XS series PLC

User manual [software] (Codesys)

Wuxi XINJE Electric., Ltd.

Data No. PS04 20220516EN 1.1

Basic description

- Thank you for purchasing the Xinje XS series programmable controller.
- This manual mainly introduces the software of XS series programmable controllers.
- Before using the product, please read this manual carefully and programming on the premise of fully understanding the contents of the manual.
- Please deliver this manual to the end user.

Notes to users

- Only operators with certain electrical knowledge can conduct wiring and other operations on the product. If there is any unknown place, please consult our technical department.
- The examples listed in the manual and other technical data are only for users' understanding and reference, and do not guarantee certain actions.
- When using this product in combination with other products, please confirm whether it conforms to relevant specifications and principles.
- When using this product, please confirm whether it meets the requirements and is safe.
- Please set up backup and safety functions by yourself to avoid possible machine failure or loss caused by the failure of this product.

Statement of responsibility

- Although the contents of the manual have been carefully checked, errors are inevitable, and we cannot guarantee complete consistency.
- We will often check the contents of the manual and make corrections in subsequent versions. We welcome your valuable comments.
- The contents described in the manual are subject to change without notice.

Related manuals

For the hardware related and advanced motion control instruction application of XS series PLC, please consult the following manuals.

- XS series motion control manual
- XS series hardware manual

November 2021

Catalog

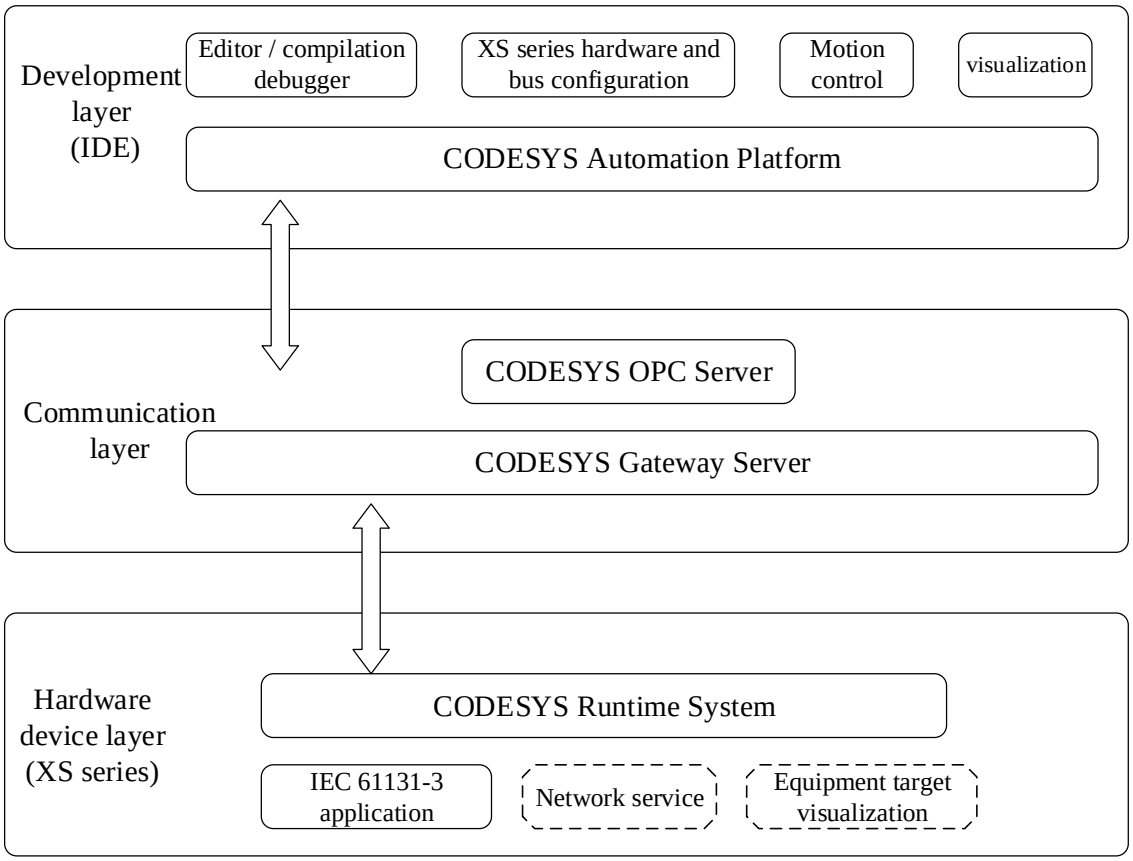
1. Codesys overview and installation

1-1. Codesys overview

The Codesys programming platform of German 3S company is selected for the Xinje XS series controller. Codesys is an industrial information technology, automation programming software and intelligent manufacturing equipment programming development platform, which provides global users with an open, flexible, stable and reliable series of advanced industrial information technology, software products and industry solutions. At present, about 350 control system manufacturers worldwide are Codesys users. The platform fully supports PLCopen specification and provides all editors defined and supported by IEC international standards for automated application development.

1-2. Codesys software architecture

Codesys software has powerful functions, high reliability and good openness. It integrates PLC programming, visual HMI, safety PLC, controller real-time core, fieldbus and motion control. It is a complete automation software. Codesys software can be divided into three layers in terms of architecture: application development layer, communication layer and device layer, as shown in the figure:



1-2-1. Development layer

Codesys Development System (It has perfect online and offline programming functions), compiler and its accessories, visual interface programming components, etc, at the same time, the optional motion control module, safety module and other components make Codesys more complete and powerful.

- Editor

Codesys provides six programming languages defined by IEC61131-3: function block diagram (FBD), ladder diagram (LD), instruction list (IL), structured text (ST), sequential function diagram (SFC) and continuous function diagram (CFC).

- Compiler

Responsible for converting the application program in Codesys into machine code and optimizing the performance of the programmable controller. When users input wrong application code, they will immediately receive syntax error warnings and error messages from the compiler, so that programmers can quickly make corresponding corrections. Users can use different Codesys based hardware devices (systems) for engineering development without changing the programming mode.

- XS series hardware and bus configuration

For XS series hardware devices and different fieldbus protocols, this part is responsible for setting corresponding parameters in Codesys.

- Visual interface programming

Visual programming (HMI) can be realized in Codesys, and the system has integrated a visual editor.

- Motion control module

The motion control function has been integrated into Codesys to form the softmotion (CNC) software package. The toolkit based on PLCopen can realize single axis and multi-axis motion, electronic cam transmission, electronic gear transmission, complex multi-axis CNC control, etc.

1-2-2. Communication layer

The communication between the application development layer and the hardware device layer is realized by the gateway server in Codesys, in which the OPC server is installed.

- Codesys gateway server

It functions between the application development layer and the hardware device layer. It can use TCP/IP protocol or CAN and other bus to realize remote access. It is an integral part of Codesys development kit.

- Codesys OPC server

For the Codesys based controller, it does not need to consider the hardware CPU. It has integrated and realized the multi-client function of OPC v2.0 specification, and can access multiple controllers at the same time.

1-2-3. Device layer

XS series PLC is the hardware equipment layer of the system. Codesys runtime system has been installed, which can meet the real-time response and accurate control requirements of the industry. At the same time, functional expansion can also be realized by using optional components of Codesys, such as Codesys target visual programming module or network visual programming module.

1-3. Xinje PLC supported by Codesys

XSDH series, XS3 series, M series and visual industrial computer.

1-4. Codesys installation and uninstallation

1-4-1. System requirements

Hardware and software requirements

- windows 8 or windows 10 64-bit OS
- Memory 4GB and above
- Hard disk space above 12GB

1-4-2. Obtain the Codesys

Download from the official Codesys store, website is http://www.Codesys.cn/.

1-4-3. Codesys installation

1. Basic requirements for hardware and software

Since Codesys v3.5 software is relatively large and has a lot of processing information, it has certain requirements for PC hardware and software. The required minimum configuration and recommended configuration are shown in the following table:

Item	Minimum configuration	Recommended configuration
OS	Windows 2000 （Windowns XP/Windows Vista/Windows7）	Windows10
Memory	4GB	4GB
Hard disk	12GB	12GB
CPU	Pentium V, Centrino>1.8GHz, Pentium M >1.0GHz	Pentium V, Centrino>3.0GHz, Pentium M>1.5GHz

2. Installation

Run Codesys 64 3.5.16.0.exe as an administrator to enter the installation, and the installation assistant will guide the user to install throughout the installation process.

Note: It is not recommended that users install the software on disk C.

1-4-4. Codesys version management

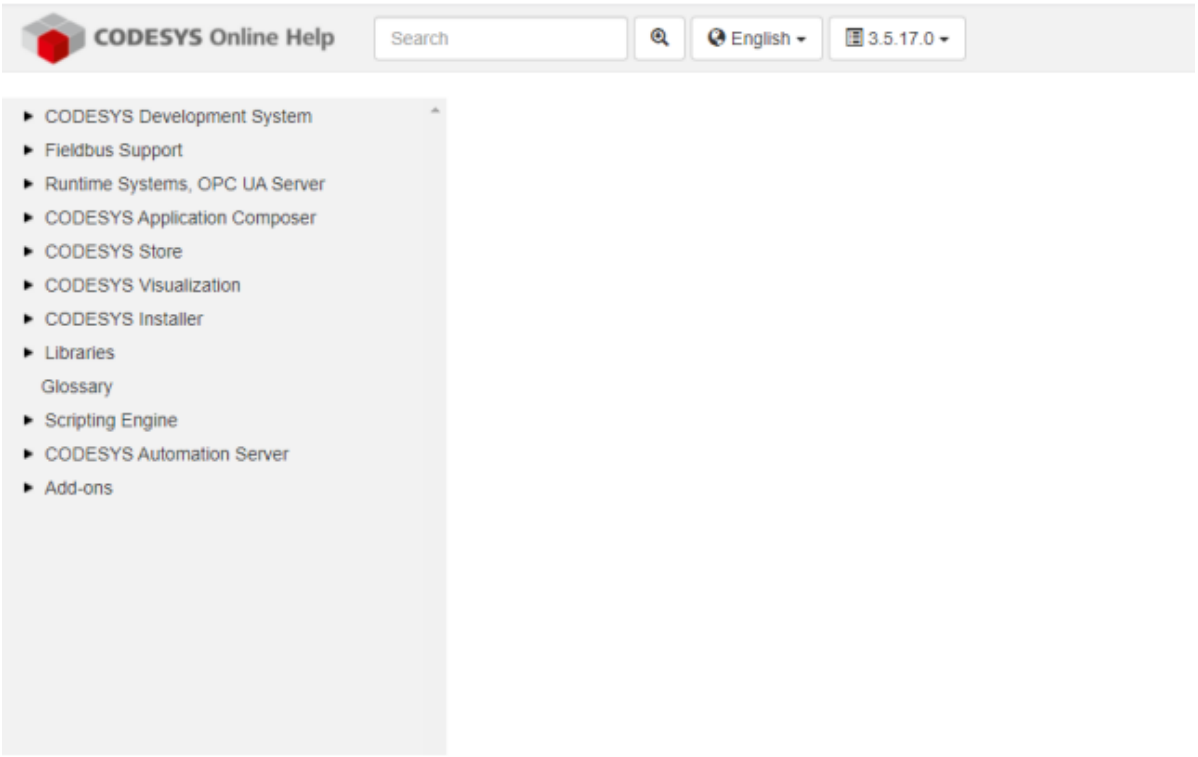
The upper computer of Codesys supports the installation of multiple versions at the same time. The compiler also supports the installation of multiple versions. Version 3.5.16.40 is recommended. Using other versions of the upper computer may cause abnormal use of some functions.

1-4-5. Codesys uninstallation

The Codesys programming software can be uninstalled through the windows control panel. Open control panel - > Add / remove programs, select Codesys, click the delete button, and complete the uninstallation according to the prompt.

1-5. Codesys help

After opening the Codesys application, users can find the help menu and click "contents" to open the online help. Users can quickly find the required content according to the index or search keywords, as shown in the figure:

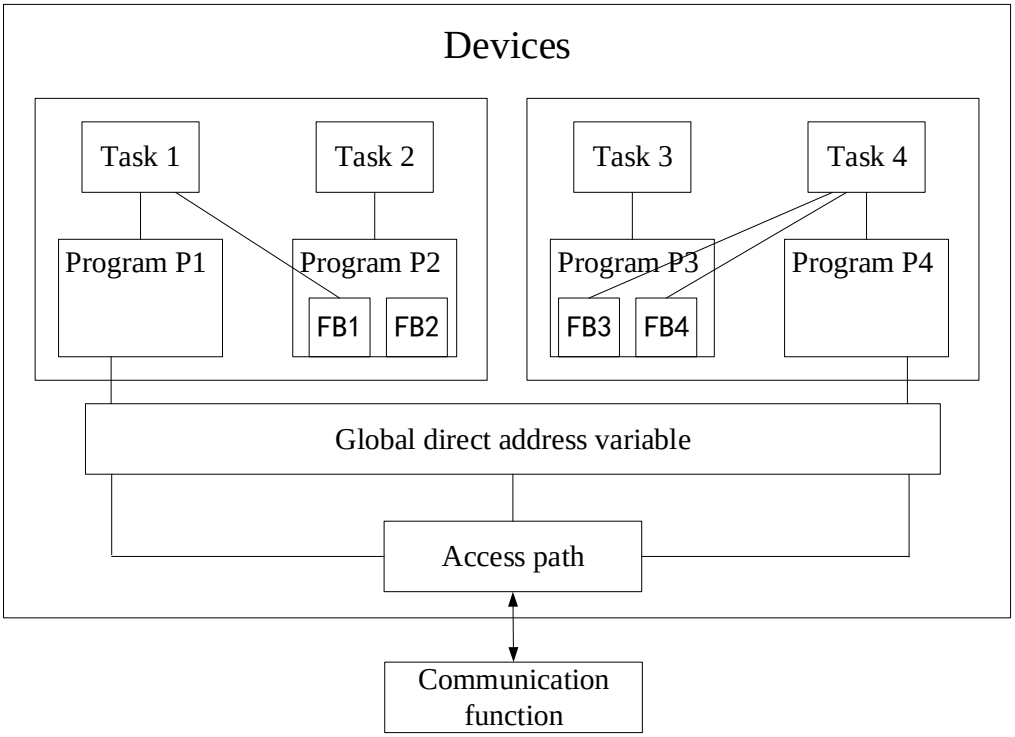


2. Codesys structure

2-1. Software model

2-1-1. Software model introduction

The software model of Codesys describes the basic software elements and their relationships, which are expressed in a hierarchical structure. Each layer contains many characteristics of its underlying layers, and its internal structure is shown in the following figure. Among them, the software elements include: equipment, application, task, global variable, access path and application object. They are the software foundation of modern soft PLC. The software model is consistent with the software model of IEC 61131-3 standard.



The software model describes how to decompose a complex program into several small manageable parts in principle, and there is a clear and standardized interface method between the decomposed parts. The software model describes how a programmable controller can run several independent programs at the same time, and how to fully control the program execution.

- Devices

At the top layer of the model is "equipment", which can be equivalent to all the software required by a PLC. For large and complex application systems, such as the automation of the whole product line, multiple PLCs may be required for online communication. It is necessary to realize bus communication between one PLC and multiple other equipment interfaces. At this time, "equipment" can be understood as a specific type of control system, which includes hardware devices, processing resources, I/O address mapping and system memory storage capacity, that is, it is equivalent to a PLC.

- Application

In the PLC system, the equipment combines all "applications" into groups to provide a means of data exchange for "applications". In each device, there are one or more "applications", which are located in the second layer of the software model. "Application" not only provides a support system for running programs, but also reflects the physical structure of PLC and provides an interface between programs and PLC physical I/O channels.
The application is allocated in the CPU of a PLC, so the application can be understood as a microprocessor unit in

a PLC. Global variables defined within an application are valid within the application. The main members of the application include global variables, tasks, and program organizational units (POU).

- Access path

The main function of access path is to link global variables, direct representation variables and input/output variables of program organization unit to realize information storage. It provides a method to exchange data and information between different applications. Variables in each application can be accessed through other remote configurations.

- Communication function

Provide communication with other systems, such as other programmable controller systems, robot controllers, computers and other devices, for program transmission, data file transmission, monitoring, diagnosis, etc. Generally, communication methods conforming to international standards (such as RS232, RS485) or industrial field buses such as CANopen, EtherCAT, MODBUS, Ethernet/IP, DeviceNet, etc. are adopted.

2-1-2. Characteristics of software model

The Codesys software model has the following features:

- Codesys software model can load, start and execute multiple independent programs in one PLC at the same time.
- Codesys software model can realize full control over program execution. Through the standard task mechanism, the PLC system can fully control the program execution. The traditional PLC program can only scan the execution program in sequence, and can not execute a certain program regularly according to the actual requirements of the user. The task mechanism in the software model allows different parts of the program to execute in parallel at different times and at different rates, which greatly expands the application scope of PLC.
- Codesys software model is an international standard software model, which can adapt to different PLC structures. It is not only for specific PLC system, but has strong applicability. It is suitable for both small PLC systems and large distributed systems.
- Codesys software model supports the reusability of program organization unit: software reusability is an important advantage of Codesys.
- Codesys software model supports hierarchical design: a complex software can be decomposed into manageable program units through layer by layer.

2-2. Device

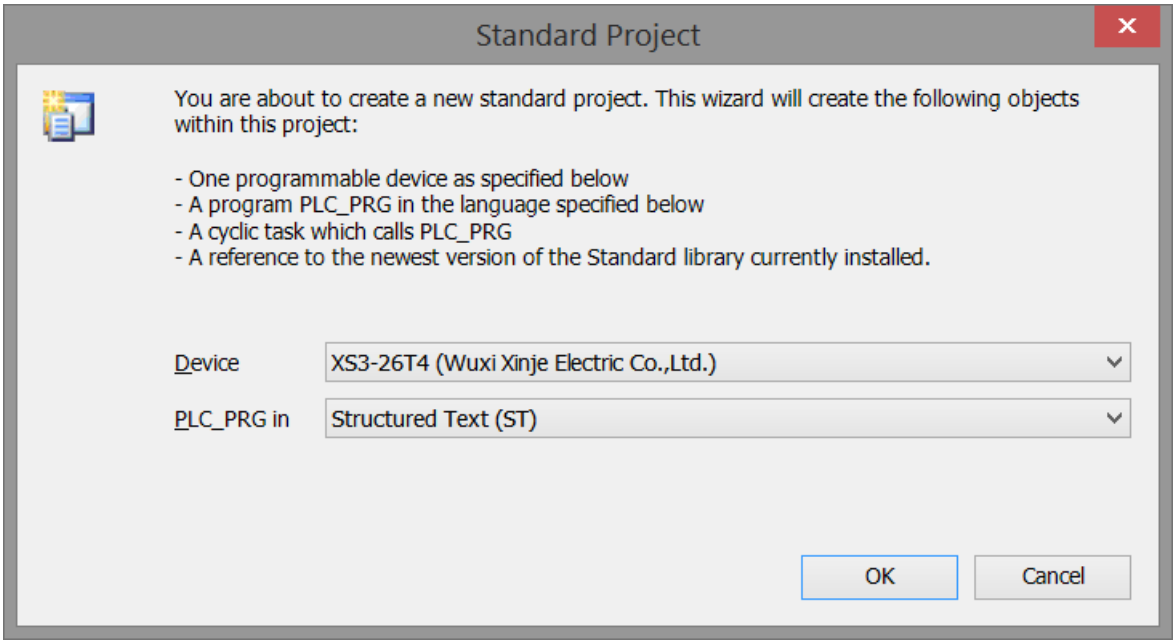
The device represents a specific target, that is, the hardware object, which is located at the top of the Codesys software model. The hardware object can be a controller, a fieldbus site, a bus coupler, a driver, an input / output module or a touch screen. Each device is defined by a "device description" file, which is installed in the Codesys native system for insertion under the device tree (the "device tree" here represents the tree list in the device window). The device description document determines the relevant configuration, programmability and interconnection with other devices. Device is a structural element, which is located at the top level of the software model. It is a large language element inside the software.

2-2-1. Device management

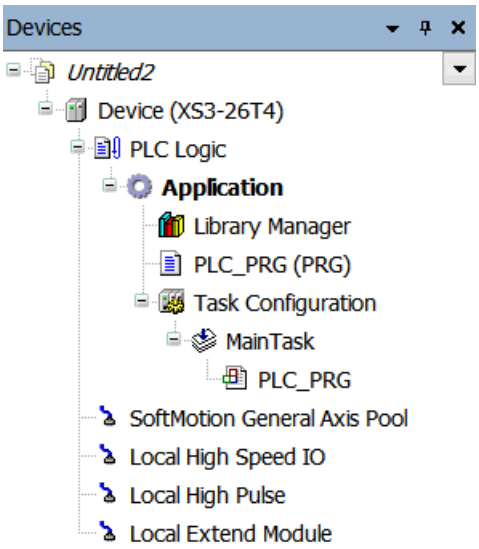
The management of equipment includes the addition of equipment, the management of installation package and the management of equipment library.

1. Add device

When creating a new project, a dialog box will pop up automatically, as shown in the following figure. You can select to create an empty project or a standard project in the template option. When selecting a standard project, you need to select the actual connected hardware device.



Click OK to get the following device tree.



2. Package manager

All "devices" must be installed in the "package manager" in advance. The package manager can be selected in the "tools" menu, and users can add or delete packages.

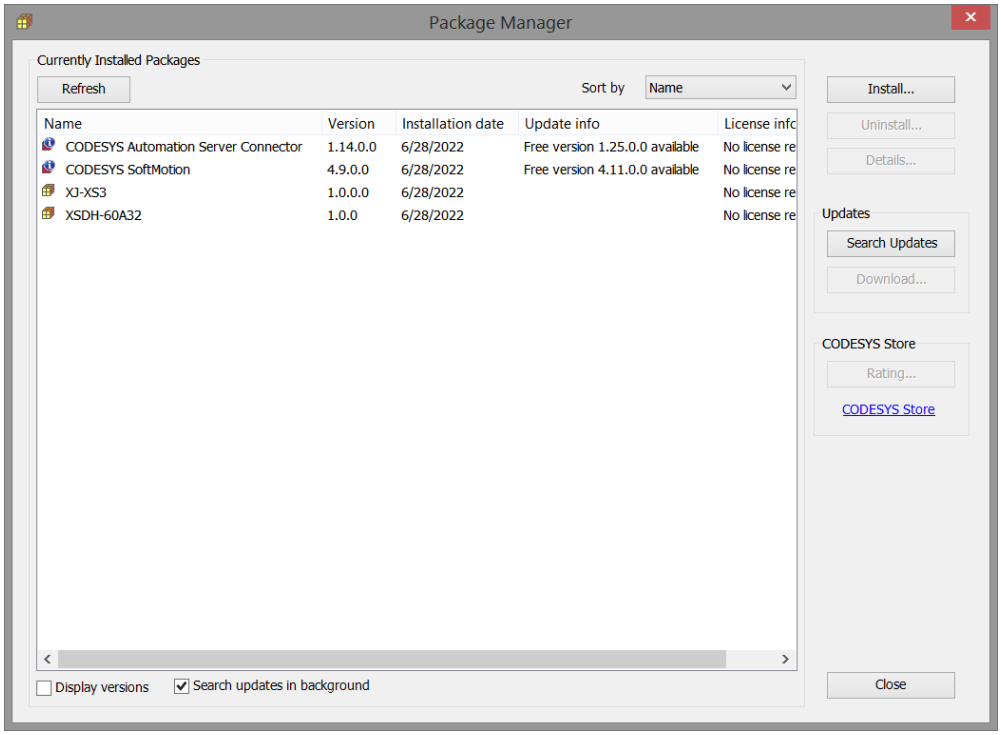
Different hardware configuration parameters are required for different hardware devices. The parameters that must be configured include code generator, memory management, PLC function, I/O module configuration. In addition, the library, gateway driver, INI files for error messages and relevant information of PLC browser must be linked. In addition, the package integrates special functions, including corresponding library files, device description files, etc.

The package manager installation process for this product is as follows:

Open "tools" and select "package manager".

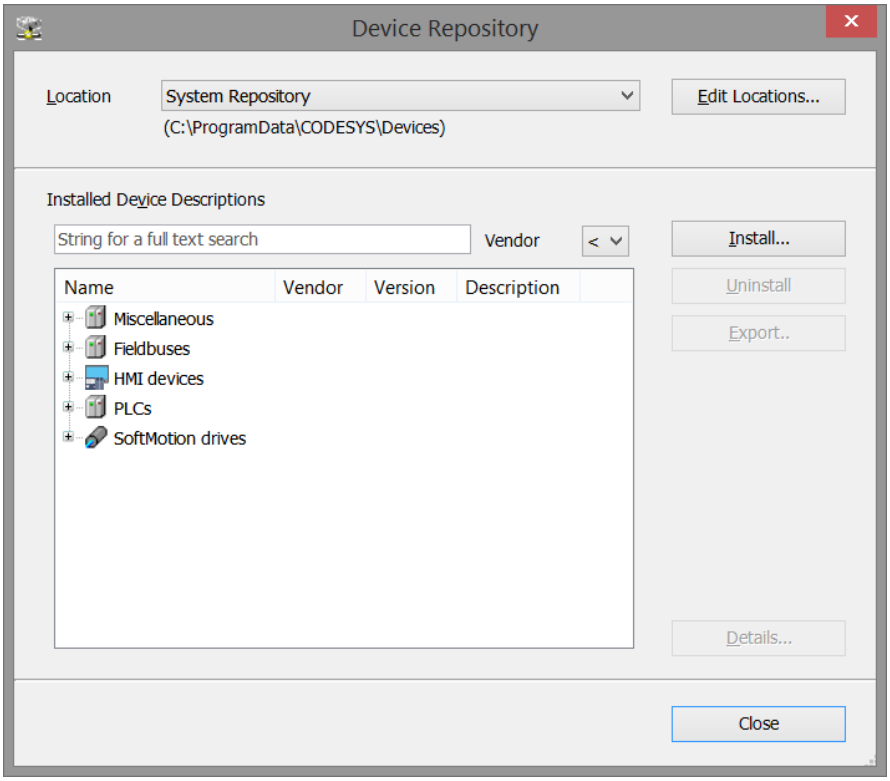
Click "Install" and find the corresponding installation package in the directory. This example uses XJ_XS3.package.

Click OK, the installation is successful, and the "XJ_XS3" icon will be displayed in the package manager, as shown in the following figure:



3. Device library management

The device library is the operation that the user needs to do when adding or deleting hardware device information. The device library is the database of the device. All data after installation is imported into the user's local system for Codesys development. The device library dialog box is shown in the following figure:



The device library can be used to add all hardware devices. After importing the corresponding files in this option, the corresponding data can be generated in the local system for easy calling in the project. The device that can be added include the supplier's PLC, softmotion motion control equipment (encoder, driver, etc.), fieldbus, special interface and other equipment.
The device description files that can be added to this product include the device description files of the ontology and extension modules officially provided by Xinje, the XML files of EtherCAT, the EDS and DCF files of CANopen, the IODD of IO-Link and the GSD files of Profibus DP, etc.

2-2-2. Device Editor

The device editor is a dialog box for configuring devices. Open by selecting the device icon, right clicking the edit object command, or double clicking the device object entry in the device window.
The main dialog box is named by the device name according to the device type. This product provides tabs containing the following sub dialog boxes, as shown in the following table:

Communication setting	Configuration related to the connection between the target device and other programmable devices (PLCs)
Application	Display the configuration of device parameters respectively
Backup and restore	Backing up application specific files on the PLC
File	Configuration of file transfer between host and PLC
Log	Display log file of PLC
PLC setting	Application related to I/O operation, I/O status in stop state, configuration of bus cycle options
PLC command	PLC can be configured through shell command
Users and groups	User management related to equipment access during operation (not to be confused with engineering user management)
Access rights	Configuration of access rights for running objects and files by special user groups
IEC object	Access to device "objects" through IEC applications
Clock I/O mapping	Provide real time clock
Task deployment	Displays input and output tables and their assignments to defined tasks
Status	Detailed status and diagnostic information of equipment
Information	Basic information of equipment (name, supplier, version, serial number, etc.)

2-3. Application

An application is a collection of objects required to run a program on a hardware device (such as a PLC). These objects are independent of the hardware device platform, and users can manage them in the program organization unit (POU). Then instantiate them in the device window and assign them to specific devices. This method accords with the idea of object-oriented programming.

Application objects include tasks, program organization units, task configurations, global variables, library managers, and sampling traces. The resource objects in Codesys v3.x can only be managed in the device tree. After adding objects to the device tree, it is necessary to map with the controlled device according to certain "rules". The effective range of objects (such as libraries and global variable lists) in the project depends on the hierarchical relationship between applications and device objects in the device tree. Generally speaking, an object in an application is also valid for its "sub applications" and can be used.

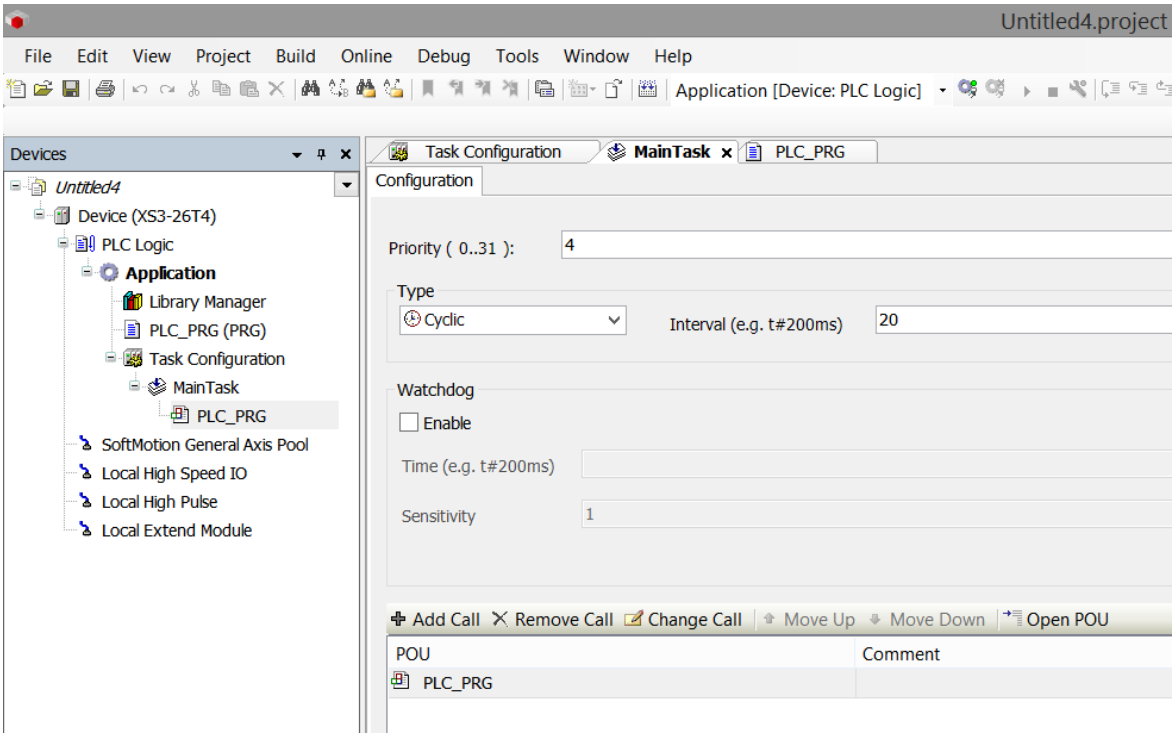
2-3-1. Task

1. Overview

A program can be written in different programming languages. A typical program is composed of many interconnected function blocks, which can exchange data with each other. The execution of different parts of a program is controlled by "tasks". After the "task" is configured, a series of programs or function blocks can be executed periodically or triggered by a specific event.

There is a task manager tab in the device tree, which can be used in addition to declaring specific PLC_PRG, it can also control the execution and processing of other subprograms in the project. Task is used to specify the attributes of the program organization unit at run time. It is an execution control element with the ability to call. In a task configuration, multiple tasks can be established, and in a task, multiple program organization units can be called. Once the task is set, it can control program cycle execution or start execution by triggering specific events.

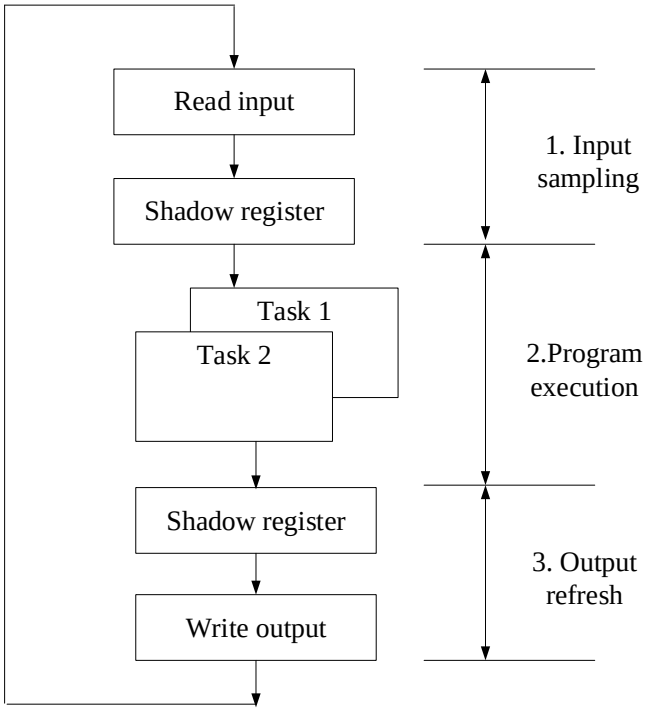
In task configuration, it is defined by name, priority and task startup type. This startup type can be defined by time (periodic, random) or by internal or external trigger task time, such as using the rising edge of a Boolean global variable or a specific event in the system. For each task, you can set a series of programs started by the task. If this task is executed in the current cycle, these programs will be processed within the length of one cycle. The combination of priority and condition will determine the timing of task execution. The task setting interface is shown in the following figure:



Users should follow the following rules when configuring tasks:
(1) The maximum number of circular tasks is 100
(2) The maximum number of free running tasks is 100
(3) The maximum number of event triggered tasks is 100
(4) According to the target system, PLC_ PRG may be executed as a free program in any case without inserting into the task configuration
(5) Processing and calling programs are executed from top to bottom in the task editor.

2. PLC program execution process

The following figure describes in detail the complete process of executing the program inside the PLC, which is mainly composed of three important steps: input sampling, program execution and output refresh.



(1) Input sampling

At the beginning of each scanning cycle, PLC detects the status of input devices (switches, buttons, etc.) and writes the status into the input image area. In the program execution stage, the running system reads data from the input image area for program solution. The refresh of the input image area only occurs at the beginning of a scan. During the scan, even if the output state changes, the input state will not change.

(2) Program execution

In the execution program stage of the scanning cycle, the soft PLC reads the status and data from the input image area or the output image area, and performs logical and arithmetic operations according to the instructions. The results of the operations are saved in the corresponding units of the output image area. At this stage, only the contents of the input image register remain unchanged, and the contents of other image registers will change with the execution of the program.

(3) Output refresh

The output refresh stage is also called the write output stage. The PLC transmits the status and data of the output image area to the output point, isolates and amplifies the power in a certain way, and drives the external load.
In addition to completing the tasks of the above three stages within a scanning cycle, PLC also completes auxiliary tasks such as internal diagnosis, communication, public processing and input / output services.
According to the scanning mode of PLC, in order to quickly respond to the changes of input and output data and complete the control task, the scanning time of PLC is relatively short, and the scanning time of PLC is generally

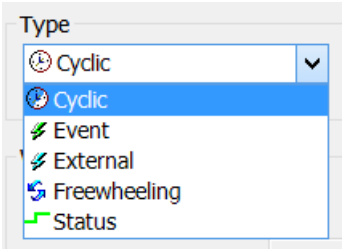
controlled in ms. therefore, it is necessary to develop a stable, reliable and fast response real-time system for the PLC operation system.

The PLC repeats the above processes (1) to (3), and the time for each repetition is a working cycle (or scanning cycle).
From the working process of PLC, it can be seen that since PLC adopts the circular working mode, the input signal will only be refreshed at the beginning of each cycle, and the output will be output intensively at the end of each cycle. Therefore, the lag between the output signal and the input signal is inevitable.
It takes a period of time from the change of a signal input at the PLC input to the response of the PLC output to the change of the input signal. Lag time is an important parameter that should be understood when designing PLC control system.
The lag time is related to the following factors:

- The filter time of the input circuit is determined by the time constant of the RC filter circuit. The input delay time can be adjusted by changing the time constant.
- The lag time of output circuit is related to the mode of output circuit. The lag time of relay output mode is generally about 10ms, and the lag time of transistor output mode is less than 1ms.
- The working mode of PLC is cycle scanning.
- The arrangement of statements in a user program.

3. Task execution type

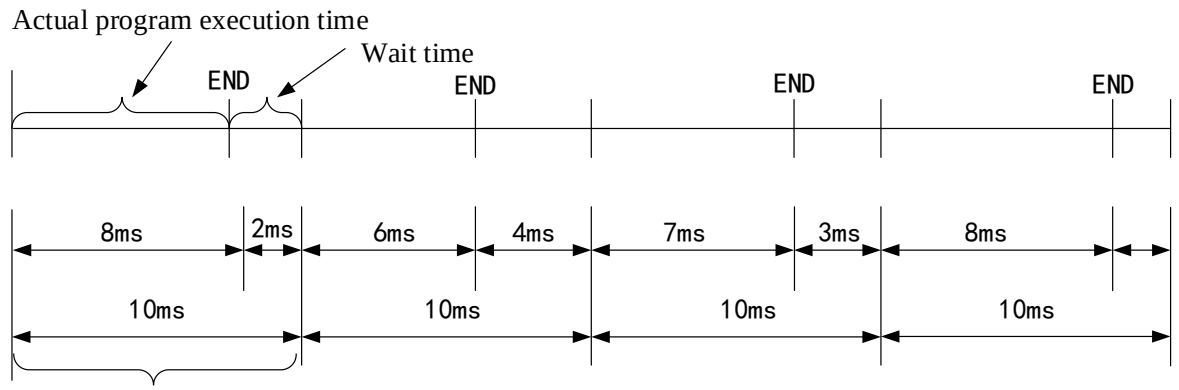
There is a "task configuration" at the top of the task configuration tree. The following are the currently defined tasks, each represented by a task name. The POUs calling for a specific task is not displayed in the task configuration tree.
For each independent task, you can edit and configure its execution type. It includes cyclic, event, external, freewheeling and status. See the figure below for details.



