Программируемый логический контроллер

PROMPOWER

серии PMP3xx

Руководство по эксплуатации

(Программное)



PROM POWER

Введение

Общее описание

В данном руководстве представлены аппаратные характеристики программируемых контроллеров серии PMP3xx. Перед использованием изделия внимательно прочтите данное руководство и выполняйте подключение, исходя из полного понимания его содержания. Пожалуйста, передайте данное руководство конечному пользователю.

Информация для пользователей

К работе с оборудованием допускаются только квалифицированные сотрудники. Если возникнут какие-либо проблемы, пожалуйста, свяжитесь с нашим техническим отделом. Приведенные примеры используются для того, чтобы помочь пользователям лучше понять процесс работы с оборудованием. При подключении ПЛК к другому оборудованию, убедитесь, что технические характеристики и принципы работы ПЛК соответствуют требованиям. При использовании ПЛК соблюдайте меры безопасности. Ошибки оператора могут привести к повреждению оборудования.

Исключение ответственности

Содержание руководства было тщательно проверено, однако потенциальные ошибки при работе не могут быть исключены. В связи с этим мы не гарантирует полного соответствия.

Мы регулярно проверяем руководство и исправляем возникающие проблемы в последующих версиях. Мы всегда рады обратной связи.

Авторское право PROMPOWER

Не копируйте руководство или информацию из него без письменного разрешения PROMPOWER. Нарушители понесут ответственность за убытки. Пожалуйста, сохраняйте все авторские права нашей компании, включая практические модули и разработанные патенты.

Содержание

1 Обзор и установка Codesys 5
1.1 Обзор Codesys 5
1.2 Установка и удаление Codesys 8
1.3 Помощь Codesys 9
2 Архитектура Codesys 10
2.1 Модель программного обеспечения 10
2.2 Устройство 13
2.3 Приложение 17
2.4 POU 32
2.5 Объект приложения 43
3 Основные инструкции 46
3.1 Битовые логические инструкции 46
3.2 Инструкции к таймеру 47
3.3 Инструкции для счетчиков 48
3.4 Инструкции по обработке данных 49
3.5 Инструкция по эксплуатации 51
4 Специальные функции 54
4.1 Высокоскоростной счёт 54
4.2 Внешнее прерывание 60
4.3 PLC SHELL 61
4.4 Часы 67
5 Примеры проектов Codesys 69
5.1 Основные операции программирования 69

5.2 Отображение ввода/вывода 73
5.3 Конфигурация задачи 74
5.4 Загрузка/чтение программ 80
5.5 Отладка программы 84
5.6 Моделирование 87
5.7 Функция сценария ПЛК 88
6 Технология промышленной полевой шины 89
6.1 Связь по протоколу MODBUS 89
6.2 MODBUS TCP 95
6.3 OPC UA 100
6.4 Свободный формат 102
6.5 TCP/IP 104
7 Общие проблемы и решения 107
7.1 Таргет-файлы (.package) 107
7.2 Модуль локального расширения серии PMP3xx 108
7.3 Для обращения ко входам и выходам ПЛК PMP301 110

1 Обзор и установка Codesys

1.1 Обзор Codesys

Для контроллера серии PROMPOWER PMP3xx выбрана платформа Codesys компании 3S.

Codesys — это современная и мощная среда разработки программного обеспечения для промышленных контроллеров, основанная на стандарте IEC 61131-3. Она предлагает широкий спектр возможностей для создания сложных автоматизированных систем управления (АСУ), включая логические, функциональные и структурированные текстовые схемы.

Основные языки программирования Codesys

Структурированный текст (ST): Это язык программирования, который позволяет писать код в виде последовательности инструкций, аналогично традиционным языкам программирования, таким как C или Pascal.

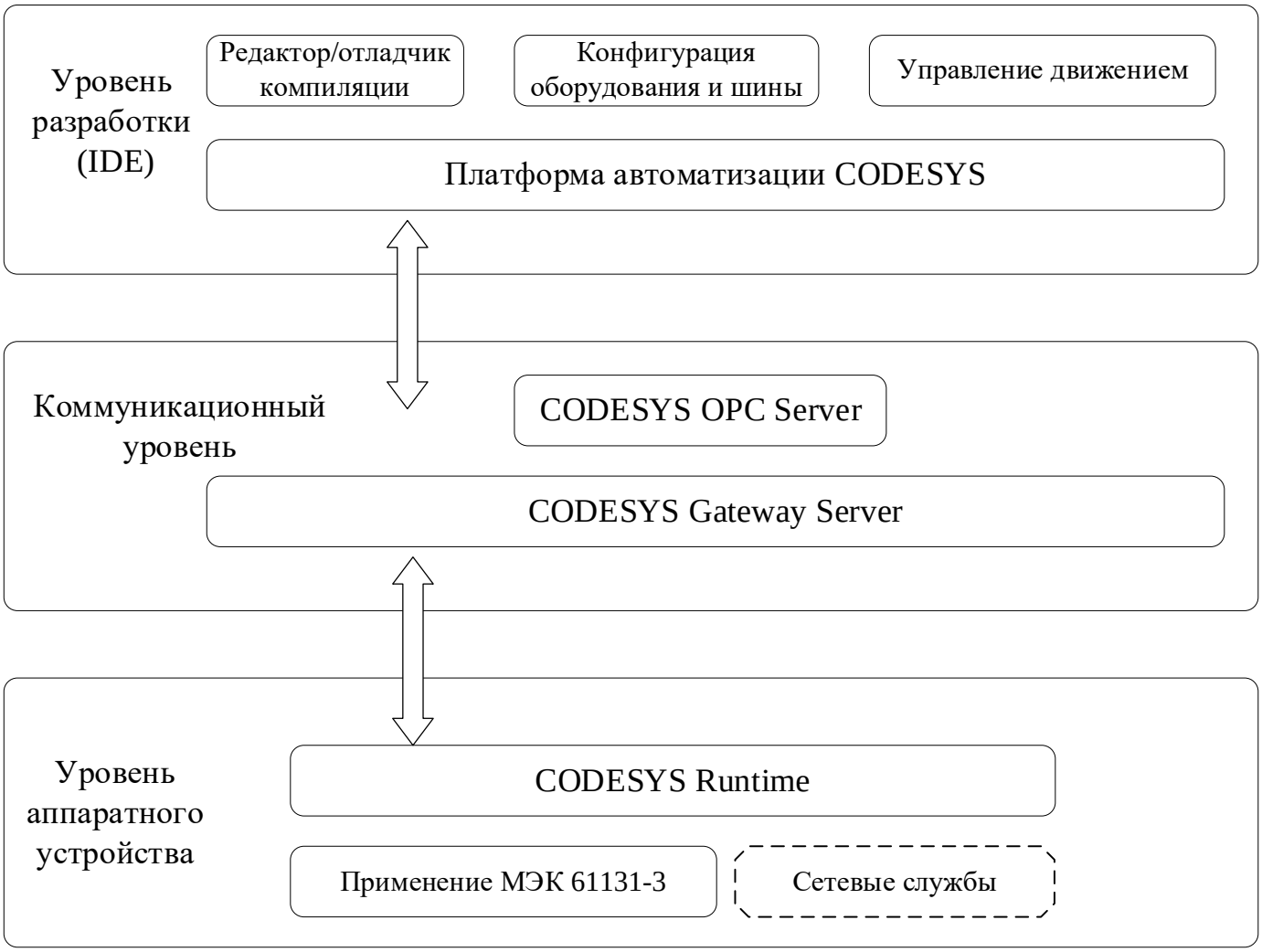
Логический выражение (LAD): Предоставляет графический интерфейс для написания программ с использованием логических элементов, таких как ИЛИ, И, НЕ, операции сравнения и т.д., представленные в виде блоков.

Функциональный блок (FBD): Позволяет использовать функциональные блоки для описания логики системы, используя графическое представление потока данных между различными элементами.

