
3	16#6011:16#01	Command_1	16449	16	☐	☐	0	Command_1
4	16#6011:16#02	Command_2	16449	16	☐	☐	0	Command_2
5	16#8101:16#00	Command_0	1081344	32	☐	☐	0	Command_0
6	16#7102:16#01	Command_1	255	8	☐	☐	0	Command_1
7	16#7102:16#02	Command_2	255	8	☐	☐	0	Command_2

Note: The startup parameters of EtherCAT slave station cannot be modified. If you need to modify the configuration of the control word of the expansion module, please double-click the corresponding expansion module in the menu tree, and download the configuration to the CPU after modifying the startup parameters, and the relevant configuration changes will take effect immediately.

⑤ “EtherCAT IEC Object” tab

This dialog box is used to display the EtherCAT IEC object



For modules attached under ETC-00 to read and write parameters, do the following: Debug the Module to which it is attached in the Module I/O Mapping TAB of the ETC-00 slave station (check Always in Bus Cycle Task). Take the Digital Output 16 Bits input Module as an example. Click DQ-16_16DQ (Digital Output 16 Bits) below ETC-00 and select Module I/O mapping so that the parameters can be read and written.



Startup Parameters		Find Filter Show all					
Information		Variable	Mapping	Channel	Address	Type	Description
Module I/O Mapping				OutByte0	%QB14	USINT	OutByte0
				Bit0	%QX14.0	BOOL	
				Bit1	%QX14.1	BOOL	
				Bit2	%QX14.2	BOOL	
				Bit3	%QX14.3	BOOL	
				Bit4	%QX14.4	BOOL	
				Bit5	%QX14.5	BOOL	
				Bit6	%QX14.6	BOOL	
				Bit7	%QX14.7	BOOL	
				OutByte1	%QB15	USINT	OutByte1
				Module State	%IB32	USINT	Module State
				Bit0	%IX32.0	BOOL	
				Bit1	%IX32.1	BOOL	
				Bit2	%IX32.2	BOOL	
				Bit3	%IX32.3	BOOL	
				Bit4	%IX32.4	BOOL	
				Bit5	%IX32.5	BOOL	
				Bit6	%IX32.6	BOOL	
				Bit7	%IX32.7	BOOL	
				Module Err Num	%ID9	UDINT	Module Err Num

Note: The address assigned to each extension module is not fixed. If you add/remove modules, the corresponding address assigned will change.

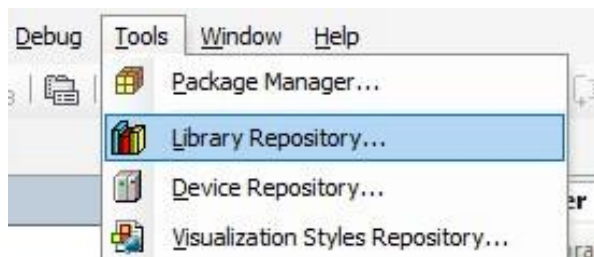
G ExtBus library

G.1 Install ExtBus library files

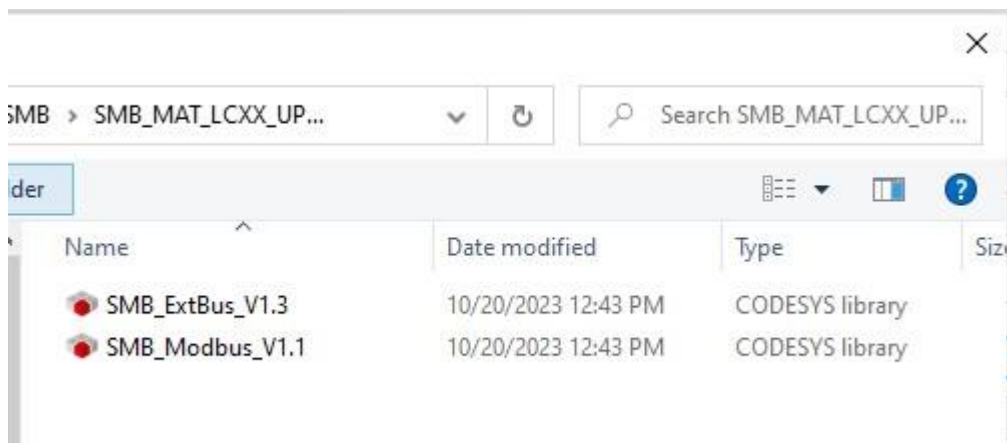
To use the high-speed counter provided by the CPU and the high-speed counter extension module, first install the ExtBus library file (download the library file from the SMB TECHNICS website: (<https://www.smbtechnics.com>), and then run the "Add Library" instruction to use the library.

1) Install library

To boot CODESYS, and select menue “Tools” → “Library Repository” :



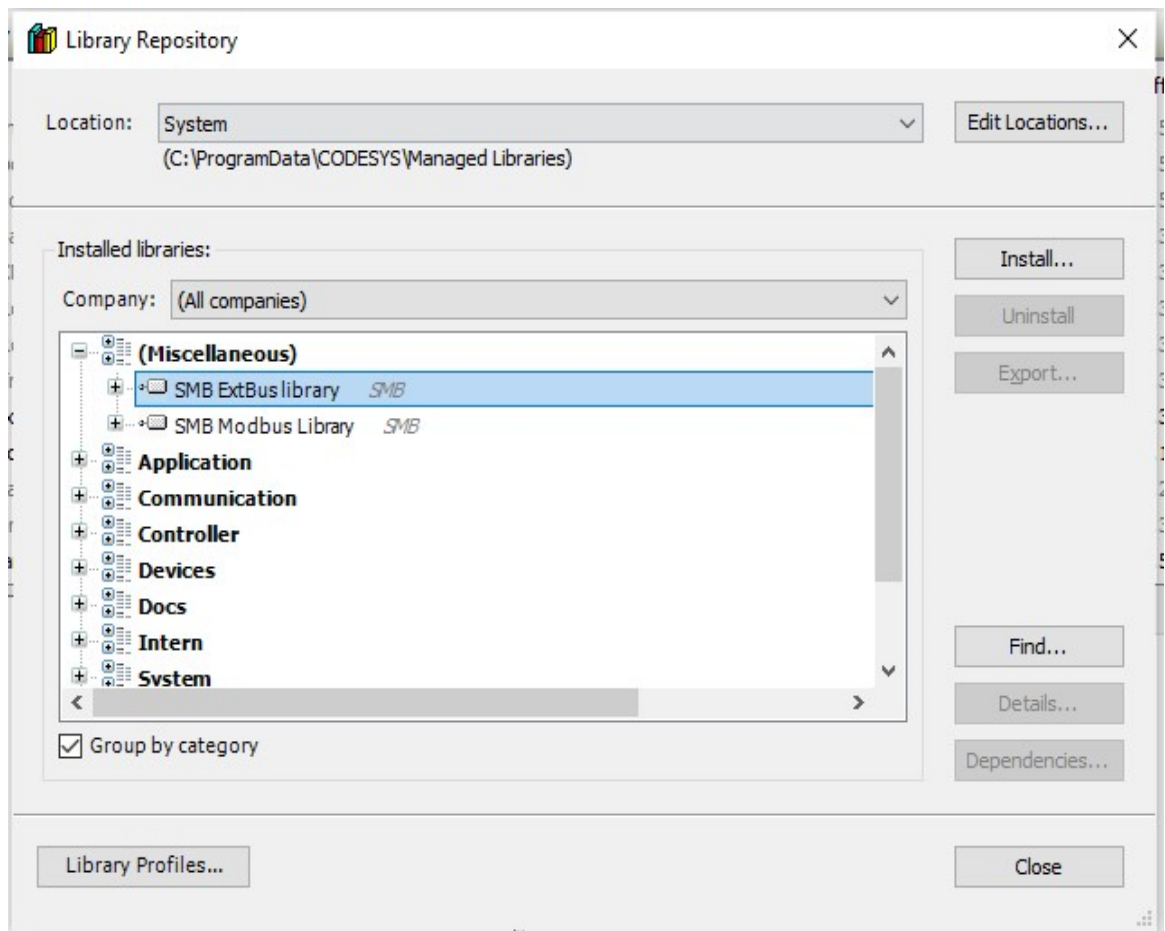
In the pop-up window, browse to the folder where " SMB_ExtBus_V1.3.library" is located, select " SMB_ExtBus_V1.3.library", and click the "Open" button. That means the SMB_ExtBus_V1.3 library has been installed successfully.