(1) Cyclic

According to whether the instructions used in the program are executed or not, the processing time of the program will be different, so the actual execution time will change differently in each scanning cycle, and the execution time will vary from long to short. By using the fixed cycle mode, the program can be executed repeatedly for a certain cycle time. Even if the execution time of the program changes, a certain refresh interval can be maintained. Here, it is also recommended that you give priority to the fixed cycle task startup mode.

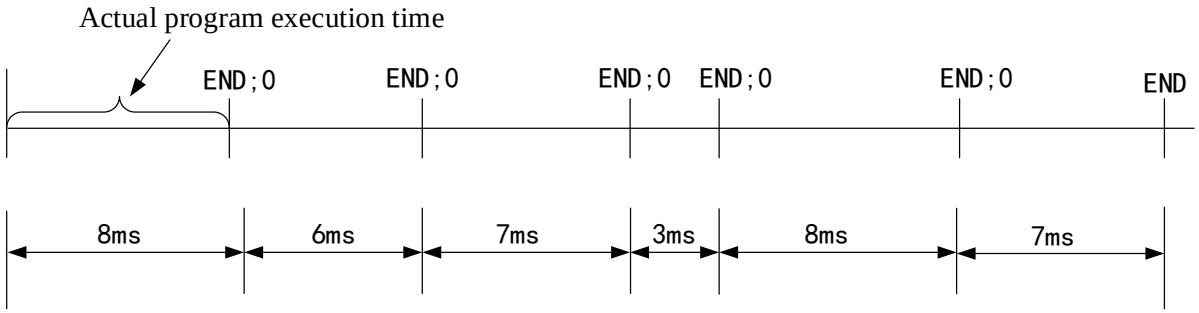
For example, suppose that the task corresponding to the program is set to the fixed cycle mode, and the interval time is set to 10ms, the sequence diagram of the actual program execution is shown in the following figure:



If the actual execution time of the program is completed within the specified fixed cycle setting time, the spare time is used as waiting. If there are tasks with lower priority in the application that have not been executed, the remaining waiting time is used to execute tasks with lower priority. See the description of task priority for details.

(2) Freewheeling

The task will be processed as soon as the program starts running. After one running cycle, the task will be automatically restarted in the next cycle, as shown in the following figure. It is not affected by the program scanning cycle (interval time). That is to ensure that each time the last instruction of the program is executed, the next cycle is entered. Otherwise, the program cycle will not end.



This execution method has no fixed task time, so the execution time may be different each time. Therefore, the real-time performance of the program cannot be guaranteed, and this method is rarely used in practical applications.

(3) Event

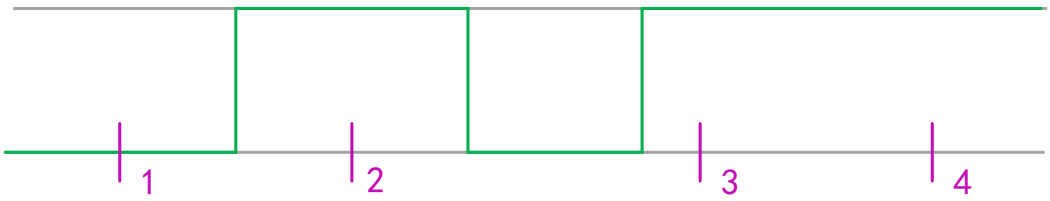
If the variable in the event area gets a rising edge, the task starts.

(4) Status

If the variable in the event area is true, the task starts.

The status triggering method is similar to the event triggering function, except that the program will be executed as long as the trigger variable of status triggering is true, and will not be executed if it is false. The event trigger only collects the effective signal of the rising edge of the trigger variable.

The following figure compares event triggering and status triggering respectively. The green solid line is the boolean variable status selected by the two triggering methods. The following table shows the comparison results.



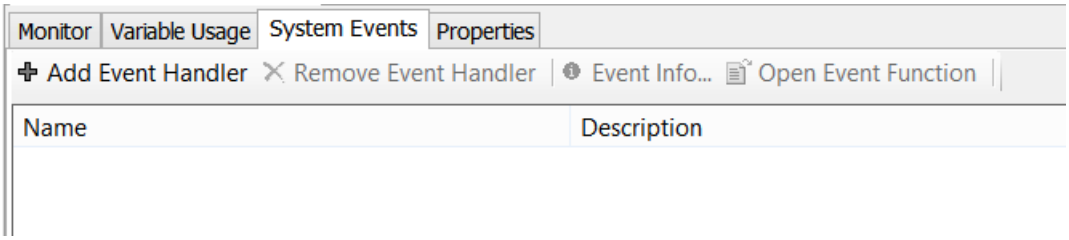
Different types of tasks showed different responses at sampling points 1-4 (purple). This specific event completes the condition of the state driven task for true. However, an event driven task requires the event to change from false to true. If the sampling frequency of the task plan is too low, the rising edge of the event may not be detected.

Execution point	1	2	3	4
Event	Not execute	Execute	Execute	Execute
Status	Not execute	Execute	Not execute	Not execute

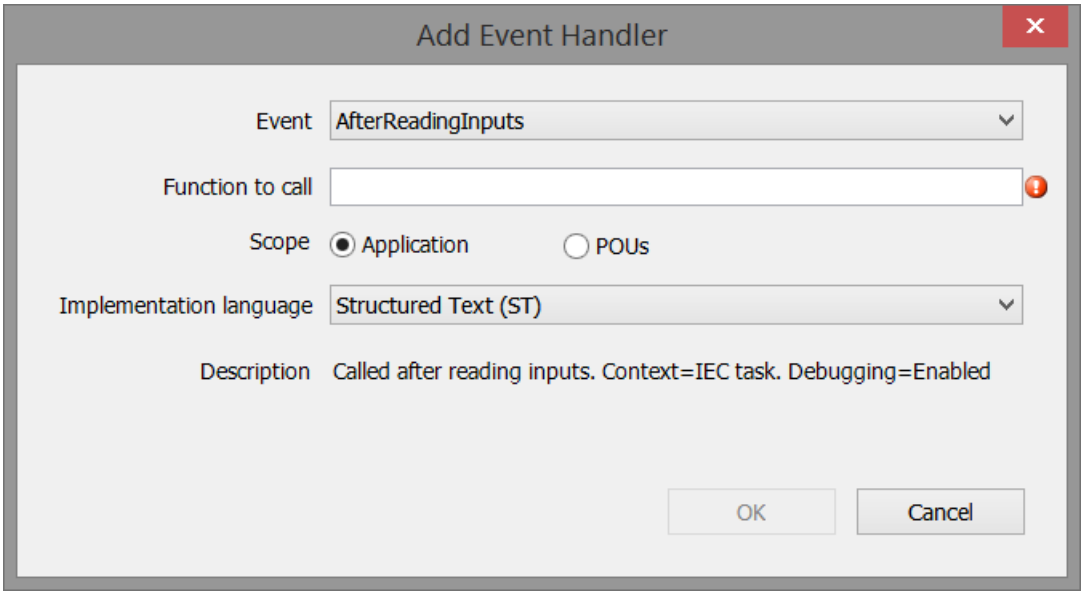
4. System events

The system events that can be selected by the user depend on the actual target system. The corresponding library file of the target system provides the corresponding system events. Therefore, the system events corresponding to different target hardware devices may be different. Common system events include: stop, start, login, change, etc. In task configuration, you can set system events in task configuration.

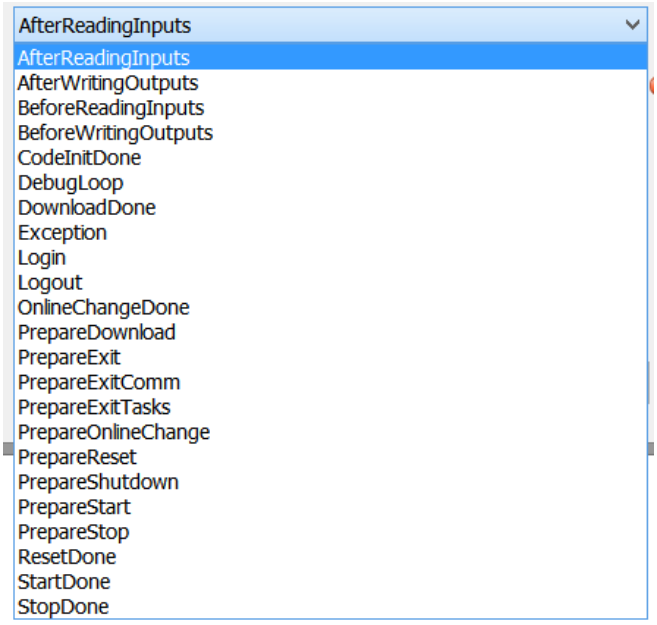
The user can select "task configuration" - > "system event" through the mouse to enter the interface shown in the figure below.



Select the "add event handler" button to add system events. The opened interface is shown in the following figure.

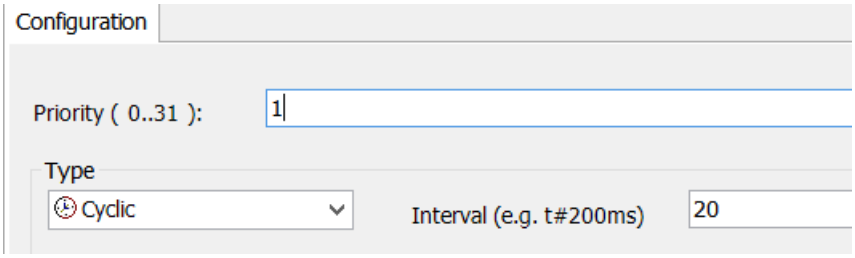


The "event" types that can be selected are shown in the following figure. You must create a new function name in "function to call" instead of using functions that already exist in the POU. "Implementation language" is the programming language of the corresponding function. Click "OK" after setting.



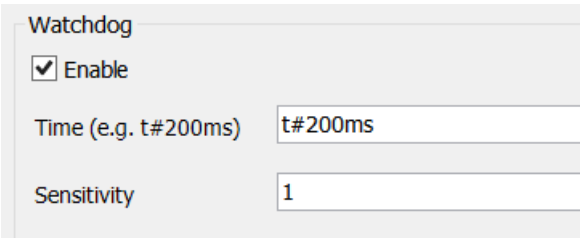
5. Task priority

Codesys software can set the priority of tasks. There are 32 levels in total (a number between 0 and 31. 0 is the highest priority and 31 is the lowest priority). When a program is executing, the task with high priority takes priority over the task with low priority. High priority task 0 can interrupt the execution of the program with lower priority in the same resource, so that the execution of the program with lower priority is slowed down.
If the task type is "cyclic", it will be executed according to the time cycle in "interval". The specific settings are shown in the following figure.



6. Watchdog

The watchdog is a kind of controller hardware timing device, which can be enabled through "task configuration" in Codesys. The watchdog function is not used by default.
The main function of the watchdog is to monitor the exception during program execution or the failure of the internal clock. For example, when the system crashes or the program enters the dead cycle, the watchdog timer will send a reset signal to the system or stop the program currently running by the PLC. We can understand it vividly as a puppy needs its owner to feed it regularly. If it is not fed after the specified time, it will be hungry immediately. To configure the watchdog, you must define two parameters, time and sensitivity. The configuration of the watchdog is shown in the following figure.



(1) Time

Codesys can configure independent watchdog for each task. If the target hardware supports long watchdog time setting, the upper and lower limits can be set. The default watchdog time unit is milliseconds (MS). If the program execution cycle exceeds the watchdog trigger time, the watchdog function will be activated and the current task will be aborted.

(2) Sensitivity

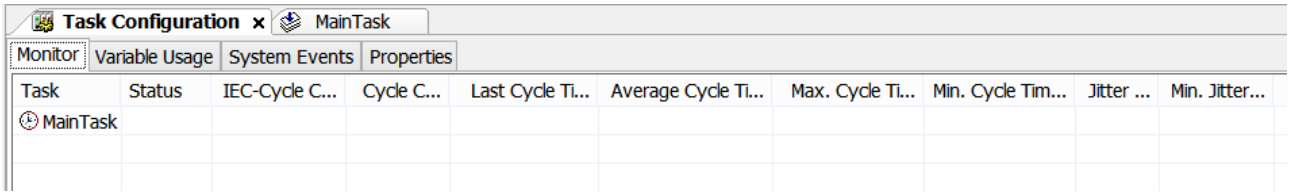
Sensitivity is used to define the number of task watchdog exceptions that must occur before the controller detects an application error. The default value is 1. Please refer to the following table.

Sensitivity	Multiple of set time exceeded
0,1	1
2	2
……	……
n	n

Final watchdog trigger time = time × sensitivity. If the actual execution time of the program exceeds the watchdog trigger time, the watchdog is activated. For example, if the time is 10ms and the sensitivity is set to 5, the watchdog trigger time is 50ms. As long as the task execution time exceeds 50ms, the watchdog will be activated immediately and the task will be aborted.

7. Task running status monitoring

Each task can be directly enabled or disabled, and the system will automatically configure a task monitor. After entering the online mode, the user can use the monitor provided by the system to monitor the task execution related parameters such as the average / maximum / minimum cycle time of the task. As shown in the following figure:



At the initial stage of the project, the maximum / minimum / average cycle time can be tested, which can be used to measure the stability of the program and optimize the task cycle time set by the program. See the following table for the specific definitions of each parameter in the monitoring window:

Parameter	Description	Parameter	Description
Task	Task name defined in task configuration	Average cycle time (μs)	Average execution time of task, unit: μs
Status	They have the following states: Not created: the consistency is not established after the program is downloaded. This state may occur when trigger task in the used time Create: the task has been established in the real-time system, but has not been officially run Effective: the task is being executed Exception: an exception occurred in the task.	Max /min cycle time (μs)	Task maximum/minimum execution time, unit: μs
IEC cycle count	The cumulative count of cycles since the program started running. '0' means the target system is not supported.	Jitter (μs)	Jitter value measured in the last cycle, unit: μs
Cycle count (μs)	Count of cycles that have been run. Depending on the target system, it can be equal to the IEC cycle count, or greater. In this case, even if the	Min/max jitter (μs)	Measured maximum/minimum jitter time, unit: μs

	application is not running, the cycle is also counted.		
Last cycle time (μs)	Task execution time of the previous cycle, unit: μs		

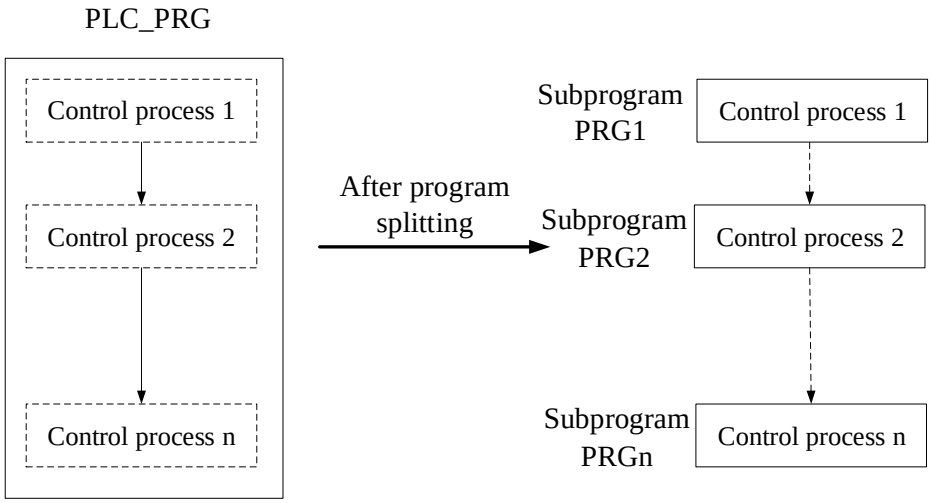
After understanding the definitions of the above times, the following time setting relationship should be followed. According to this setting method, the program task cycle and watchdog time can be better optimized to ensure the stability of the program and the real-time performance of the program.

Watchdog trigger time > fixed cycle time > program maximum cycle time

When the cycle time is longer than the fixed cycle time, the CPU will detect that the program has exceeded the count. At this time, the real-time performance of the program will be affected. If the program cycle time is longer than the watchdog time setting, the CPU will detect the watchdog fault and stop the execution of the program.

8. Running of multiple subprograms

In actual engineering projects, the program can usually be divided into many subroutines according to the control flow or according to the object of the equipment. Therefore, designers can program according to each processing unit. As shown in the figure below, the main program is divided into several subroutines with different processes by the control process. The purpose of splitting is to make the main program conditioning clearer and facilitate future debugging.



The right half of the figure above shows the subprograms PRG1, PRG2..PRGn classified by process, the left half of the figure is the main program PLC_ PRG, PRG1..PRGn can be called respectively in the main program.
There are two ways to run multiple subroutines. The first is to add subroutines to the task configuration. The second method is to call subroutines in the main program, which is also a common and flexible way.

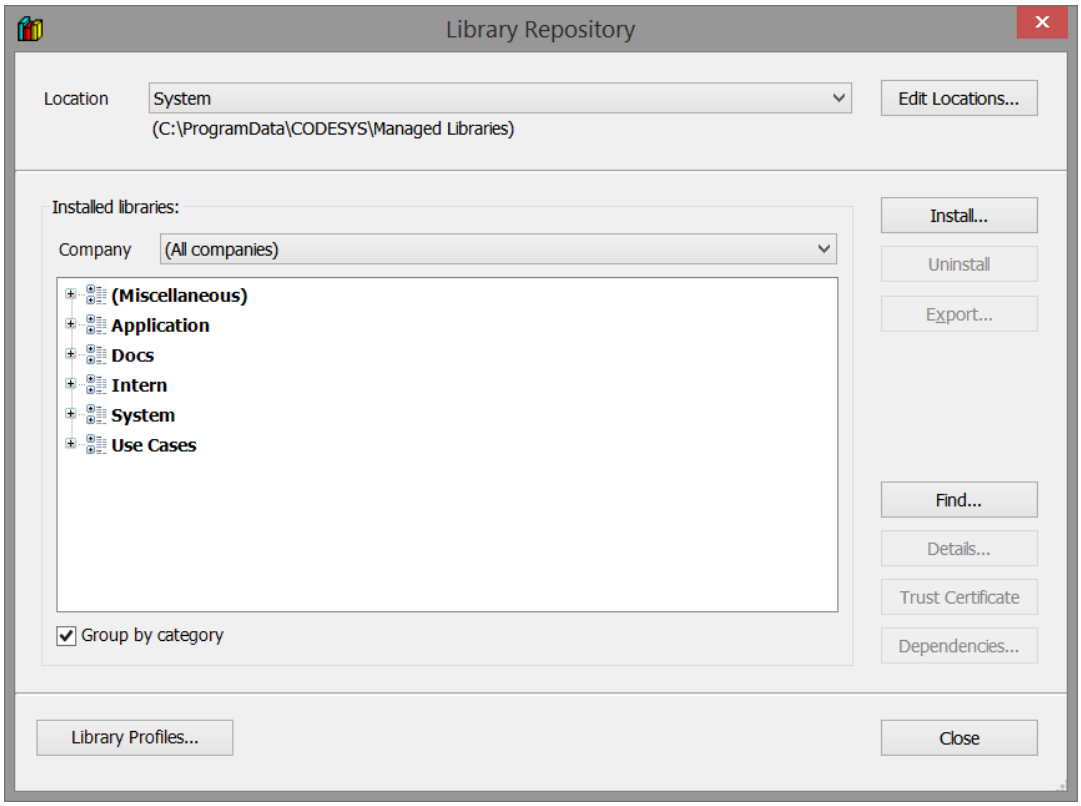
2-3-2. Library files

Library files are used to store program organization units (POU) that can be used multiple times in Codesys. Codesys provides a basic library. Users can construct a new library based on the basic library and reference it in the program by loading.
Library file is a collection of functions, function blocks and programs, which also contains some specially defined structures, enumeration types, etc. In terms of function, library files can be divided into system library files, application library files and manufacturer defined library files. Among them, the system library file is a file that supports Codesys software system, including support for software structure and syntax writing, as well as support for standard I/O. Application library file is a file library that supports basic applications, including data operation

function, timer, counter, edge detection, etc. The vendor defined library file is a specially made library file according to the product specifications of different manufacturers.

1. Management of library files

The library manager displays all libraries related to the current project. The POU, data type and global variables of the library can be like user-defined POU and data class. The library manager is opened through the library manager command, and relevant information including the library is saved together with the project.
If you need to install the library file on the computer or call the library file provided by the supplier, you need to use the library file management. Library file management is defined by using the menu command "tools" - > "library repository". The following figure shows the view of library file management.



The categories of displayed library files include application, communication, controller, device, system, etc.

The use process of library files is as follows:
(1) Installation of library files. Before using a library file, you must first "Install" it in the Library dialog box. After installation, the library can be called in the project.
(2) Call of library file. After installing the library file, you need to add the library file through the library manager to realize the call of the project to the library file.

2. Properties of library files

The library file needs to realize the uniqueness and security of access.

(1) Access uniqueness. If several modules or variables in a project have the same name, the paths to access variables with the same name must be different (that is, "unique access"), otherwise compilation errors will occur. This rule applies to local projects, libraries, and modules or variables in libraries referenced by other libraries. Users can achieve unique access by adding a namespace before the module or variable name.
(2) Access security. Codesys provides library file encryption function to protect the source code of developer library files. By adding permission information to the library file in the project settings and saving it as a "compiled function library", the user needs to log in with a password to open the library file next time. If the password is wrong, the library file cannot be used and opened, and a log alarm is triggered.

2-3-3. Access path

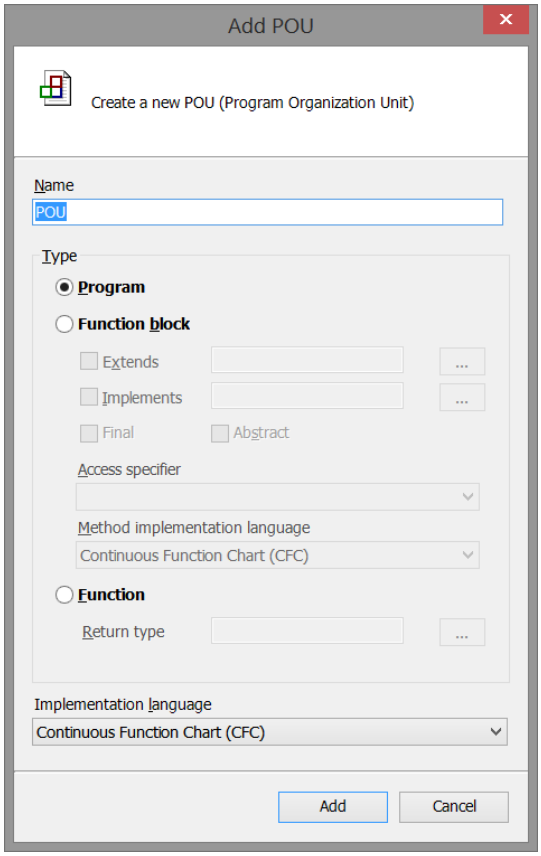
The access path is used to connect global variables, direct representation variables, input/output of function blocks and local variables to realize the storage of information. It provides a method to exchange data and information between different configurations. Many variables with specified names in each configuration can be accessed through other remote configurations.

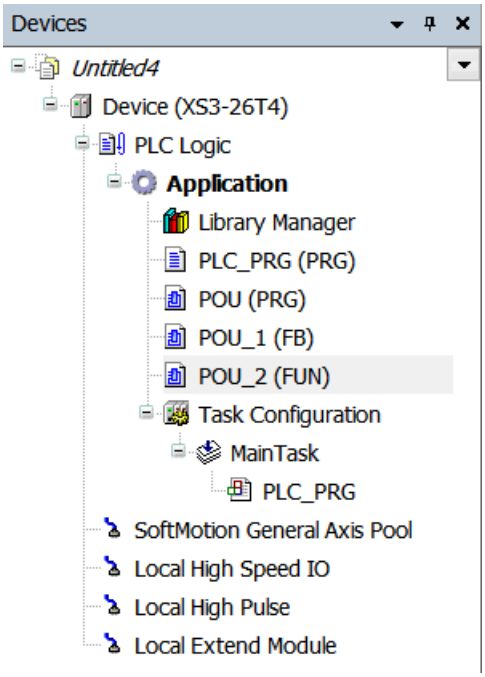
The access path function has been integrated into Codesys. Users do not need to operate it. All access operations will be carried out automatically in the background of Codesys.

2-4. POU

Program organization unit (POU) is the smallest program unit of user program, which is composed of declaration area and code area. It is the basis for a comprehensive understanding of new language concepts. According to function, program organization unit (POU) can be divided into function (FUN), function block (FB) and program (PRG).

Right click “application”, click “add object…” --- “POU”, which will pop up below figure. In the dialog box, users can choose to add programs, function blocks or functions, and the corresponding programming language can be selected in the drop-down menu. After adding, you can view the corresponding attributes in the brackets of the POU in the project device tree on the left. FB is the function block, FUN is the function, and PRG is the program.



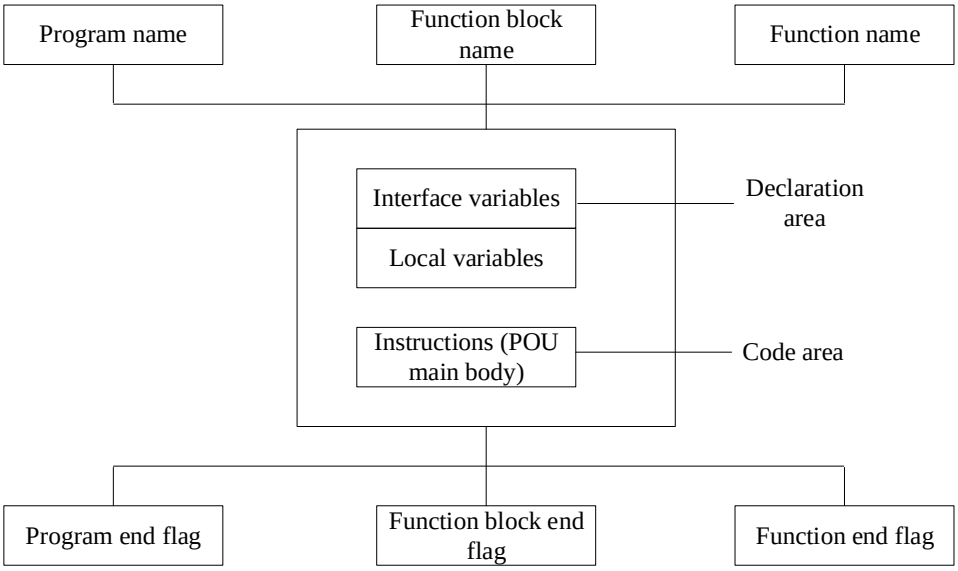


The program organization unit has the following characteristics:

- User's function block library can be set for each application field, which is convenient for engineering application. For example, establish a library of motion control function blocks
- Function blocks can be tested and recorded
- It can provide global library inventory retrieval function
- It can be used repeatedly, and the number of times of use is unlimited
- The programming can be changed to establish the function block network.

2-4-1. POU structure

A complete POU consists of three parts: POU type and naming, variable declaration part and code instruction part (POU body). The structure diagram is as follows:



In the above figure, from the perspective of specific functions, the program (PRG) on the left, the intermediate function block (FB) and the function (fun) on the right can be formed respectively. From the structure of each function, it can be divided into declaration part and code part.

All variables declared by the user are ultimately used by the program organization unit. Interface variables and local variables can be declared in the variable declaration.

1. Declaration area

The variable declaration area is used to specify the name, type and initial value of variables.

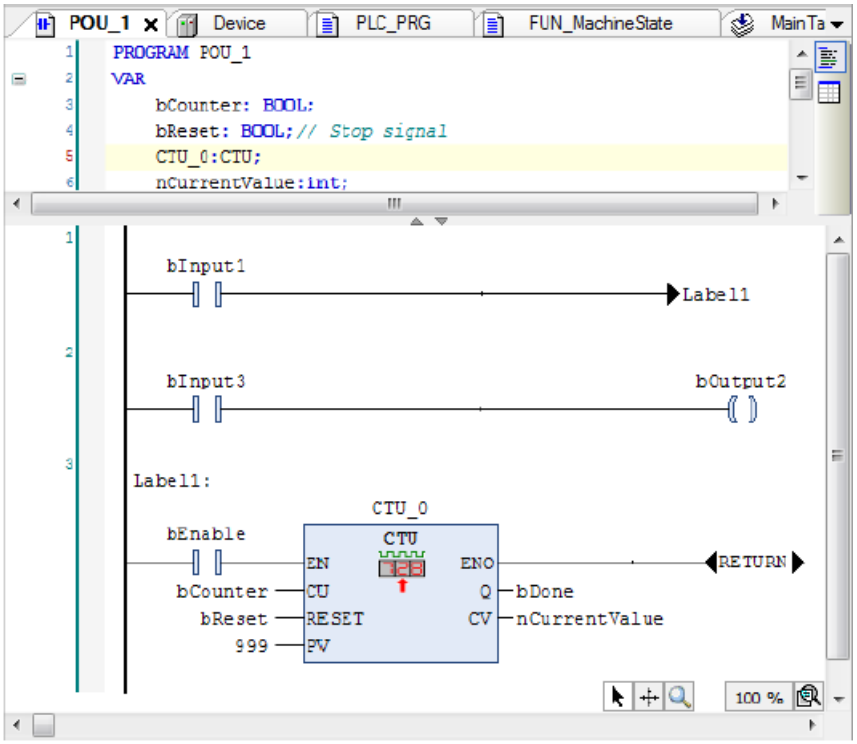
The variable declaration editor is used to declare POU variables and data types. The declaration part is usually a text editor or a table editor. All variables to be used in this POU are declared in the declaration part of the POU, including input variables, output variables, input / output variables, local variables, added variables and constants. The declaration format is based on IEC61131-3 standard. The declaration of variables adopts the following format:

< identifier >{AT<Address>}: <data type>{: =< initialization >}:

Part of {} is optional.

2. Code area

In the code area, Codesys supports two text languages: instruction list (IL) and structured text (ST). Four graphical languages: function block diagram (FBD), ladder diagram (LD), sequential function diagram (SFC) and continuous function diagram (SFC). Users can choose one or several languages to program in the main part. The main editor interface is shown in the figure below, in which ladder diagram (LD) program language is used.



2-4-2. Function

For the application of PLC programming language, function (FUN) is also defined as a program organization unit. Function is a program organization unit that can be assigned parameters but has no static variables. That is, when a function is called with the same input parameters, the function can always generate the same result as the function value (return value). An important feature of functions is that they cannot use internal variables to store values, which is completely different from function blocks.

Function (FUN) is a basic algorithm unit with no internal state (no memory allocation at runtime). In other words, as long as the same input parameters are given, the calling function must get the same operation result, and there is absolutely no ambiguity. Various mathematical operation functions we usually use, such as sin (x), sqrt (x), etc., are typical function types.

A function is a basic algorithm unit with at least one input variable, no private data, and only one return value.

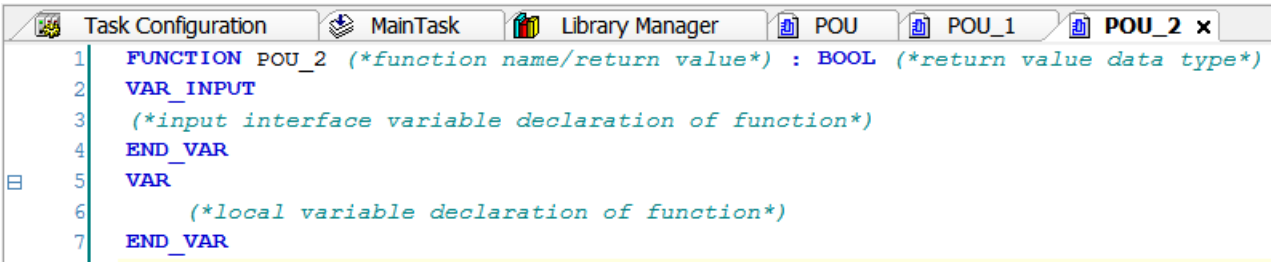
Standard functions are already pre-existing in the standard library of Codesys.

Functions can be used by functions, function blocks, and programs.

1. Representation and declaration of functions

(1) Representation of custom functions

The internal logic part of the function can use any of the six programming languages. The function name is the return value of the function, which can also be understood as the output value of the function, as shown in the following figure:



(2) Declaration of variables in functions

When users customize functions, they should pay attention to the following matters:

- A function can have many input variables, but only one return value (output variable). However, there is no restriction on the data type of the return value, so it can be a structure as the return value.
- The important feature of functions is that they cannot store values in internal variables, which is different from function blocks.
- The function has no specified memory allocation and does not need to be instantiated like a function block.
- Functions can only call functions, not function blocks.
- The argument configured to VAR_INPUT can be empty, constant, variable or function call. When the function is called, the function is called as the actual argument.

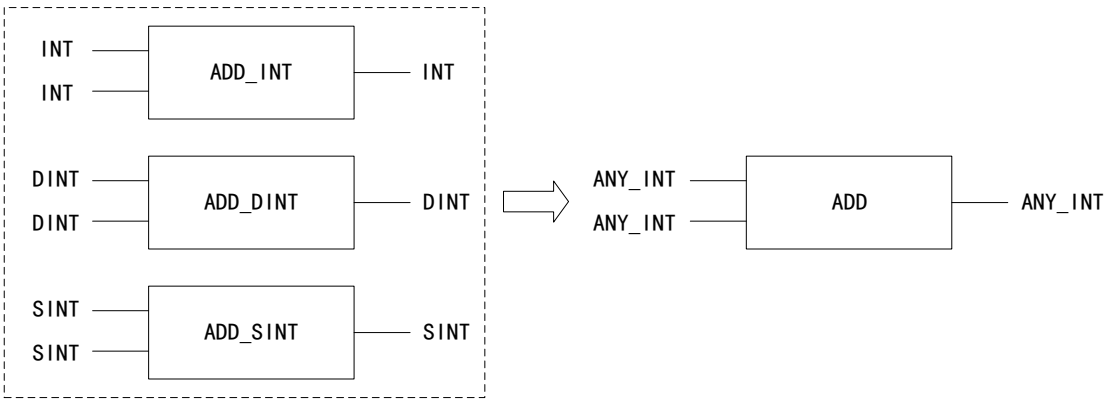
2. Standard functions

Codesys supports all IEC class 8 standard functions. In addition, the following functions not specified in IEC standards can be used: ANDN, ORN, XORN, INDEXOF, SIZEOF, ADR, BITADR, etc. Codesys supports the following 11 types of functions. The use and description of specific functions will be introduced in detail in Chapter 6.

3. Properties of function

(1) Overloaded property

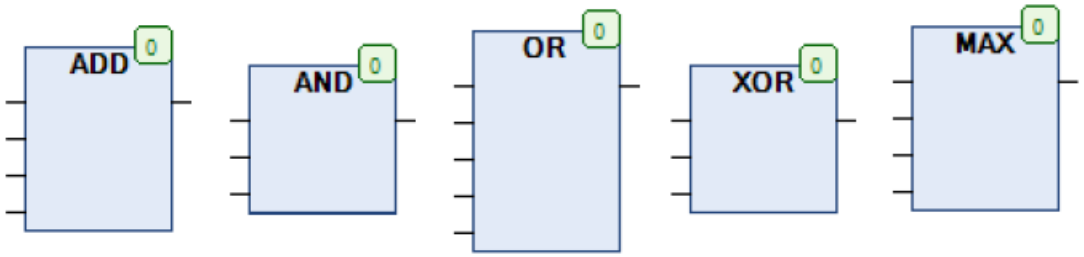
For a function, if its input is described by generic data type, it is called overloaded function. This means that the input of this function is not limited to a single data type, but can be used for different data types. All standard functions of Codesys have overload properties, which can be applied to different data types. If a function is only applicable to a certain data type, it needs to be declared in the function name, which is called function typing. For example, if a PLC can recognize INT, DINT and SINT, it supports overload function ADD of generic data type ANY_INT (including BYTE, WORD, DWORD, SINT, USINT, REAL, etc.). For example, ADD_INT is an INT addition function limited to data types. It is a typed function. In this way, the overload function is independent of type. The description of overloaded functions is shown in the following figure:



When using overloaded functions, the system will automatically select the appropriate data type. For example, if the called ADD argument data type is DINT, the system will call ADD_DINT standard functions.

(2) Scalability

The property that the number of input variables of a function can be extended is called the extensible property of a function. For example, the input variables of the ADD function can be more than two. It can realize the addition of multiple input variables. Therefore, the add function can be said to have extensible properties. Not all standard functions have extensible attributes. The extension limit of this function is subject to the upper limit imposed by PLC, the height limit of the box in the graphic programming language, or the function definition limit of the function itself. For example, DIV function has this attribute. Functions with extensible properties can simplify the program and reduce the required storage space. The following figure is an example of some functions with extensible properties.

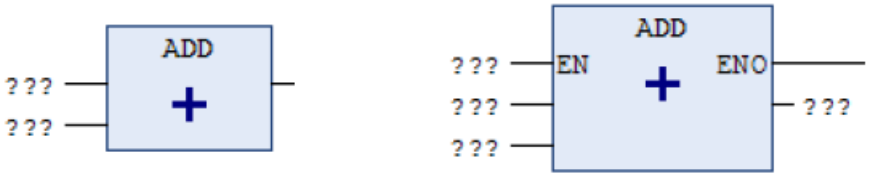


(3) EN and ENO

This attribute is valid only in ladder and function block diagram programming languages. EN and ENO are the input enable and output enable of the function respectively. All functions can enable or disable this property.
The application principles of enable input and enable output are as follows:

- When the input function is called, the value of EN is false, then the operation defined by the function body will not be executed by the program, and the value of ENO is false.
- When EN is true, the function is called, the operation defined by the function body is executed, and the value of ENO is true.
- EN and ENO attributes are additional attributes, which can be enabled or disabled according to actual needs.