Инструкции по шагам (SCL): Язык высокого уровня, основанный на C, который предоставляет более гибкие возможности для написания сложной логики и алгоритмов.

График блока (LD): Графический язык, позволяющий визуализировать процесс управления с помощью диаграмм состояний и переходов. Архитектура программного обеспечения Codesys

Программное обеспечение Codesys обладает мощными функциями, высокой надежностью и открытостью. Оно объединяет программирование ПЛК, ядро контроллера реального времени, полевую шину и управление движением. Это комплексное программное обеспечение для автоматизации. С точки зрения архитектуры программное обеспечение Codesys можно разделить на три уровня: уровень разработки приложений, коммуникационный уровень и уровень устройств, как показано на рисунке:



1.1.1 Уровень разработки

- **Редактор**

Codesys предоставляет шесть языков программирования, определенных стандартом IEC61131-3: функциональная блок-схема (FBD), лестничная диаграмма (LD), список инструкций (IL), структурированный текст (ST), диаграмма последовательных функций (SFC) и диаграмма непрерывных функций (CFC).

- **Компилятор**

Отвечает за преобразование управляющей программы Codesys в машинный код и оптимизацию работы контроллера. При вводе неверного кода пользователи немедленно получают предупреждения о синтаксических ошибках и сообщения об ошибках от компилятора, для внесения соответствующих исправлений.

- **Аппаратное обеспечение и конфигурация шины серии PMP3xx**

Для PMP3xx установка соответствующих параметров производится в Codesys.

- **Модуль управления движением**

Функция управления движением была интегрирована в Codesys для программного пакета Softmotion (ЧПУ). Инструментарий на базе PLCopen позволяет реализовать одноосевое и многоосевое движение, электронную кулачковую передачу, электронную зубчатую передачу, сложное многоосевое управление ЧПУ и т.д.

1.1.2 Коммуникационный уровень

Связь между уровнем разработки и уровнем аппаратных устройств осуществляется с помощью серверной части, в которой установлен OPC-сервер.

- **Серверная часть Codesys Gateway Server**

Функционирует между уровнем разработки и уровнем аппаратных устройств. Может использовать протокол TCP/IP или другие шины для реализации удаленного доступа.

- **OPC-сервер Codesys**

Благодаря OPC-серверу реализована функция мультиклиента, предусмотренная спецификацией OPC v2.0.

1.1.3 Уровень аппаратного устройства

ПЛК серии PMP3xx является аппаратным уровнем системы, на котором установлена система исполнения Codesys, которая удовлетворяет требованиям к реагированию и точному управления в реальном времени. В то же время функциональное расширение может быть реализовано за счет использования дополнительных компонентов Codesys.

1.2 Установка и удаление Codesys

1.2.1 Системные требования

Требования к аппаратному обеспечению:

- Windows 8 или Windows 10 64-разрядная ОС
- Оперативная память 4 ГБ и выше
- Объем жесткого диска более 12 ГБ

1.2.2 Где скачать Codesys?

Официальный магазин Codesys: https://store.codesys.com.
На сайте PROMPOWER: https://prompower.ru/docs/plc/pmp30/soft/CODESYS-64-3.5.16.40.zip

1.2.3 Установка Codesys

1) Основные требования к аппаратному и программному обеспечению

Поскольку программное обеспечение Codesys v3.5 имеет относительно большой размер и содержит много обрабатываемой информации, Codesys предъявляет определенные требования к обеспечению ПК. Необходимая минимальная конфигурация и рекомендуемая конфигурация приведены в следующей таблице:

Наименование	Минимальная конфигурация	Рекомендуемая конфигурация
ОС	Windows 2000 (Windows XP / Windows Vista / Windows 7)	Windows 10
Оперативная память	4 ГБ	4 ГБ
Жесткий диск	12 ГБ	12 ГБ
ЦПУ	Pentium V, Centrino > 1,8 ГГц, Pentium M > 1,0 ГГц	Pentium V, Centrino > 3,0 ГГц, Pentium M > 1,5 ГГц

2) Установка

Запустите Codesys 64 3.5.16.40.exe от имени администратора, чтобы начать установку, и следуйте рекомендациям помощника по установке.

Примечание: Не рекомендуется устанавливать программное обеспечение на диск С.

1.2.4 Управление версиями Codesys

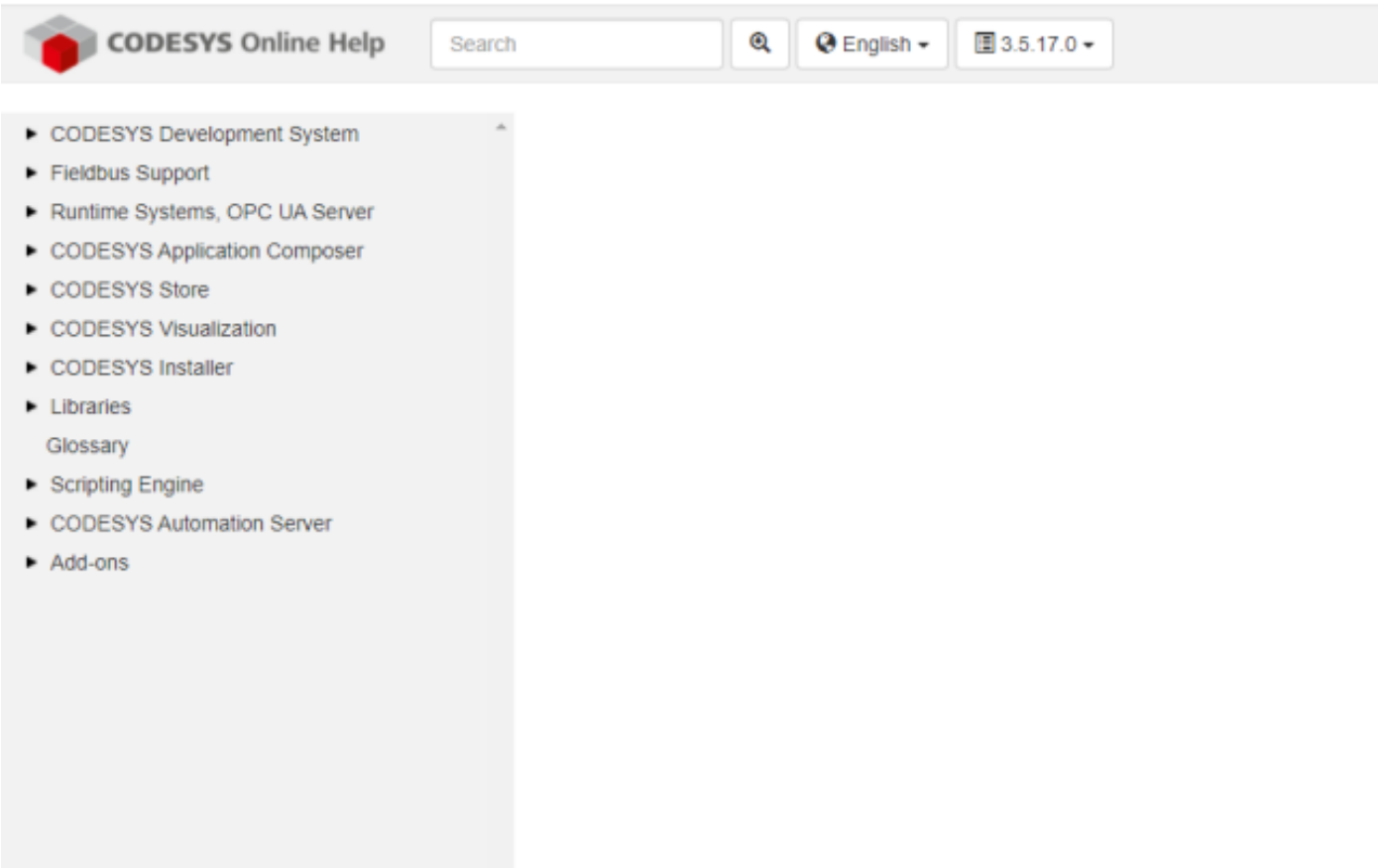
Codesys поддерживает одновременную установку нескольких версий. Компилятор также поддерживает установку нескольких версий. Рекомендуется использовать версию 3.5.16.40. Использование других версий может привести к некорректному поведению некоторых функций.