2) Add library

SMB_ExtBus_V1.3 library After the library file is installed successfully, you need to perform Add Library if you want to use the library in the current project. The detailed operation steps are as follows:

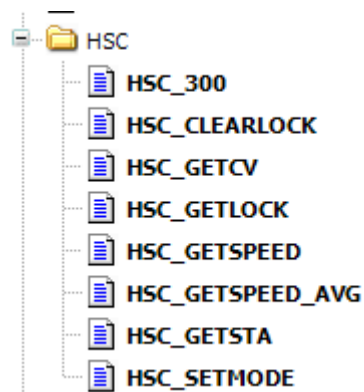
Expand "Device" → "PLC" → "Application" in the Device view, double-click to open "Library Manager", then click "Add Library" button in the library manager dialog box, and the following "Add Library" dialog box will pop up:



Expand "Miscellaneous" in the above dialog box, select "SMB ExtBus library" in the sublist and click "OK" button to add the ExtBus library to the current project.

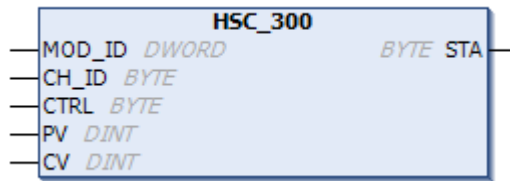
G.2 ExtBus library instruction description

The instruction library (ExtBus) supported by the high-speed counter, the description of each instruction refers to the following description.



1、Set the counter parameter instruction

Function: HSC_300



Parameter	I/O	Description	Type	Initialization	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
CTRL	IN	Control byte	BYTE	16#F9	See following table
PV	IN	Preset value	DINT	0	
CV	IN	Current value	DINT	0	
STA	OUT	Return status	BYTE	0	Module status byte 0: OK 5: Module parameter error 7: Module does not respond 8: Module link layer verification error.

Control byte (R/W)

7	6	5	4	3	2	1	0
Count enable	Current value update	Preset value update	Count direction update	Count direction	Quadrature count selection		Reset level

Reset level: 1-high-level reset, 0-low-level reset

Quadrature count options: 00-4x multiples, 01-2x multiples, 10-1x multiples

Counting direction: 0 - count down, 1 - count up

Counting direction update: 0 - no update, 1 - update

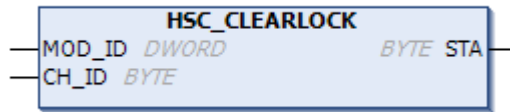
Default value update: 0-no update, 1-update

Current value update: 0 - no update, 1 - update

Counting enable: 0-disable, 1-enable

2. Clear the latched value

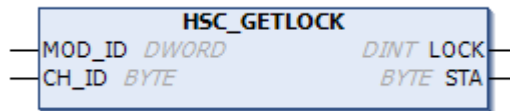
Function: HSC_CLEARLOCK



Parameter	I/O	Description	Type	Initialization	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 5: Module parameter error 7: Module does not respond 8: Module link layer error.

3. Obtain current latch value

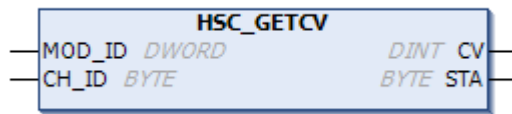
Function: HSC_GETLOCK



Parameter	I/O	Description	Type	Initialization	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
LOCK	OUT	Latch value	DINT	0	Latch value
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 2: invalid parameter 5: Incorrect module parameter 7: Module does not respond 8: Module link layer error.

3. Obtain current count value

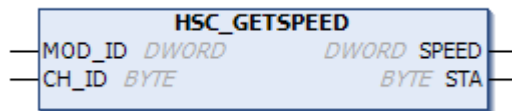
Function: HSC_GETCV



Parameter	I/O	Description	Type	Initialization	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
CV	OUT	Current count value	DINT	0	Current value
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 2: invalid parameter 5: Incorrect module parameter 7: Module does not respond 8: Module link layer error.

5. Obtain current counting speed

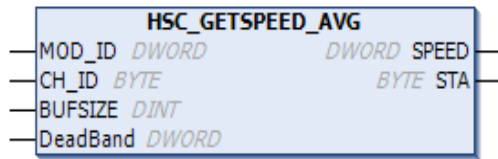
Function: HSC_GETSPEED



Parameter	I/O	Description	Type	Initialization	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
SPEED	OUT	Count speed	DWORD	0	Hz
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 2: invalid parameter 5: Incorrect module parameter 7: Module does not respond 8: Module link layer error.

5. Obtain current speed by average value

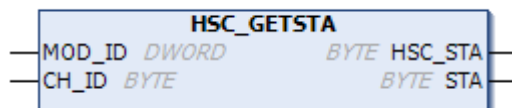
Function: HSC_GETSPEED-AVG



Parameter	I/O	Description	Type	Initiali zation	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
BUFSIZE	IN	Average value buffer	DI1, NT	16	Average buffer size is greater than 0 and less than 64
DeadBand	IN	Dead zone value	DWORD	20000	If the difference between the average value and the existing value is less than the dead zone value, the average value shall prevail, and the value greater than the dead zone average value is the existing value
SPEED	OUT	Count speed	DWORD	0	Hz
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 2: invalid parameter 5: Incorrect module parameter 7: Module does not respond 8: Module link layer error.

7. Obtain current count status

Function: HSC_GETSTA

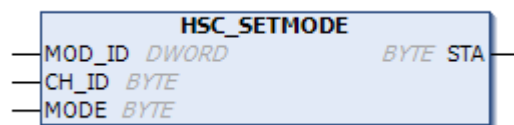


Parameter	I/O	Description	Type	Initiali zation	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration

CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)
HSC_STA	OUT	Count status	BYTE	0	Bit0~Bit3: current mode Bit4: reserved Bit5: HSC0 current count direction bit: 1 = count up Bit6=1: The current value is equal to the preset value bit Bit7=1: The current value is greater than the preset value bit
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 2: invalid parameter 5: Incorrect module parameter 7: Module does not respond 8: Module link layer error.