The following figure compares the ADD function with EN/ENO with the ordinary ADD function.



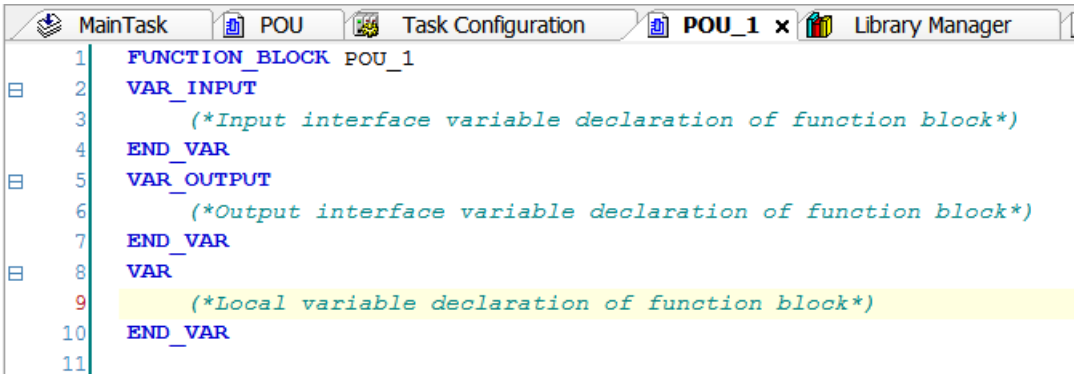
2-4-3. Function block

Function block is to convert some program blocks that are used repeatedly into a general component. It can be called by any programming language in the program and used repeatedly, which not only improves the development efficiency of the program, but also reduces the errors in programming, thus improving the quality of the program.
A program organization unit that can generate one or more values when a function block is executed. The function block retains its own special internal variables, and the controller target execution system must allocate memory to the internal state variables of the function block, which constitute its own state characteristics.
The execution logic of the function block constitutes its own object behavior characteristics. Therefore, for the input variable value of the same parameter, there may be different internal state variables, so different calculation results may be obtained. In the control system, the function block can be some kinds of control algorithm, such as PID function module is used for closed-loop control, and other function blocks can be used for counters, slopes, filters, etc.

1. Representation and declaration of function blocks

(1) Representation of custom function blocks

Like functions, the internal logic part of function blocks can use any of the six programming languages. The function name is the return value of the function, which can also be understood as the output value of the function. The following figure is the syntax expression of the function block.



(2) Declaration of variables in function blocks

Variable declarations in function blocks are similar to those in functions. When writing, you should pay attention to the following matters:

- The internal and output variables of a function block can use the qualified attribute RETAIN to indicate that the variable has a hold function. Input variables can only be declared with retain properties at the time of invocation.
- It is generally not allowed to assign values to function block input variables. Only when the input is the calling part of the function block, it is allowed to assign a value to the input variable of the function block.
- Since function blocks can call functions and function blocks, you can also call function block instances as variables of instances of other function blocks. Such as DB_ FF(S1:=DB_ON.Q, R:=DB_OFF.Q).
- The input of function blocks is not assigned, which means that their initial values are maintained.
- To ensure that the function block does not depend on hardware, address variables with fixed addresses (such as %IX1.1, %QD12) are not allowed to be used as local variables in the variable declaration of the function block, but they can be assigned values when called.
- Use VAR_INPUT and VAR_OUTPUT will occupy too much memory. Therefore, VAR_IN_OUT can be used as much as possible when programming function blocks to reduce the occupation of storage area.

2. Standard function block

Bistable elements, edge detection, timers and other functional blocks have been included in the standard library.

3. Attributes of function blocks

(1) Instantiation

According to IEC61131-3 standard, the type of function block is the definition of abstract structure type, rather than real data entity. If it is not defined and instantiated, it cannot be called and executed by the program. Therefore, function blocks need to be instantiated before they can be used.

The instantiated function block is an independent structural variable that has private data, can complete specific functions according to the established logic, and is completely encapsulated. Thus, the previous abstract type definition is transformed into a data entity.

(2) Scalability

Codesys supports object-oriented programming, so function blocks can also derive "sub" function blocks. In this way, the "child" function block has the attribute of the "parent" function block, and can have its own additional characteristics. It can be visually considered that the "child" function block is an extension of the "parent" function block. So in this article, we call this "function block extension".

Add the keyword "extends" when declaring the function block to use the extended function. You can also expand by selecting the "extends" option when adding a function block in the "add object" dialog box.

(3) EN and ENO

Function blocks have the subsidiary attributes of EN and ENO, which are similar to the use of EN and ENO in functions.

(4) Differences between function blocks

To sum up, the obvious differences between functions and function blocks are summarized in the following table:

	Function (FUN)	Function block (FB)
Memory allocation	No specified memory allocation address	All data allocated memory address
Input/output variables	Only one output variable is allowed	Multiple output variables or no output variables
Calling relationship	Functions can be called, but function blocks cannot be called	Callable function block or function

2-4-4. Program

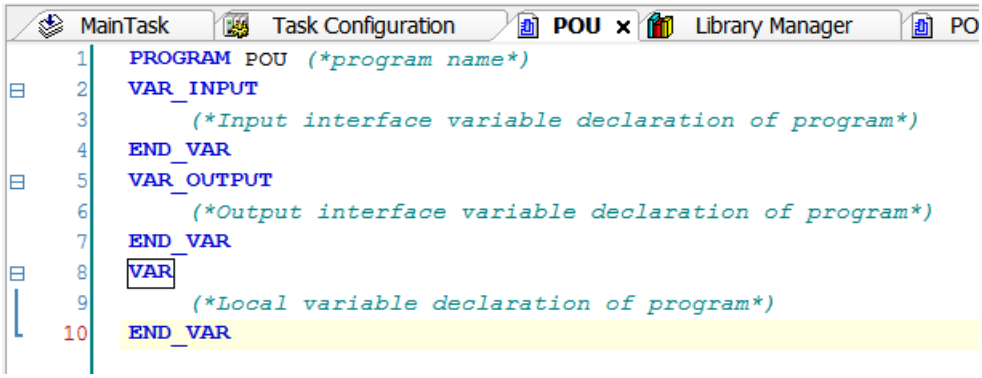
Program is the main core of planning a task. The program has the greatest call right and can call function blocks and functions.

Generally speaking, it is divided into main program and subroutine. In a broad sense, it also includes hardware configuration, task configuration, communication configuration and target setting information.

Generally, general global variables, mapped hardware address global variables and local variables are defined in the program. The application logic is realized by calling between programs.

1. Representation and declaration of program

The program is expressed by the following syntax expression, and the logic part of the program can use any of the six programming languages.



2. Program performance

(1) A program can contain the configuration of addresses. It is allowed to declare the direct representation variables that store the physical address of PLC, and the direct representation address configuration is only used for the declaration of internal variables in the program. Direct representation variables allow hierarchical addressing mode descriptions, such as the following representations.

You can fill in the program declaration in the following format.

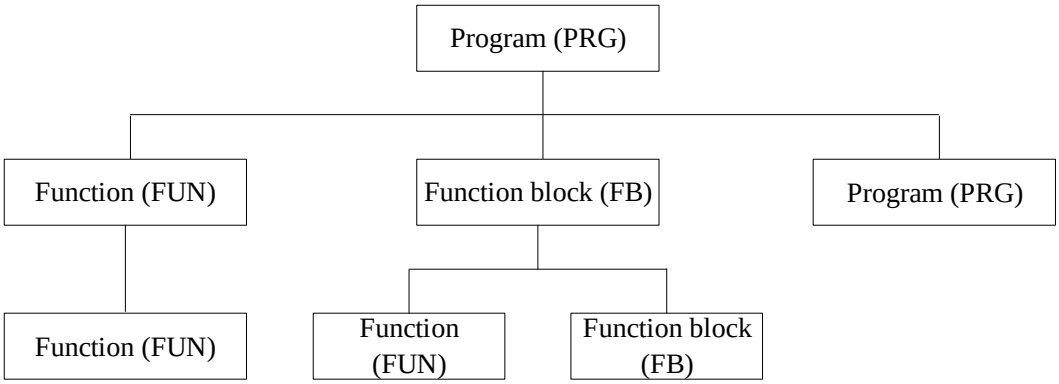
bTest AT %QX10.3:BOOL:=TRUE;

(2) A program organization unit cannot call itself directly or indirectly, that is, a program organization unit cannot call an instance of a program organization unit with the same type and name

(3) Programs are instantiated only in resources. Declared in the resource. An instance of a program only needs to combine the program with a task, otherwise it will not be executed. Function blocks can only be instantiated in programs or other function blocks.

3. Program calling relationship

It is allowed to call function block instances, function and other programs in the program, as shown in the following figure:



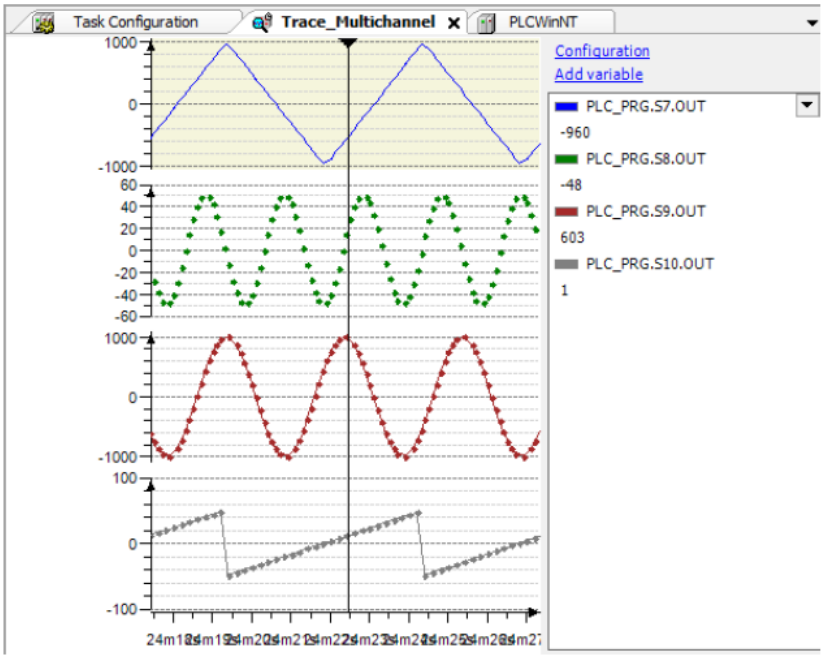
According to the above figure, functions and function blocks are used to form subroutines, and programs are used to form user main programs. Therefore, programs are considered global. Program is the largest form of program organization unit, which can call functions, function blocks and programs.

Function blocks can call other function blocks and functions. Since there are no private variables in the function, the function can only call other functions, not function block instances.

2-5. Application object

2-5-1. Sample tracking

The function of sampling and tracking is to monitor and track the history of variable values on the controller. The working mode of sampling tracking is similar to that of digital sampling oscilloscope. It is a very practical and effective debugging tool in the process of program debugging and diagnosis. The user can add the "tracking object" and set the "tracking configuration" in the tracking manager to record the command word, status word, motor speed, position and other parameters used in the execution of the program. The user can understand the whole process of the program running in the control system by observing these parameters. This function is shown in the following figure:

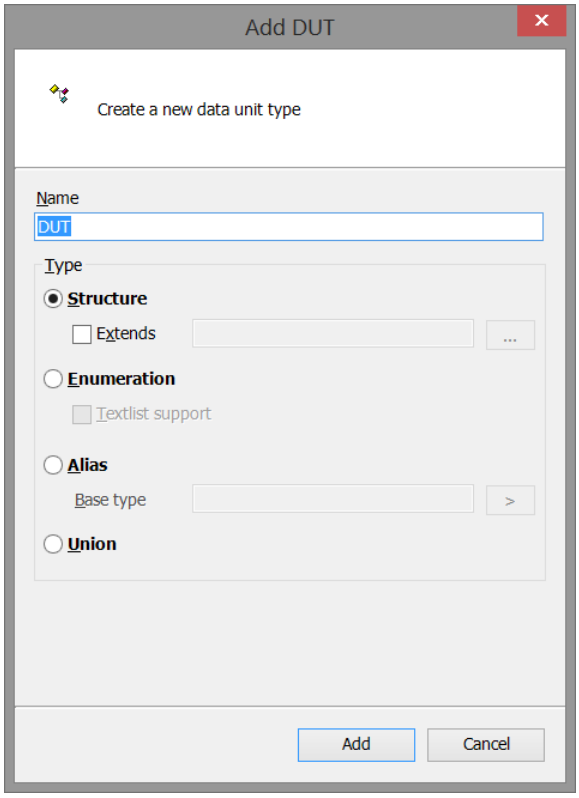


2-5-2. Persistent variable

The function of persistent variable is to save the data to the storage unit after the system is shut down or abnormal interruption, and call it out after power on again, and it can continue to be used by the program. In order to adapt to the on-site working conditions, when designing the PLC control system, it is necessary to consider the storage and recovery of data after power failure or abnormal interruption. Users can register the data that needs to be maintained during power failure in the list by adding the persistentvars list, which can realize the continuous variable function.

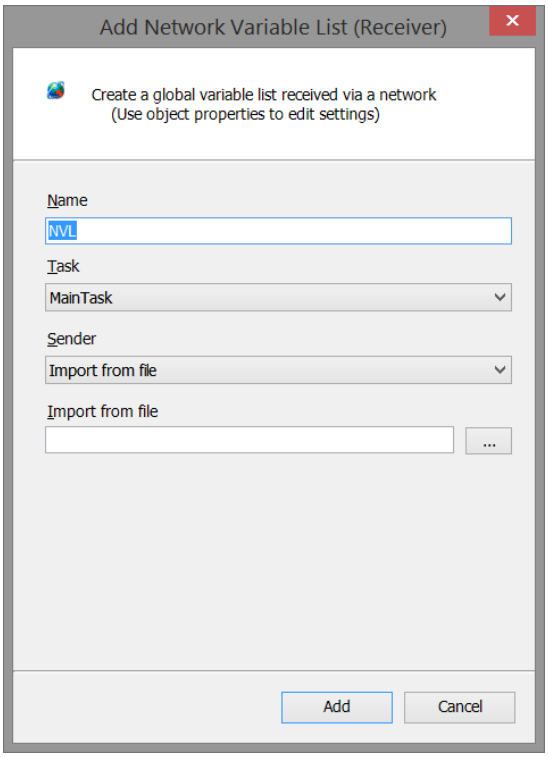
2-5-3. Data unit type

The function of data unit type (DUT) is to provide users with a user-defined data type, including structure, enumeration, alias and union, as shown in the following figure. The use of data unit type plays a role in standardizing programming process, improving programming efficiency, optimizing programming format, and realizing object-oriented programming.



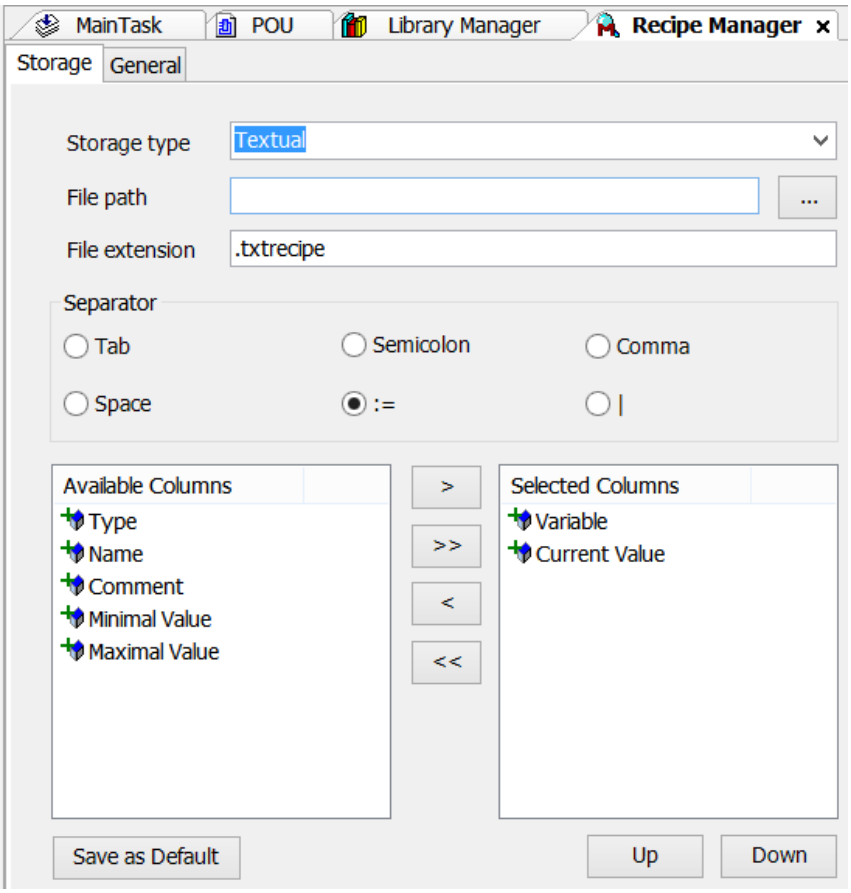
2-5-4. Global network variables

The global network variable list (GNVL) is divided into two forms: sender and receiver. The function of the sender is to declare and list the global variables of the network variable list (receiver) that should be sent to other devices or network items. The function of the receiver is to list the received network variables and display information (network, transmission information, sender, etc.). Users can add the global network list of sender and receiver to the device tree by configuring the global network variable editor to realize the interaction of global variables in the network.



2-5-5. Recipe manager

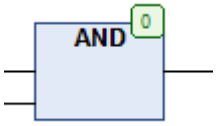
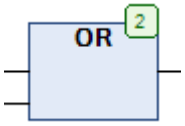
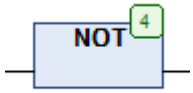
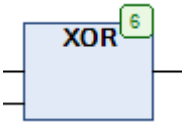
The function of the recipe manager is to provide a list of user-defined variables (recipe definitions). Users can configure the storage location, storage method and storage category through the recipe manager, as shown in the following figure. After the recipe manager is configured successfully, users can upload and download recipe definitions.



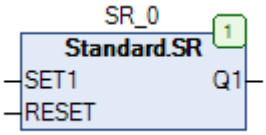
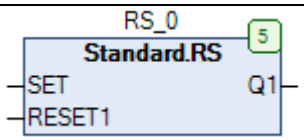
3. Basic instructions

3-1. Bit logic instructions

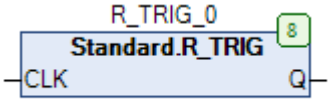
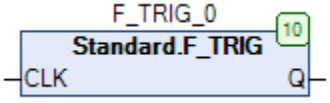
3-1-1. Basic logic instructions

Instruction	Command icon	Function
AND		Operator AND
OR		Operator OR
NOT		Operator NOT
XOR		Operator XOR

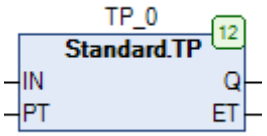
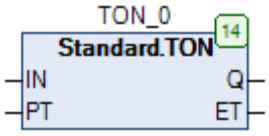
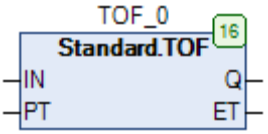
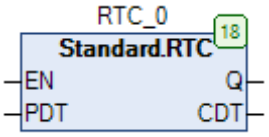
3-1-2. Set priority and reset priority trigger instructions

Instruction	Command icon	Function
SR		Set priority trigger: set bistable trigger, set priority
RS		Reset priority trigger: reset bistable trigger, reset priority

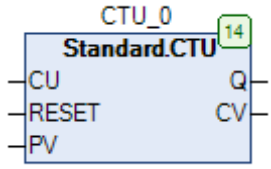
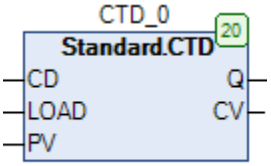
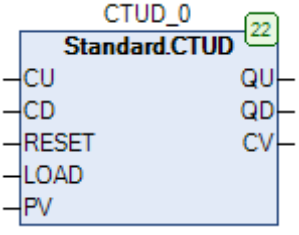
3-1-3. Data unit type

Instruction	Command icon	Function
R_TRIG		Rising edge trigger
F_TRIG		Falling edge trigger

3-2. Timer instructions

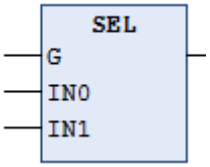
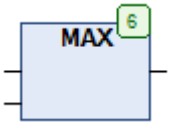
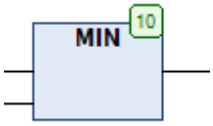
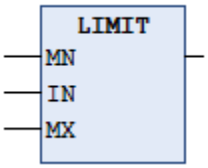
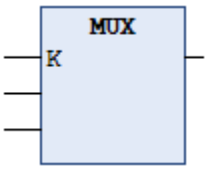
Instruction	Command icon	Function
TP		Pulse timer: once IN becomes TRUE, Q is true, and the time will start counting in milliseconds in ET until its value is equal to PT, then Q is FALSE
TON		Power on delay timer: once IN becomes TRUE, the time will start counting in milliseconds in ET until its value is equal to PT, then Q is TRUE
TOF		Power off delay timer: when IN is FALSE and ET is equal to PT, Q is FALSE. Otherwise, it is TRUE
RTC		Real time clock: starts at a given time and returns the date and time

3-3. Counter instructions

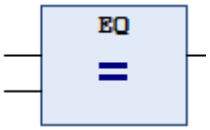
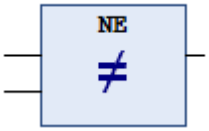
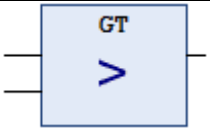
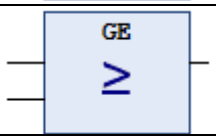
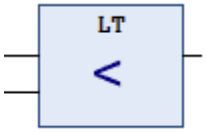
Instruction	Command icon	Function
CTU		Increment counter: if RESET is TRUE, initialize to 0. The rising edge of CU always increases by 1. Once CV > = PV, Q will be set to TRUE
CTD		Minus counter: if LOAD is TRUE, CV will be set to the starting value given by PV. The rising edge of CD always increases by 1, the counter value (CV) decreases by 1 until 0, and Q will be set to TRUE
CTUD		Up/down bidirectional counter

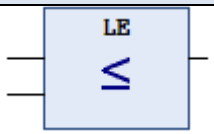
3-4. Data processing instructions

3-4-1. Select operation instructions

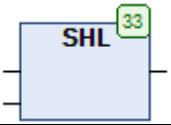
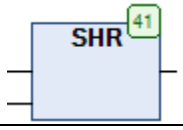
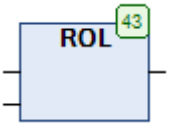
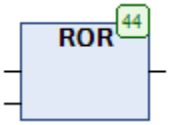
Instruction	Command icon	Function
SEL		One out of two instruction: when the selection switch is FALSE, the output is the first input data; when the selection switch is TRUE, the output is the second data
MAX		Take the maximum value
MIN		Take the minimum value
LIMIT		Limit value: if the IN value is higher than the upper limit of Max, LIMIT generates Max. If the value of IN is lower than the lower limit of Min, the result is Min
MUX		Choose one from many: MUX selects the Kth value from a group of values. The first value is K=0. If K is greater than the number of other inputs (n), Codesys passes the last value

3-4-2. Compare instructions

Instruction	Command icon	Function
EQ		Equal to
NE		Not equal to
GT		Greater than
GE		Greater than or equal to
LT		Less than

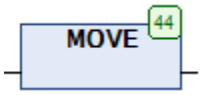
Instruction	Command icon	Function
LE		Less than or equal to

3-4-3. Shift instruction

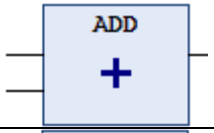
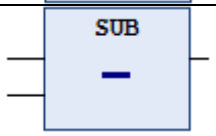
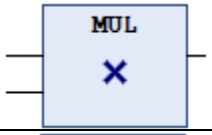
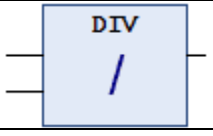
Instruction	Command icon	Function
SHL		Shift left by bit
SHR		Shift right by bit
ROL		Rotate left
ROR		Rotate right

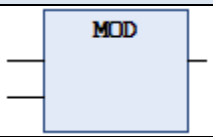
3-5. Operation instruction

3-5-1. Assignment instruction

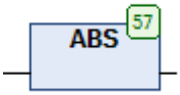
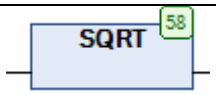
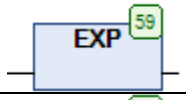
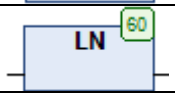
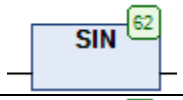
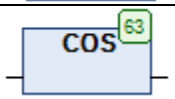
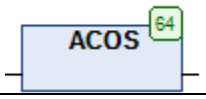
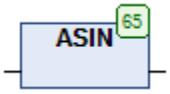
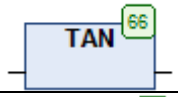
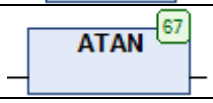
Instruction	Command icon	Function
MOVE		Assignment

3-5-2. Arithmetic operation

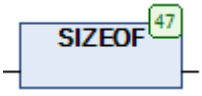
Instruction	Command icon	Function
ADD		Addition
SUB		Subtraction
MUL		Multiplication
DIV		Division

Instruction	Command icon	Function
MOD		Residual

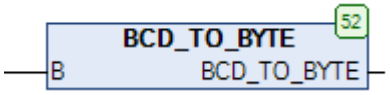
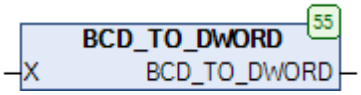
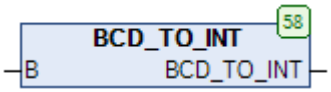
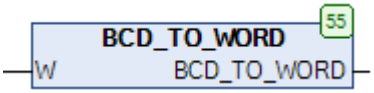
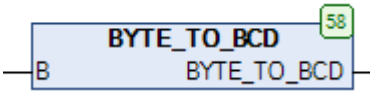
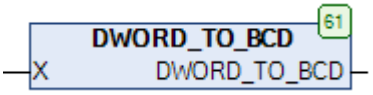
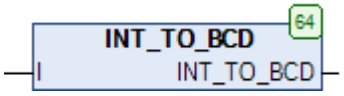
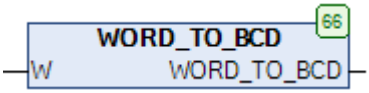
3-5-3. Mathematical operation instruction

Instruction	Command icon	Function
ABS		Absolute value instruction
SQRT		Square root instruction
EXP		Exponent instruction
LN		Natural logarithm instruction
LOG		Common logarithmic instruction
SIN		Sine command
COS		Cosine instruction
ACOS		Arccosine instruction
ASIN		Arcsine command
TAN		Tangent instruction
ATAN		Arctangent instruction

3-5-4. Address operation instruction

Instruction	Command icon	Function
SIZEOF		Data type size
ADR		Adress operator
BITADR		Bit address operator

3-5-5. Data conversion instruction

Instruction	Command icon	Function
BCD_TO_BYTE		BCD convert to BYTE
BCD_TO_DWORD		BCD convert to DWORD
BCD_TO_INT		BCD convert to INT
BCD_TO_WORD		BCD convert to WORD
BYTE_TO_BCD		BYTE convert to BCD
DWORD_TO_BCD		DWORD convert to BCD
INT_TO_BCD		INT convert to BCD
WORD_TO_BCD		WORD convert to BCD

4. Special functions

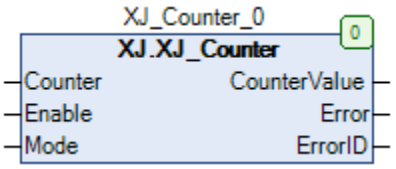
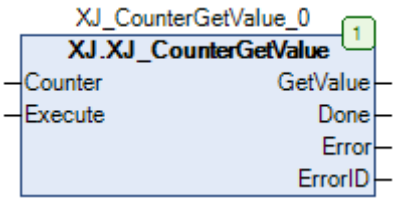
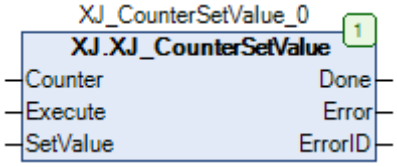
4-1. High speed counting

4-1-1. Function overview

XS series PLC has high-speed counting function. By selecting different counters, it can measure high-speed input signals such as measurement sensors and rotary encoders, and its maximum measurement frequency can reach 200kHz.

4-1-2. Function block introduction

1. Command format

Command	Name	Graphic representation	ST performance
XJ_Counter	High speed counter		XJ_Counter(Counter:= , Enable:= , Mode:= , CounterValue=> , Error=> , ErrorID=>);
XJ_CounterGetValue	Read high speed counter		XJ_CounterGetValue(Counter:= , Execute:= , GetValue=> , Done=> , Error=> , ErrorID=>);
XJ_CounterSetValue	Write high speed counter		XJ_CounterSetValue(Counter:= , Execute:= , SetValue:= , Done=> , Error=> , ErrorID=>);

2. Related variables

【XJ_Counter】

(1) Input variables

Input variables	Name	Data type	Effective range	Initial value	Description
Counter	Counter	COUNTER_REF	-	-	High speed counter, which specifies the high-speed counting input and initial value
Enable	Enable	BOOL	TRUE, FALSE	FALSE	Normally open enable counting
Mode	Counting mode	Mode	AB_Mode, Single_Mode	FALSE	High speed counting mode: MODE=XJ.AB_Mode, is AB phase high speed counting

					MODE=XJ.Single_Mode, is single phase high speed counting

(2) Output variables

Output variables	Name	Data type	Effective range	Initial value	Description
CounterValue	Counter value	DINT	Data type	0	High speed counter value
Error	Error flag	BOOL	TRUE, FALSE	FALSE	
ErrorID	Error type	UINT	-	0	

【XJ_CounterGetValue】

(1) Input variables

Input variables	Name	Data type	Effective range	Initial value	Description
Counter	Counter	COUNTER_REF	-	-	High speed counter, which specifies the high-speed counting input and initial value
Execute	Enable	BOOL	TRUE, FALSE	FALSE	Trigger on the rising edge to read the current high-speed count value

(2) Output variables

Output variables	Name	Data type	Effective range	Initial value	Description
GetValue	Read value	DINT	Data range	0	Present counter value
Done	Completed flag	BOOL	TRUE, FALSE	FALSE	After reading, the flag bit is TRUE
Error	Error flag	BOOL	TRUE, FALSE	FALSE	
ErrorID	Error type	UINT	-	0	

【XJ_CounterSetValue】

(1) Input variables

Input variables	Name	Data type	Effective range	Initial value	Description
Counter	Counter	COUNTER_REF	-	-	High speed counter, which specifies the high-speed counting input and initial value
Execute	Enable	BOOL	TRUE, FALSE	FALSE	Trigger on rising edge, write high-speed count value, write the value of SetValue to CounterValue
SetValue	Write in value	DINT	Data range	0	Write high speed count setting value

(2) Output variables

Output variables	Name	Data type	Effective range	Initial value	Description
Done	Complete flag	BOOL	TRUE, FALSE	FALSE	After writing, the flag bit is TRUE
Error	Error flag	BOOL	TRUE, FALSE	FALSE	
ErrorID	Error type	UINT	-	0	

Note: if the displayed value of ErrorID is 2, it is because the range of CounterID is not 0-3.

3. Function description

(1) The high-speed counting function has three function blocks: high-speed counting function block, read high-speed counting function block and write high-speed counting function block. XS3 series high-speed input can only receive differential signal (DIFF) and cannot receive open collector signal (OC). Please be sure to choose the encoder of differential signal. XSDH series high-speed input is to receive open collector signal (OC).

(2) Counter is COUNTER_REF data type:

The specific description of COUNTER_REF is as follows:

Member	Name	Data type	Effective range	Initial value	Description
CounterID	Counter port	INT	0,1,2,3	0	Select high speed counter input port
CounterValue	Counter initial value	DINT	Data range	0	Set the initial value of the counter

(3) XS3 and XSDH series high-speed counting function has two modes, single phase increasing mode and AB phase mode respectively.

(a) Incremental mode (Mode= Single_Mode)

In this mode, count the input pulse signal, and the count value increases with the rising edge of each pulse signal.

(b) AB phase mode (Mode=AB_Mode)

In this mode, the high-speed count value incremented or decremented according to the pulse signal (phase A and phase B) with a phase difference of 90°, and the default counting mode is 4 times frequency.

(4) XS series high speed counter input port

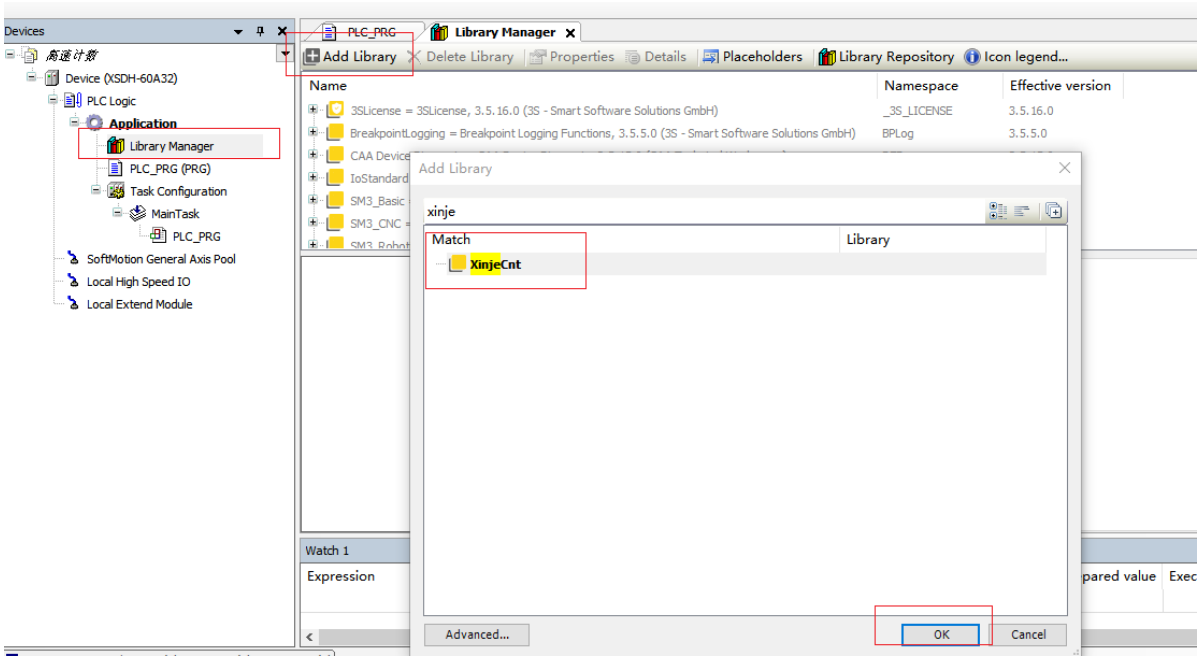
XS3-26T4								
	Single phase incremental mode				AB phase mode			
CounterID	0	1	2	3	0	1	2	3
Max frequency	200k	200k	200k	200k	200k	200k	200k	200k
X0+	U+				A+			
X0-	U-				A-			
X1+					B+			
X1-					B-			
X2								
X3+		U+				A+		
X3-		U-				A-		
X4+						B+		
X4-						B-		
X5								
X6+			U+				A+	
X6-			U-				A-	
X7+							B+	
X7-							B-	
X10								
X11+				U+				A+
X11-				U-				A-
X12+								B+
X12-								B-
X13								

XSDH-60A32-E								
	Single phase incremental mode				AB phase mode			
CounterID	0	1	2	3	0	1	2	3
Max frequency	200k	200k	200k	200k	100k	100k	100k	100k
X0	U				A			
X1					B			
X2								
X3		U				A		
X4						B		
X5								
X6			U				A	
X7							B	
X10								
X11				U				A
X12								B
X13								

4-1-3. Parameter setting

Add library file:

Add “XinjeCnt” in Library Manager. High speed counting function can be used after adding.



4-1-4. Application example

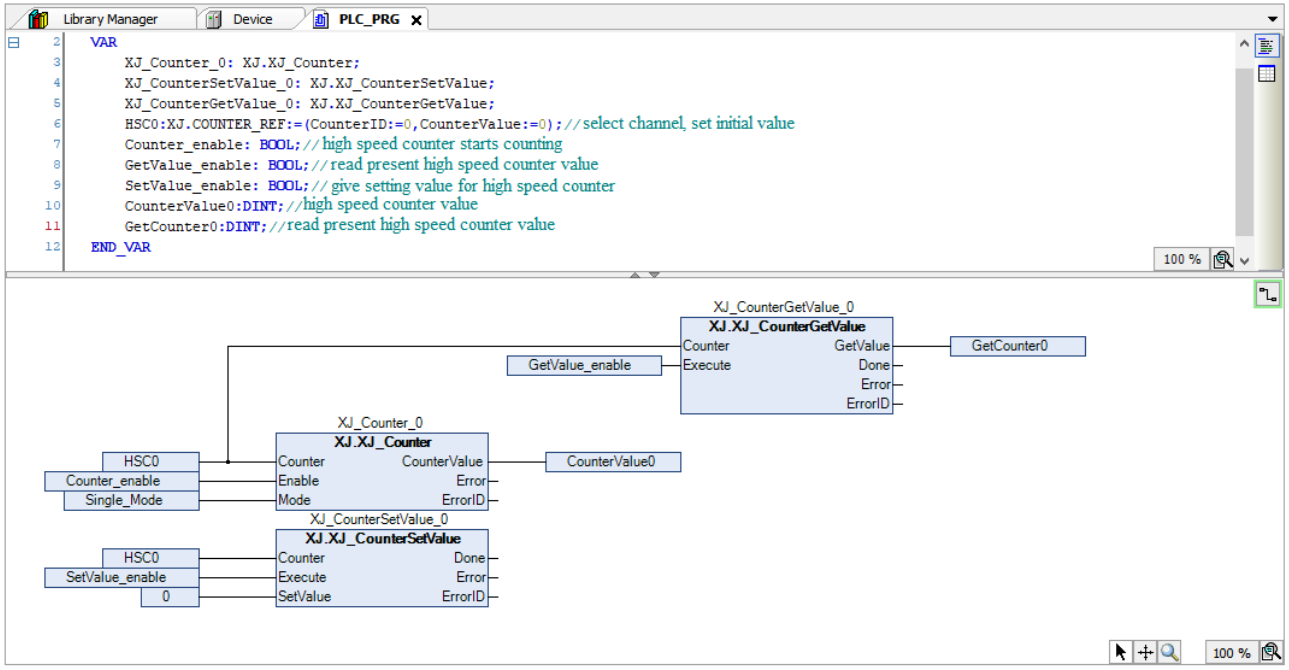
Example 1: use the first channel of high-speed counter, read the current count value in the counter, and modify the current high-speed count value.