1.2.5 Деинсталляция Codesys

Программное обеспечение Codesys можно удалить через панель управления windows. Откройте панель управления > Добавить/удалить программы, выберите Codesys, нажмите кнопку удаления и завершите деинсталляцию в соответствии с подсказкой.

1.3 Помощь Codesys

После открытия приложения Codesys пользователи могут найти меню помощи и нажать "содержание", чтобы открыть онлайн-справку. Пользователи могут быстро найти нужный контент по индексу или ключевым словам поиска, как показано на рисунке:

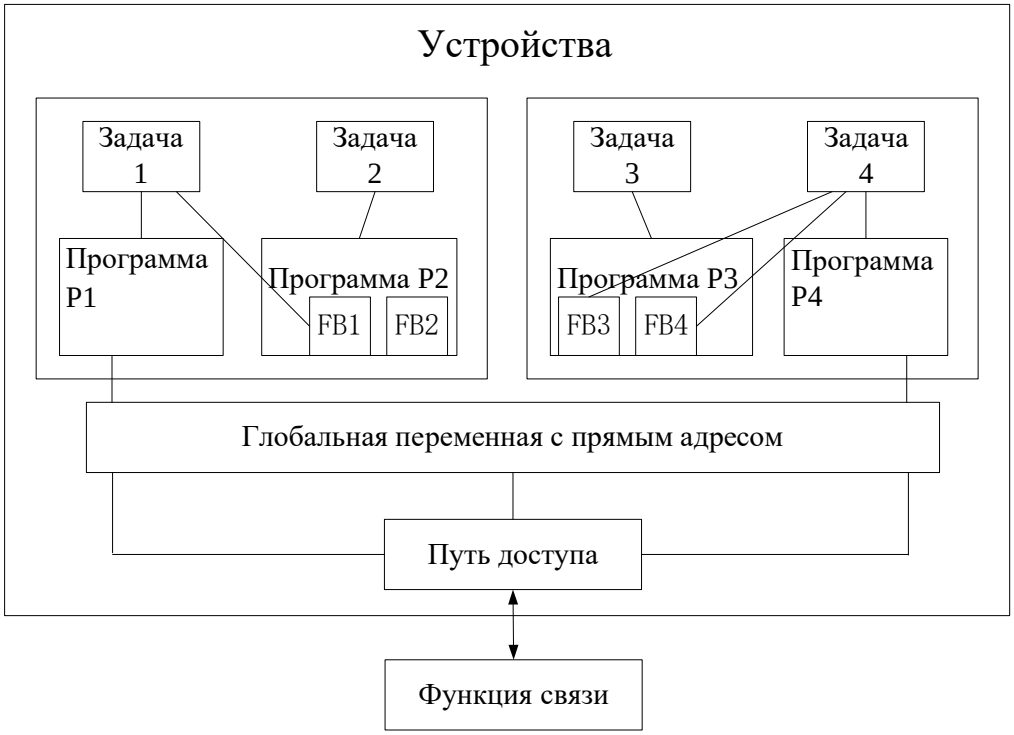


2 Архитектура Codesys

2.1 Модель программного обеспечения

2.1.1 Программная модель

Программная модель Codesys описывает основные элементы программного обеспечения и их взаимосвязи, которые выражены в иерархической структуре. Каждый слой содержит множество характеристик нижележащих слоев, а его внутренняя структура показана на следующем рисунке. К программным элементам относятся: оборудование, приложение, задача, глобальная переменная, путь доступа и объект приложения. Они являются программной основой современных программных компонентов ПЛК. Модель программного обеспечения соответствует стандарту IEC 61131-3.



Программная модель описывает, как программируемый контроллер может выполнять несколько независимых программ одновременно, и как отслеживать выполнение программы.

- **Устройства**

В среде разработки CODESYS, устройства могут быть как физическими, так и виртуальными. Устройства могут быть интегрированы в систему для выполнения различных функций. Среди этих устройств можно выделить промышленные контроллеры (ПЛК), датчики, приводы, серверы и т.д.

- **Приложение**

Объект "Application" (Приложение) представлен в виде узла в дереве устройств. Он содержит объекты, необходимые для исполняемой программы управления.

В контексте среды разработки CODESYS, слово "приложение" обычно означает программное обеспечение, предназначенное для запуска на ПЛК. Такое приложение может быть составлено из различных инструкций, написанных на языках программирования: Structured Text (ST), Ladder Diagram (LAD), Function Block Diagram (FBD), Sequential Function Chart (SFC) и Instruction List (IL). В рамках CODESYS, приложение может содержать управляющую логику, обработку данных и функции взаимодействия с внешними устройствами и системами.

В каждом устройстве имеется одно или несколько "приложений", которые располагаются на втором уровне программной модели. "Приложение" не только обеспечивает систему поддержки для выполнения программ, но и отражает физическую структуру ПЛК и обеспечивает интерфейс между программами и физическими каналами ввода/вывода ПЛК.

Приложение размещается в центральном процессоре ПЛК, поэтому приложение можно понимать как микропроцессорный блок ПЛК. Глобальные переменные, определенные в приложении, действительны в пределах этого приложения. Основными членами приложения являются глобальные переменные, задачи и программные организационные единицы (POU).

- **Путь доступа**

Основная функция пути доступа - связывать глобальные переменные, переменные представления и входные/выходные переменные блока POU. Доступ к переменным в каждом приложении может осуществляться через другие удаленные конфигурации.

- **Функция связи**

Обычно используются методы связи, соответствующие международным стандартам (например, RS232, RS485), или промышленные шины, такие как EtherCAT, MODBUS, Ethernet/IP и т. д.

2.1.2 Характеристики модели программного обеспечения

Codesys имеет следующие особенности:

- Codesys позволяет загружать, запускать и выполнять несколько независимых программ в одном ПЛК одновременно.
- Codesys позволяет реализовать полный контроль над выполнением программы. Благодаря механизму задач, ПЛК может полностью контролировать выполнение программы. Механизм задач позволяет различным частям программы выполняться параллельно и в разное время с разной скоростью, что значительно расширяет возможности ПЛК.
- Codesys - это международная стандартная модель программного обеспечения, которая может адаптироваться к различным ПЛК. Codesys подходит как для небольших систем с ПЛК, так и для больших распределенных систем управления.
- Codesys поддерживает возможность повторного использования POU.
- Codesys поддерживает иерархическое проектирование: сложная программа может быть декомпозирована на POU слой за слоем.