8. Set the counter mode

Function: HSC_SETMODE



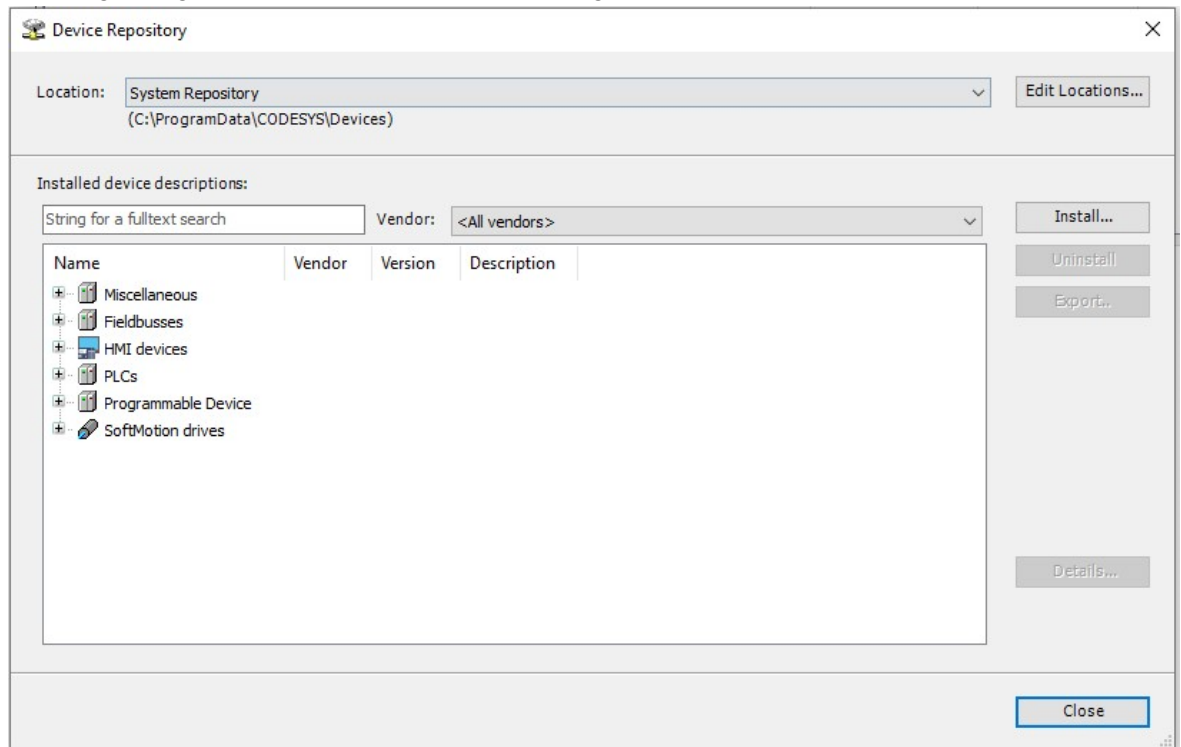
Parameter	I/O	Description	Type	Initialization	Remark
MOD_ID	IN	Module address	DWORD	0	PLC 16# 0a000605, Module Id of hardware configuration
CH_ID	IN	Channel address	BYTE	0	PLC local value (0-5) HSC module value (0, 1)

MODE	IN	Count mode	BYTE	0	Bit0~Bit3: HSC counting mode (this machine supports 0, 1, 3, 4, 6, 7, 9, 10; HSC-02 supports 0-11 mode) Mode 0~2: Single-phase counter with internal direction control. Modes 3~5: Single-phase counter with external direction. Modes 6~8: Two-phase counter with 2 clock inputs. Mode 9~11: A\B phase quadrature counter. Bit4: Z signal latch function, 0: latch, 1: not latch Bit5: Z signal clearing function, 0: clearing, 1: not clearing Bit6: reserved Bit7: Clear the latch value 0: Invalid, 1: Valid
STA	OUT	Module status byte	BYTE	0	Module status byte 0: OK 2: invalid parameter 5: Incorrect module parameter 7: Module does not respond 8: Module link layer error.

H Install the device description file in CODESYS

H.1 Install EtherCAT servo slave device description file

1. Open the CODESYS software, select the menu "Tools" → "Device Repository" to open the following dialog box, and click "Install" in the dialog box.

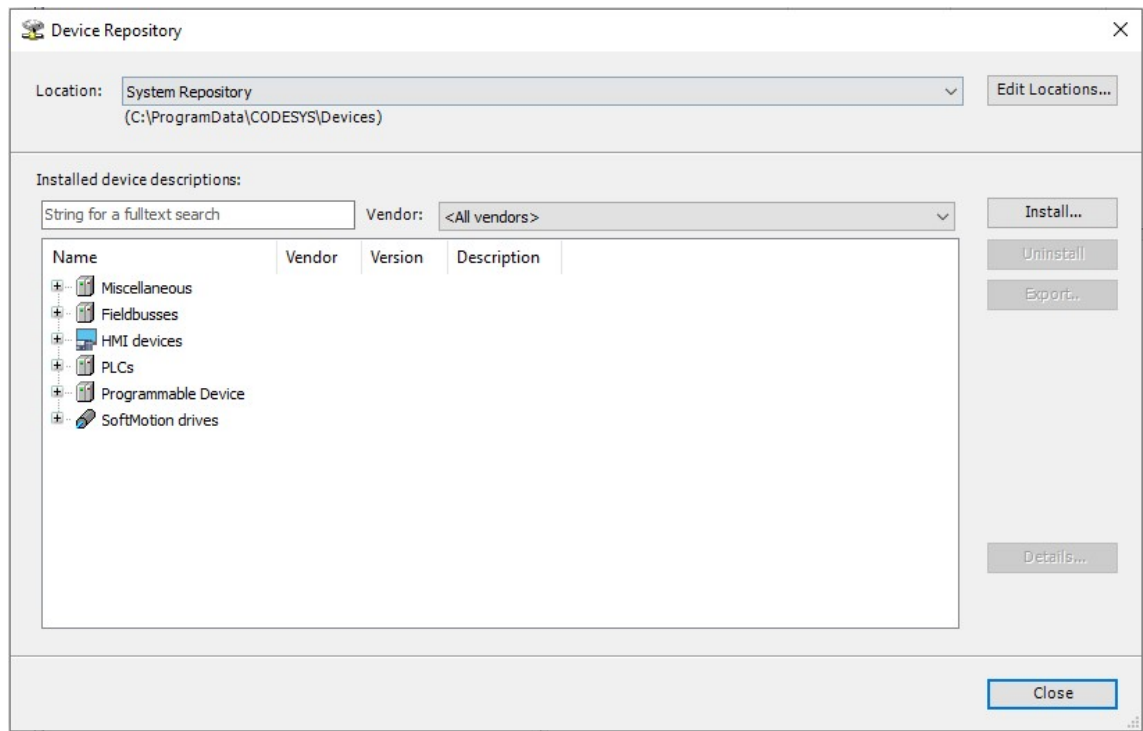


2. After performing the above operations, a dialog box will pop up as follows. Please select the file type "EtherCAT XML Device Description Configuration file (*.xml)", select the file "SMB_SD_LP_E_V1_xx.xml".

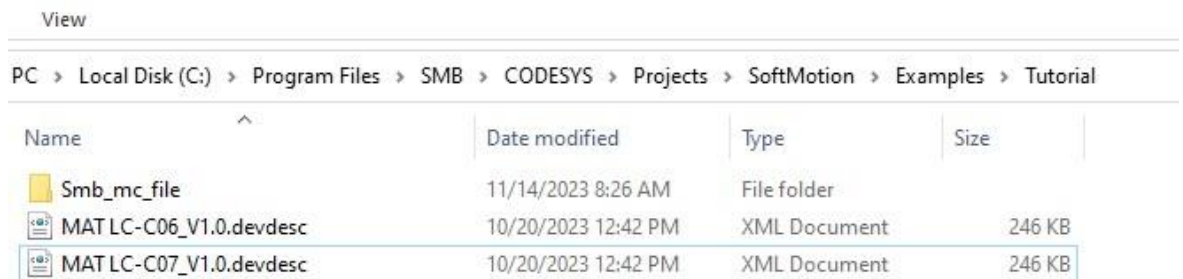
Name	Date modified	Type
SMB_SD_LP_E_V1_02	9/18/2022 5:29 PM	XML Document
SMB_SD_LP_E_V1_07	9/13/2022 4:14 PM	XML Document
SMB_SD_LP_E_V1_09	2/2/2023 4:13 PM	XML Document

H.2 Install PLC/402 axis device description file

1. Open the CODESYS software, select the menu item "Tools" → "Device Repository" to open the following dialog box, and click "Install" in the dialog box.