Program operation:

(1) Install the library to be used according to the steps in section 4-1-3.

(2) Write a high-speed counting program.

Programming: use the function blocks "XJ.XJ_Counter", "XJ.XJ_CounterGetValue", "XJ.XJ_CounterSetValue". Set the high-speed counting port, high-speed counting mode and high-speed counting value in the program.



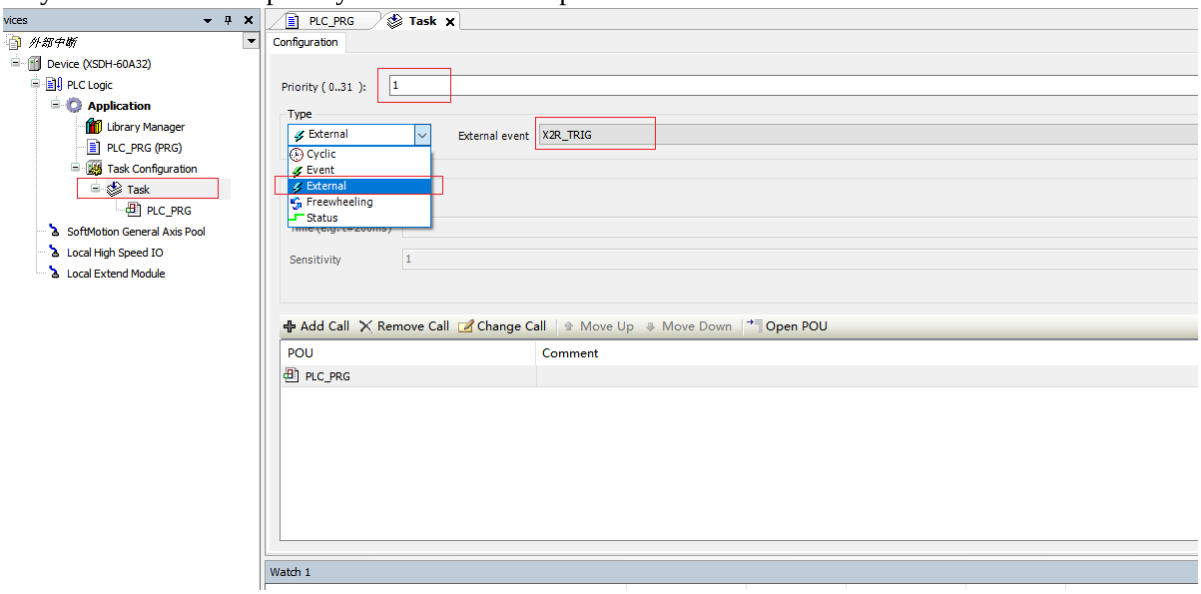
4-2. External interrupt

4-2-1. Function overview

XS series PLC supports X-terminal interrupt, and the same terminal supports rising edge and falling edge interrupt. In Codesys, interrupt is used in the form of external events in task type. Such as X2R_TRIG stands for X2 rising edge interrupt, X2F_TRIG represents the falling edge interrupt. For the number and type of interrupts supported by each model, see the "external event" option.

4-2-2. Application example

Double click "task" and set it to external event "external" in the pop-up interface - external interrupt uses terminal X, and you can also set the priority of external interrupt events.



4-3. PLC SHELL

4-3-1. Function overview

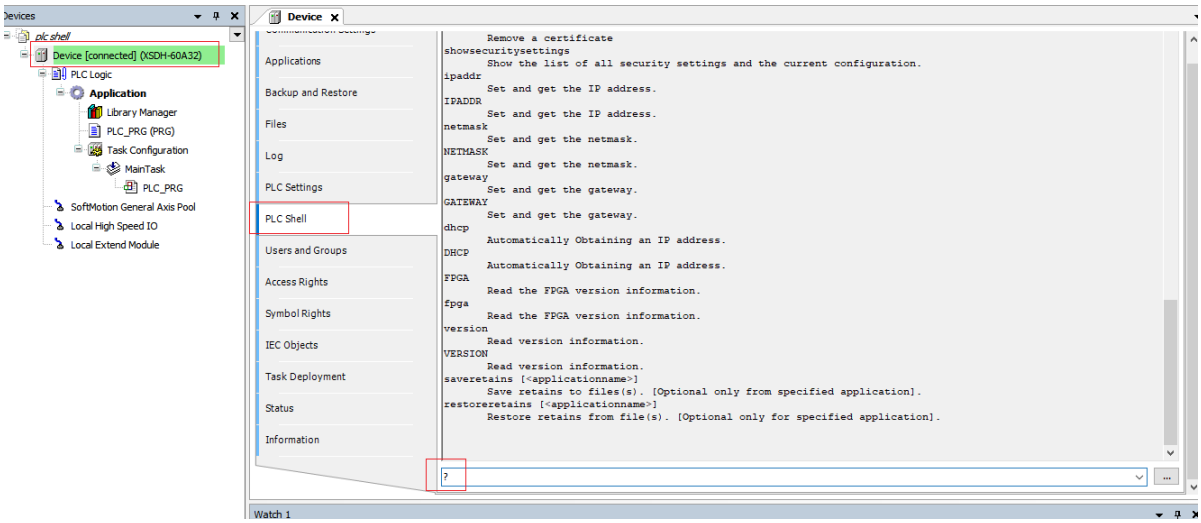
PLC shell function is a text-based control monitor, which can be used to query the specific information of the controller, input the specified command in the input window, and receive the response from the controller in the result window.

4-3-2. Command list

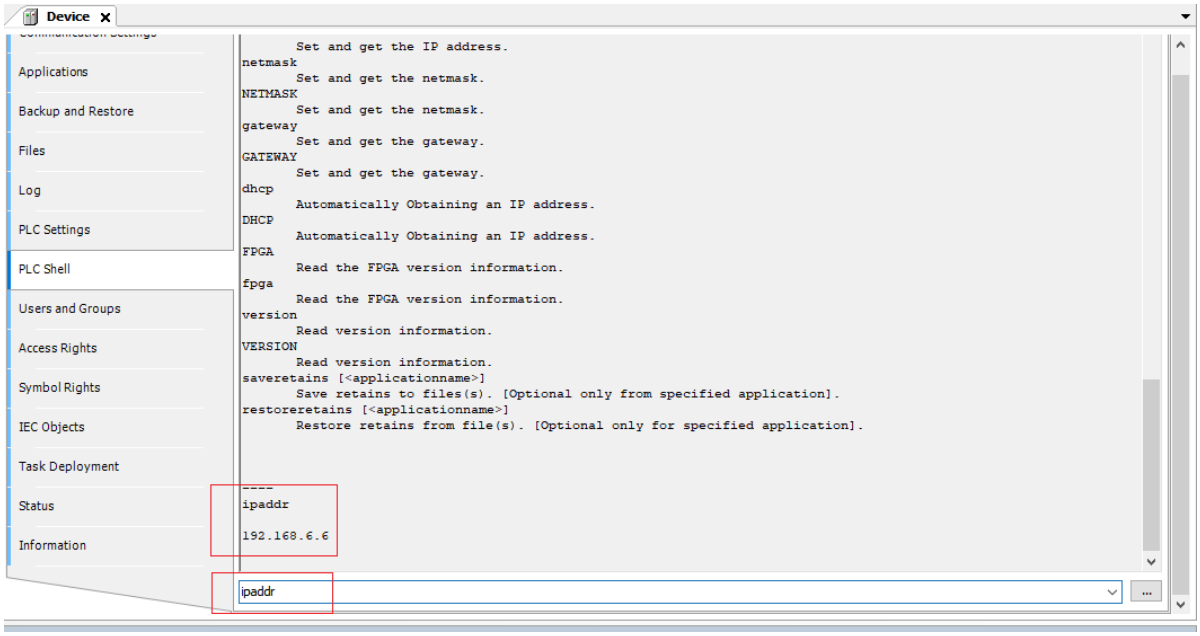
Command name	Function
ipaddr / IPADDR	Get/set the IP address of PLC
netmask / NETMASK	Get/set the subnet mask of PLC
gateway / GATEWAY	Get/set the gateway of PLC
dhcp / DHCP	Set IP to automatic acquisition
fpga / FPGA	Get FPGA version of PLC
version / VERSION	Get firmware version of PLC
rtc-get / RTC-GET	Get the current UTC time
rtc-set / RTC-SET	Set UTC time

4-3-3. Application example

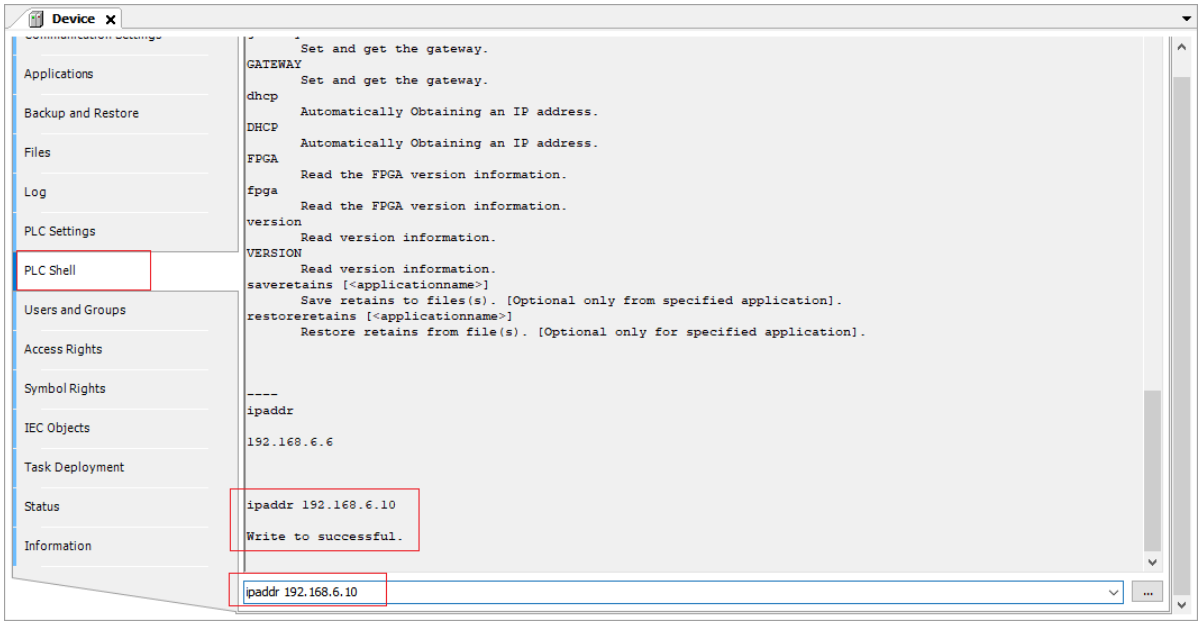
Double click “Device”, input “?” in “PLC Shell”, it will show all the functions. Here you can modify the IP, obtain the firmware version, set / read the clock information, and so on.



For example, enter "ipaddr" to get the current IP address of PLC.

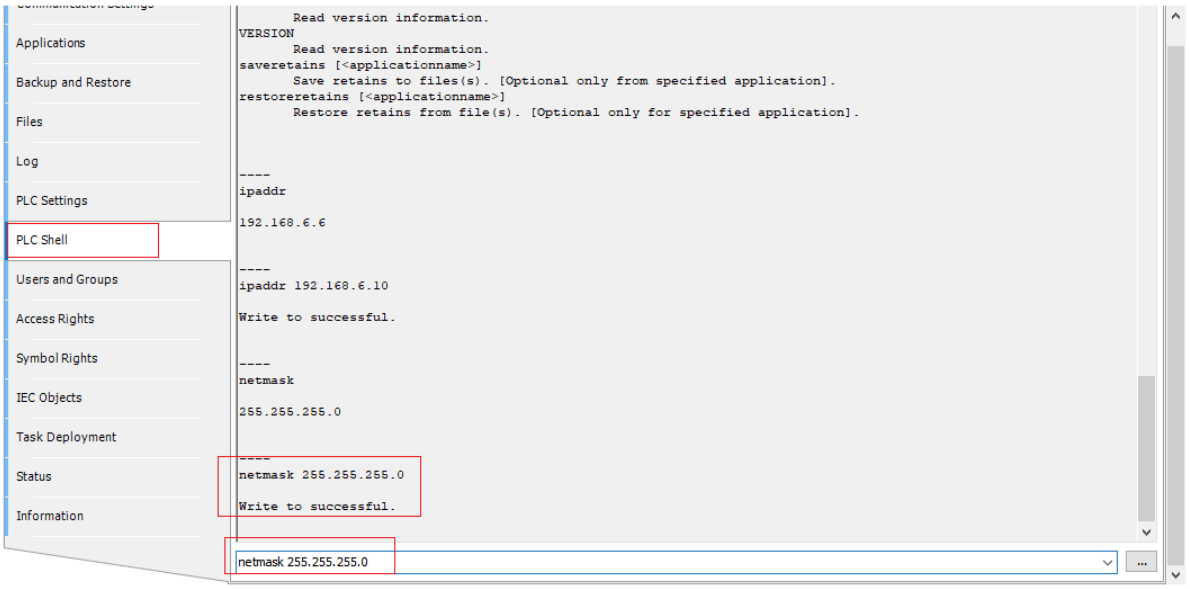


Input “ipaddr 192.168.61.196”, set PLC IP address. If "write to successful" is displayed, the writing is successful.

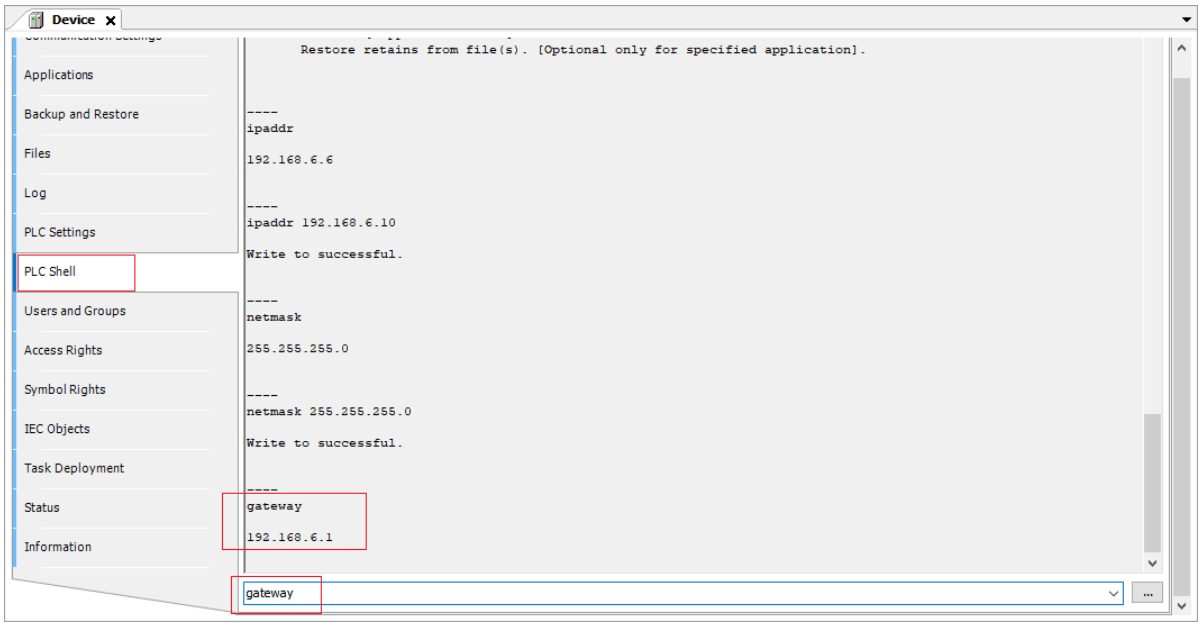


Input “netmask” can get the current subnet mask of PLC.

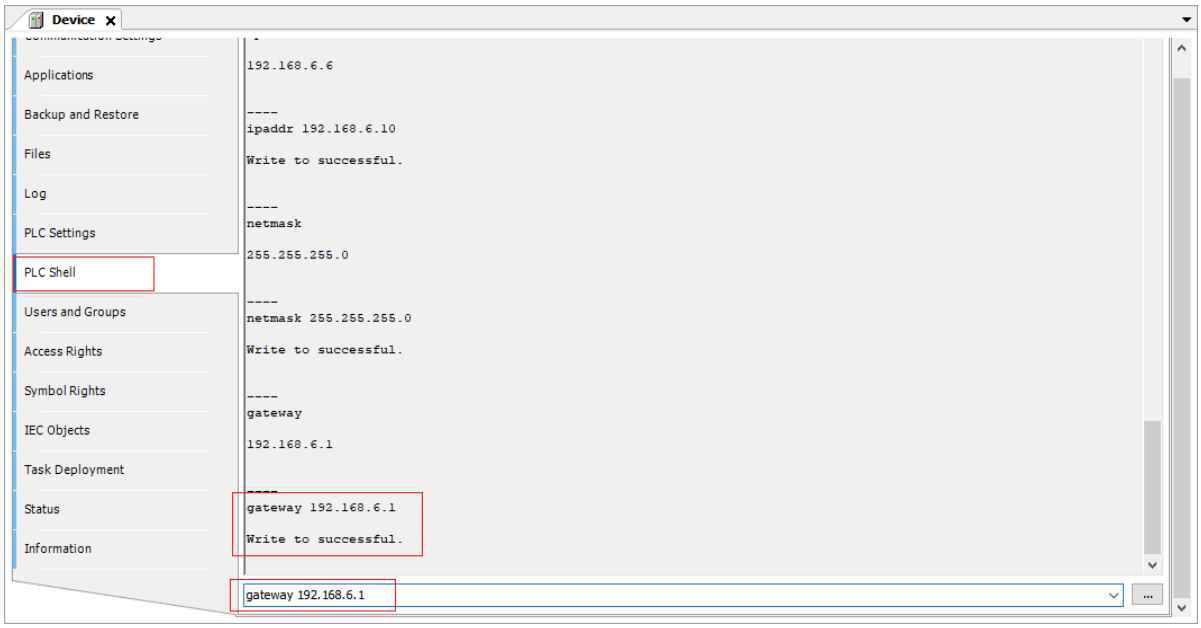
Enter "netmask 255.255.254.0" to set the subnet mask of PLC. If "write to successful" is displayed, the writing is successful.



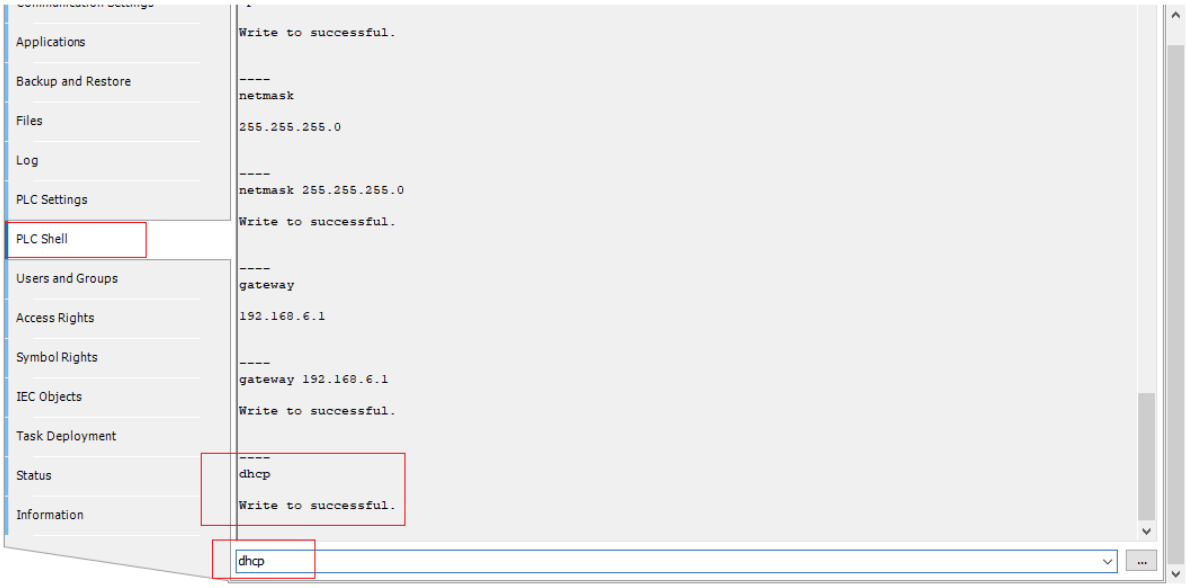
Enter "gateway" to get the current default gateway of PLC.



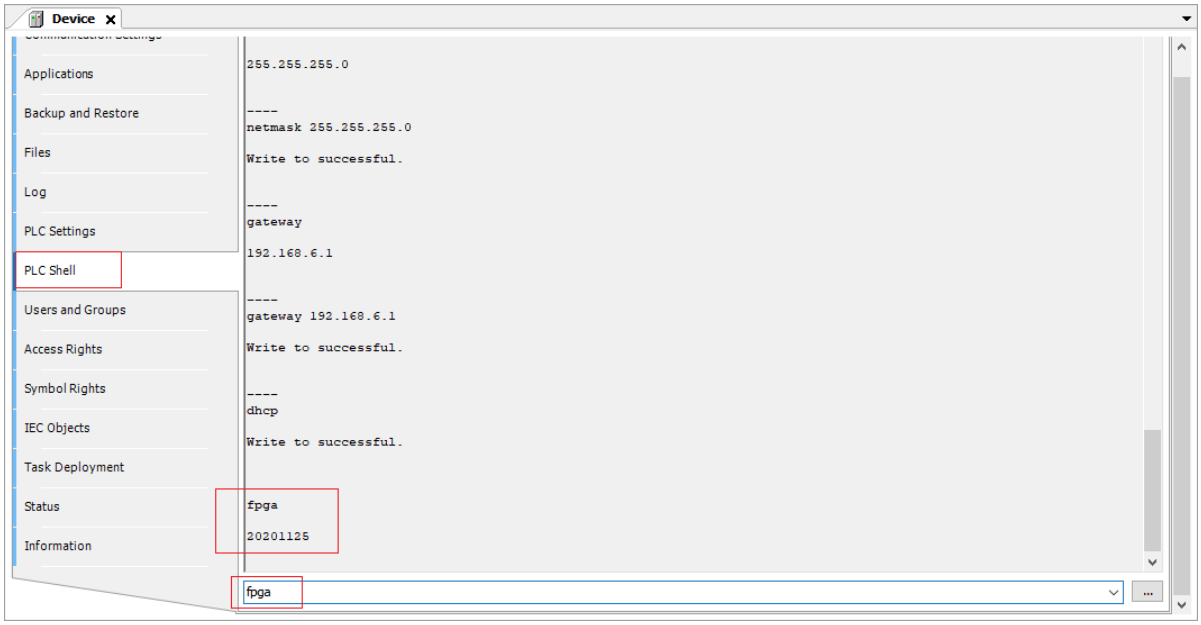
Enter “gateway 192.168.60.1” to set PLC gateway, if "write to successful" is displayed, the writing is successful.



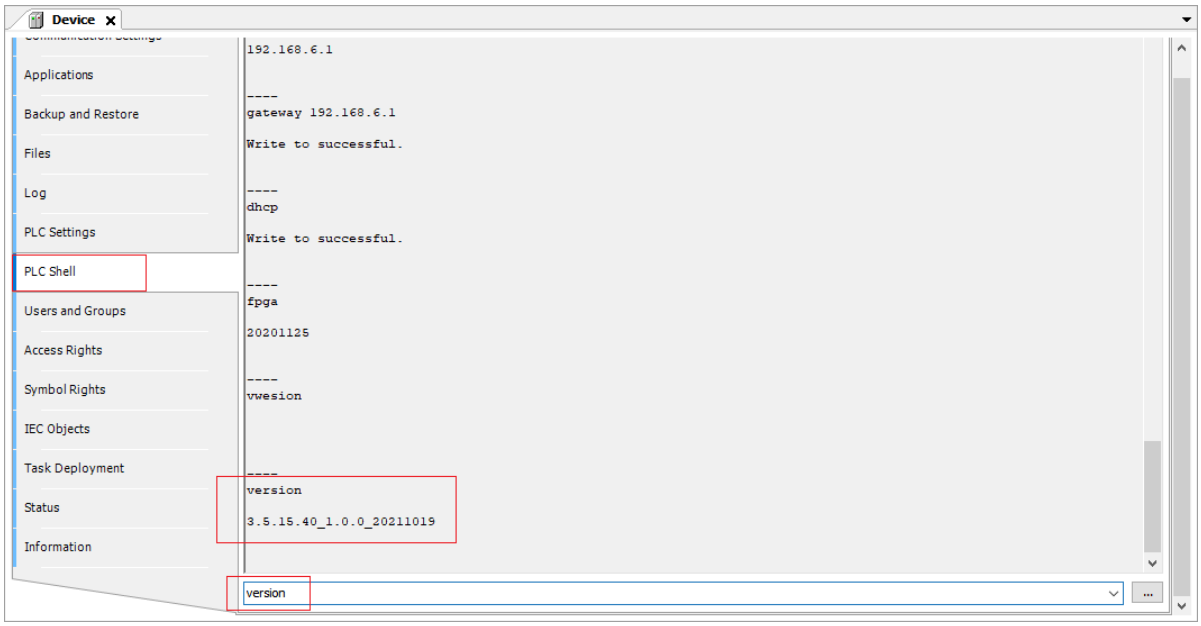
Enter "dhcp" and set the IP acquisition method of PLC to automatic acquisition. If "write to successful" is displayed, the writing is successful. When the IP acquisition method is automatic, it is necessary to ensure a good network environment.



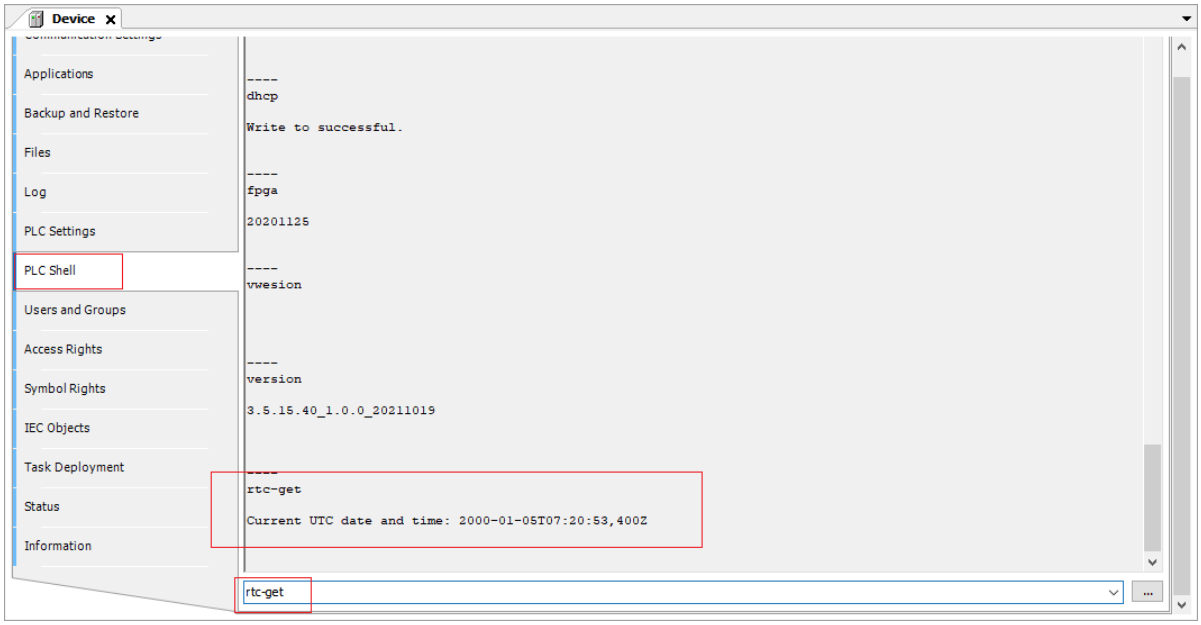
Enter "FPGA" to get the current FPGA version of PLC.



Enter "version" to get the current firmware version of PLC.



Enter "rtc-get" to get the current UTC time.



Enter “rtc-set 2021-10-25T18:24:30” to set UTC time. If “RTC successfully set to 2021-10-25T18:24:30,000Z” is displayed, the writing is successful. “000Z” display content is not fixed.

4-4. Clock

4-4-1. Function overview

XS series PLC integrates RTC, which is used to record the current system time. The clock is powered by battery, which can ensure the accuracy of time. At the same time, it also supports users to modify RTC time manually.

4-4-2. Application example

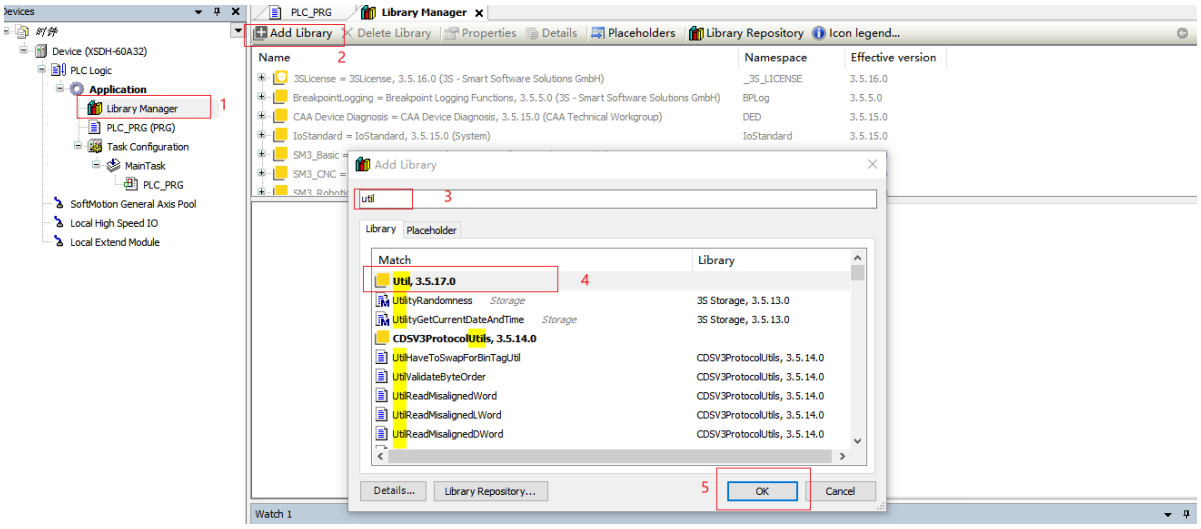
How to get events:

1. Double click “Device”, enter “rtc-get” in “PLC Shell” to get the current time.

2. Use clock instruction

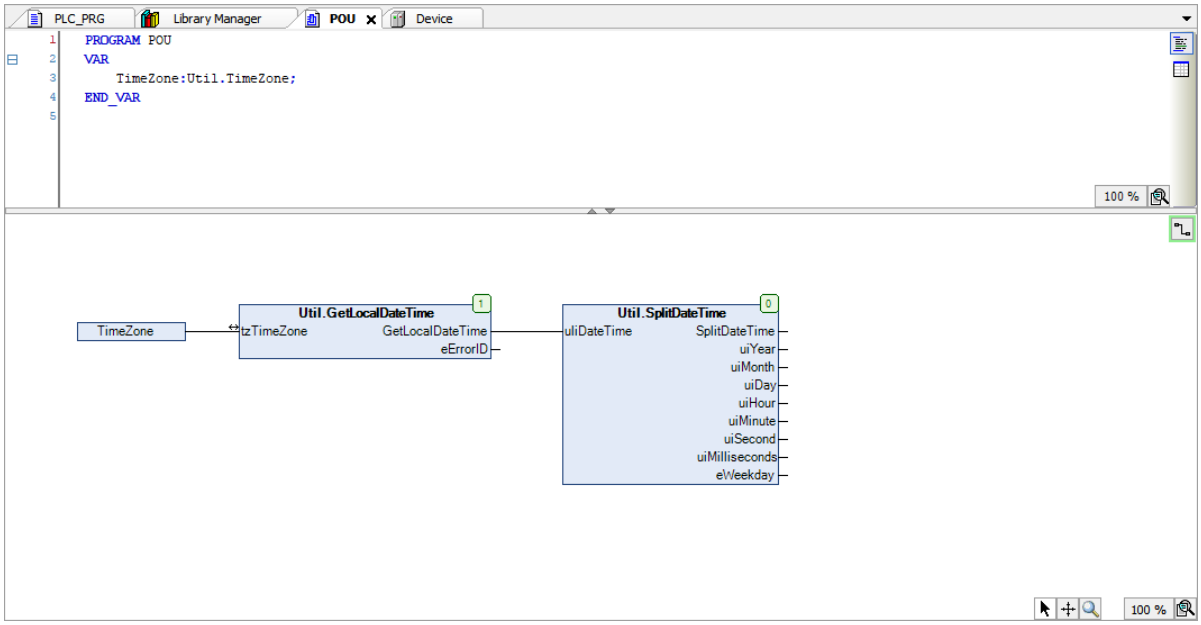
(1) Add related library file

Add “Util” in “Library Manager”. After adding, you can use the clock function.



(2) Make the clock program

Obtain the current time by using the function block “Util.GetLocalDateTime”, “Util.SplitDateTime”. There are other function blocks about clock in this library, which can be viewed in the library "Util".


PLC_PRG
Library Manager
POU
Device
1 PROGRAM POU
2 VAR
3 TimeZone:Util.TimeZone;
4 END_VAR
5
100 %
TimeZone
Util.GetLocalDateTime
tzTimeZone
GetLocalDateTime
eErrorID
Util.SplitDateTime
uliDateTime
SplitDateTime
uiYear
uiMonth
uiDay
uiHour
uiMinute
uiSecond
uiMilliseconds
eWeekday
100 %

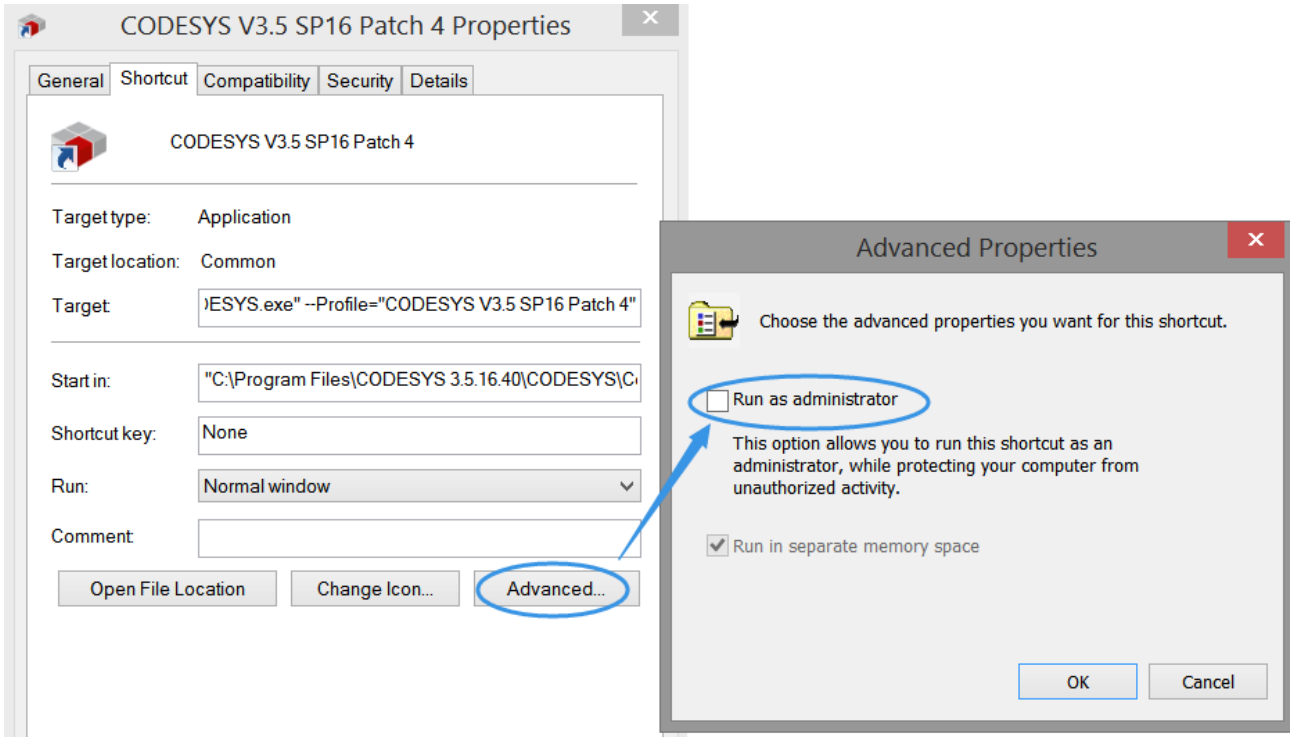
5. Codesys project examples

5-1. Basic programming operation

1. Start Codesys

(1) Set administrator permissions

Right click the Codesys v3.5 software, click properties, select “Run as administrator”.