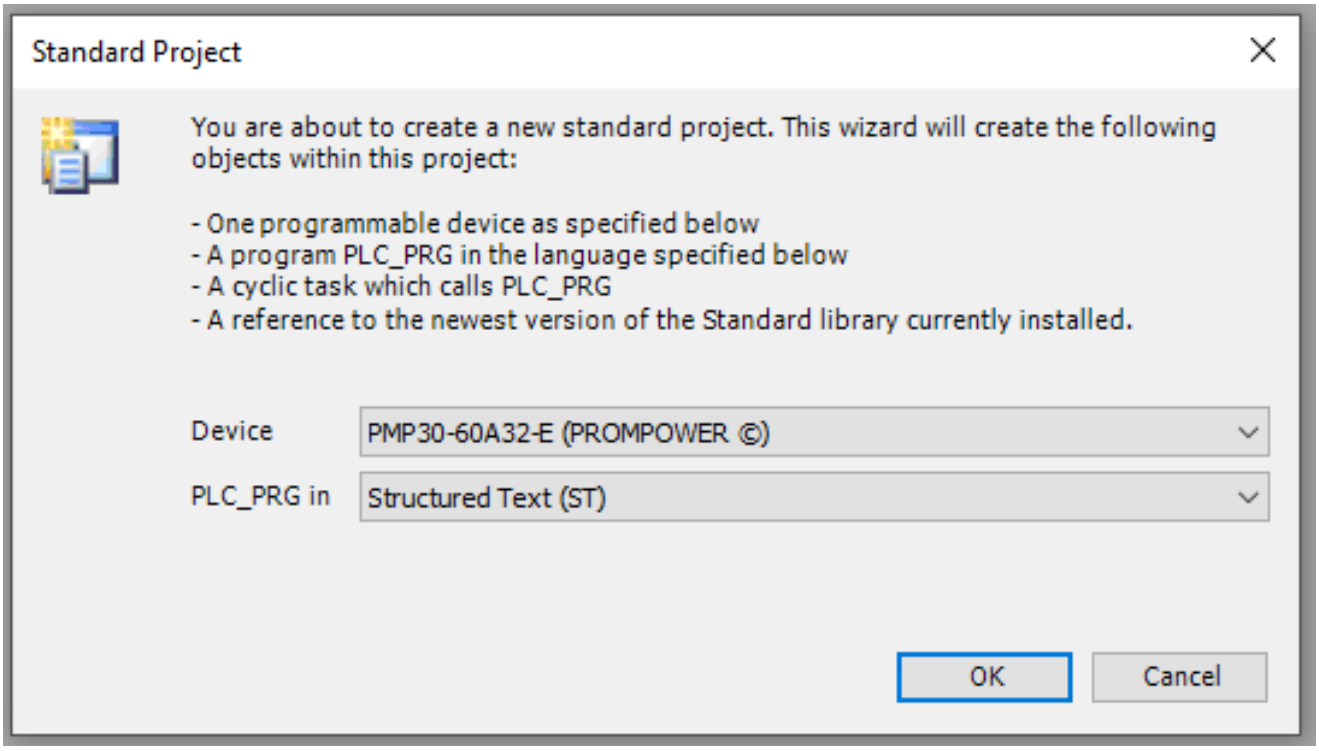
2.2 Устройство

В среде разработки Codesys, устройства — это физические или виртуальные элементы, которые могут быть интегрированы в систему управления. К таким устройствам относятся промышленные контроллеры (ПЛК), датчики, приводы, серверы и прочее оборудование, совместимое с Codesys. В Codesys устройства могут быть как физическими, так и виртуальными, и их можно настроить и управлять ими через среду разработки.

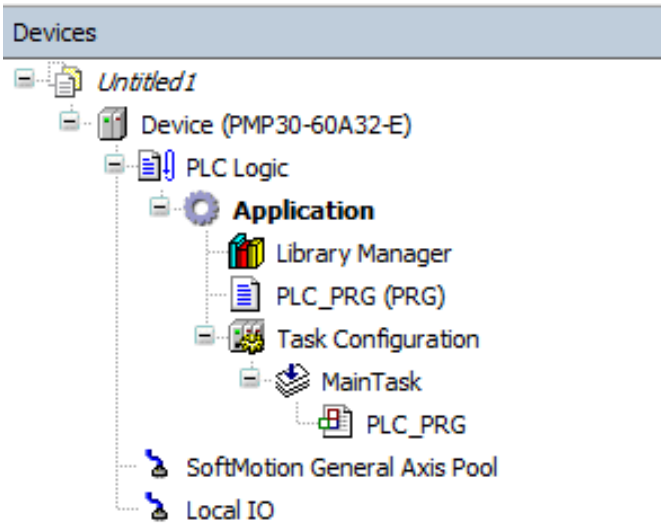
2.2.1 Управление устройствами

1) Подключение устройства

При создании нового проекта автоматически появляется диалоговое окно, как показано далее. В опции шаблона можно выбрать создание пустого проекта или стандартного проекта. При выборе стандартного проекта необходимо выбрать подключенное устройство.



Нажмите OK, чтобы получить дерево устройств.

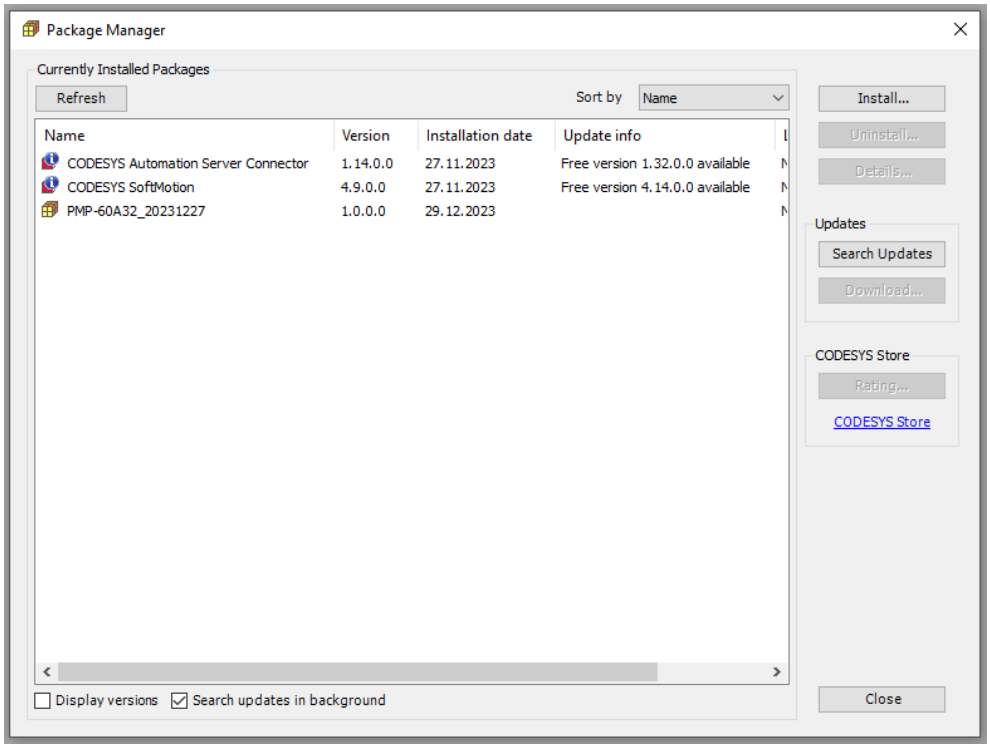


2) Менеджер пакетов

Все "устройства" должны быть заранее установлены в "менеджере пакетов". Менеджер пакетов (Package Manager) можно выбрать в меню "инструменты", а пользователи могут добавлять или удалять пакеты.