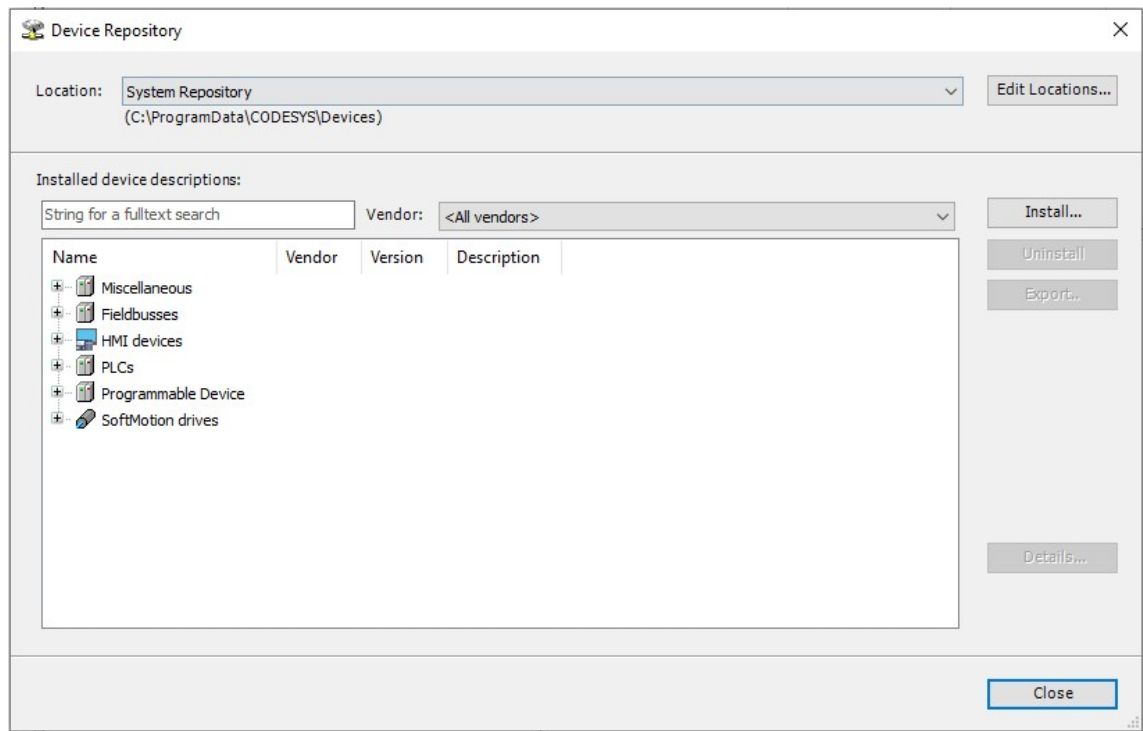


2. After the above operation, a dialog box will pop up as follows. Please select the file type "Device Description Configuration file (*.devdesc.xml)". Select MAT LC-C06_V1.0.devdesc.xml or MAT LC-C07_V1.0.devdesc.xml and click Open. The installation is successful.

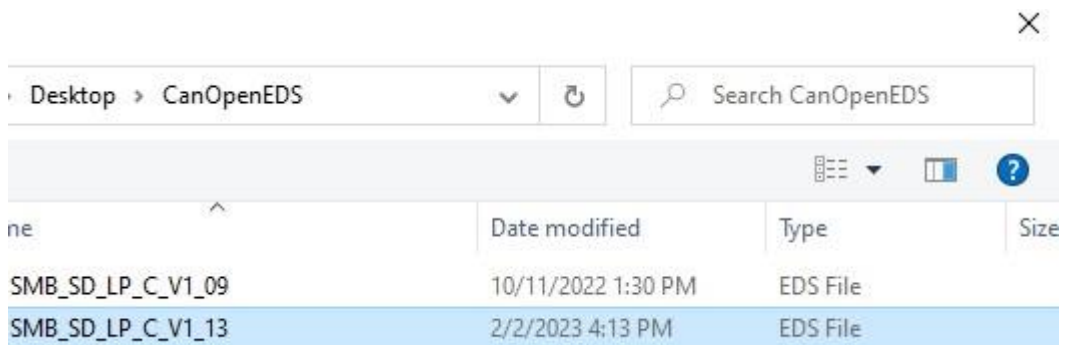


H.3 Install CANopenslave EDS file

1. Open the CODESYS software, select the menu "Tools" → "Device Repository" to open the following dialog box, and click "Install" in the dialog box.



2. A dialog box is displayed as follows. Select the file type EDS and DCF file (*.eds;*.DCF) and the suffix SMB_SD_LP_C_V1_XX.eds Click "Open" to install the EDS file successfully.



I Instruction Quick Check

I.1 Operator summary in CODESYS

The following table summarizes the operators that exist in the Standard.lib and Util.lib libraries of CODESYS, respectively.

Note the 'IL' column: only lines that use this symbol are displayed. A necessary condition is that the first set of necessary operands has been successfully loaded on the previous line (e.g., LD in) The 'Mod.IL' column shows modifiers that are possible in IL:

C	Modifier for the result of the preceding expression being TRUE.
N	For JMPC, CALC, and RETC: If the instruction output is FALSE, run this instruction.
N	Otherwise: negates operand (non-cumulator)
(	Operator in parentheses: The operation starts only after the parentheses appear.

Table I-1 Operator summary table

In ST	In AWL	Mod. AWL	Description
'			string delimiter
.. []			Array domain size
:			The delimiter between operands and types in the declaration.
;			Description terminated.
^			Designator obsolete.
	LD var1	N	Loading of variable 1 value in buffer.
:=	ST var1	N	Store the actual result in variable 1.
	S boolvar		When the actual result is TRUE, exactly set the operand boolean variable to TRUE.
	R boolvar		When the actual result is FALSE, exactly sets the operand boolean variable to TRUE.
	JMP label	CN	Skip directly to the tab.
<Program name>	CAL prog1	CN	Call the program prog1.
<Instance name>	CAL inst1	CN	Call the program inst1.
<Fctname>(vx, vy,...)	<Fctname> vx, vy	CN	Call the function fctname and transmit the variables vx,vy.
RETURN	RET	CN	Leave POU and go back to caller
	(		The values after the parentheses are treated as operands, and the operations before the parentheses are not performed before the expressions inside the parentheses.

	)		Now execute the set operation
AND	AND	N,(	Bit AND
OR	OR	N,(	Bit OR
XOR	XOR	N,(	bit specific OR
NOT	NOT		bit NOT
+	ADD	(	addition
-	SUB	(	subtraction
*	MUL	(	multiplication
/	DIV	(	division
>	GT	(	Greater than
>=	GE	(	great than or equal to
=	EQ	(	equal to
<>	NE	(	unequal to
<=	LE	(	Less than or equal to
<	LT	(	Less than
MOD(in)	MOD		Module division
INDEXOF(in)	INDEXOF		Internal index of POU IN1
SIZEOF(in)	SIZEOF		We know the number of bytes required for the IN data type
SHL(K,in)	SHL		operator in bit shift left by k
SHR(K,in)	SHR		operator in bit shifts right by k
ROL(K,in)	ROL		operator in bit rotates k to the left
ROR(K,in)	ROR		operator in bit rotates k to the right
SEL(G,in0,in1)	SEL		Binary selection between two operands in0 (G is FALSE) and in1 (G is TRUE).
MAX(in0,in1)	MAX		Restoration of the larger of the 2 values.
MIN(in0,in1)	MIN		Restoration of the smaller of the two values in0 and in1.
LIMIT(MIN,in,Max)	LIMIT		Limit range (in resets to MIN, or MAX to avoid going out of range).
MUX(K,in0,...in_n)	MUX		Select the Kth value from a set of values (in to In_n).
ADR(in)	ADR		The address of the operand in DWORD.
ADRINST()	ADRINST()		The call operator is the address of the function block operator.
BITADR(in)	BITADR		Bit balance of operands in DWORD.
BOOL_TO_<type>(in)	BOOL_TO_<type>		Type conversion of boolean operands.
<type>_TO_BOOL(in)	<type>_TO_BOOL		Type converted to BOOL.
INT_TO_<type>(in)	INT_TO_<type>		INT operand type is converted to another basic type.
REAL_TO_<type>(in)	REAL_TO_<type>		REAL operand type is converted to another basic type.
LREAL_TO_<type>	LREAL_TO_<type>		LREAL operand type is converted to