(2) Start Codesys

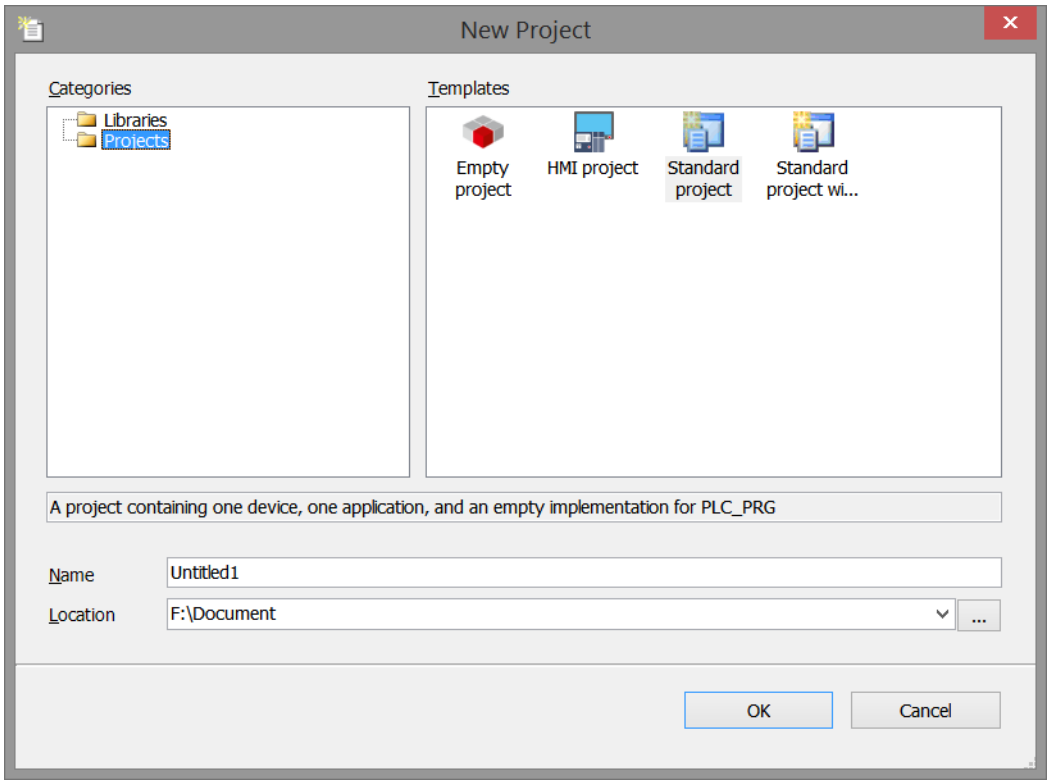
Double click Codesys software on the desktop.

(3) Create a project

Select the new project in the file menu to create a new project, as shown in the figure.

(4) Select the project

User can build empty project or standard project. And enter the name and path for the project file, then click ok.

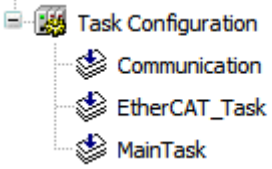


2. Create PLC program file

The establishment of PLC program file is not only the establishment of operation sequence of operation structure, but also the establishment of programming mode, and even includes the segmentation of data area. Before establishing the program file, the operation structure should be divided in detail, the continuous, periodic and event triggered tasks should be determined, and the priority of periodic and event triggered tasks should be arranged. After creating a Codesys project, a default continuous task will be automatically generated, under which there is a default program and PLC_ PRG.

(1) Create a task

First of all, manage tasks in "task configuration". In general project applications, it can be divided into main logical tasks and communication tasks. Communication will put it at a higher task priority and a shorter cycle time because it needs to update the data source. In addition, if motion control is involved in the project, it will also be separated into a task and placed at the highest priority, as shown in the figure:

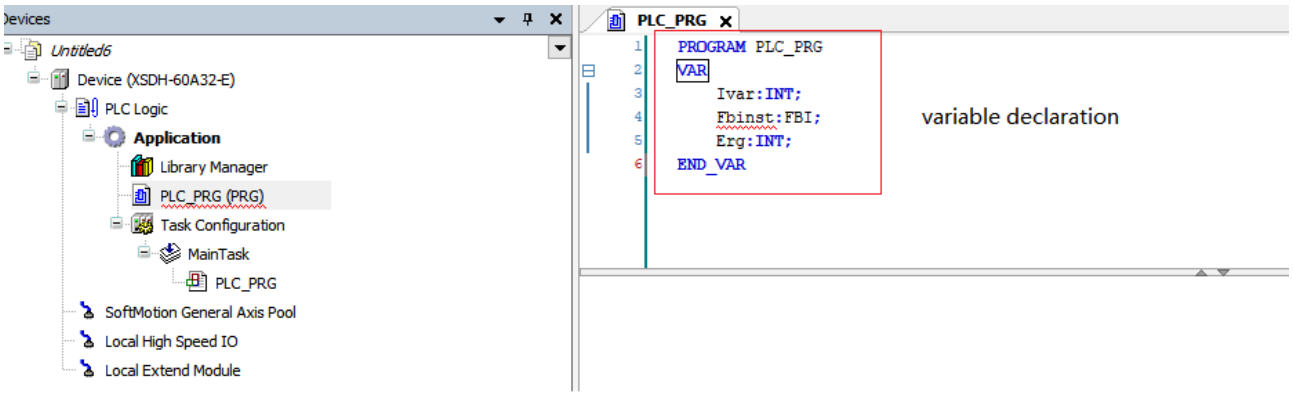


(2) Create POU

Click “Application”-->right click “add object”, select POU.

(a) Variable declaration

In the device window, the default POU is "PLC_PRG". Double click "PLC_PRG" in the device tree to automatically open it in the ST language editor in the middle of the Codesys user interface. The language editor consists of a declaration part (upper part) and an implementation part (lower part), separated by an adjustable dividing line. The declaration part includes: the line number, POU type and name (such as "PROGRAM PLC_PRG") displayed, and the variable declaration between the keywords "VAR" and "END_VAR", as shown in the following figure. In the declaration section of the editor, move the cursor after VAR and click enter. Insert a new blank line, declare INT variable "Ivar", INT variable "Erg", FB1 variable "Fbinst".



(b) Input the program

Enter the following code in the program editing area under the declaration area:

```
Ivar:=Ivar+1;
Fbinst(in:=11,out =>Erg);
```

(c) Custom function / function block

In the variable declaration area, you can see that the function block "FB1" is called, but "FB1" is not a standard function block, so you need to customize the function block. Select the add object command from the Project menu. Select "POU" on the left side of the "add object" dialog box, enter the name: FB1, and activate the "function block (b)" option in the type option. Select "structured text (ST)" as the implementation language. Click the "open" button to confirm the object setting.

The edit window for the new function block FB1 opens. Like the variable declaration of PLC_PRG, the following variables are declared here:

```
FUNCTION_BLOCK FBI
VAR_INPUT
    in:INT;
END_VAR
VAR_OUTPUT
    out:INT;
END_VAR
VAR
    ivar:INT:=2;
END_VAR
```

Enter the following in the program editor implementation section:

```
out:=in+ivar;
```

The function is to add "2" to the input variable "in" and assign it to the output "out".

5-2. I/O mapping

In Codesys application, when variable mapping with I / O module of programmable logic controller or network communication with external equipment is required, two methods can be adopted:

① Bind the parameters defined in the POU to the variables

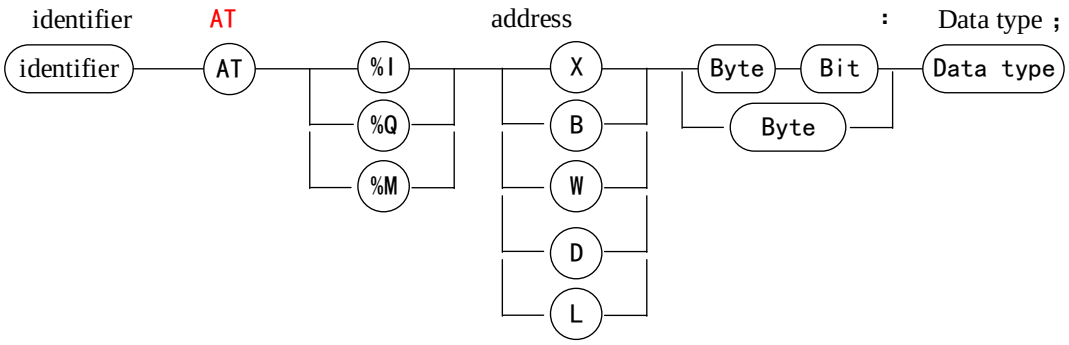
② Use the keyword AT to directly link the variable to the determined address. The direct variable must comply with the following rules:

AT<address>:

< identifier> AT<address>:<data type>{:=< initialize value>};

{}is an optional part.

Start with "%", followed by the position prefix symbol and size prefix symbol. If there is a grade, use an integer to represent the grade, and use the decimal point symbol ".", such as %IX0.0, %QW0. The specific format of direct variable declaration is shown in the following figure:

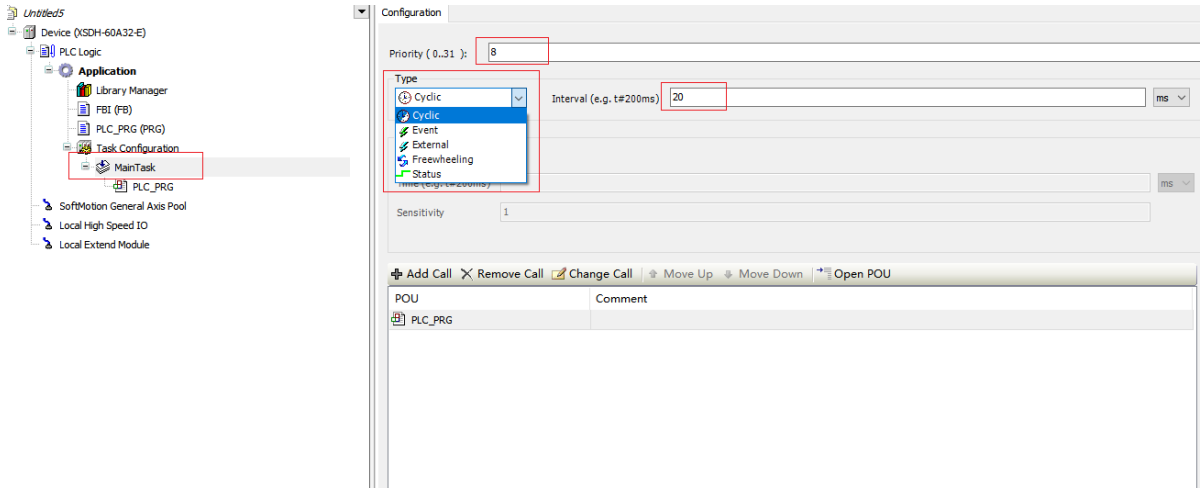


5-3. Task configuration

1. Overview

A program can be written in different programming languages. A typical program is composed of many interconnected functional blocks, which can exchange data with each other. The execution of different parts of a program is controlled by "tasks". After the "task" is configured, a series of programs or function blocks can be executed periodically or triggered by a specific event to start executing the program. There is a task manager tab in the device tree, which can be used to declare a specific PLC_PRG and control the execution of other subprograms in the project. Task is used to specify the attributes of the program organization unit at runtime. It is an execution control element with the ability to call. In a task configuration, multiple tasks can be established, and in a task, multiple program organization units can be called. Once a task is set, it can control program cycle execution or start execution through specific events.

In task configuration, it is defined by name, priority and start type of task. This startup type can be defined by time (periodic, random) or by internal or external trigger task time, for example, using the rising edge of a Boolean global variable or a specific event in the system. For each task, a series of programs started by the task can be set. If this task is executed in the current cycle, these programs will be processed within the length of one cycle. The combination of priority and condition will determine the sequence of task execution. The task setting interface is shown in the following figure:



Since Codesys v3.x has the following attributes during task configuration, programmers should follow the following rules:

- The maximum number of cycle tasks is 100.
- The maximum number of free running tasks is 100.

- The maximum number of event triggered tasks is 100.
- According to the target system, PLC_ PRG may be executed as a free program in any case without inserting into the task configuration.
- The processing and calling program are executed according to the top-down sequence in the task editor.

2. Task priority

In Codesys, you can set the priority of tasks. There are 32 levels (a number between 0 and 31, 0 is the highest priority and 31 is the lowest priority). When a program is executing, the task with high priority takes precedence over the task with low priority. The high priority task 0 can interrupt the execution of the lower priority program in the same resource, so that the execution of the lower priority program is slowed down.

Note: when assigning a task priority level, do not assign tasks with the same priority. If there are other task views that precede tasks with the same priority, the results may be uncertain and unpredictable.

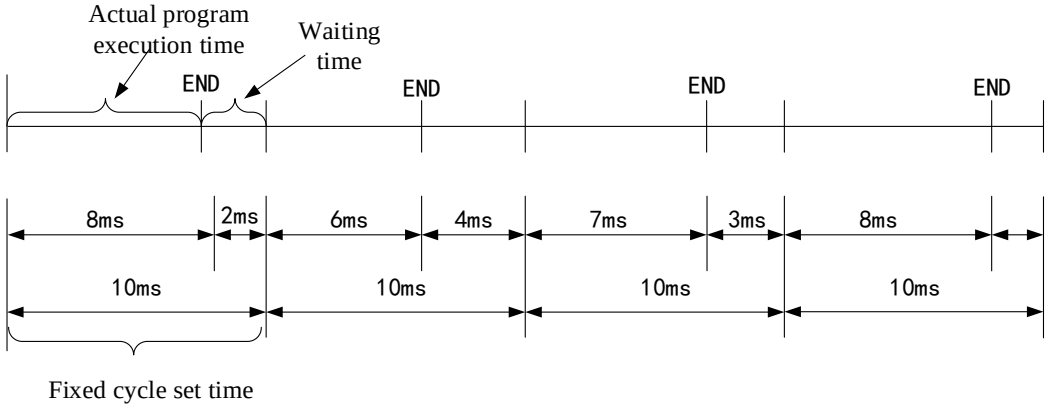
3. Task execution type

Type editing and configuration can be performed for each independent task. Including fixed-cycle cycle, event trigger, external trigger, free running and state trigger.

(1) Cyclic

According to whether the instructions used in the program are executed or not, the processing time of the program will be different, so the actual execution time will change differently in each scanning cycle, and the execution time will vary. By using the fixed cycle mode, the program can be repeatedly executed with a certain cycle time. Even if the execution time of the program changes, a certain refresh interval can be maintained. Here, we also recommend that you preferentially choose the fixed cycle task startup mode.

For example, if the task corresponding to the program is set to the fixed cycle mode and the interval time is set to 10ms, the sequence diagram of the actual program execution is shown in the following figure.

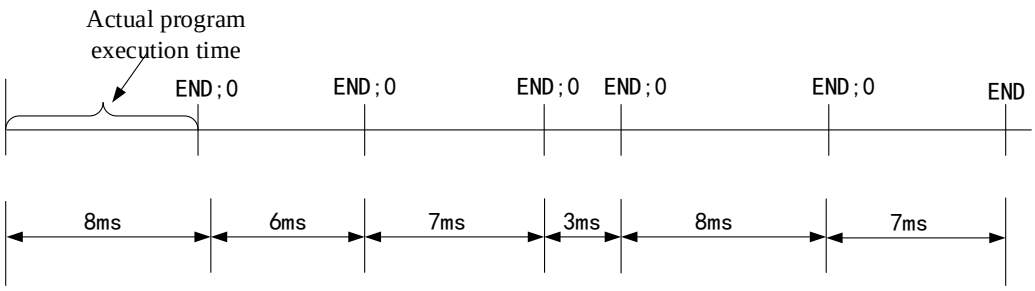


If the actual execution time of the program is completed within the specified fixed cycle setting time, the spare time is used as a waiting time. If there are tasks with lower priority in the application that are not executed, the remaining waiting time is used to execute tasks with lower priority.

(2) Freewheeling

The task will be processed as soon as the program starts to run. After the end of one running cycle, the task will be automatically restarted in the next cycle.

It is not affected by the program scanning cycle (interval time). That is, ensure that the next cycle is entered after the last instruction of the program is executed every time. Otherwise, the program cycle will not be ended.



Since there is no fixed task time in this execution mode, the execution time may be different each time. Therefore, the real-time performance of the program cannot be guaranteed, and this method is rarely used in practical applications.

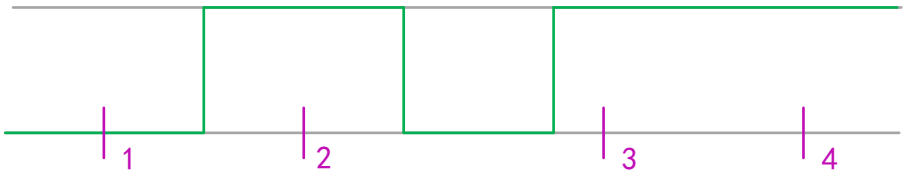
(3) Event

If the variable in the event area gets a rising edge, the task starts.

(4) Status

If the variable in the event area is true, the task begins.

In the following figure, the event trigger and status trigger are compared respectively. The green solid line is the boolean variable status selected by the two trigger methods. The following table is the comparison result.



Task input trigger signal

The state triggering mode is similar to the event triggering function, the difference is that the program will be executed as long as the trigger variable of the state triggering is true, and will not be executed if it is false. While the event trigger only collects the rising edge effective signal of the trigger variable.

Different types of tasks showed different responses at sampling points 1-4 (purple). This specific event is true to complete the condition of the state driven task. However, an event driven task requires the event to change from false to true. If the sampling frequency of the task plan is too low, the rising edge of the event may not be detected.

Execution point	1	2	3	4
Event	Not execute	Execute	Execute	Execute
Status	Not execute	Execute	Not execute	Not execute

(5) External interrupt

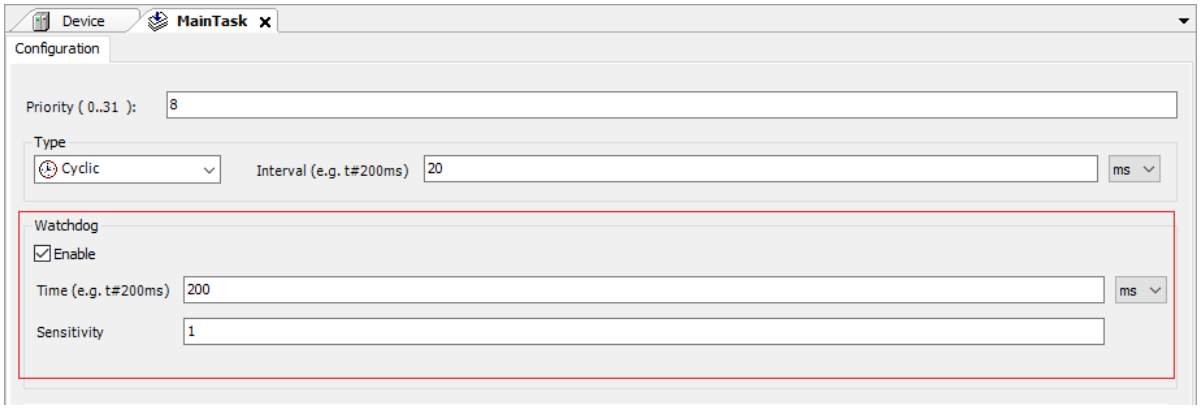
If the variable in the event area gets the rising edge or falling edge of an external interrupt signal X, the task starts. The input terminal X can be used as an input of an external interrupt. Each input terminal corresponds to an external interrupt, and the rising edge or falling edge of the input can trigger an interrupt.

(6) Watchdog

The watchdog is a controller hardware type timing device. It can be enabled by "task configuration" in Codesys. The watchdog function is not used by default.

The main function of the watchdog is to monitor the abnormality during the execution of the program or the failure of the internal clock. If the system crashes or the program enters the dead cycle, the watchdog timer will send a reset signal to the system or stop the program currently running by the PLC. We can vividly understand it as a puppy needs its owner to feed it regularly. If it is not fed within the specified time, it will be hungry

immediately. To configure the watchdog, you must define two parameters, time and sensitivity. The configuration of the watchdog is shown in the figure.



① Time

Codesys can configure independent watchdog for each task. If the target hardware supports long watchdog time setting, the upper and lower limits can be set. The default watchdog time unit is milliseconds (ms). If the program execution cycle exceeds the watchdog trigger time, the watchdog function will be activated and the current task will be aborted.

② Sensitivity

Sensitivity is used to define the number of task watchdog exceptions that must occur before the controller detects an application error. The default is 1.

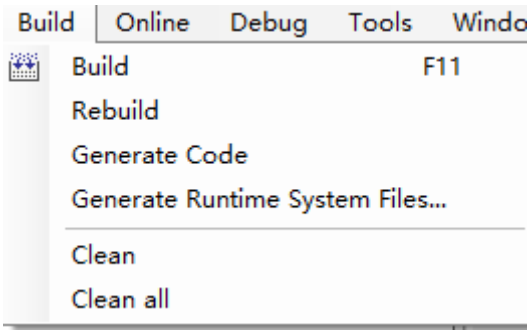
Final watchdog trigger time = time × Sensitivity. If the actual execution time of the program exceeds the watchdog trigger time, the watchdog is activated. For example, if the time is 10ms and the sensitivity is set to 5, the watchdog trigger time is 50ms. As long as the task execution time exceeds 50ms, the watchdog will be activated immediately and the task will be suspended.

5-4. Program download/read

5-4-1. Compile

After the program is written, the program needs to be compiled before downloading. The compile command checks the syntax of the written program and compiles only the programs added to the task. If the created POU is not added to the task, the compilation command does not check the syntax of the POU.

The compilation instruction does not generate any code, and only checks the syntax of POU. When the device login command is directly executed, the system will also execute the compilation command by default (equivalent to manually executing the compilation command first), and then execute the connection login command after compiling and checking that there is no syntax error. Similarly, no syntax check is performed on POU that are not added to the task during compilation. Executing the login command generates code at the same time.



(1) Compile: compile the current application.

(2) Recompile: if you need to recompile the compiled application, you can recompile it.

(3) Generate code: after executing this command, the machine code of the current application is generated. When executing the login command, the generated code is executed by default.
(4) Clear: deletes the compilation information of the current application. You need to regenerate the compilation information when logging in to the device again.
(5) Clear all: deletes all compilation information in the project.

After executing the compile command, you can see that the " PLC_PRG " added to the task is displayed in blue, and the " PLC_PRG " not added to the task is displayed in gray. The compilation instruction does not check the syntax of the gray POU because the program unit is not in the active state. The compilation instruction only checks the syntax of the POU in the active state. If the program unit that needs to be run is displayed in gray during the compilation process, you can check whether the program unit has been successfully added to the task that needs to be run.

After the compilation command is executed, you can see the information generated by compilation in the message bar, where you can see whether the compiled program has errors or warnings, and the number of errors and warnings. If there are errors and warnings, you can view and find them in the message window and modify the program according to the prompt information.

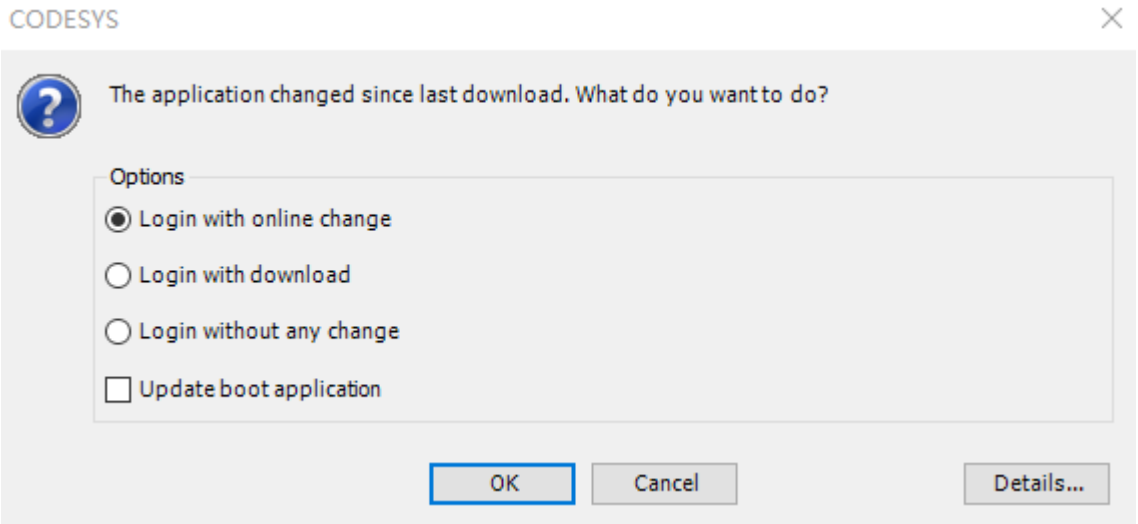
5-4-2. Login download

1. Login

Login enables the application to establish a connection with the target device and enter the online state. The prerequisite for correct login is to correctly configure the communication settings of the device and the application must be free of compilation errors.
For login with the current active application, the generated code must be free of errors and the device communication settings must be configured correctly. After login, the system will automatically select program download.

2. Download

Download command is valid in online mode. It includes compiling the current application and generating object code. In addition to syntax checking (compilation processing), application object code is generated and loaded into PLC.



(1) Login with online change

When the user selects this option, the changed part of the project is loaded into the controller. Use the "login with online change" operation to prevent the controller from entering the stop state.

Note:

①The user has performed at least complete download once.

②The pointer data will update the value of the latest cycle. If the data type of the original variable is changed, the accuracy of the data cannot be ensured. At this time, the pointer data needs to be reallocated.

(2) Login with download

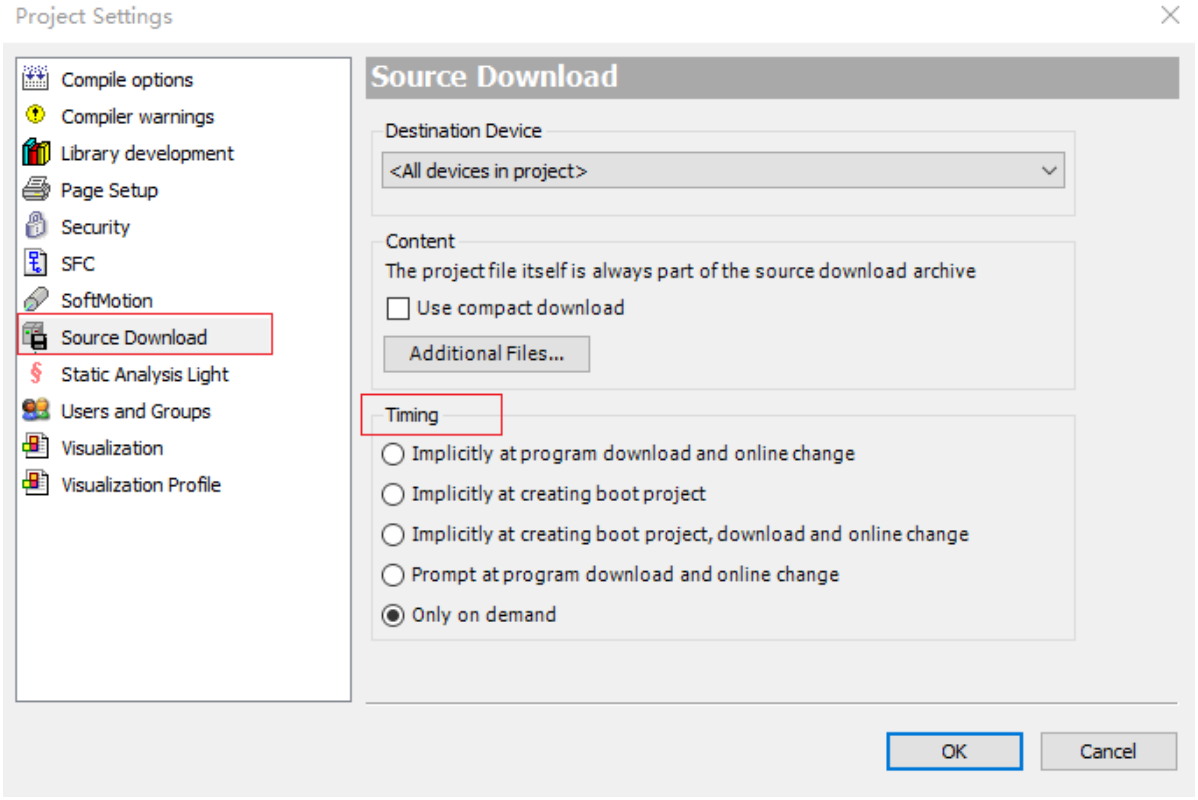
After selecting login and download, reload the entire project into the controller. The biggest difference from "login with online change" is that after downloading, the controller will stay in the stop mode and wait for the user to send the run command or restart the controllerto run the program.

(3) Login without any change

When you log in, you do not change the program that was last loaded into the controller.

5-4-3. Source code download

Codesys does not download the source code automatically by default for the protection of the programmer's source code. If you need to download the source code, you need to manually set it. Click "online" - > "source download to connected device". You can also set this attribute in project - > project settings - > source download - > timing.



5-4-4. Read program

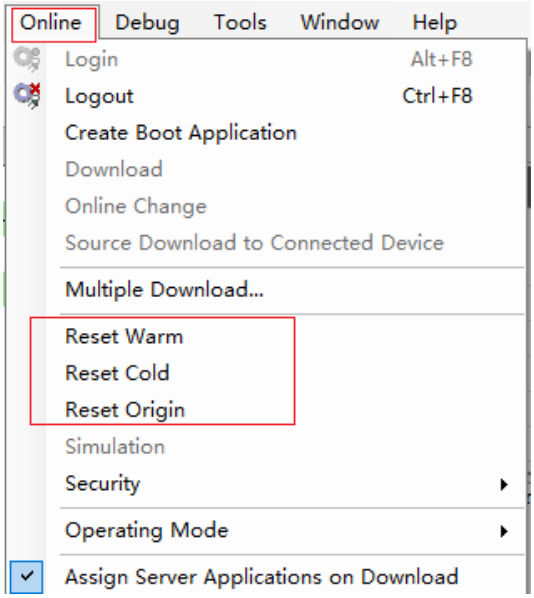
Open a device selection dialog box in the menu "File > source upload". The user selects the network path connected to the PLC and clicks "OK". If the archive file already exists under the path selected by the user, a prompt will be given whether to overwrite it.

It should be noted here that before reading the program, it is necessary to ensure that "source download to connected device" has been done in the previous download process. Otherwise, the data in the controller cannot be read.

5-5. Program debugging

5-5-1. Reset

There are three ways to reset the Codesys program, which can be selected in the "online" menu.



1. Reset warm

After the warm reset, except for the holding type variables (PERSISTENT and RETAIN variables), other currently applied variables are reinitialized. If the variables with initial values are set, the values of these variables will be restored to the setting initial values after warm reset. Otherwise, the variables will be set to the standard initial value 0.

2. Reset cold

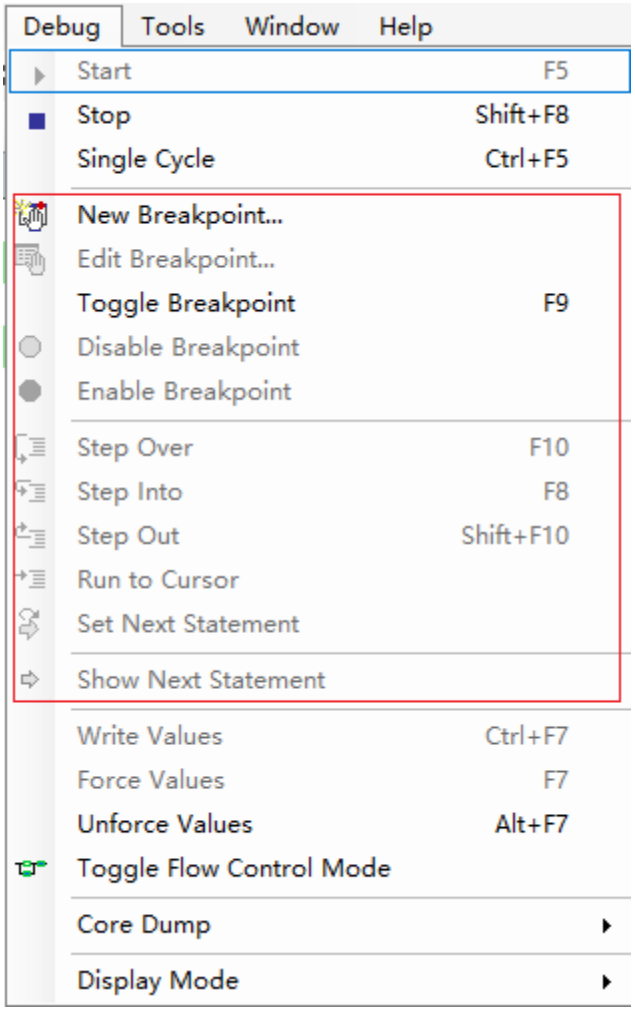
Unlike "warm reset", the cold reset command not only sets the value of the common variable to the initial value of the currently active application, but also sets the value of the holding variable (RETAIN variable) to the initial value of 0. The persistent variable remains unchanged.

3. Reset origin

When a programmable device is selected in the device tree, this command can be used either offline or online. Using this command will reset the device to the initial state, that is, any applications, boot projects and remaining variables in the device will be cleared.
Since all project information has been cleared, it is necessary to "download" the program again and "start" it after relogin.

5-5-2. Program debugging

The view of "debug" menu in Codesys is as shown in the figure. The main operations involve breakpoint setting and single cycle.



1. Breakpoint

Breakpoint is the function of processing stop in the program. When the program stops, the program R&D personnel can observe the contents of its variables and I/O and other related variables when the program reaches the breakpoint position, which is helpful to deeply understand the mechanism of program operation and find and eliminate program faults.

Breakpoints can be set in all programming languages in Codesys. In the text editor ST language, the breakpoint is set on the line. Set it on the network number in the FBD and LD editor. While in SFC, it is set at step.

2. Step

After the breakpoint is set, the program can be executed step by step. This function allows the program to run step by step, which is convenient for programmers to debug and check the logic errors in the program.

(1) Step over

This command will execute the current instruction in the program and stop after execution. When POU is not called, the step over and step into commands have the same effect. However, if the POU is called, step over will not enter the POU. Instead, the POU calling is regarded as a complete step and executed at one time. Step into will enter the POU. If the SFC language is used, step over will treat an action as a complete step and execute it at one time. If you want to jump to the called POU for single-step debugging, you must use step into.

(2) Step into

When executed, the current instruction position is indicated by a yellow arrow. If the current instruction does not call POU, using this command has the same effect as using the skip command.

(3) Step out

When single-step debugging is performed in a POU, the remaining instructions of the POU will be executed at one time by using step out, and then the next instruction at the place where the POU is called will be returned. Therefore, if the POU is called down layer by layer, the step out will return up layer by layer, one layer at a time. If the program does not contain any POU calls, the step out cannot be returned to the upper layer, and it will return to the beginning of the program.

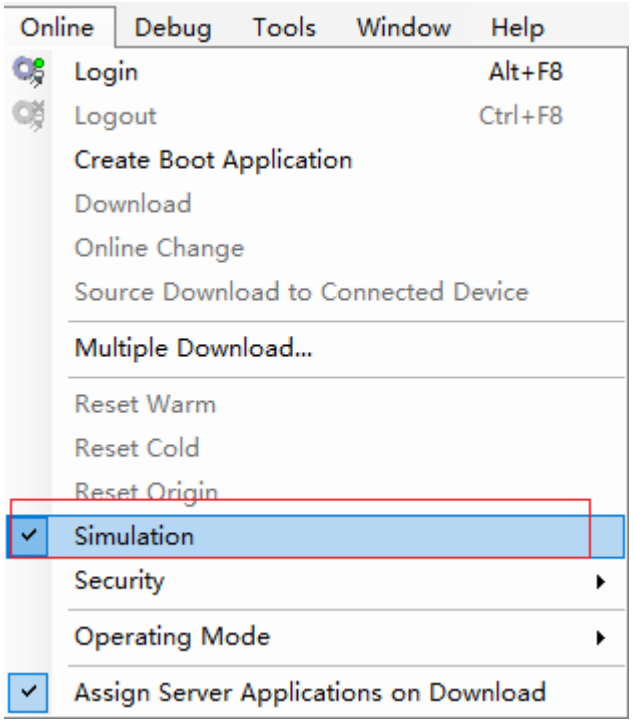
3. Single cycle

Select "single cycle" in "debug", so that the program can run step by step. That is, press once to run, and the program will stop after one cycle and wait for the next run instruction.

5-6. Simulation

Offline simulation

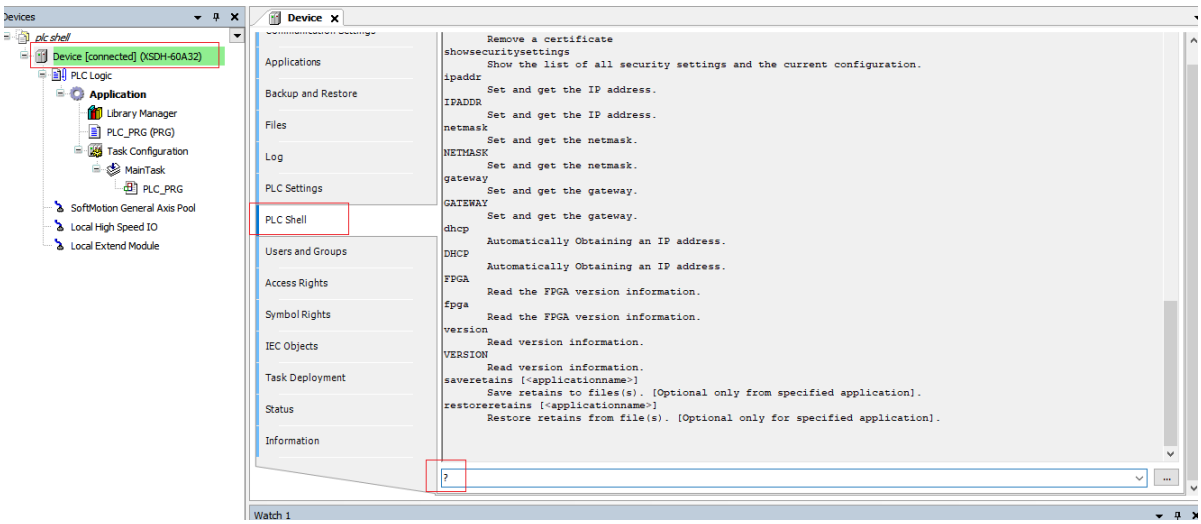
Select "simulation" in the menu "online" to enter the program running process in the simulation mode. After confirming that the option "simulation" has been marked, compile the program and enter the simulation mode after there is no error.