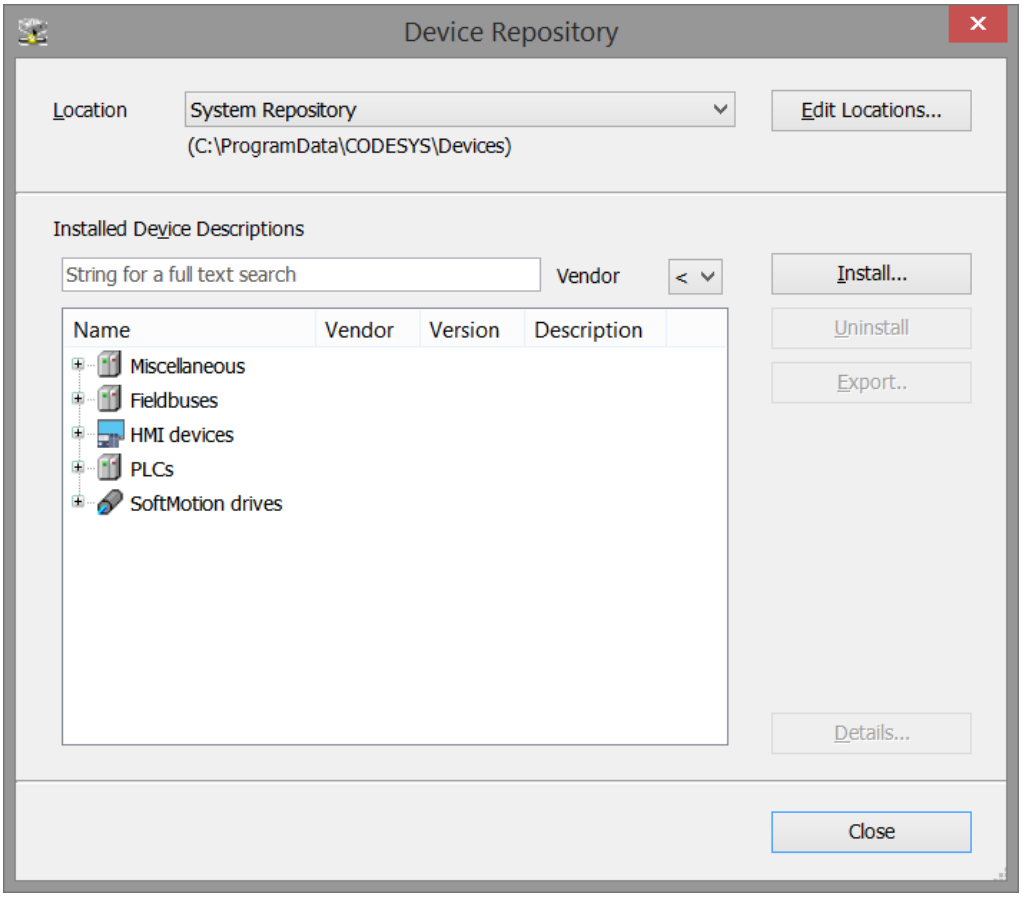
Процесс установки менеджера пакетов для этого продукта выглядит следующим образом:

Откройте "Инструменты" и выберите "Менеджер пакетов".
Нажмите кнопку "Установить" и найдите в каталоге соответствующий установочный пакет. В данном примере используется пакет PMP-60A32.package.
Нажмите OK, установка пройдет успешно, и в менеджере пакетов появится значок "PMP-60A32", как показано на следующем рисунке:



3) Управление репозиторием устройств

Репозиторий устройств (Device repository) необходим для добавления информации об аппаратной части устройств. Репозиторий устройств — это база данных устройств. Все данные после установки импортируются в локальную систему Codesys. Диалоговое окно репозитория устройств показано на следующем рисунке:



После импорта файлов в окне, необходимые данные будут записаны в локальной системе для вызова в проекте.

Файлы описания устройств, которые можно добавить, включают файлы описания устройств и модулей расширения, официально предоставляемых Prompower, EtherCAT XML-файлы, EDS-файлы и и т.д.

2.2.2 Редактор Device

Редактор Device — это диалоговое окно для конфигурирования устройств. Открывается при выборе значка устройства Device, щелчком правой кнопки мыши по команде "Редактировать объект".

Ниже перечислены вкладки, содержащие следующие поддиалоговые окна, как показано в следующей таблице:

Установки соединения	Конфигурация, связанная с соединением между целевым устройством и другими программируемыми устройствами (ПЛК)
Приложения	Отображение конфигурации параметров устройства соответственно
Резервное копирование и восстановление	Резервное копирование файлов ПЛК, специфичных для конкретного приложения
Файлы	Конфигурация передачи файлов между хостом и ПЛК
Журнал	Отображение журнала событий ПЛК
Установки ПЛК	Применение, связанное с работой ввода/вывода, состояние ввода/вывода в состоянии останова, конфигурация опций цикла шины
Оболочка ПЛК	Управление оболочкой ОС ПЛК
Пользователи и группы	Управление пользователями, связанное с доступом к оборудованию во время эксплуатации (не путать с управлением пользователями инженерных систем)
Права доступа	Настройка прав доступа к запущенным объектам и файлам
Символьные права (IEC)	Доступ к "объектам" устройства через приложения IEC
Common.PCI Конфигурация (Clock I/O mapping)	Часы реального времени
Размещение задачи	Отображает таблицы входов и выходов и их назначение определенным задачам
Состояние	Подробная информация о состоянии и диагностике оборудования
Информация	Основная информация об оборудовании (название, поставщик, версия, серийный номер и т.д.)

2.3 Приложение

Приложение — это набор объектов, необходимых для выполнения программы на аппаратном устройстве (например, ПЛК). Эти объекты не зависят от платформы устройства, пользователи могут управлять ими в POU. Инстанция происходит в окне устройства, назначается конкретным устройствам. Такой подход к организации программы соответствует идее объектно-ориентированного программирования (ООП).

Объекты приложений включают задачи, организационные единицы программы POU, конфигурации задач, глобальные переменные, менеджеры библиотек и трассировки (Trace). Объекты ресурсов в Codesys v3.x могут управляться только в дереве устройств. После добавления объектов в дерево устройств их необходимо сопоставить с управляемым устройством в соответствии с определенными "правилами". Диапазон объектов (таких как библиотеки и списки глобальных переменных) в проекте зависит от иерархических отношений между приложениями и объектами устройств в дереве проекта. Как правило, объект в приложении также действителен для его "подпрограмм" и может быть использован.