(in)			another basic type.
TIME_TO_<type> (in)	TIME_TO_<type>		TIME operand type is converted to another basic type.
TOD_TO_<type> (in)	TOD_TO_<type>		TOD operand type is converted to another basic type.
DATE_TO_<type> (in)	DATE_TO_<type>		DATE operand type is converted to another basic type.
DT_TO_<type> (in)	DT_TO_<type>		DT operand type is converted to another basic type.
STRING_TO_<type> (in)	STRING_TO_<type>		The string operand type is converted to another primitive type, and in must contain a valid value of the requested type.
TRUNC (in)	TRUNC		Convert from REAL to INT.
ABS (in)	ABS		Absolute value of operand in.
SQRT (in)	SQRT		The square root of the operand in.
LN (in)	LN		The natural logarithm of the operand in.
LOG (in)	LOG		The base 10 logarithm of the operand in.
EXP (in)	EXP		The power of the operand in.
SIN (in)	SIN		Sine of operand in.
COS (in)	COS		Cosine of operand in.
TAN (in)	TAN		Tangent of operand in.
ASIN (in)	ASIN		Arc sine of operand in.
ACOS (in)	ACOS		Arc cosine of operand in.
ATAN (in)	ATAN		Arc tangent of operand in.
EXPT (in,expt)	EXPT expt		The power of the operand in with expt.

<Remark> Please get detailed usage information in the relevant appendix of the CODESYS library.

I.2 Standard.lib library

Table I-2 Standard.lib file instruction

In ST	In AWL	Description
LEN (in)	LEN	The string length of the operand in
LEFT (str,size)	LEFT	The first string to the left of the known size of the string str.
RIGHT (str,size)	RIGHT	The first string to the right of the known size of the string str.
MID (str,size,pos)	MID	Partial str string of known size at POS.
CONCAT ('str1','str2')	CONCAT 'str2'	The union of two subsequent strings.
INSERT ('str1','str2',pos)	INSERT 'str2',p	Insert string str1 into string str2 at pos
DELETE ('str1',len,pos)	DELETE len,pos	Delete part of the string (length len), starting at pos of str1.
REPLACE ('str1','str2',len,pos)	REPLACE 'str2',len,pos	The replacement length string is str2, and the starting position is pos of str1.
FIND ('str1','str2')	FIND 'str2'	Find the partial string str2 in str1.

SR	SR	Bistable FB is set to dominant.
RS	RS	Bistable FB reset.
SEMA	SEMA	FB: Arm board software (interruptible)
R_TRIG	R_TRIG	FB: The rising boundary is monitored.
F_TRIG	F_TRIG	FB: The falling boundary is monitored.
CTU	CTU	FB: Appreciation count.
CTD	CTD	FB: Depreciation count.
CTUD	CTUD	FB: Count up and down.
TP	TP	FB: trigger
TON	TON	FB: On-Delay lock
TOF	TOF	FB: Off-Delay clock
RTC	RTC	FB: Realistic clock.

I.3 Util.lib library

Table I-3 Util.lib library instruction

BCD_TO_INT	Byte conversion: BCD to INT format.
INT_TO_BCD	Byte conversion: INT to BCD format.
EXTRACT(in,n)	The Nth bit of DWORD in is restored as BOOL.
PACK	Up to 8 bits are pushed into a byte.
PUTBIT	A bit in DWORD in is set to a certain value.
UNPACK	Bytes are recovered as individual bits.
DERIVATIVE	Local derivation.
INTEGRAL	Integer.
LIN_TRAFO	Conversion of REAL values.
STATISTICS_INT	Min, Max and Average in INT format.
STATISTICS_REAL	Minimum, maximum and average values in REAL format.
VARIANCE	Variation
PD	PD controller
PID	PID controller
BLINK	Pulse signal
FREQ_MEASURE	Measured frequency of boolean input signal.
GEN	Periodic function.
CHARCURVE	Linear function
RAMP_INT	The rise of the limited supply function (INT) weakens.
RAMP_REAL	Restricted supply function (REAL) weakened rise.
HYSTERESIS	Hysteresis
LIMITALARM	Note: whether the input value is outside the limits of the defined range.

SM3_Basic Library

This library is the base library for CODESYS SoftMotion applications, so it must be included in the SoftMotion project. This will happen automatically when a SoftMotion device is plugged in.

The library provides the following function blocks and functions:

- PLCopen function block allows both single-axis motion control and two-axis synchronous motion. In addition to the library elements for status checking, parameterization and general operation, there are function blocks for driving the axis with defined velocity, acceleration parameters. There are also modules for synchronizing master, slave and drive shafts.
- DriveInterface basic driver function blocks (AXIS_REF_SM3, AXIS_REF_VIRTUAL_SM3, AXIS_REF_MAPPING_SM3)
- Utility functions such as file serving and error reporting
- A function block or implementation of a specific drive driver, an extension of the basic drive function block. These special driver function blocks (named AXIS_REF_*) are usually included in a separate library that is referenced by the corresponding device description for this device.

The function blocks are organized as follows:

- Drive interface

Instruction Name	Function
AXIS_REF	This driver interface is a function block. It is provided by the "SM3_Basic.library" library and includes work as a driver. Each SoftMotion axis is an extension of the AXIS_REF_SM3 instance.

- Data types, global variables

Instruction Name	Function
MC_DIRECTION	It is used as input to several function blocks and specifies the direction of movement. Note that not all modes are available for all function blocks and axis types (modulo/finite).
MC_TAPPETMODE	It is used in the function block MC_DigitalCAMSwitch to determine whether the tappet value should refer to the spindle setpoint (1), or the actual position (2). In the case of automatic detection (0), the decision of the function block depends on whether the control state of the drive uses the setpoint (bRegulatorOn = TRUE) or the actual value (bRegulatorOn = FALSE).
SMC_BRAKESETSTATE	It is used in the function block SMC3_BrakeControl and determines how the drive sets the mechanical brake.
SMC_CAMTAPPETACTION	It determines which switching actions are performed when the cam has been passed.
SMC_CAMTAPPETTYPE	It is used to determine which direction the cam must go in order to be activated.

SMC_CONTROLLER_MODE	It describes which quantity of the motor is controlled.
SMC_ERROR	It includes all possible error numbers returned by the SoftMotion function block.
SMC_HOMING_MODE	It defines the homing sequence when used by SMC_Homing.
SMC_INT_STATUS	It describes the state of the interpolator instance.
SMC_LANGUAGE_TYPE	It is used in the function SMC_ErrorString to determine the language.

● Data types, structures

Instruction Name	Function
MC_CAM_ID	It is an internal data structure that describes the CAM workbench. It is generated by MC_CAMTableSelect and used as input to MC_CamIn.
MC_CAM_REF	It represents a normal CAM and contains the following elements:
MC_CAMSWITCH_REF	It describes the CAMSwitches table and is used as input to MC_DigitalCAMSwitch.
MC_CAMSWITCH_TR	It describes the toggle position of the follower.
MC_OUTPUT_REF	It defines the tappet output in the form of an array of 32 booleans. TYPE MC_OUTPUT_REF : ARRAY [0..31] OF BOOL; END_TYPE
MC_TRACK_REF	

● DriveBasic

Instruction Name	Function
SMC3_BrakeControl	This function block determines the behavior of the mechanical brake if supported by the drive and its SoftMotion driver.
SMC3_BrakeStatus	It reads the state of the mechanical brake.
SMC_SetControllerMode	This module can be used to switch to other controller modes if supported by the drive device.
SMC_GetMaxSetAccDec	It is used to measure the maximum value of the absolute value of the acceleration (or deceleration) of an axis.
SMC_GetMaxSetVelocity	It is used to measure the maximum value of a shaft speed.
SMC_GetTrackingError	It measures the actual and maximum hysteresis error used to compensate for dead time, which may occur during fieldbus communications and will last for many cycles (byDeadTimeCycles).
SMC_InPosition	It checks if the actual position of the axis is within a certain position value for a certain period of time.
SMC_MeasureDistance	It is used on a rotary axis to measure distance traveled (the casing is also counted).