5-7. PLC script function

The PLC script is a text-based control monitor (terminal). The command with specific information obtained from the controller is input in an input line and sent to the controller as a string. The result display of the relevant string in the browsing window is returned. This function is used for diagnosis and debugging purposes.

Double click the mouse to select "device", find "PLC shell" in the right view, and enter the corresponding command in the command input box below. Enter ?, Press enter to display all commands supported by the controller. Refer to section 4-3.



Note: to use the script function, you must log in the PLC before using the corresponding command.

6. Industrial fieldbus technology

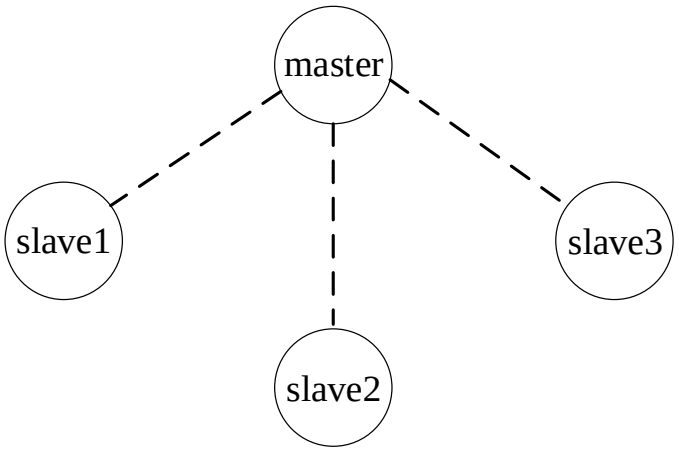
6-1. MODBUS communication

6-1-1. MODBUS overview

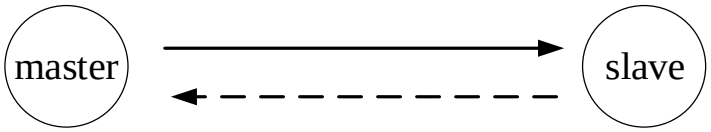
XS series programmable controller body supports Modbus protocol communication in the form of master and slave.

Master station form: when the programmable controller is used as the master station device, it can communicate with other slave devices using Modbus protocol. Data exchange with other equipment. Example: Xinje XS series PLC can control the frequency converter through communication.
Slave station form: when the programmable controller is used as the slave station equipment, it can only respond to the requirements of other master stations.

Master slave concept: in the RS485 network, at a certain time, there can be one master and multiple slaves (as shown in the figure below). The master station can read and write to any of the slave stations, and the slave stations cannot directly exchange data. The master station needs to write a communication program to read and write to one of the slave stations. The slave station does not need to write a communication program, and only needs to respond to the reading and writing of the master station. (wiring mode: all 485 + are connected together, and all 485 - are connected together)



In the RS232 network (as shown in the figure below), only one-to-one communication is available, and there is only one master and one slave at a time.

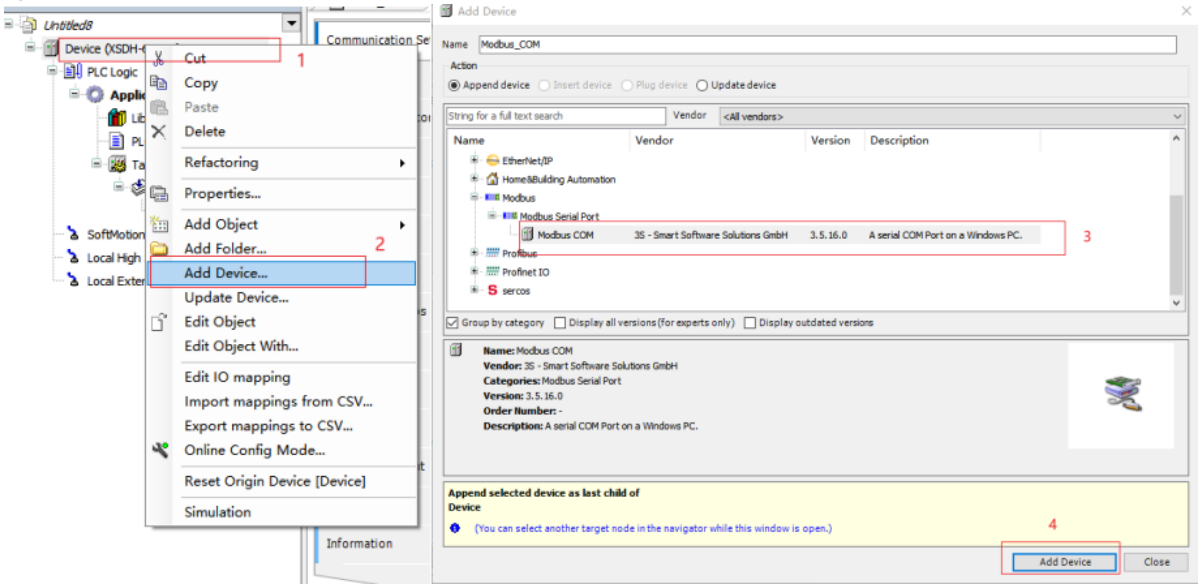


The reason why there are dotted arrows in the figure (including in RS485 network) is that theoretically, in the two networks, as long as each PLC does not send data, any PLC in the network can be used as the master station and other PLCs as the slave station. However, since there is no unified clock reference among multiple PLCs, it is easy to send data from multiple PLCs at the same time, which will lead to communication conflict failure. Therefore, it is not recommended to use this method.

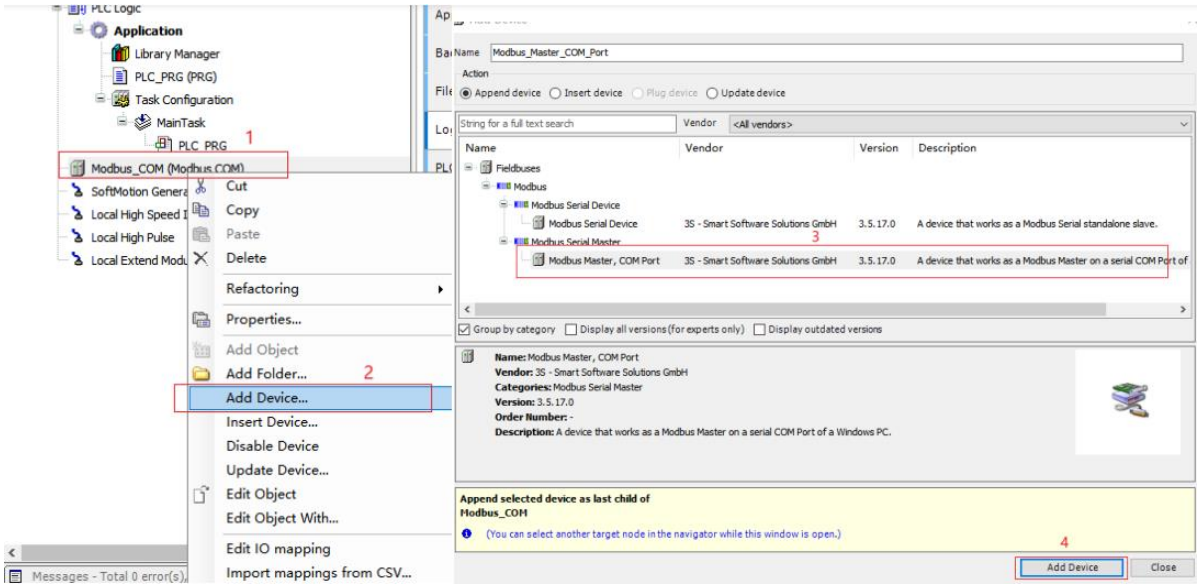
6-1-2. Parameter configuration

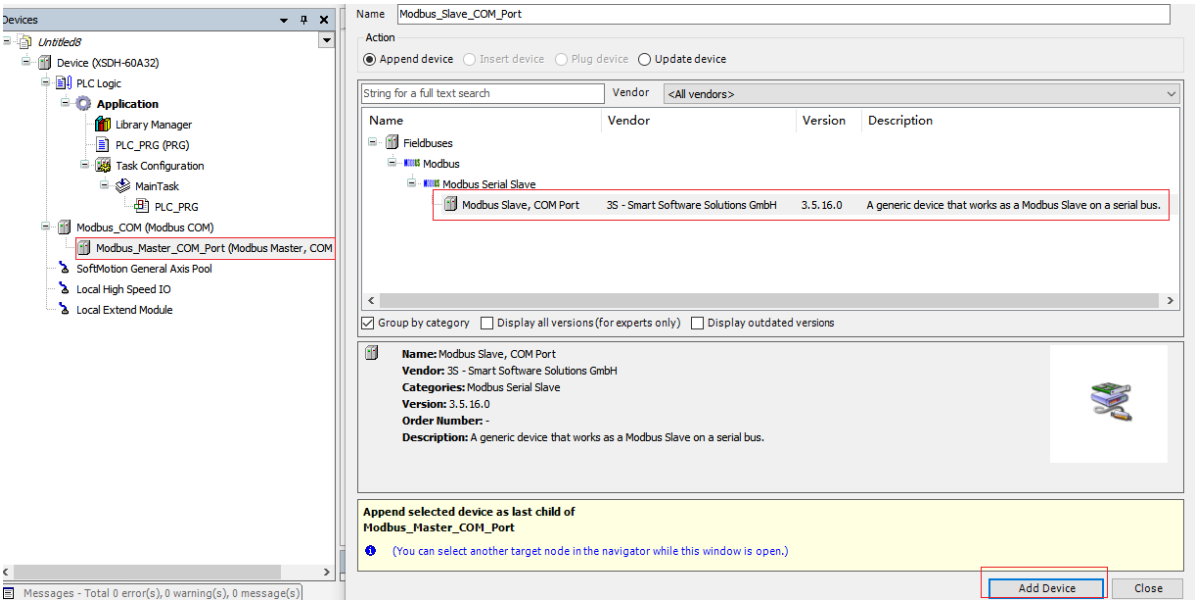
1. Modbus master station configuration

① In the applied project, right-click the device, click "add device", and click "MODBUS com" in the pop-up dialog box to add.

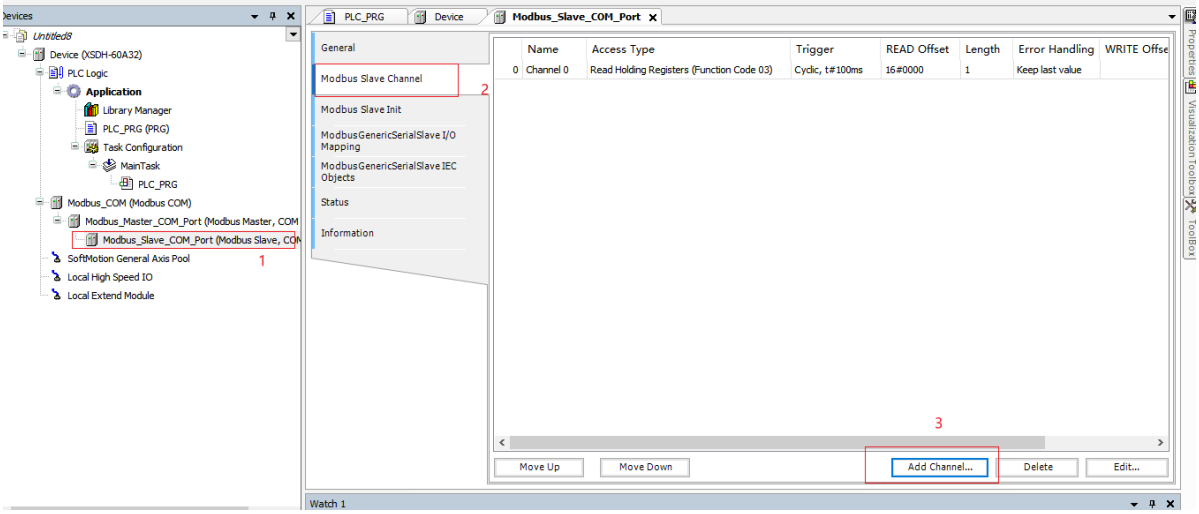


② After adding successfully, you can see "MODBUS COM" under the device, click "add device", select "MODBUS master, COM port" in the pop-up dialog box, and click "add device" to add. Select "MODBUS master, COM port", select "MODBUS slave, COM port" in the pop-up dialog box, and click "add device" to complete the addition.



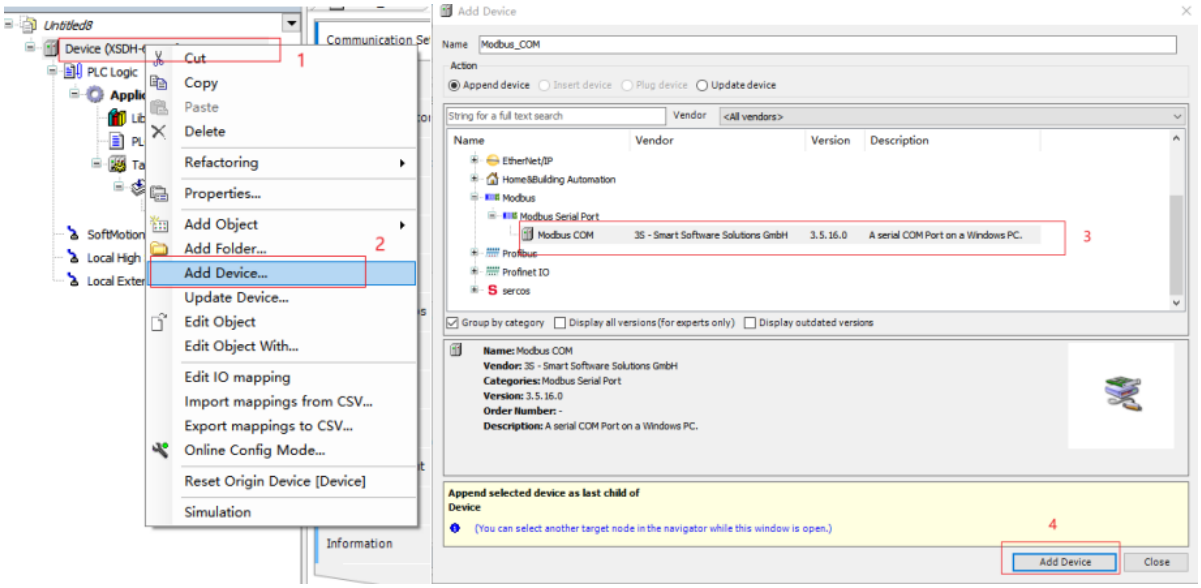


③ After adding, you can see the addition of the Modbus COM master station in the left device bar. Double click "modbus_slave_com_port" to configure the reading and writing in the "MODBUS slave channel" on the right.

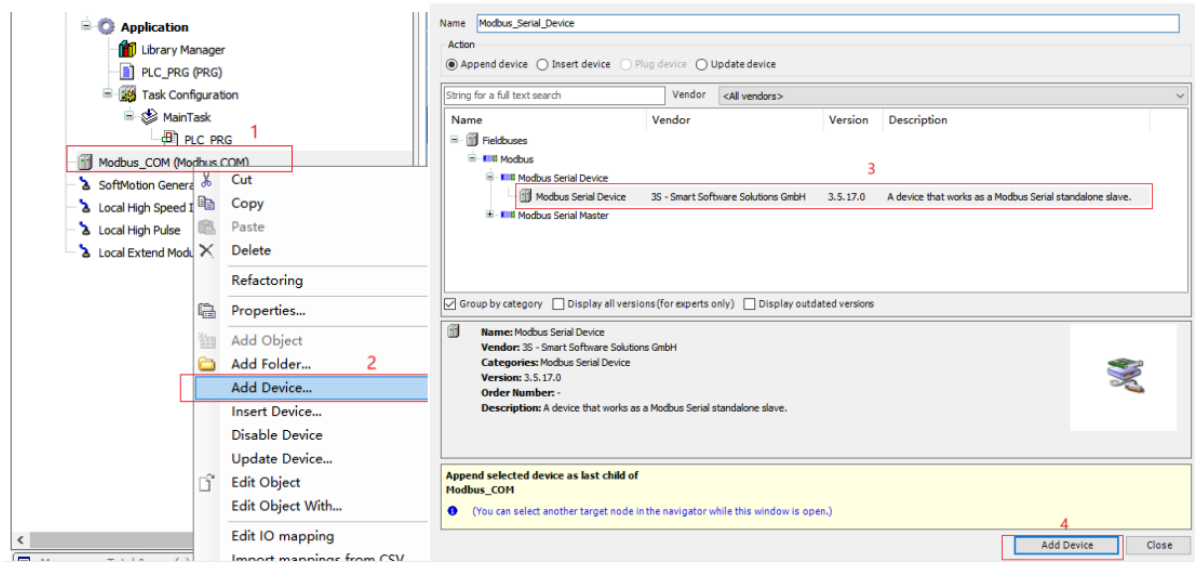


2. Modbus slave station configuration

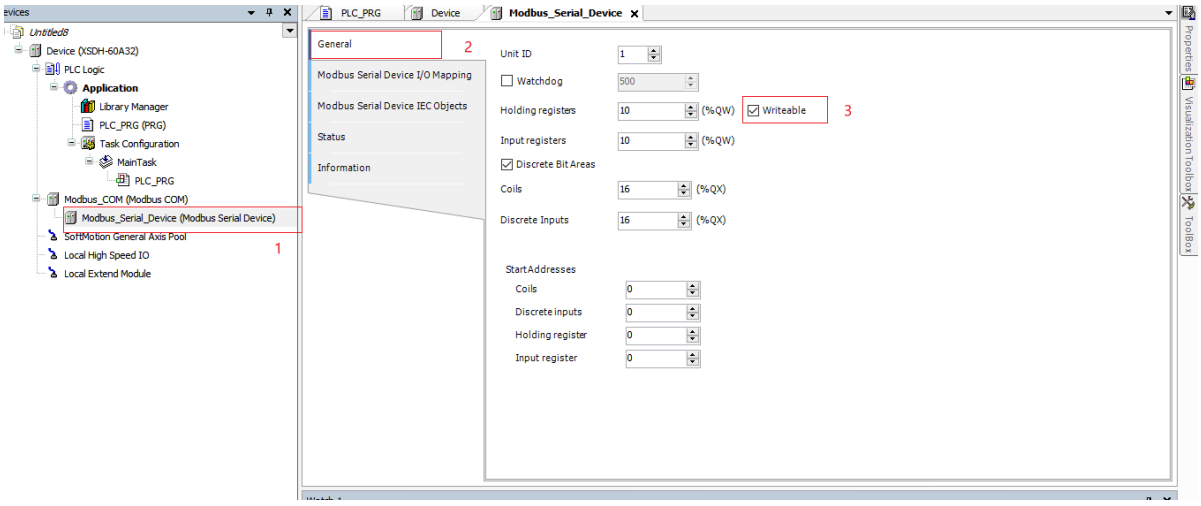
① In the applied project, right-click the "device", click "add device", and click "MODBUS COM" in the pop-up dialog box to add.

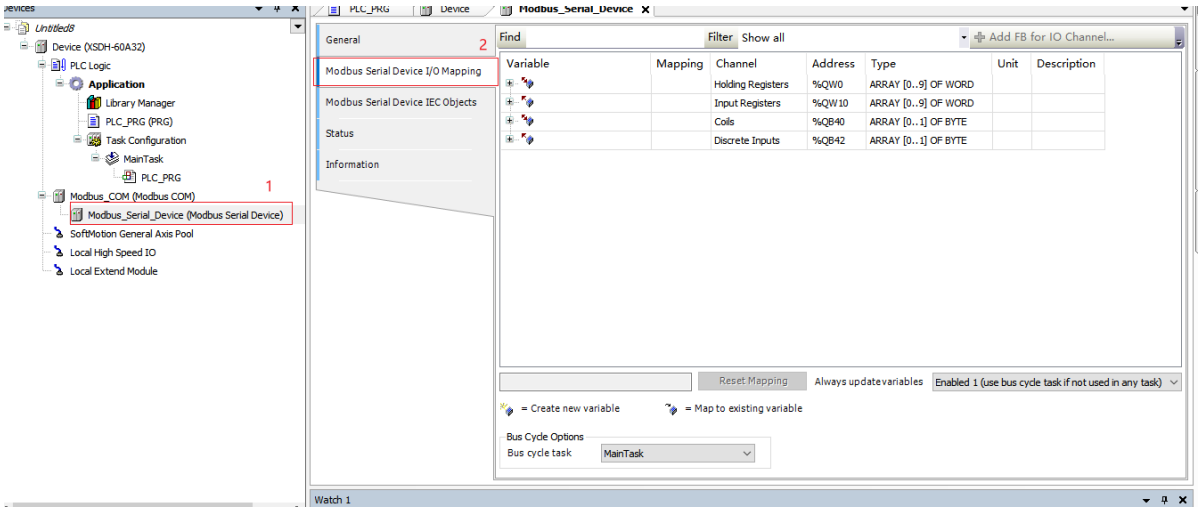


②After adding successfully, you can see "MODBUS COM" under the device. Right click "add device" and select "MODBUS serial device" in the pop-up dialog box. As shown in the following figure:



③After adding, you can see the addition of Modbus com slave station in the left device bar. Double click "MODBUS serial device" to configure registers and coils in "General". After configuration, you can monitor the reading and writing data of master station to XS slave station in "MODBUS serial device I/O mapping".





6-2. MODBUS TCP

6-2-1. MODBUS TCP overview

Modbus TCP uses TCP/IP to transmit MODBUS messages between stations. Modbus TCP combines the TCP/IP protocol and Modbus protocol as the data representation method of application protocol standard. Modbus TCP communication messages are encapsulated in Ethernet TCP/IP packets. Compared with the traditional serial port mode, Modbus TCP inserts a standard MODBUS message into the TCP message without any data parity and address.

XS series programmable controller supports Modbus TCP protocol communication in the form of master and slave.

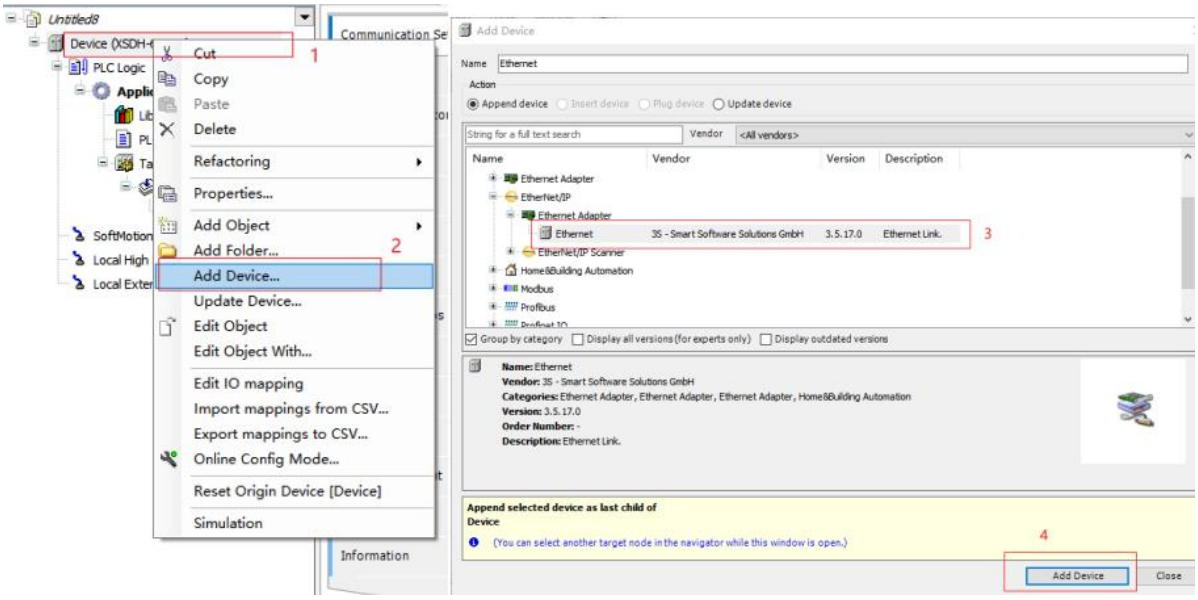
Master station form: when the programmable controller is the master station device, it can communicate with other slave devices using Modbus TCP protocol. A master station can connect up to 64 slave stations.

Slave station form: when the programmable controller is used as the slave station equipment, it can only respond to the requirements of other master stations.

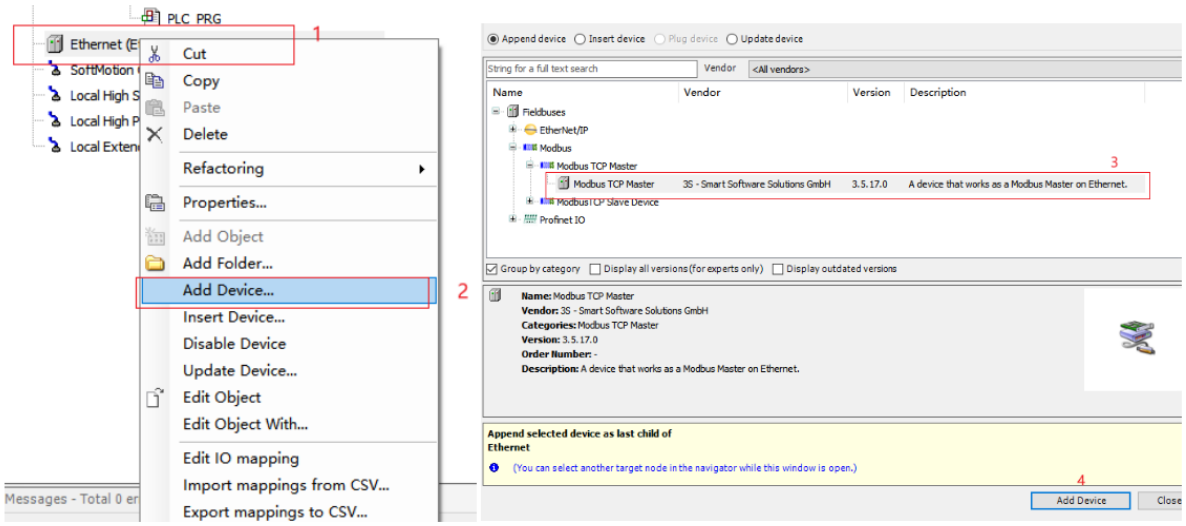
6-2-2. Parameter configuration

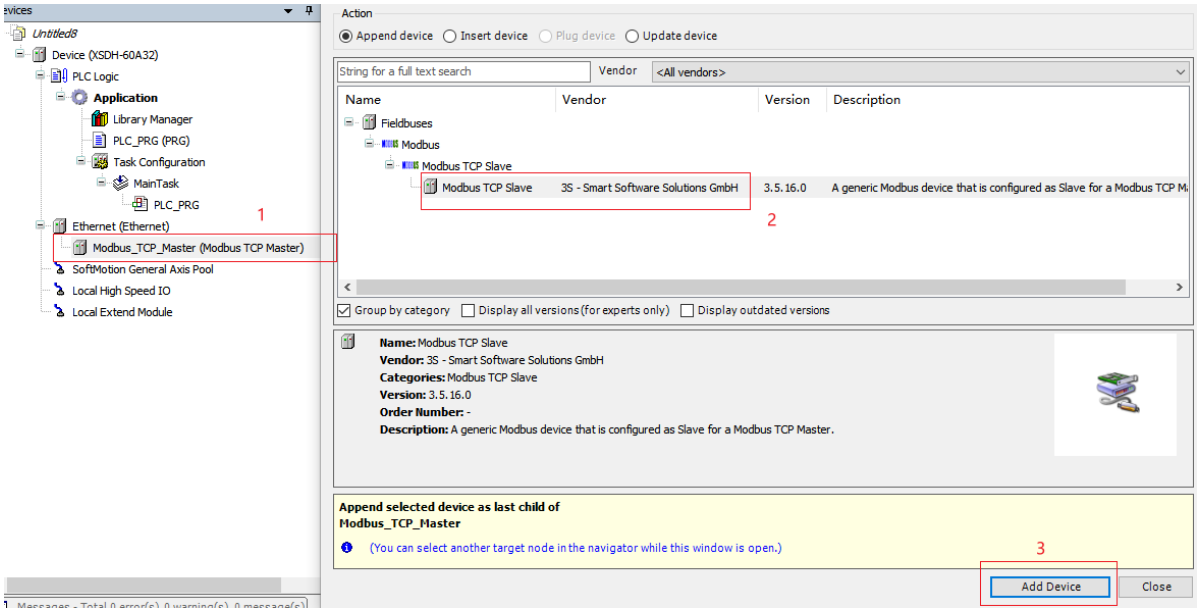
1. Modbus TCP client configuration

① In the applied project, right-click the "device", click "add device", and click "Ethernet" in the pop-up dialog box, as shown in the figure:

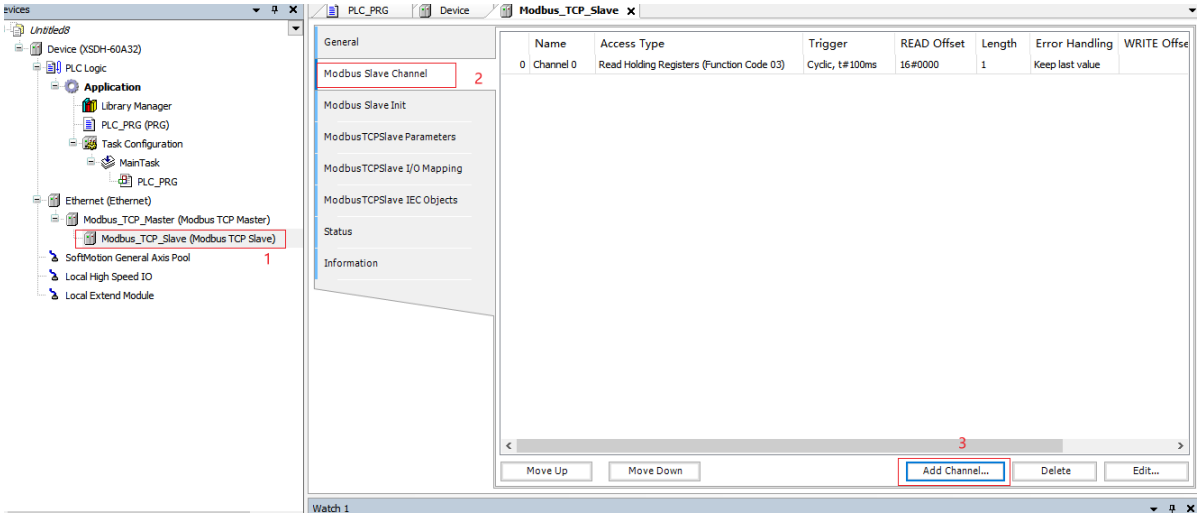


② After adding successfully, you can see "Ethernet" under the device. Right click "Ethernet", click "add device", select "Modbus TCP master" in the pop-up dialog box, and click "add device" to add. Select "Modbus TCP master", select "Modbus TCP slave" in the pop-up dialog box, and click "add device" to complete the addition.



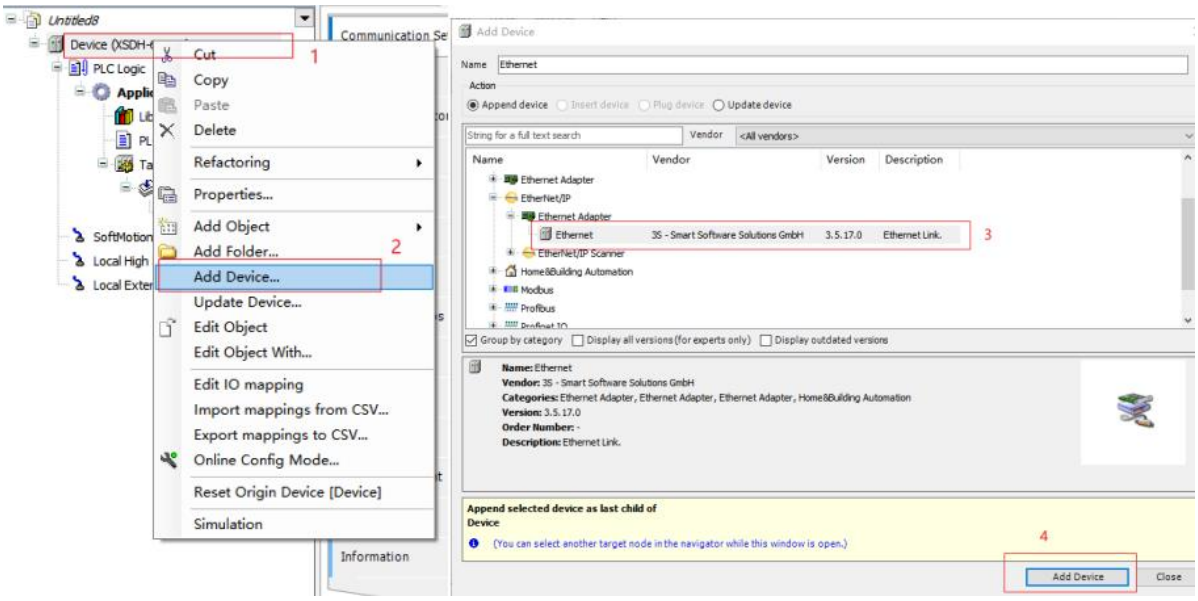


③ After adding, you can see the addition of Modbus TCP client in the left device bar. Double click "modbus_tcp_slave" to configure the read/write in the right "MODBUS slave channel".

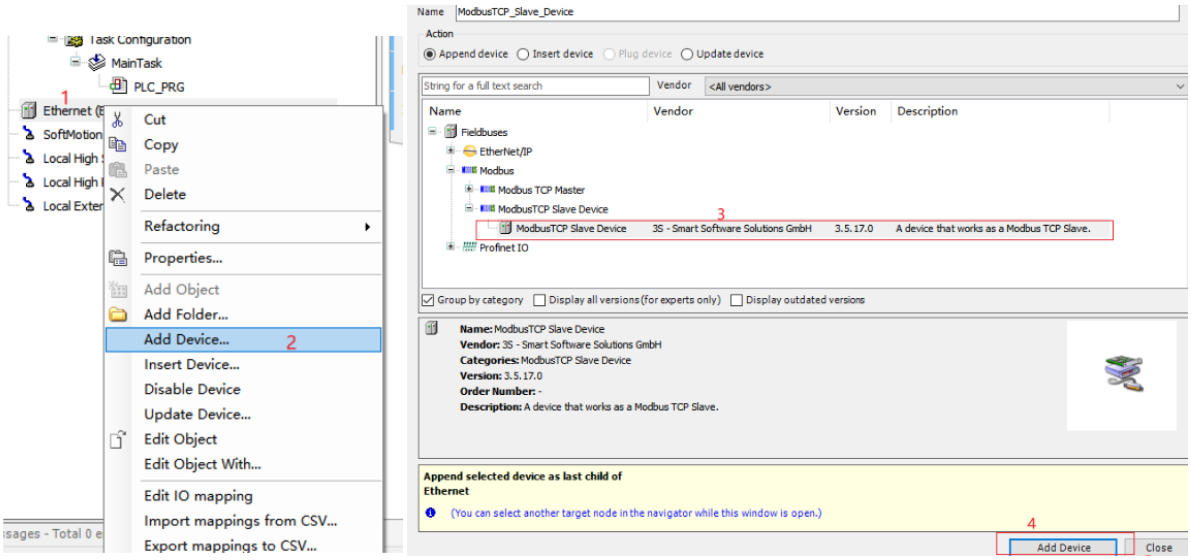


2. Modbus TCP server configuration

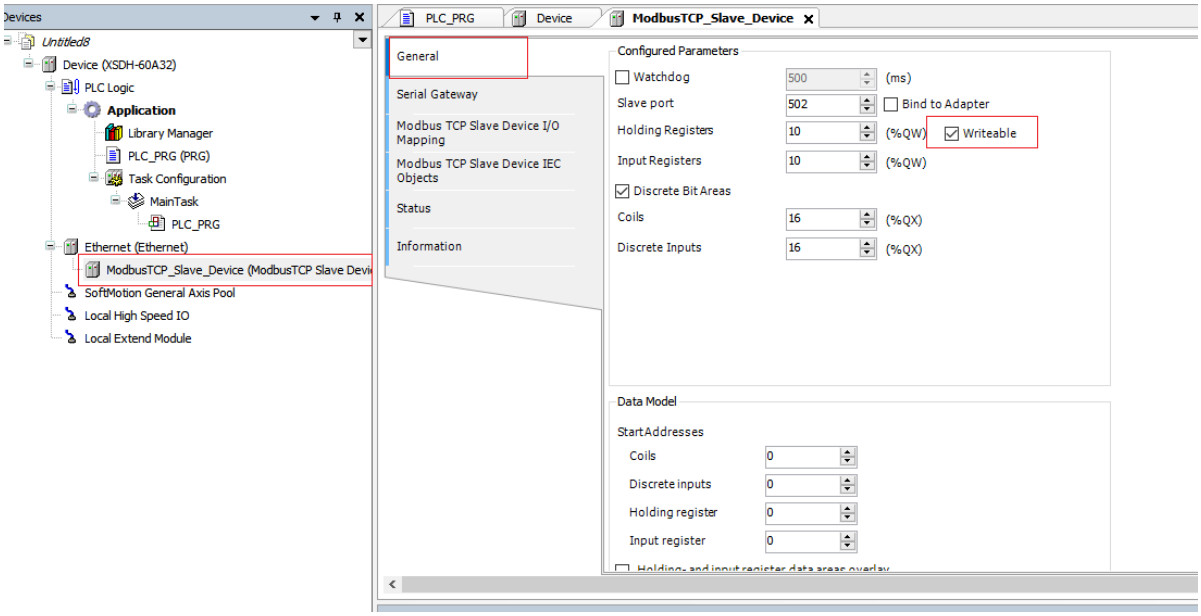
① In the applied project, right-click the "device", click "add device", and click "Ethernet" in the pop-up dialog box, as shown in the figure:

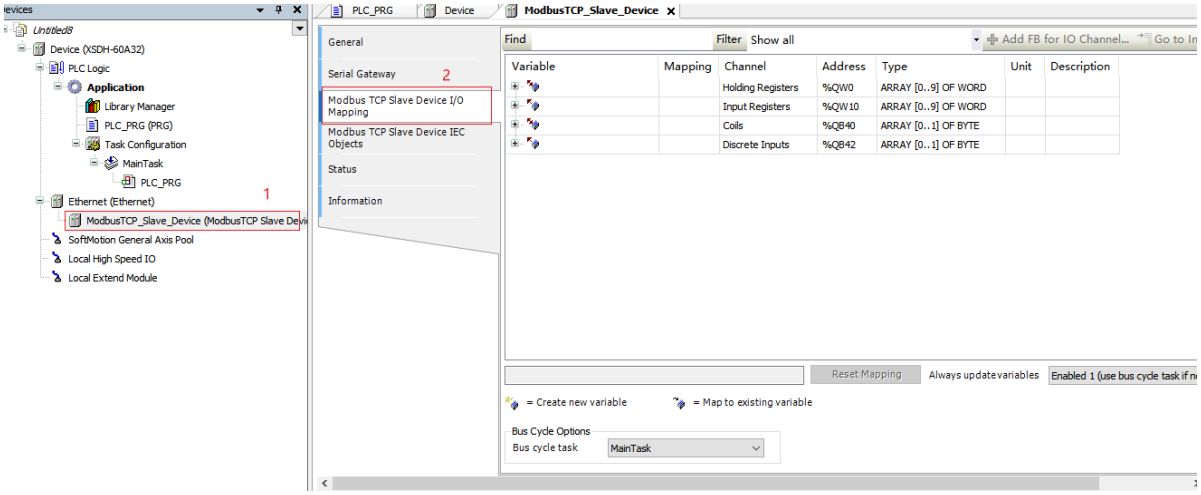


② After adding "Ethernet", right-click "add device" and select "Modbus TCP slave device" in the pop-up dialog box. As shown in the following figure:



③After adding, you can see the addition of Modbus TCP server in the left device bar. Double click "ModbusTCP_slave_device" to configure registers and coils in "General". After configuration, you can monitor the reading and writing data of the client to the XS server in "Modbus TCP slave device I/O mapping".