2.3.1 Задание

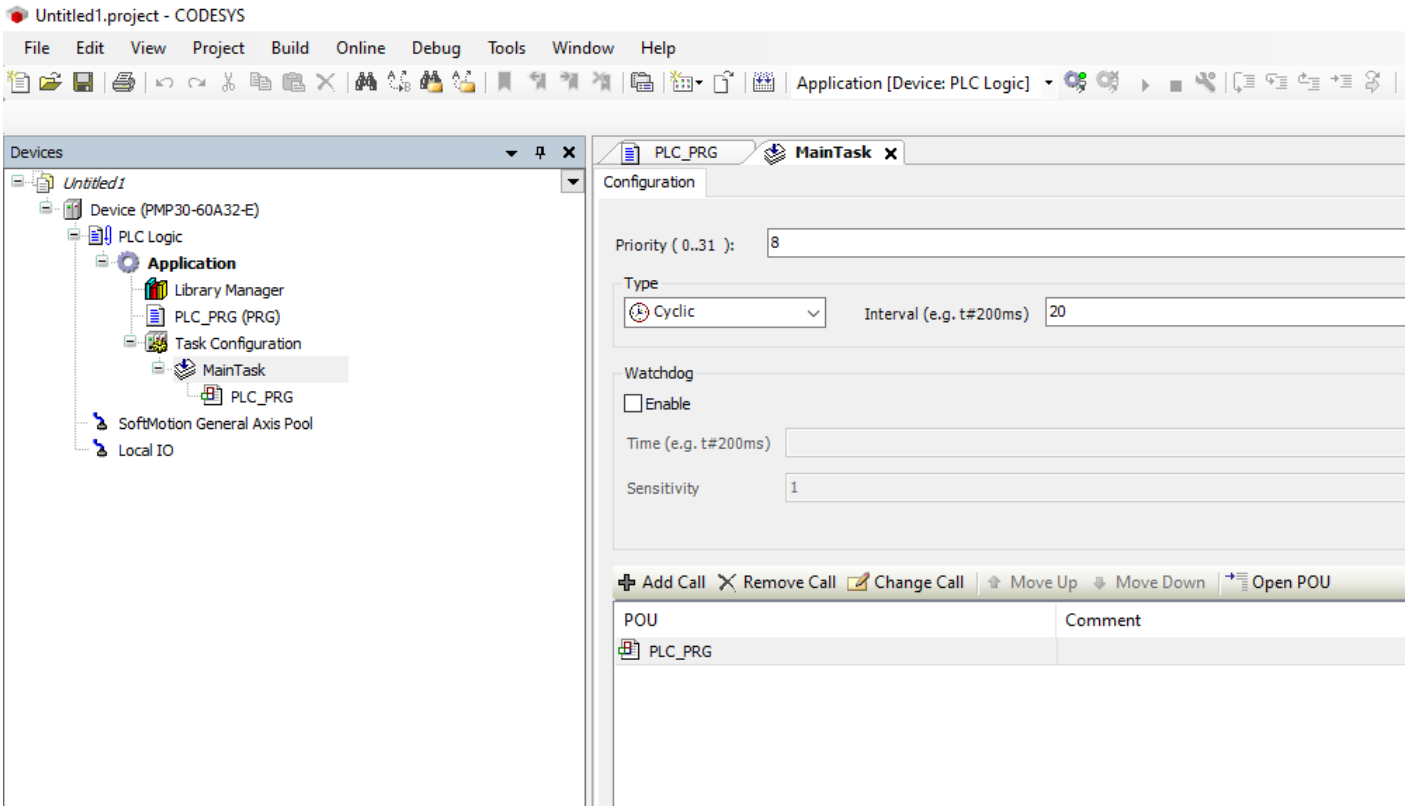
1) Обзор

Управляющая программа может быть написана на различных языках программирования. Типичная программа состоит из множества взаимосвязанных функциональных блоков, которые могут обмениваться данными друг с другом. Выполнение различных частей программы контролируется "задачами". После настройки "задачи" серия программ или функциональных блоков может выполняться периодически или запускаться по определенному событию.

В дереве устройств есть вкладка диспетчера задач, которая может использоваться не только для объявления конкретных PLC_PRG, но и для управления выполнением и обработкой других подпрограмм в проекте. Задача используется для задания атрибутов POU (Это элемент управления выполнением с возможностью вызова). В конфигурации задачи может быть установлено несколько задач, а в задаче может быть вызвано несколько POU. Как только задача установлена, она может управлять выполнением программного цикла или запускать выполнение по триггеру определенных событий.

В конфигурации задачи она определяется именем, приоритетом и типом запуска задачи. Тип запуска может быть задан по времени (периодический, случайный) или по времени внутреннего или внешнего триггера задачи, например,

по нарастающему фронту глобальной булевой переменной или по определенному событию в системе. Для каждой задачи можно задать серию программ, запускаемых этой задачей. Если задача выполняется в текущем цикле, эти программы будут обработаны в течение одного цикла. Сочетание приоритета и условия определяет время выполнения задачи. Интерфейс настройки задачи показан на следующем рисунке:

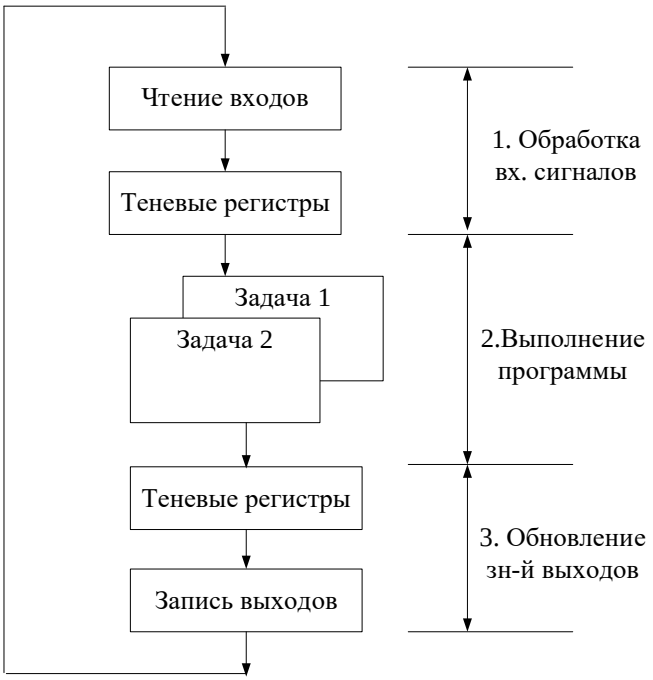


При настройке заданий пользователям следует придерживаться следующих правил:

(1) Максимальное количество циклических заданий - 100
(2) Максимальное количество свободно выполняемых задач - 100
(3) Максимальное количество задач, запускаемых событиями, составляет 100
(4) В соответствии с целевой системой ПЛК, PLC_ PRG может выполняться как свободная программа без вставки в конфигурацию задачи
(5) Программы обработки и вызова выполняются сверху вниз в редакторе задач.

2) Процесс выполнения программы ПЛК

На следующем рисунке подробно описан полный процесс выполнения программы в ПЛК, который состоит в основном из трех важных этапов: выборка входных данных, выполнение программы и обновление выходных данных.



(1) Обработка входных сигналов

В начале каждого цикла сканирования ПЛК определяет состояние устройств ввода (переключателей, кнопок и т. д.) и записывает его в область отображения входов. Обновление области отображения входов происходит только в начале сканирования. Во время сканирования, даже если состояние выхода изменится, состояние входа не изменится.

(2) Выполнение программы

На этапе выполнения программы цикла сканирования программная модель ПЛК считывает статус и данные из области входного и выходного отображения и выполняет логические и арифметические операции в соответствии с инструкциями. Результаты операций сохраняются в соответствующих блоках области выходного изображения. На этом этапе неизменным остается только содержимое регистра отображения входов, а содержимое других регистров отображения будет меняться по мере выполнения программы.

(3) Обновление выходов

Этап обновления выходных сигналов также называется этапом вывода. ПЛК передает состояние и данные области отображения выходов в точки выхода,

изолирует и усиливает мощность определенным образом, приводя в действие внешнюю нагрузку.

Помимо выполнения задач трех вышеперечисленных этапов в рамках цикла сканирования, ПЛК также выполняет вспомогательные задачи, такие как внутренняя диагностика, коммуникация, обработка информации и операции ввода/вывода.

В соответствии с режимом сканирования ПЛК, для того чтобы быстро реагировать на изменения входных и выходных данных и завершить задачу управления, время сканирования ПЛК относительно невелико, и время сканирования ПЛК обычно отсчитывается в мс.

ПЛК повторяет описанные выше процессы (1) - (3), и время каждого повторения составляет рабочий цикл (или цикл сканирования).

Из процесса работы ПЛК видно, что поскольку ПЛК использует циклический режим работы, входной сигнал обновляется только в начале каждого цикла, а выходной сигнал формируется в конце каждого цикла. Поэтому неизбежна задержка между выходным и входным сигналом.

От момента изменения сигнала на входе ПЛК до реакции выхода ПЛК на изменение входного сигнала проходит определенный промежуток времени. Время запаздывания - важный параметр, который необходимо понимать при проектировании системы управления ПЛК.

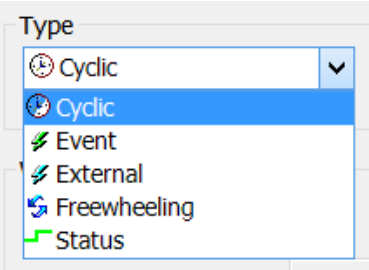
Время задержки зависит от следующих факторов:

- Время фильтрации входной цепи определяется постоянной времени RC-фильтра. Время задержки входного сигнала можно регулировать, изменяя соответствующую постоянную времени.
- Время задержки выходной цепи зависит от режима работы выходной цепи. Время запаздывания у релейных выходов обычно составляет около 10 мс, а время запаздывания у транзисторных выходов - менее 1 мс.
- Режим работы ПЛК - циклическое сканирование.

3) Тип вида выполнения задачи

В верхней части дерева конфигурации задач находится "Конфигурация задач". Ниже перечислены определенные на данный момент задачи, каждая из которых представлена именем задачи. POU, вызывающие конкретную задачу, не отображаются в дереве конфигурации задач.