SMC_CheckLimits	It can be used to check if the actual setpoint of the drive exceeds the maximum value configured in the controller. The result of the check will be indicated by the bLimitsExceeded output.
SMC_FollowPosition	It will write the position set point to the axis without any checks.
SMC_FollowPositionVelocity	Its use is equivalent to SMC_FollowPosition, except for possibly additionally defined speeds.
SMC_FollowSetValues	It will write the set value to the axis without any checking.
SMC_FollowVelocity	It will write the set speed to the axis without any checks.
SMC_ClearFBError	It is used to delete the oldest function block error message.
SMC_ReadFBError	It is used to read the oldest message of function block errors.
SMC_Homing	It performs the reference movement of the axis. As the reference switch, a boolean value is used, which is typically input by a hardware.
SMC_ChangeGearingRatio	With the help of this module, gear ratios and drive equipment types can be modified.

- Error

Instruction Name	Function
SMC_ErrorString	Depending on the input ErrorID and language, the function SMC_ErrorString will return a string representation of the error.

- File

Instruction Name	Function
SMC_ReadCam	It is used to download the CAM at runtime and make it available to MC_CamTableSelect and MC_CamIn.
SMC_WriteCam	This function block writes a CAM created in the CAM editor to *.CAM-file at runtime.
SMC_AxisDiagnosticLog	It is used to cyclically write a series of parameter values belonging to an axis to a file. This output file is the basis for troubleshooting.

- PLCopen

Instruction Name	Function
SMC_ReadSetPosition	It is used to read the current set position of the drive device.
SMC_SetTorque	It can be used to create a torque if the drive is in controller mode "torque".
SMC_CAMBounds	It calculates the maximum position, velocity and acceleration values of a slave device that is connected in absolute mode to a master device that moves at a specified maximum velocity and acceleration/deceleration.
SMC_CAMBounds_Pos	In contrast to SMC_CAMBounds, this function block only calculates the maximum and minimum positions of the slave

	device, which is connected to a master device in absolute mode, and the master device moves at the specified maximum speed and acceleration/deceleration.
--	---

SMC_CAMRegister	It represents a tappet control unit. It acts on the MC_CAM_REF structure like MC_CamIn, cancels the original track information and only reads the tappet information.
SMC_GetCamSlaveSetPosition	This module calculates the current target position of the slave axis if the (slave) axis is linked to the motion of the other (master) axis via CAM. So neither axis will be moved or affected.
SMC_GetTappetValue	It evaluates the output tappet of the MC_CamIn function block and contains the current tappet state.
MC_CamIn	It occupies CAM.
MC_CamOut	It will immediately release the slave axis from the master axis.
MC_CamTableSelect	It selects a CAM table by setting up a connection to the relevant table.
MC_GearIn	It selects a CAM table by setting up a connection to the relevant table.
MC_GearInPos	It selects a CAM table by setting up a connection to the relevant table.
MC_GearOut	It selects a CAM table by setting up a connection to the relevant table. It disengages the slave axis from the master axis.
MC_Phasing	It will occupy CAM.
MC_AccelerationProfile	It is used to control a time-acceleration locked motion profile.
MC_Halt	It is used to command a controlled movement to stop. The function block will terminate the execution of all in-progress function blocks.
MC_Home	Its execution causes the axis to execute the "search home" sequence. The details of this sequence depend on the machining process and can be set by the parameters of the axis.
MC_MoveAbsolute	It is used to command controlled movement to a specified absolute position.
MC_MoveAdditive	It is used in discrete motion states to command a controlled motion that adds the specified relative distance to the newly requested position.
MC_MoveRelative	It is used to command a controlled movement that, when executed, specifies a distance from the actual position of the axis.
MC_MoveSuperImposed	It is used to command a controlled movement that adds a specified relative distance to an existing movement. Existing movements are not interrupted, but additional movements are superimposed.

MC_MoveVelocity	It is used to command a never-ending controlled movement at a specific speed.
MC_PositionProfile	It is used to command a time-position locked motion profile.
MC_Power	It is used to control the power stage ("on" or "off").
MC_ReadActualPosition	It will return the actual position of the reference axis.
MC_ReadAxisError	It is used to describe common axis errors not related to function blocks.
MC_ReadBoolParameter	It returns a parameter of type BOOL specified by the manufacturer.
MC_ReadStatus	It returns the detailed status of the axis regarding the motion in the current process.
MC_ReadParameter	It returns vendor-specified parameters.
MC_Reset	It converts ErrorStop to StandStill state by resetting all internal errors about the axis - this affects the output of the function block instance.
MC_Stop	Its execution will cause the controlled motion to stop and set the axis to state "Stopping".
MC_VelocityProfile	It is used to command a time-velocity locked motion profile.
MC_WriteBoolParameter	It can be used to modify factory-set Boolean-type parameters.
MC_WriteParameter	It can be used to modify factory-set parameters.
MC_AbortTrigger	It is used to terminate a function block connected to a trigger event.
MC_DigitalCamSwitch	Similar to a switch on a motor shaft: the function block controls a set of discrete output bits to switch a set of switches like a mechanical cam controlled switch attached to the shaft. Both forward and backward movements are permitted.
MC_ReadActualTorque	When Enable remains TRUE, the function block will return the actual torque or force value.
MC_ReadActualVelocity	When Enable remains TRUE, the function block will return the value of the actual speed.
MC_SetPosition	It is used to move the coordinate system of an axis.
MC_TouchProbe	It is used to record the position of the axis when the trigger event occurs.
SMC_MoveContinuousAbsolute	It performs absolute movement, but unlike MC_MoveAbsolute, it does not reach the target position with velocity=0, but with the specified velocity.
SMC_MoveContinuousRelative	It performs relative movement, but unlike MC_MoveRelative, it does not reach the target position with velocity=0, but with a specified velocity.

- SimpleTest

Instruction Name	Function
SMC_StartupDrive	It contains a set of frequently used function blocks which are used to test and commission an axis.

SM3_CNC Library

SM3_CNC.Library library are used (according to DIN 66025) to decode trajectories created in the CNC editor. Decoding is to transform the trajectory into a structure so that it can be used by IEC programs (trajectory preprocessing), or can be modified, interpolated, and transformed so that it is readable by the driving device (for example, SMC_NCDecoder, ToolCorr, SMC_AvoidLoop, SMC_SmoothPath, SMC_RoundPath, SMC_Interpolator).

Functional failure of modules allows certain components - such as decoders - to be replaced by modules designed for special needs. In addition, other modules - such as certain trajectory preprocessing or optimization - can be added to existing components without any problems, so that the compatibility between components in the library does not need to be considered. Therefore, the program logic is completely handled by the PLC program, and only pure action information is handled by the library module.

The data structures of the library SM3_CNC.Library (eg SMC_POSINFO, SMC_VECTOR3D) are used to describe positions, trajectory segments and vectors, and to manage GEOINFO objects (QUTQUEUE structures).