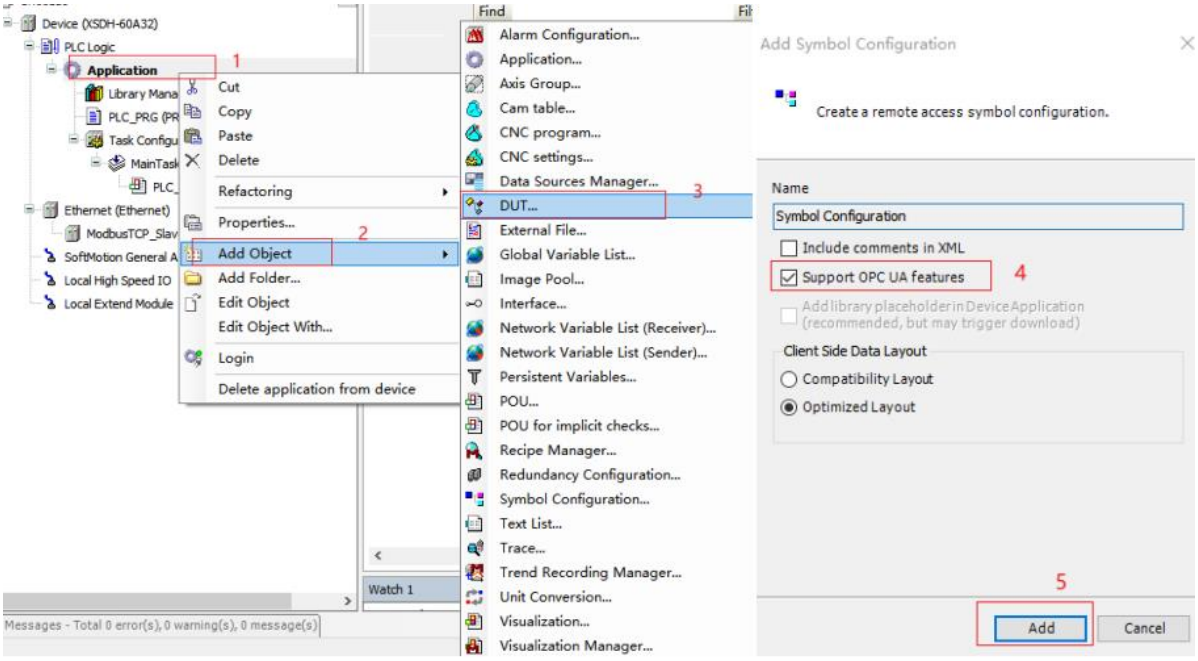
6-3. OPC UA

6-3-1. OPC UA communication overview

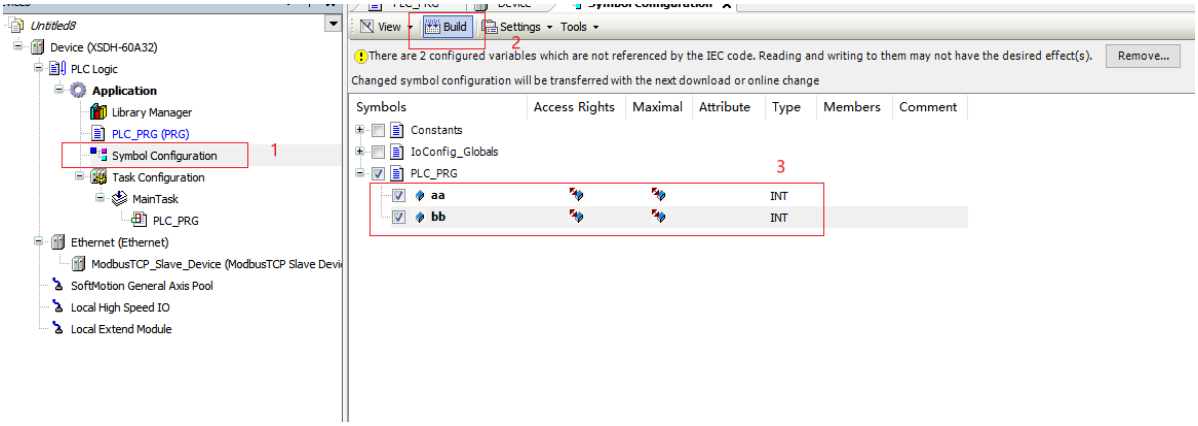
OPC UA was released in 2008. It is a platform independent service-oriented architecture that integrates all functions of various OPC classic specifications into an extensible framework.
In XS series PLC, OPC UA server is integrated, which can support users to access data in PLC through OPC UA client.

6-3-2. Parameter configuration

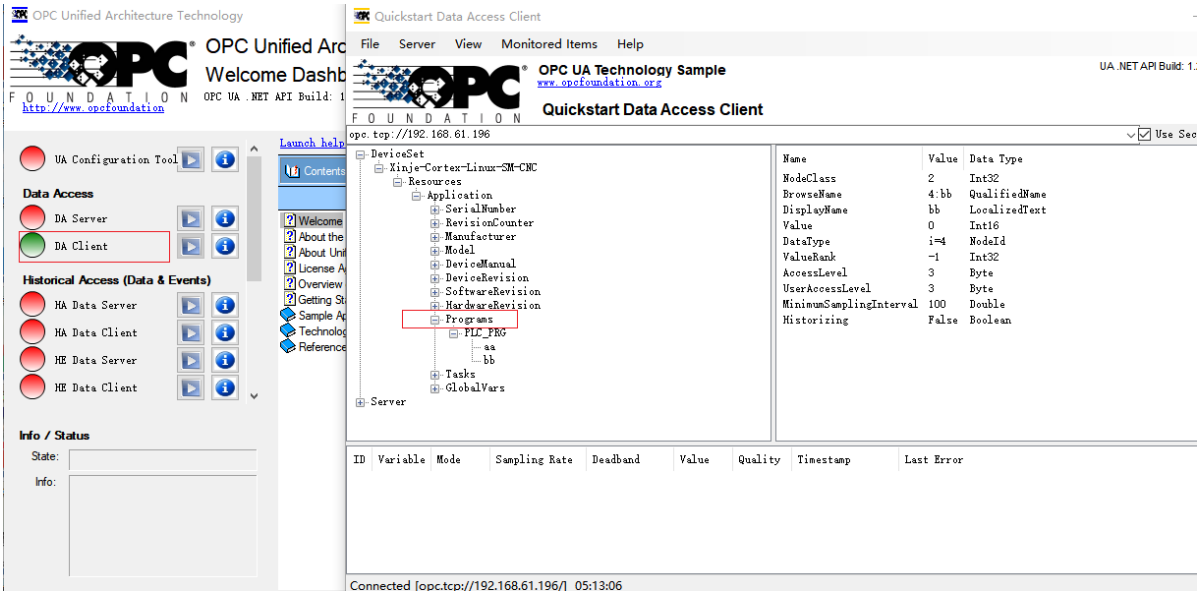
① In the applied project, right-click "application", select "add object" - "symbol configuration..", and select "support OPC UA feature" in the pop-up dialog box to add, then the function of OPCUA will be enabled.

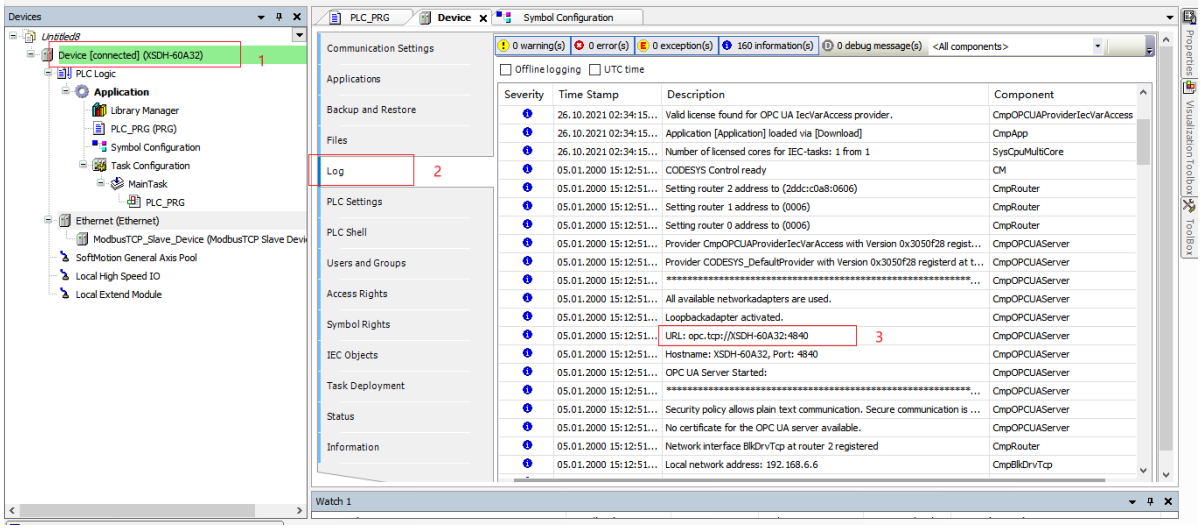


② double click “Symbol Configuration”, click "build" in the pop-up interface and select the parameters to be monitored.



③ After downloading the program, open the software of OPC UA, select "DA client", and enter the "IP" address of industrial control in the pop-up interface (for example, opc.tcp://192.168.61.196) Or you can enter the address in "log" and find "programs" in "DeviceSet". There are the parameters just checked. You can right-click the parameter -- monitor -- to read and write the parameter.





6-4. Free format

6-4-1. Free format overview

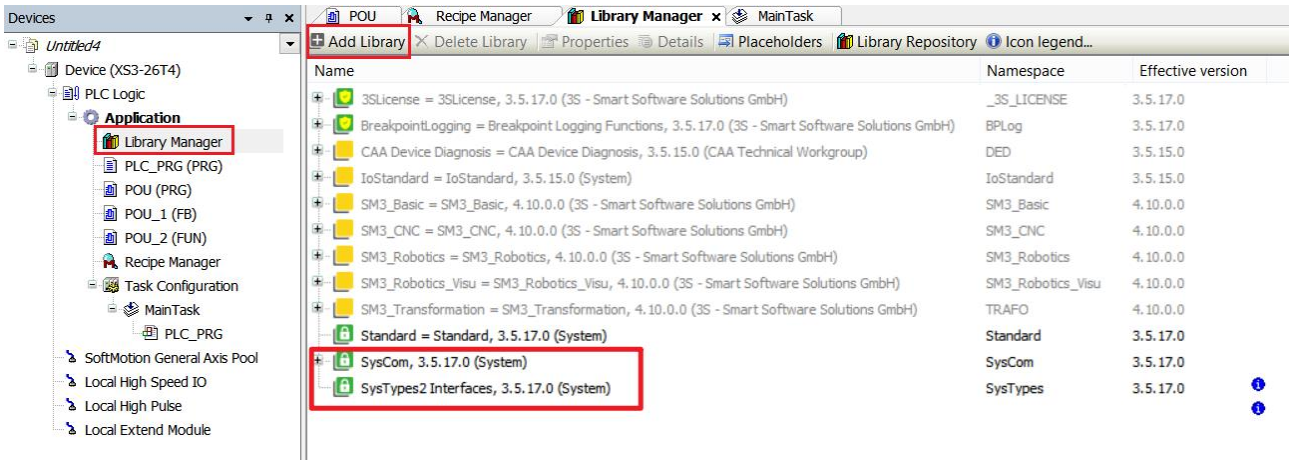
When Xinje PLC communicates with other equipment, if it is the lower computer, the upper computer must exchange data with it according to the data format of Modbus RTU. If Xinje PLC is the upper computer, when the lower computer also supports the Modbus RTU protocol, it can directly use the relevant communication instructions to communicate, making the program writing simpler and more efficient. If the lower computer does not directly support the Modbus RTU protocol, it can use free format communication.

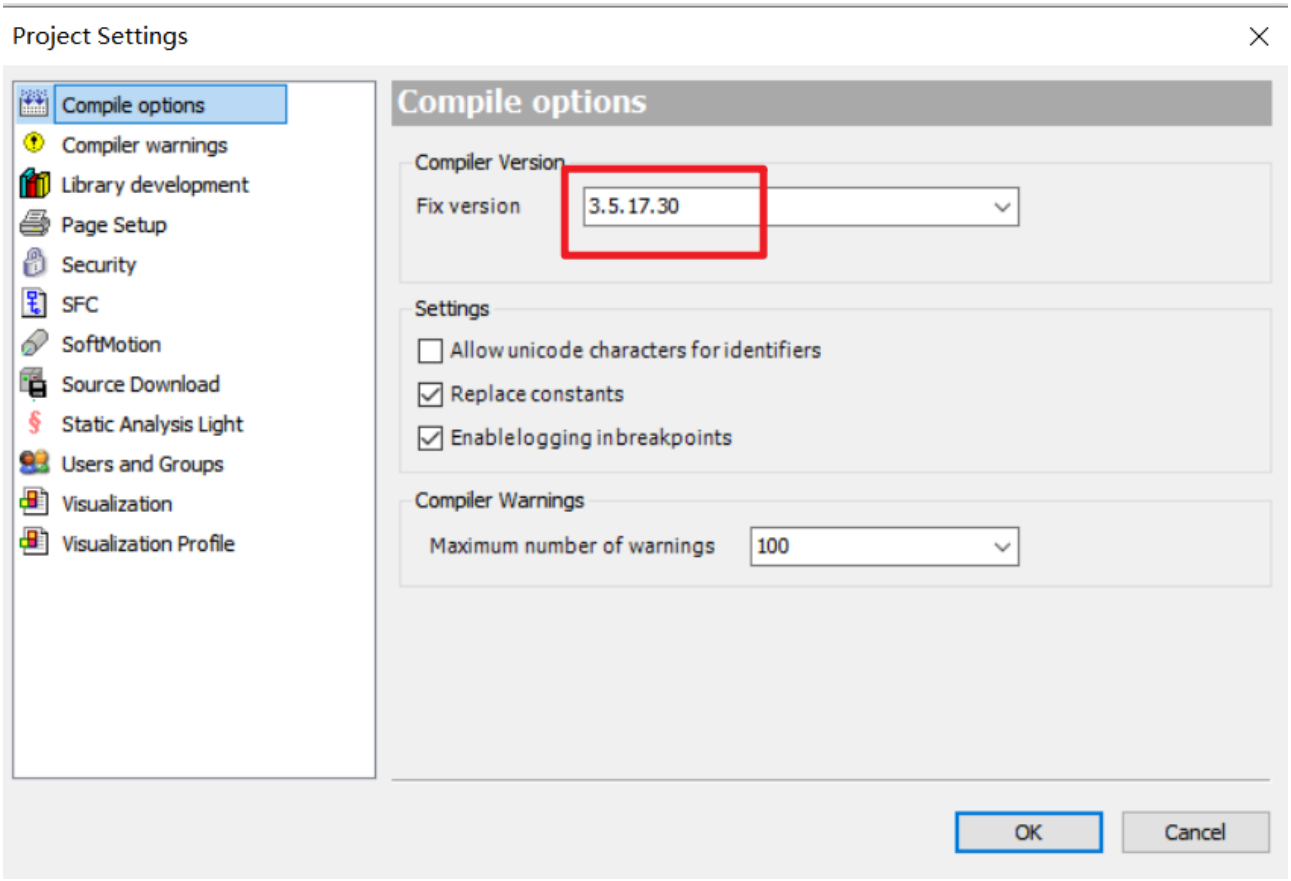
The so-called free format means that when the communication protocol of the lower machine does not match the PLC protocol, the PLC internally defines the data format to send data, so that it can communicate with many lower machines.

Free format communication is to transmit data in the form of data blocks. Each block can transmit up to 256 bytes. At the same time, each block can be set with or without a start and end character.

6-4-2. Parameter setting

① Add two libraries in the library manager -- Syscom and Systypes, and add the library version corresponding to the upper computer version. And the compiler version should also correspond to the upper computer version.





6-4-3. Application

Example 1: free format communication between two PLCs and data transmission / reception are realized through the following program.

Note: when free format communication is used for data reception, either the cycle of the corresponding task needs to be extended when it is normally on, or the rising edge triggering is used for reception.

Program operation:

(1) Install the library to be used according to the steps in section 6-5-2.

(2) Write a free-form program.

```
PROGRAM POU
VAR
    SysCom2Settings:SysCom.SysComSettings;
    SysCom2SettingsEx:SysCom.SysComSettingsEx;
    StartSetting:BOOL:=1;
    SendData:ARRAY[0..19] OF BYTE;
    result:(*POINTER TO *)SysTypes.RTS_IEC_RESULT;
    Send_start:BOOL;
    Ton0:Standard.TON;
    i:INT;
    RCVData:ARRAY[0..19] OF BYTE;
    RCV_start:BOOL;
    hCom:SysTypes.RTS_IEC_HANDLE:=SysTypes.RTS_INVALID_HANDLE;
    size : UDINT;
//  close_en:BOOL;
END_VAR
```

```
IF StartSetting THEN
        hCom:=SysCom.SysComOpen(sPort:=SysCom.SYS_COMPORT2,pResult:=ADR(result));
    IF hCom<>RTS_INVALID_HANDLE THEN
        result := SysComGetSettings(hCom:= hCom, pSettings:= ADR(SysCom2Settings), pSettingsEx:= ADR(SysCom2SettingsEx));//call the interface
        SysCom2Settings.sPort:=SysCom.SYS_COMPORT2;//port
        SysCom2Settings.ulBaudrate:=SysCom.SYS_BR_19200;// baud rate
        SysCom2Settings.byStopBits:=SysCom.SYS_ONESTOPBIT;// stop bit
        SysCom2Settings.byParity:=SysCom.SYS_EVENPARITY;// parity
        SysCom2Settings.ulTimeout:=SYS_NOWAIT;
        //SysCom2Settings.ulBufferSize:=8;
        SysCom2SettingsEx.byByteSize:=8;// data bit
        SysCom2SettingsEx.bOutX := FALSE;// flow control
        SysComSetSettings(hCom:= hCom, pSettings:= ADR(SysCom2Settings), pSettingsEx:= ADR(SysCom2SettingsEx));
        SysComPurge(hCom:= hCom);
    END_IF
    StartSetting:=0;// port settings cannot be always ON
END_IF
Ton0(IN:=NOT  Ton0.Q ,PT:=T#1S,Q=>,ET=>);
FOR i:=0 TO 19 BY 1 DO
 IF Ton0.Q THEN
    SendData[i]:=SendData[i]+1;
 END_IF
END_FOR
    // start sending data
IF Send_start AND Ton0.Q THEN
    SysCom.SysComWrite(hCom:=hCom,pbyBuffer:=ADR(SendData),ulSize:=SIZEOF(SendData),ulTimeout:=SysCom.SYS_NOWAIT,pResult:=ADR(result));
END_IF
//receiving the data
IF RCV_start    THEN
    size := SysCom.SysComRead(hCom:=hCom,pbyBuffer:=ADR(RCVData),ulSize:=SIZEOF(RCVData),ulTimeout:=SysCom.SYS_NOWAIT,pResult:=ADR(result));
END_IF
```

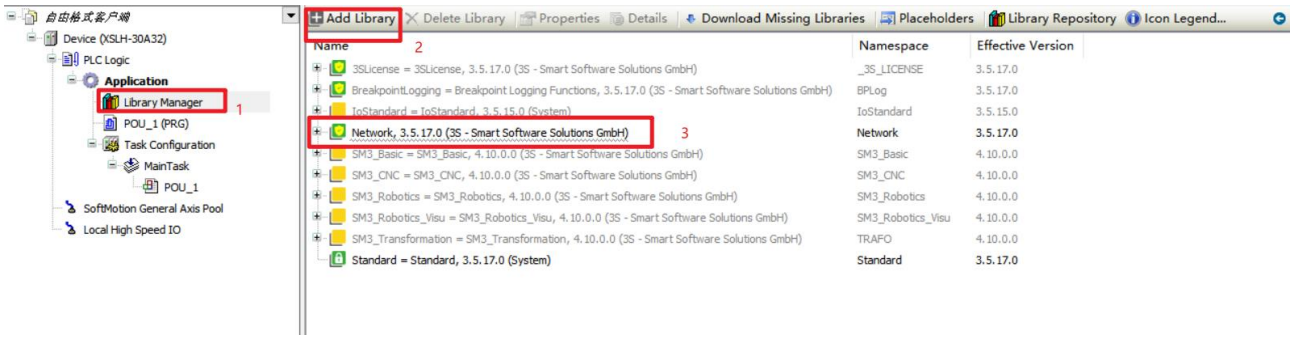
6-5. TCP/IP

6-5-1. TCP/IP overview

TCP / IP protocol is a common Ethernet communication protocol. Compared with the open interconnection model ISO, it adopts a more open way and is widely used in practical projects. TCP / IP protocol can be used on a variety of channels and underlying protocols (such as T1, X.25 and RS232 serial interface). Specifically, TCP / IP protocol is a protocol group including TCP protocol, IP protocol, UDP protocol, ICMP protocol and other protocols.

6-5-2. Parameter configuration

Add the TCP / IP library network in the library manager, and add the network version corresponding to the upper computer version.



Define relevant variables in POU and write programs.

6-5-3. Application

Example 1: through the following procedure, TCP/IP communication between two PLCs is realized to receive and send data. The IP address of PLC 1 is 192.168.6.17, and the IP address of PLC 2 is 192.168.6.18.

Note: the server needs to open the connection first and wait for the client to connect. Otherwise, TCP/IP communication may not be established successfully.

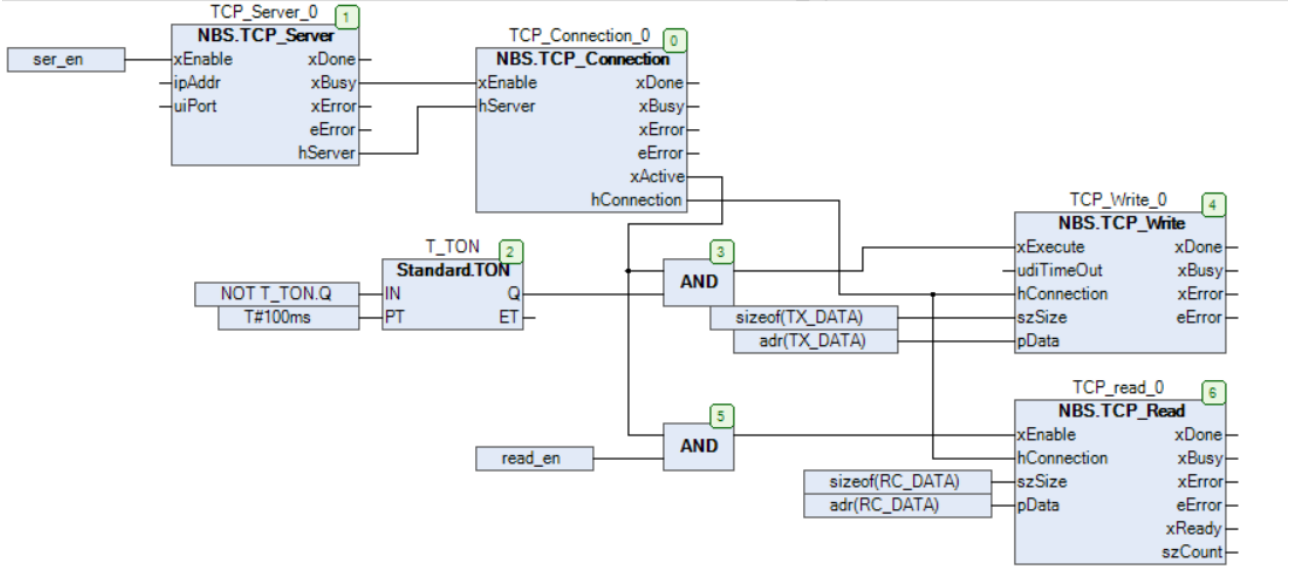
Program operation:

(1) Install the library to be used according to the steps in section 6-6-2.

(2) Write TCP/IP programs.

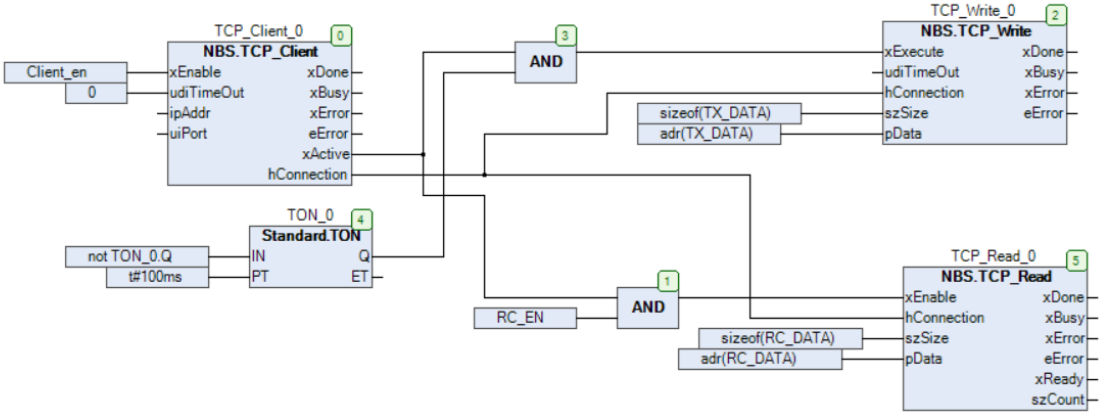
Programming: use the function blocks " NBS.TCP_Server ", " NBS.TCP_Client ", " NBS.TCP_Connection ", " NBS.TCP_Write ", " NBS.TCP_Read " to set the server, client, sending and receiving parameters of the TCP/IP used in the program.

```
PROGRAM POU
VAR
    TCP_Connection_0: NBS.TCP_Connection;
    TCP_Server_0: NBS.TCP_Server:=(ipaddr :=STRUCT(sAddr:='192.168.6.17'),uiPort:=4000);// set server IP and port
    TCP_Write_0: NBS.TCP_Write;// write data
    TCP_read_0: NBS.TCP_Read;// read data (receive data)
    ser_en: BOOL;// open the server
    T_TON:Standard.TON;
    TX_DATA:ARRAY[0..19] OF BYTE;
    RC_DATA:ARRAY[0..19] OF BYTE;
    read_en: BOOL;// start receiving
END_VAR
```




```
PROGRAM POU_1
VAR
    TCP_Client_0: NBS.TCP_Client:=(ipaddr :=STRUCT(sAddr:='192.168.6.17'),uiPort:=4000);//Set the server IP and port to be accessed by the client
    TCP_Write_0: NBS.TCP_Write;//write data
    TCP_Read_0: NBS.TCP_Read;// read data (receive data)
    Client_en: BOOL;// open client
    TX_DATA:ARRAY[0..19] OF BYTE;
    RC_DATA:ARRAY[0..19] OF BYTE;
    TON_0: Standard.TON;
    RC_EN: BOOL;// start receiving
END_VAR
```

'Auto Data Flow Mode' has been activated Properties Help



7. Common problems and solutions

7-1. Package

7-1-1. Package naming rule

Naming format: XSDH-60A32_3.5.15.40_1.0.0_P1_20211027

①　②　③ ④　⑤

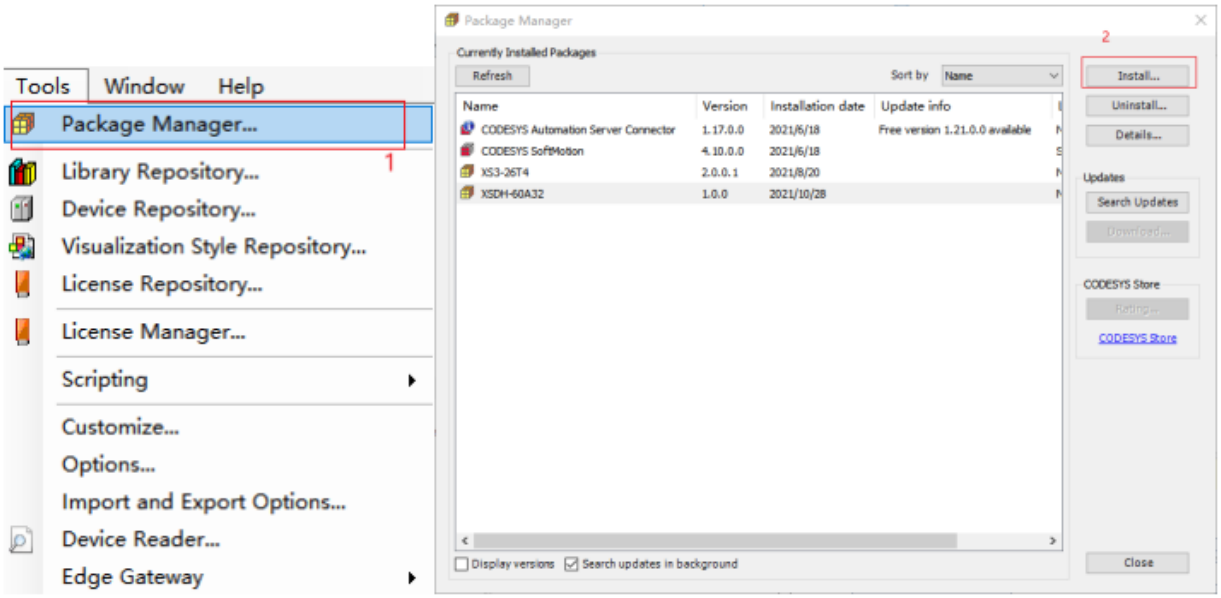
No.	Name	Note
①	XSDH-60A32	PLC model
②	3.5.15.40	Runtime version
③	1.0.0	Package production version
④	P1	The first online upgrade package after production
⑤	20211027	Package update date

7-1-2. Obtain the Package

Please contact us, email sales@xinje.com.

7-1-3. Package installation

Select "tools" - "package manager..", install the package in the pop-up interface, select "Install", and find the location of the package for installation. For example, to install the package of XSDH-60A32, it is best to uninstall the previous package before installing a new one.



7-2. XS series PLC firmware update

7-2-1. Firmware naming rule

Naming rule: XSDH-60A32_3.5.15.40_1.0.0_P1_20211027

①　②　③　④　⑤

No.	Name	Note
①	XSDH-60A32	PLC model
②	3.5.15.40	Runtime version
③	1.0.0	Firmware production version
④	P1	The first online firmware upgrade after production
⑤	20211027	Firmware update date

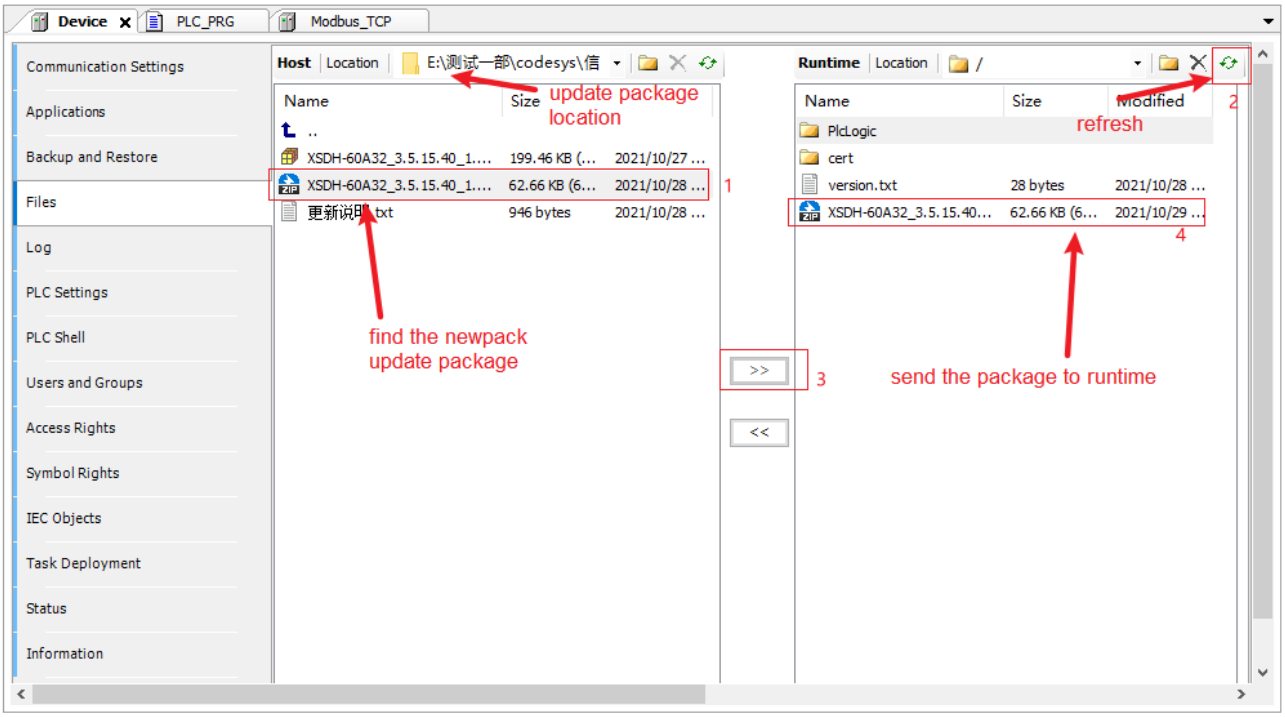
7-2-2. Obtain the firmware

Please contact us, email sales@xinje.com.