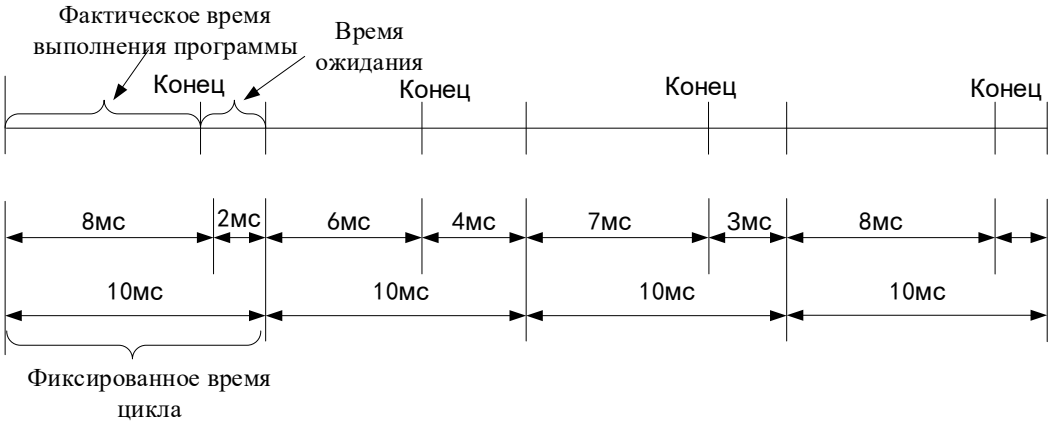
Для каждой независимой задачи можно редактировать и настраивать тип ее выполнения. К ним относятся циклический, событийный, внешний, свободный и статусный. Подробнее см. рисунок ниже.



(1) Циклическая задача (Cyclic)

В зависимости от того, выполняются или нет инструкции, используемые в программе, время обработки программы будет разным, поэтому фактическое время выполнения будет варьироваться в каждом цикле сканирования, и время выполнения будет варьироваться от длительного до короткого. При использовании режима фиксированного цикла программа может выполняться многократно в течение определенного времени цикла. Даже если время выполнения программы меняется, можно поддерживать определенный интервал обновления. Здесь также рекомендуется отдавать предпочтение режиму запуска задач с фиксированным циклом.

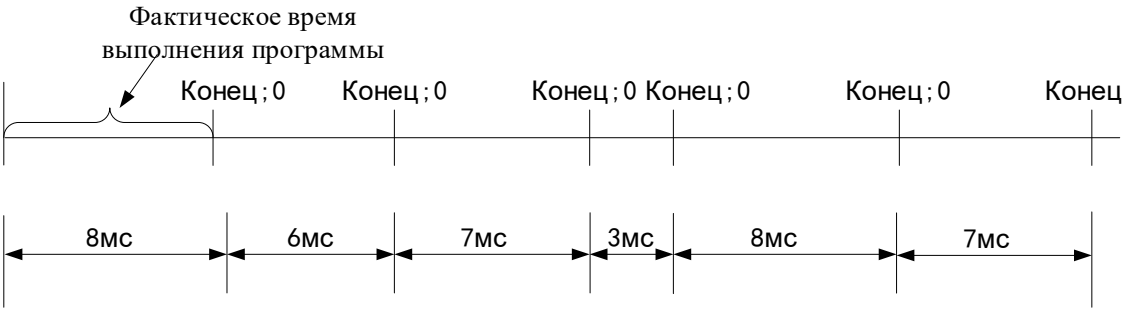
Например, предположим, что задача, соответствующая программе, установлена в режим фиксированного цикла, а время интервала установлено на 10 мс, диаграмма последовательности фактического выполнения программы показана на следующем рисунке:



Если фактическое время выполнения программы завершается в течение заданного фиксированного времени установки цикла, то оставшееся время используется как время ожидания. Если в приложении есть задачи с более низким приоритетом, которые не были выполнены, оставшееся время ожидания используется для выполнения задач с более низким приоритетом. Подробнее см. описание приоритета задач.

(2) Свободная задача (Freewheeling)

Задача будет обработана, как только программа начнет работать. После одного цикла работы задача будет автоматически перезапущена в следующем цикле, как показано на следующем рисунке. На выполнение свободной задачи не влияет цикл сканирования программы (время интервала). Это необходимо для того, чтобы каждый раз, когда выполняется последняя инструкция программы, происходил вход в следующий цикл. В противном случае цикл программы не завершится.



В этом методе выполнения нет фиксированного времени выполнения задачи, поэтому время выполнения каждый раз может быть разным. Производительность программы в реальном времени не может быть гарантирована, и этот метод редко используется в практических приложениях.

(3) Событийная задача (Event)

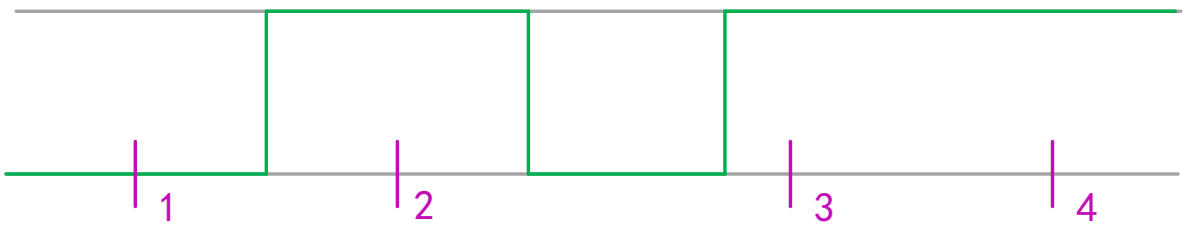
Если переменная в области событий получает нарастающий фронт, задача запускается.

(4) Статус (Status)

Если переменная в области событий равна true, задание запускается.

Метод триггера состояния аналогичен триггеру события, за исключением того, что программа будет выполняться до тех пор, пока триггерная переменная триггера состояния истинна, и не будет выполняться, если она ложна. Событийный триггер реагирует только на нарастающий фронт переменной триггера.

На следующем рисунке сравниваются триггер события и состояния. Зеленая сплошная линия — это булева переменная status, выбранная двумя методами срабатывания. В следующей таблице показаны результаты сравнения.



Основное различие между событием и состоянием заключается в том, что состояние задачи относится к её текущему положению в процессе выполнения, в то время как событие представляет собой мгновенное изменение в системе, требующее немедленного внимания или действия. Состояние задачи используется для планирования и управления временем выполнения задач, в то время как событие задачи позволяет реагировать на мгновенные изменения в системе.

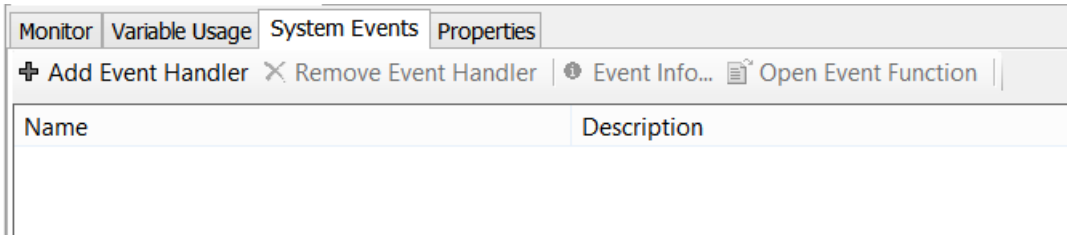
Задача, управляемая событием, требует, чтобы событие изменилось с ложного на истинное. Если частота выборки плана задач слишком мала, нарастающий фронт события может быть не обнаружен.