• Data type

Instruction Name	Function
SMC_CNC_REF	This data structure manages the parsed G-code files.
SMC_GCODE_TEXT	The data source is connected to the output of the same name of the SMC_NCDecoder module. It is used to pass the decoded string of the last line of the CNC file to the function block SMC_GCodeViewer.
SMC_GCODEVIEWER_DATA	This structure uses an array of arrays as the cache of the SMC_GCodeViewer function block; a variable of this type stores the G code file corresponding to a curve track.
SMC_GCODE_WORD	It includes data for a certain G code word such as "N20" or "G1".
SMC_GEOINFO	This structure is designed to store track objects, that is, a segment of a programmed track, such as a line, an arc, or a spline segment.
SMC_M_PARAMETERS	It can define additional parameters for the currently active M-function, which can be evaluated by SMC_GetMParameters to display the additional value of each M-function.
SMC_OUTQUEUE	This structure is used to manage GEOINFO objects in a table of a defined size.
SMC_POSINFO	For a particular location point the structure describes its coordinates and the location of additional axes.
SMC_VECTOR3D	The structure describes the velocity in three dimensions.

• Global variable

Instruction Name	Function
SMC_AL_STATUS	Its value indicates the actual state of the FB instance.
SMC_DEC_STATUS	Its value indicates the actual state of the FB instance.
SMC_DIRECTION	Its value indicates the actual interpolation direction of the function block instance.

SMC_INT_STATUS	Its value indicates the actual state of the FB instance. possible states:
SMC_INT_VELMODE	Its value determines the velocity profile.
SMC_MOVTYP	It is a member of SMC_GEOINFO, which specifies the geometry of the object.
SMC_TC_STATUS	Its value represents the actual state of the FB instance.
SMC_TOOLCORRMODE	Its value represents the mode in which the SMC_Toolcorr runs.
SMC_VECTORDIR	It is used as the input variable of SMC_CalcDirectionFromVector to define the inclination, reverse or vertical of the velocity.

Function block record

- Inside the SMC_CNC_POUs\File folder

Instruction Name	Function
SMC_READNCFILE	It reads an NC-ASCII document from the controller's file system, making the document available to SMC_NCDecoder. Thus an NC program can be read in and executed during runtime.
SMC_READNCQUEUE	It reads an OutQueue file created by the CNC editor from the PLC file system and provides an OutQueue structure, which is typically handled by the decoder.
SMC_VARLIST	IEC 61131-3 standard makes it impossible to determine the value of a variable by its symbolic name. But this is necessary, if the function of the variable is provided to the user by SMC_CNC_REF , it should also be possible to read the CNC program from a file. This can be managed by the structure SMC_VARLIST .

- Inside the SMC_CNC_POUs\SoftMotion CNC\Coordinate Transformation folder

Instruction Name	Function
SMC_DETERMINECUBOIDBEARING	It determines the position of a cube (corner sign, edge alignment) in space with the given 6 (3/2/1) points.
SMC_COORDINATETRANSFORMATIONS3D	It computes the coordinates of a point given in the old coordinate system and transformed to the new coordinate system. The coordinate transformation is defined by the transformation of the origin and the new unit vector (given in the old coordinate system).
SMC_INVCOORDINATETRANSFORMATION3D	It computes the coordinates of a point given in the new coordinate system and converts it into the old coordinate system.
SMC_TEACHCOORDINATESYSTEM	A vector that assists the user in calculating the new coordinate system.
SMC_UNITVECTORTORPY	RPY- angle from the unit vector of the new coordinate system (with reference to the old coordinate system)
SMC_ROTATEQUEUE2D	When this module executes, the trajectory stored in

	poqDataIn will be rotated around the Z axis by the given angle dPhi[°]. A positive angle is a mathematical positive rotation (counterclockwise).
SMC_SCALEQUEUE3D	When it executes, the trajectory contained in poqDataIn will be stretched by the factor fScaleFactor.
SMC_TRANSLATEQUEUE3D	The path stored in poqDataIn through its execution will be converted according to vec , which is the data type of SMC_VECTOR3D.

- SMC_CNC_POUs\SoftMotion CNC\Direct Axis Control folder

Instruction Name	Function
SMC_CONTROLAXIXBYPOS	Used to write the target position to the drive structure to test the jumped structure. It is widely used in CNC and SMC_Interpolator . For the setting value of the directional control axis, a function block starting with SMC_Follow* should be used.
SMC_CONTROLAXIXBYPOS VEL	Its use is identical to SMC_ControlAxisByPos, except that the velocity can be defined when added. It is usually used for CNC and SMC_Interpolator.

- SMC_CNC_POUs\SoftMotion CNC\GCode Viewer folder

Instruction Name	Function
SMC_GCODEVIEWER	It can be combined with SMC_NCDecoder.

- SMC_CNC_POUs\SoftMotion CNC\Posinfo Functions folder

Instruction Name	Function
SMC_POSINFO	For a particular location point the structure describes its coordinates and the location of additional axes.

- Inside the SMC_CNC_POUs\SoftMotion CNC\OutQueue Functions folder

Instruction Name	Function
SMC_APPENDOBJ	This function block is used to process the SMC_OUTQUEUE structure object.
SMC_DELETEOBJ	This boolean function is used to process an SMC_OUTQUEUE structure object .
SMC_GETCOUNT	A function of type UNIT will return the number of objects stored in the SMC_OUTQUEUE table (POQ).
SMC_GETOBJ	Provided the input poq is properly initialized and contains at least N+1 elements, the function returns a pointer to the Nth GEOINFO object in the list, counting from the beginning of the list; so for N=0 the first element is returned.
SMC_GETOBJFROMEND	Provided the input poq is properly initialized and contains at least N+1 elements, the function returns a pointer to the Nth GEOINFO object in the list, counting from the end of the list; so for N=0 the last element is returned.

SMC_OUTQUEUEINIT	It sets the variables of the SMC_OUTQUEUE data structure to its default.
SMC_RESTOREQUEUE	This function block restores an interpolated or processed structure. This is only possible if the entire trajectory can be stored into the table referenced by poq.
SMC_SETQUEUECAPACITY	for setting the SMC_OUTQUEUE data structure.

• SMC_CNC_POUs\SoftMotion CNC\SoftMotion Function Blocks folder

Instruction Name	Function
SMC_AVOIDLOOP	This function block is used for trajectory preprocessing. It copies a track, but does not cut the loops it contains.
SMC_CHECKFORLIMITS	Used to check if a track leaves its specific rectangle bounds.
SMC_CHECKVELOCITIES	This module checks the orbital velocity of a particular trajectory segment.
SMC_EXTENDEDVELOCITY CHECKS	It reduces the velocity, acceleration and deceleration of the path so that the value of the velocity acceleration and deceleration generated by the additional axes (Z, P, Q, U, V, W) does not exceed the value of the SMC_GEOINFO variables (adVelAddAx[0..6], adAccAddAx[0..6] and adDecAddAx[0..6]).
SMC_INTERPOLATOR	This function block is used to convert a continuous trajectory described by the SMC_GEOINFO object into discrete trajectory position points, which take into account the velocity profile and the time map.
SMC_INTERPOLATOR2DIR	From the function and the configuration of input and output, this module is equivalent to the function block SMC_Interpolator, but it can also interpolate a trajectory in the reverse direction.
SMC_INTERPOLATOR2DIR_SLOWTASK	This module is responsible for generating the reverse trajectory. It is split from SMC_Interpolator2Dir so that it can be encapsulated to a low priority task in a fully stretched low performance system.
SMC_LIMITDYNAMICS	This module is used to reduce the velocity, acceleration and deceleration of the trajectory so that the velocity, acceleration and deceleration of the additional axes (Z, P, Q, U, V, W) do not exceed the FB input values dMaxVel, dMaxAcc and dMaxDec .
SMC_LIMITCIRCULARVELOCITY	This module checks for a specific element in the OutQueue and limits the trajectory velocity of the circular element in the opposite direction of its radius.
SMC_NCDECODER	This module is used to convert a CNC program (DIN 6625, G code) created in the CNC editor into a table of SoftMotion-GEOINFO structure objects. One line in the program will be decoded in each loop.
SMC_ROUNDPATH	The SMC_ROUNDPATH function block is very similar to the SMC_SmoothPath block. It is the angle of the circular arc

	intercepted by the intersection of the two lines.
SMC_SMOOTHPATH	The SMC_SmoothPath function block can be used for trajectory preprocessing. It rounds the path angles, resulting in a slur path. It is used when the accuracy of the trajectory is not high, but the speed is high, and the speed is not allowed to drop to 0.
SMC_TOOLCORR	The SMC_ToolCorr is used for path preprocessing.
SMC_XINTERPOLATOR	The SMC_XInterpolator function block implements a mix of CAM and CNC. Suppose you want to cut a workpiece to a specific shape (described by G-code), according to the movement of the workpiece by another process (for example, moving along the X axis) and other axes (y, z, etc.) should follow the workpiece (X) The current position is controlled, and the target is given by the path point.
SMC_GETMPARAMETERS	The interpolator is processing an M-function, and this module is used to poll the parameters (K, L, O, see HERE) that have been set for the M-function.
SMC_PREACKNOWLEDGE MFUNCTIONS	It is used to validate an M function before the interpolator reaches the block. In this way, the pause on the M function can be avoided.