7-2-3. Firmware installation and precautions

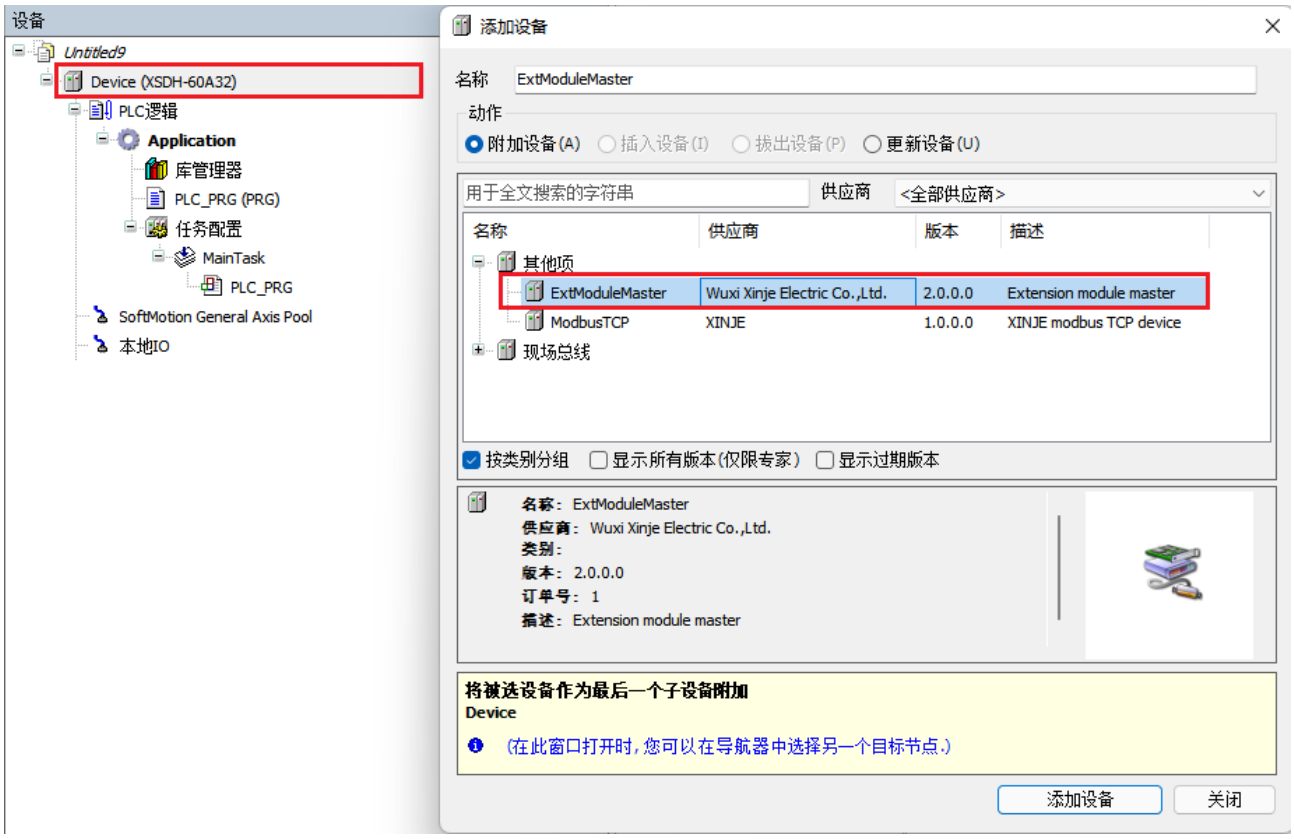
Upgrade firmware through newpack package:

Create the equipment standard project, connect the equipment, select the "Files" option in the main equipment directory, click refresh in the upper right corner, transfer the newpack upgrade package to the runtime, wait for the transfer to be completed, restart the equipment, the ERR light will be on during the upgrade of the equipment, and the ERR will be off after the update is completed. At this time, the equipment can be scanned.

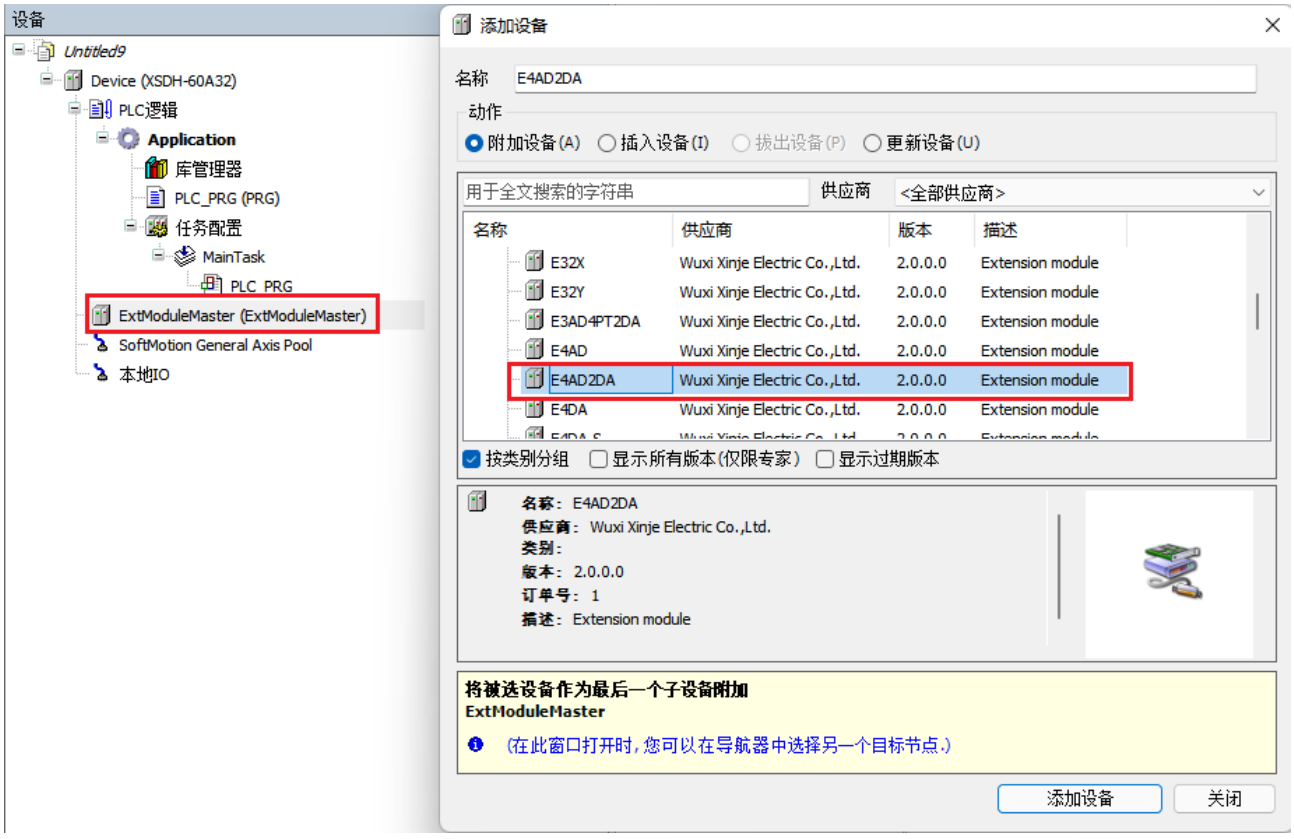


7-3. XS series local expansion module

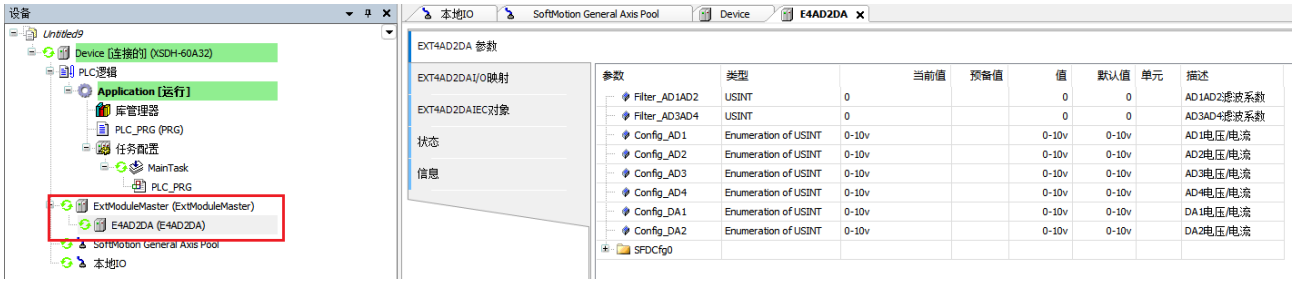
① After connecting the local expansion module with the PLC, right-click device → add device → select "exmodulemaster".



② The expansion module can be added by scanning or manually adding.

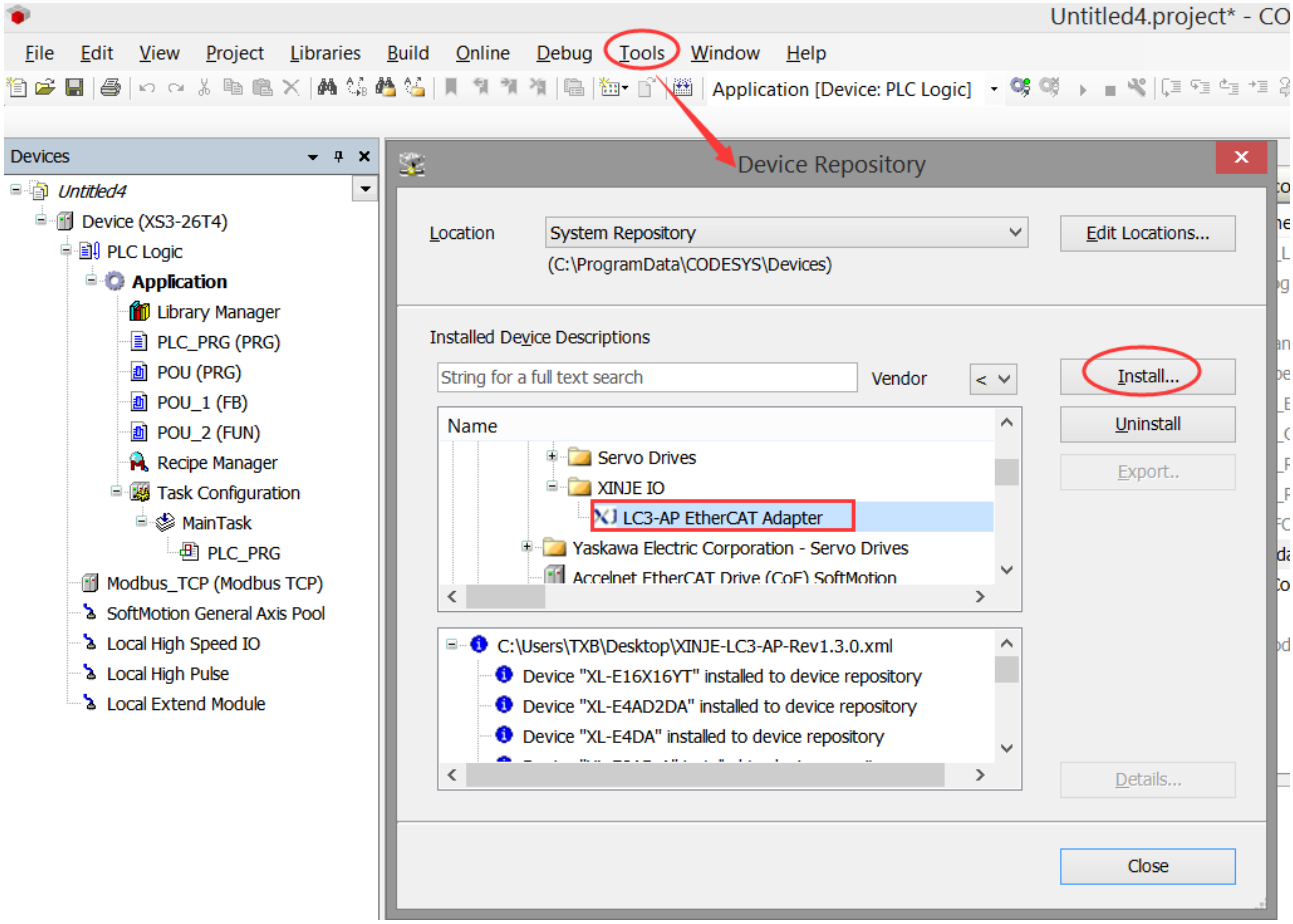


③ Testing result.

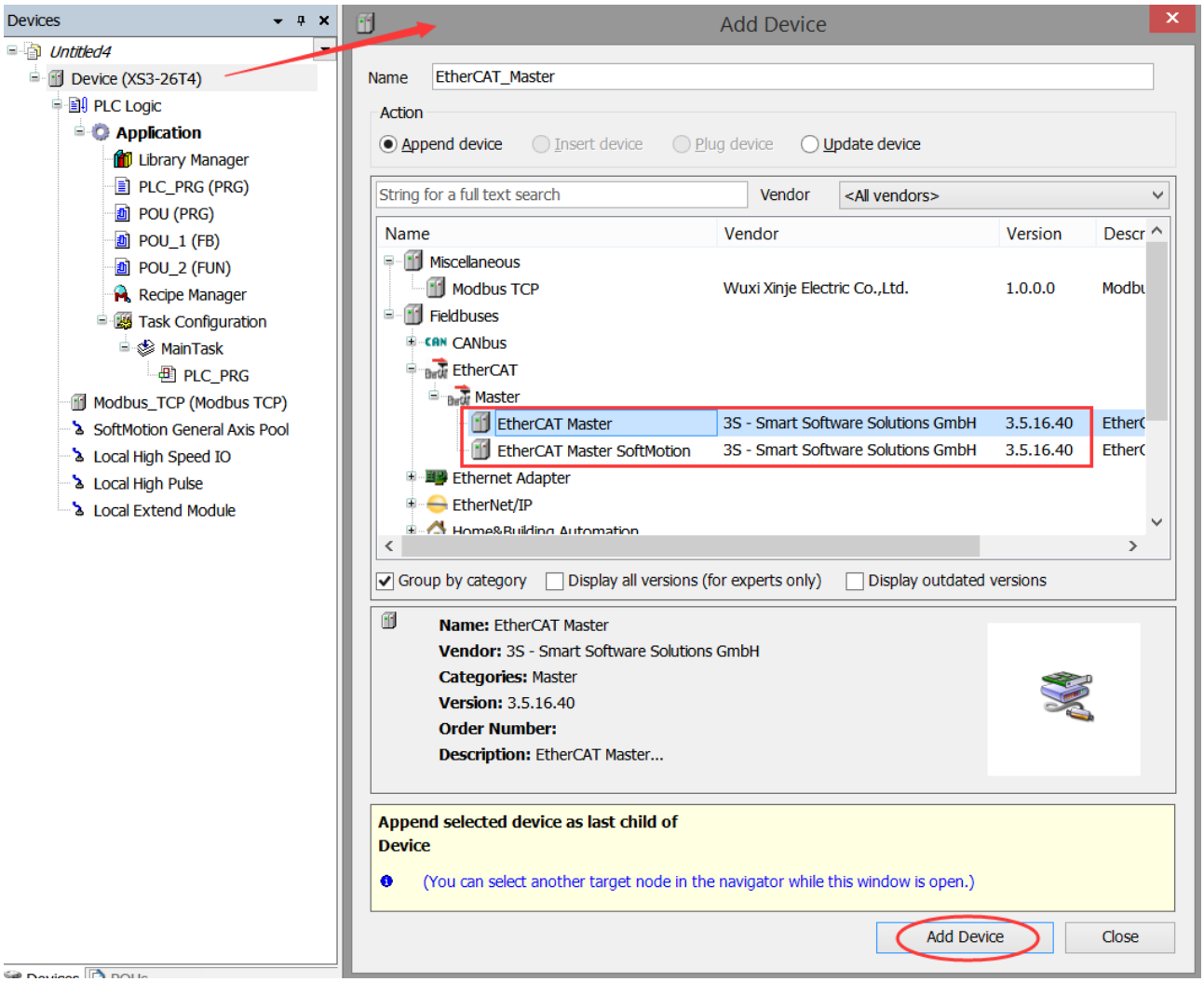


7-4. XS series remote expansion module

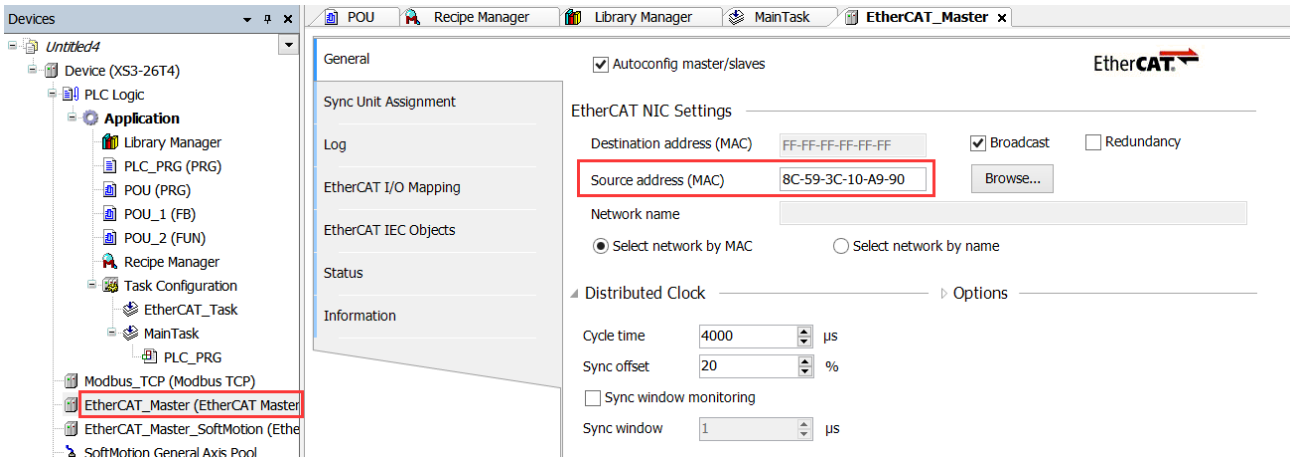
① Connect DC24V power supply with remote module LC3-AP.
② Add LC3-AP description file in the codesys software.



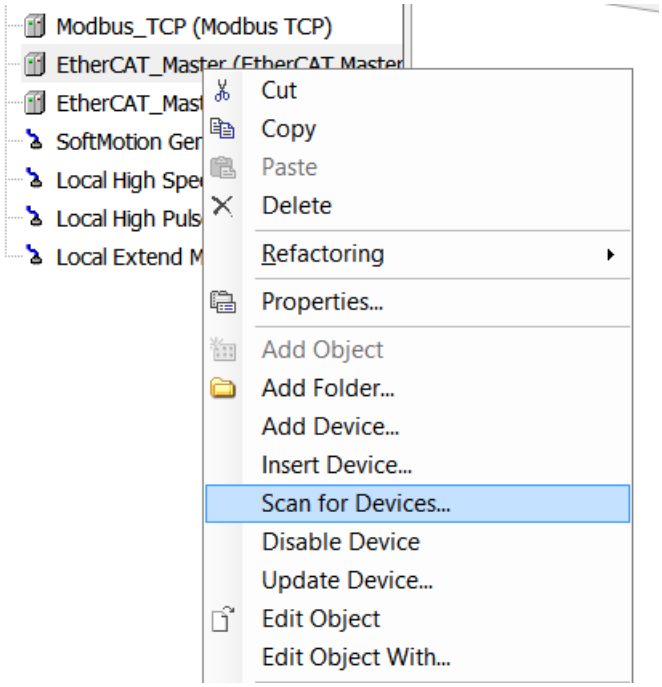
③ Add EtherCAT Master or EtherCAT Master SoftMotion.



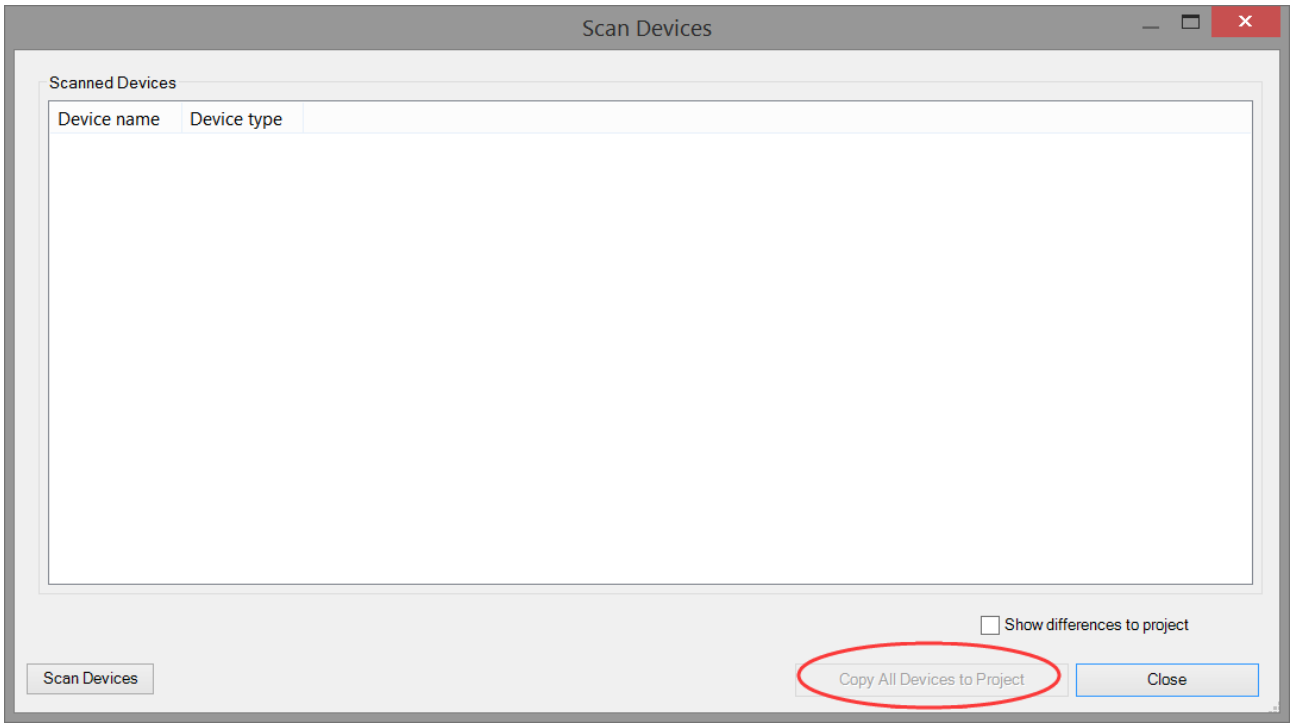
④ Select the network port for communication.



⑤ Scan the devices to add LC3-AP module.



⑥ Copy all devices to project.

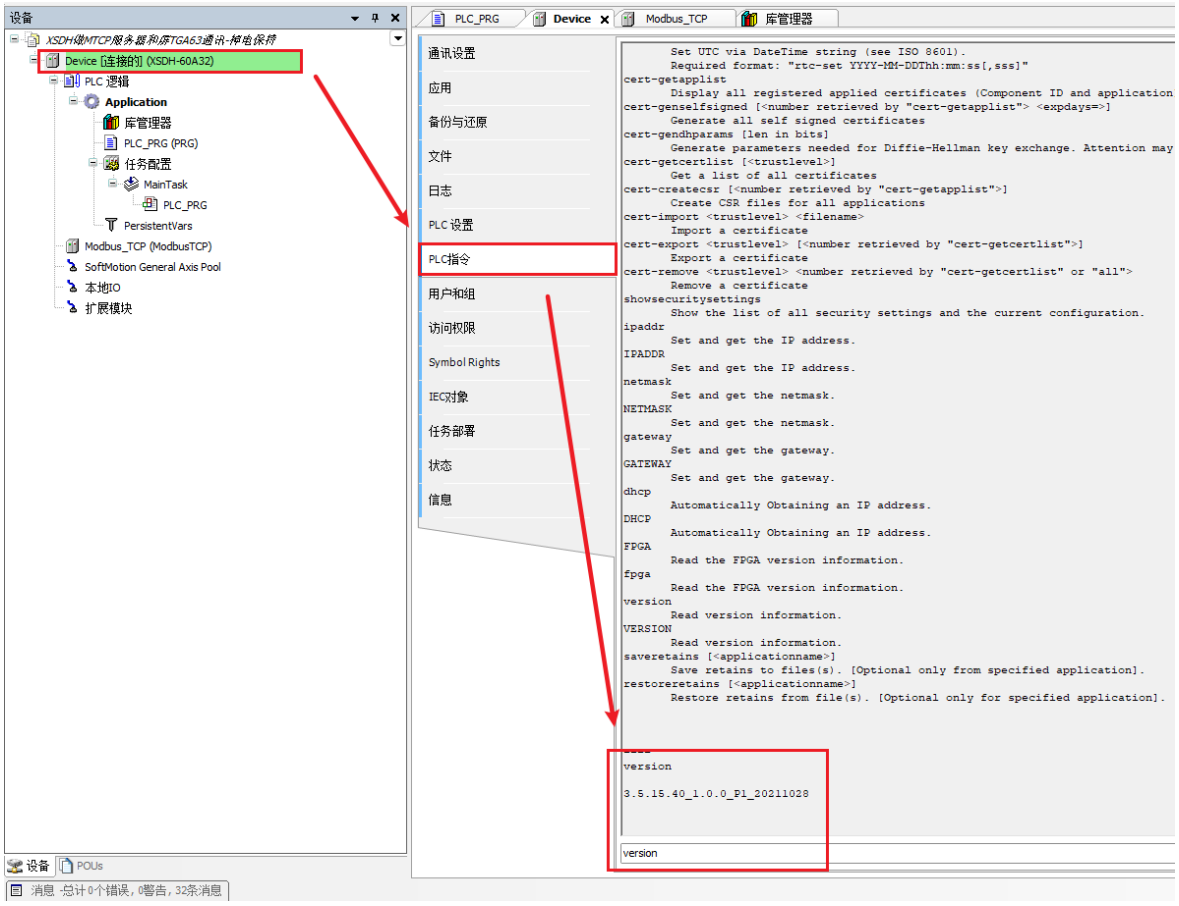


7-5. M_TCP

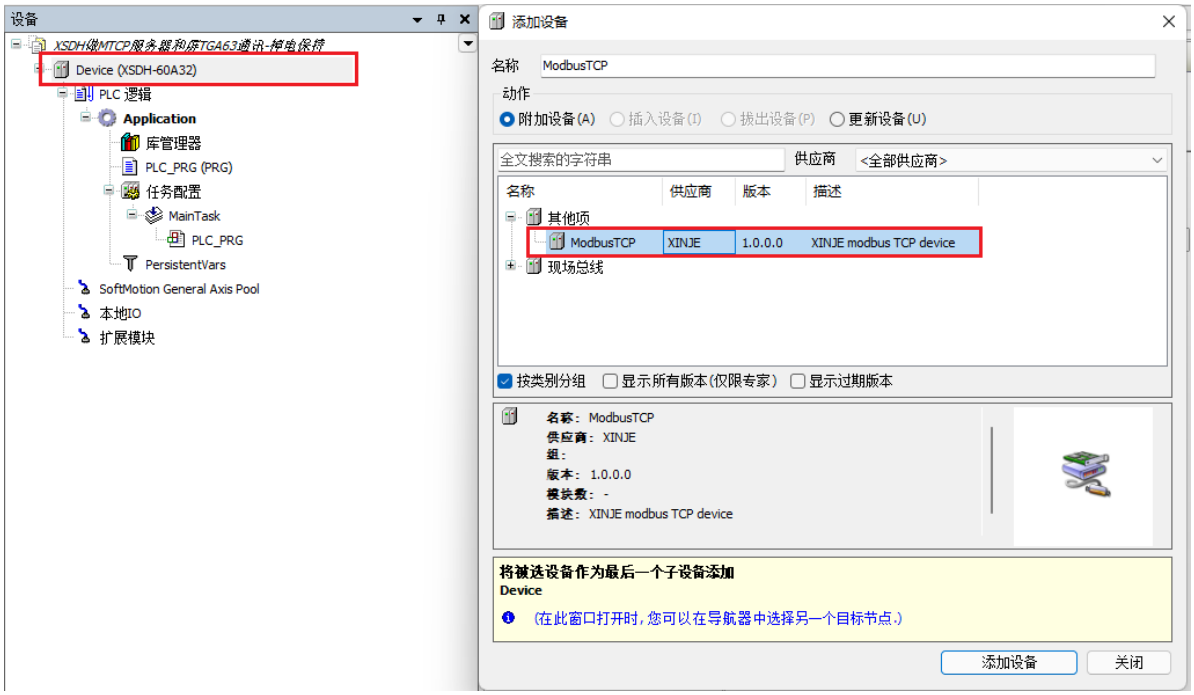
Note: the Modbus TCP developed by Xinje only supports ARM series models for the time being, and will support all Codesys models in the future.

7-5-1. Upper computer settings

① When using this communication function, please check whether the firmware version of the PLC is 3.5.15.40_ 1.0.0_ P1_ 20211028 and above, if it is not this version, please upgrade the firmware first. Refer to chapter 7-2 for details.



② Right click Device→other item→add Modbus TCP.

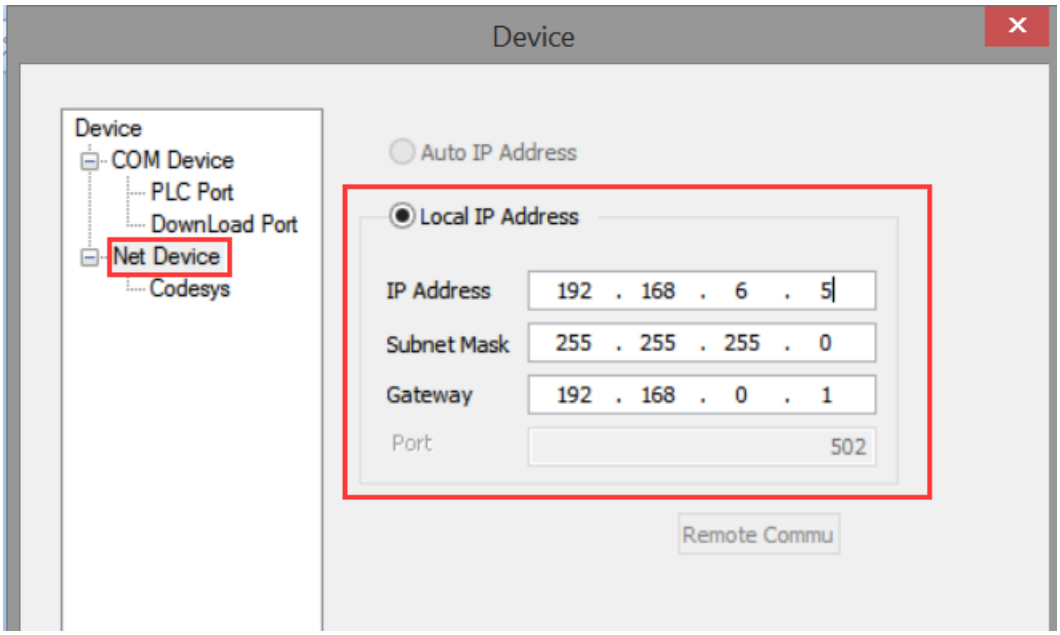


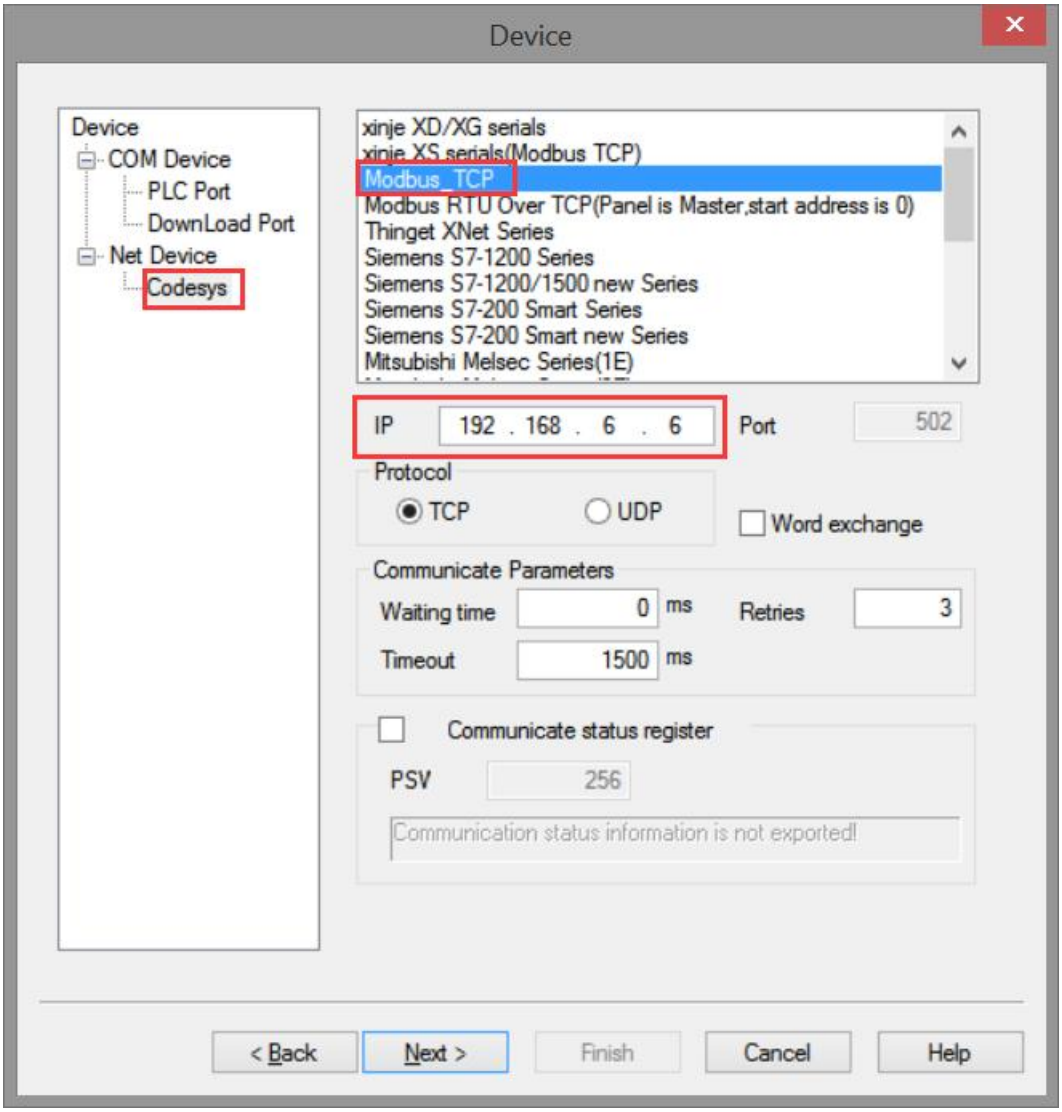
③ Associated variables.

```
PROGRAM PLC

VAR

    M AT %MB0: ARRAY[0..9] OF BOOL;   //0X and 1X on the HMI are same, MB0 is equal to 0X0 and 1X0 on the HMI

    D AT %MW40000:ARRAY[0..9] OF WORD;   //single register, 3X and 4X on the HMI are same, MW40000 is equal to 3X0 and 4X0 on the HMI

    DD AT %MW40010:ARRAY[0..9] OF REAL;   // double registers, 3X and 4X on the HMI are same, 3X10 and 4X10 will occupy MW40010+MW40011
END_VAR
```

7-5-2. HMI settings

① Set the IP address of the HMI and the device to be connected (new Ethernet device is required).



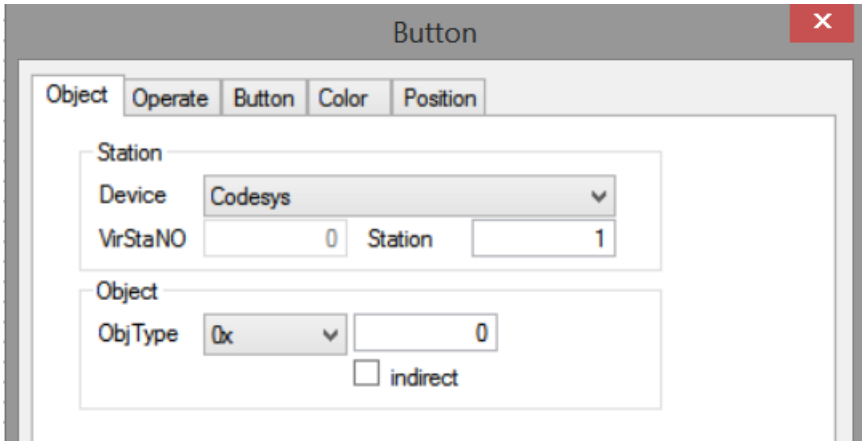


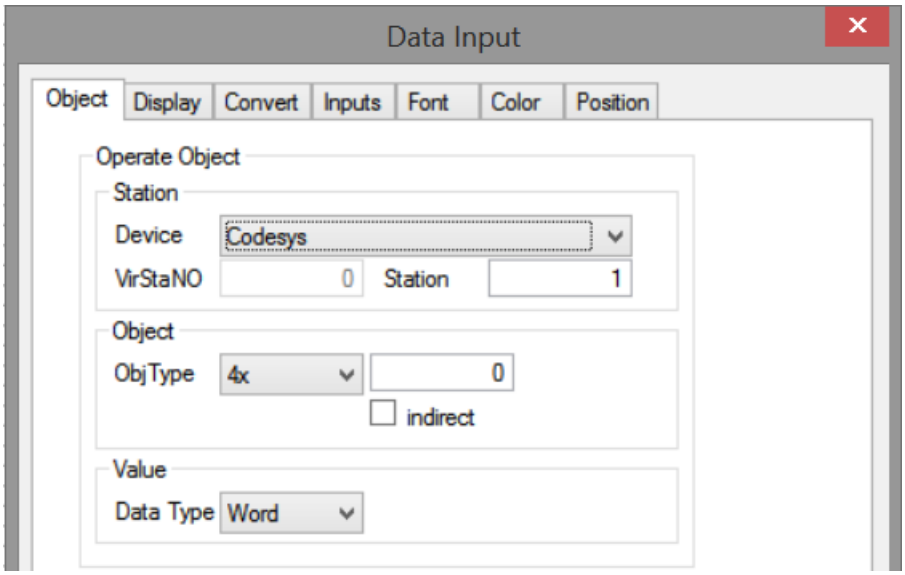
② Add the required HMI elements, select Codesys for the device, and the station number must be set to 0!

The object type of the button or indicator is 0x (readable and writable) or 1x (read-only). 0x0 and 1x0 correspond to MB0, and so on.

Select 3x (read-only) or 4x (read-write) as the object type of data input or data display. 3x0 and 4x0 correspond to MW40000, and so on.

If the data type input or display is DWORD, then 3x0 and 4x0 occupy MW40000 and MW40001 registers, and so on.





③ For the time being, the HMI can only support reading and writing of up to two registers, so floating point numbers (two registers) can be displayed at most. Double precision floating point numbers can only be converted in the PLC, and cannot be directly input or displayed on the HMI.

7-6. Dial code

XSDH-60A32-E supports the dial code function, and its specific functions are as follows:
00: Normal startup, no special processing, load user program.
10: Initialize IP.
01: Power on without loading user program.
11: Update the machine, send the SD card data to EMMC.

WUXI XINJE ELECTRIC CO., LTD.

No.816, Jianzhu West Road, Binhu District,

Wuxi City, Jiangsu Province, China

214072

Tel: 400-885-0136

Fax: (510) 85111290

www.xinje.com