- Inside the SM_Trafo_POUs\Additional FBs folder

Instruction Name	Function
SMC_CALCDIRECTIONFROMVECTOR	The orientation of the tool (dAlpha) can be calculated when using it.

- Inside the SM_Trafo_POUs\Gantry Systems folder

Instruction Name	Function
SMC_TRAFO_GANTRY2	For the gantry system, no transformation can be performed, so the module can only add offsets to the x and y axes.
SMC_TRAFO_GANTRY2TOOL1	The function block is used to reverse the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, and they are approximately on a straight line.
SMC_TRAFO_GANTRY2TOOL2	The function block is used to reverse the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, which is approximately a right angle.
SMC_TRAFO_GANTRY3	For the gantry system, no transformation can be performed, so the module can only add offsets to the x, y, and z axes.
SMC_TRAFO_GANTRYCUTTER2	Computes an inverse transformation for a 3D gantry system being controlled and indicating an axis of rotation along the tangent of the path.
SMC_TRAFO_GANTRYCUTTER3	Computes an inverse transformation for a 3D gantry system being controlled and indicating an axis of rotation along the tangent of the path.

SMC_TRAFO_GANTRYH2	This transition module is used to provide reverse transitions to address H gantry systems with fixed drives.
SMC_TRAFOF_GANTRY2	The function module is used to operate a three-dimensional gantry operating system without motion. Therefore, this module only adds an offset to each of the X and Y axes. Each instance of the SMC_TRAFOF_GANTRY2 module can be linked to a visualization template named SMC_VISU_GANTRY2.
SMC_TRAFOF_GANTRY2TO OL1	The function block is used to reverse the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, and they are approximately on a straight line.
SMC_TRAFOF_GANTRY2TO OL2	The function block is used to reverse the tool axial offset of the gantry system, that is, the axial offset relative to the Z axis, which is approximately a right angle.
SMC_TRAFOF_GANTRY3	The function module is used to operate a three-dimensional gantry operating system without motion. Therefore, this module only adds an offset to each of the X, Y, and Z axes. Each instance of the SMC_TRAFOF_GANTRY3 module can be linked to a visualization template named SMC_VISU_GANTRY3.
SMC_TRAFOF_GANTRYCUT TER2	This module will address the forward transition of a 3D gantry system with an axis of rotation pointing in a tangential direction.
SMC_TRAFOF_GANTRYCUT TER3	It will resolve the forward transition of the 3D gantry system with the axis of rotation pointing in the tangential direction of the current trajectory.
SMC_TRAFOF_GANTRYH2	This transition module is used to provide forward transitions that address H gantry systems with fixed drives.
SMC_TRAFOV_GANTRY2	An inverse transformation module is used to provide a two-dimensional gantry system that handles considering the trajectory velocity and trajectory direction as a control variable for the axis.
SMC_TRAFOV_GANTRY3	An inverse transformation module is used to provide a three-dimensional gantry system that handles considering the trajectory velocity and trajectory direction as a control variable for the axis.
SMC_TRAFOV_GANTRYCUT TER2	The inverse transformation module is used to process the two-dimensional gantry system considering the trajectory velocity and trajectory direction as the control variables of the axes.
SMC_TRAFOV_GANTRYCUT TER3	An inverse transformation module is used to provide a three-dimensional gantry system that handles considering the trajectory velocity and trajectory direction as a control variable for the axis.
SMC_TRAFOV_GANTRYH2	The inverse transformation module is used to provide a two-dimensional H-gantry system that handles considering

	the trajectory velocity and trajectory direction as the control variable of the axis.
--	---

- Inside the SM_Trafo_POUs\Parallel Systems folder

Instruction Name	Function
SMC_TRAFO_BIPOD_ARM	The conversion module is used to provide a backward conversion to handle the bipod arm.
SMC_TRAFO_TRIPOD	The conversion module is used to provide backward conversion for handling tripods.
SMC_TRAFO_TRIPOD_ARM	The conversion module is used to provide backward conversion for handling the tripod arm.
SMC_TRAFOF_BIPOD_ARM	The conversion module is used to provide forward conversion for handling the bipod arm.
SMC_TRAFOF_TRIPOD	The conversion module is used to provide forward conversion for handling tripods.
SMC_TRAFOF_TRIPOD_ARM	The conversion module is used to provide forward conversion for handling the tripod arm.

- Inside the SM_Trafo_POUs\Scara Systems folder

Instruction Name	Function
S MC_TRAFO_SCARA2	The transformation module is used to provide backward transformations for processing the second joint Scara system.
SMC_TRAFO_SCARA3	The transformation module is used to provide backward transformations for handling the third joint Scara system.
SMC_TRAFOF_SCARA2	The transformation module is used to provide forward transformations that handle the second joint Scara system.
SMC_TRAFOF_SCARA3	The transformation module is used to provide forward transformations that handle the third-joint Scara system.

J Ordering Information

Table J-1 List of ordering information

Specification description	Part Number
LC-C06 Motion Controller	
LC-C06 main control module , 32MB program data space, 64KB power-down retention data , 55M expansion bus, 1 RS485 communication port, MODBUS free port protocol , 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB ports, support SoftMotion	MAT-LC-C06
LC-C07 Motion Controller	
LC-C07 main control module, 32MB program data space, 64KB power-down retention data, 55M expansion bus, 1 RS485 communication port, MODBUS free port protocol, 1 EtherCAT interface, 1 EtherNET interface, 1 CAN port, 1 USB port, support SoftMotion, CNC	MAT-LC-C07
Power module	
PWR-020S1 power module, input 85~264V AC, output 24V DC/2A	MAT-PWR-020S1
High-speed counting module	
HSC-02 high-speed counting module, 2-way differential/single-ended signal input	MAT-LC-HSC-02
Pulse output module	
HSP-04 pulse output module, 4-channel differential/single-ended signal output	MAT-LC-HSP-04
EtherCAT Slave Module	
EtherCAT slave module , up to 8 expansion modules can be attached	MAT-LC-ETC-00S1
Digital I/O	
DIT-08 digital module, digital input 8 x 24VDC	MAT-LC-DIT-08
DIT-16 digital module, digital input 16 x 24VDC	MAT-LC-DIT-16
DQT-08 digital module, digital output 8 x 24VDC	MAT-LC-DQT-08
DQT-16 digital module, digital output 16 x 24VDC	MAT-LC-DQT-16
DQR-08 digital module, digital output 8 x relay	MAT-LC-DQR-08
DQR-16 digital module, digital output 16 x relay	MAT-LC-DQR-16
Analog I/O	
AIS-04 analog module, analog voltage and current input, 4AI x 12bit	MAT-LC-AIS-04
AQS-04 analog module, analog voltage and current output, 4AQ x 12bit	MAT-LC-AQS-04
Accessories	
PLC lithium battery, voltage 3.6 V , capacity 2700 mAh	MAT-LC-BAT-00S1