Вид задачи	1 точка	2	3	4
По событию	Не выполняется	Выполняется	Выполняется	Выполняется
По статусу	Не выполняется	Выполняется	Не выполняется	Не выполняется

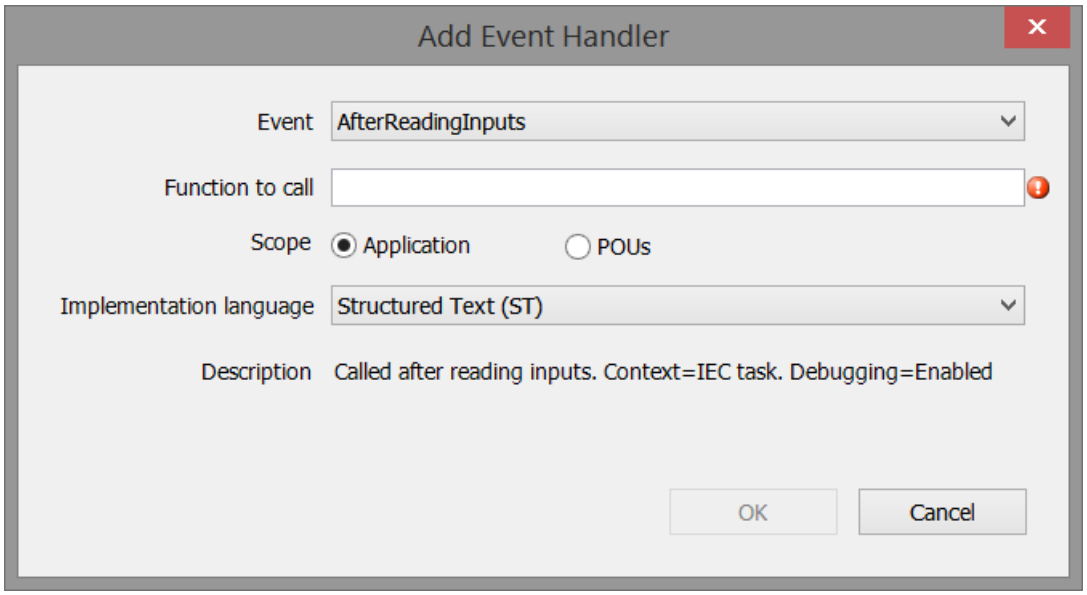
4) Системные события

Системные события, которые может выбрать пользователь, зависят от конкретного ПЛК. Соответствующий библиотечный файл ПЛК предоставляет соответствующие системные события. Поэтому системные события, соответствующие различным целевым ПЛК, могут быть разными. К общим системным событиям относятся: остановка, запуск, вход в систему, изменение и т. д. В конфигурации задачи можно задать системные события.

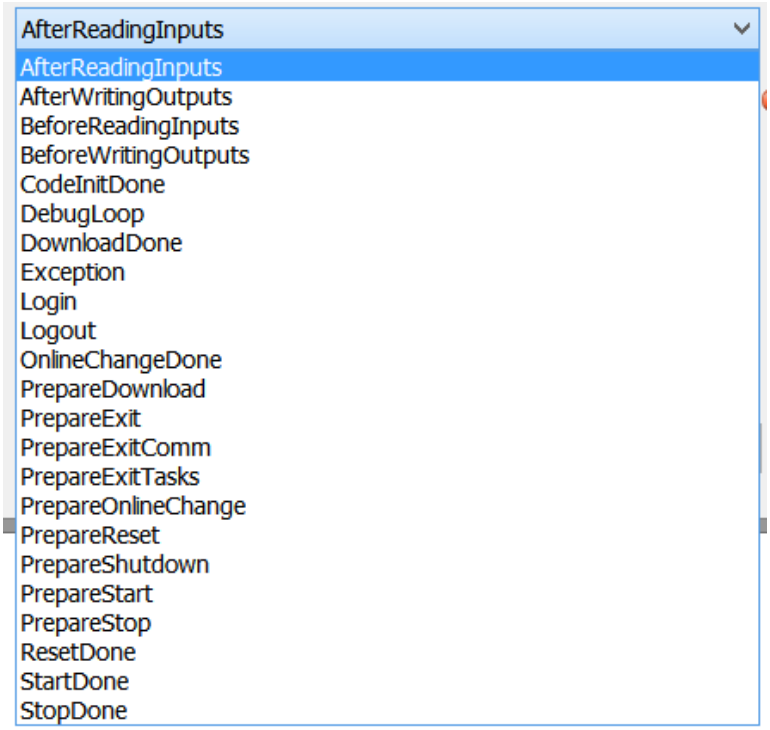
Пользователь может выбрать "конфигурация задачи" > "системное событие" с помощью мыши, чтобы войти в интерфейс, показанный на рисунке ниже.



Выберите кнопку "Добавить обработчик событий", чтобы добавить системные события. Открывшийся интерфейс показан на следующем рисунке.



Типы "событий", которые можно выбрать, показаны на следующем рисунке. Вы должны создать новое имя функции в пункте "Вызываемая функция", а не использовать функции, которые уже существуют в POU. "Язык реализации" — это язык программирования соответствующей функции. После настройки нажмите "OK".

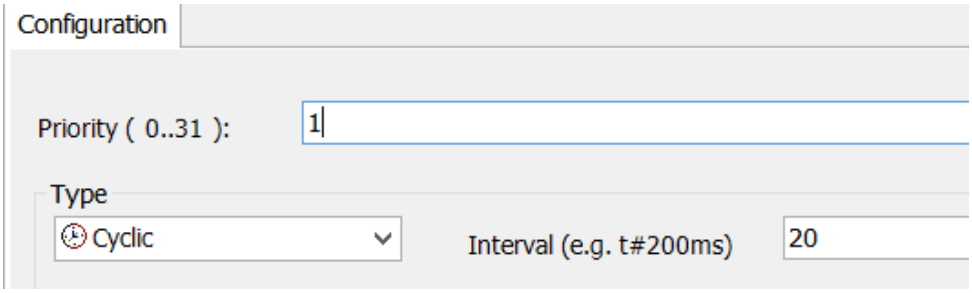


5) Приоритет задачи

В программе Codesys можно установить приоритет задач. Всего существует 32 уровня (число от 0 до 31. 0 - самый высокий приоритет, 31 - самый низкий). Во

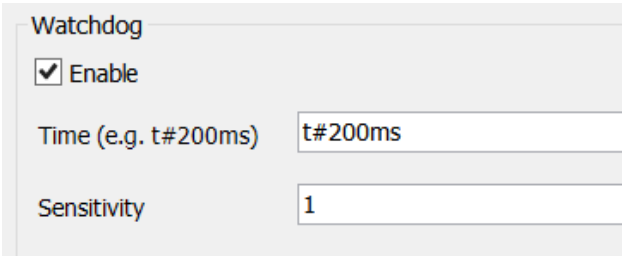
время выполнения программы задача с высоким приоритетом имеет приоритет над задачей с низким приоритетом. Задача с высоким приоритетом 0 может прервать выполнение программы с более низким приоритетом в том же ресурсе, так что выполнение программы с более низким приоритетом замедлится.

Если тип задания - "циклическое", оно будет выполняться в соответствии с временным циклом в "интервале". Конкретные настройки показаны на следующем рисунке.



6) Сторожевой таймер

Сторожевой таймер - это устройство синхронизации контроллера, которое можно включить в настройках задач в пункте "конфигурация задач" в Codesys. По умолчанию функция сторожевого таймера не используется. Основная функция сторожевого таймера заключается в отслеживании исключений во время выполнения программы или сбоя внутренних часов. Например, при сбое системы или входе программы в мертвый цикл (бесконечный) сторожевой таймер пошлет сигнал сброса в систему или остановит программу, выполняемую ПЛК в данный момент. Конфигурация сторожевого таймера показана на следующем рисунке.



(1) Время

Codesys может настроить независимый сторожевой таймер для каждой задачи. Если целевое оборудование поддерживает установку длительного времени сторожевого таймера, можно установить верхний и нижний пределы. По умолчанию единицей времени работы сторожевого таймера являются миллисекунды (MS). Если цикл выполнения программы превысит время срабатывания сторожевого таймера, функция сторожевого таймера будет активирована, и текущая задача будет прервана.

(2) Восприимчивость

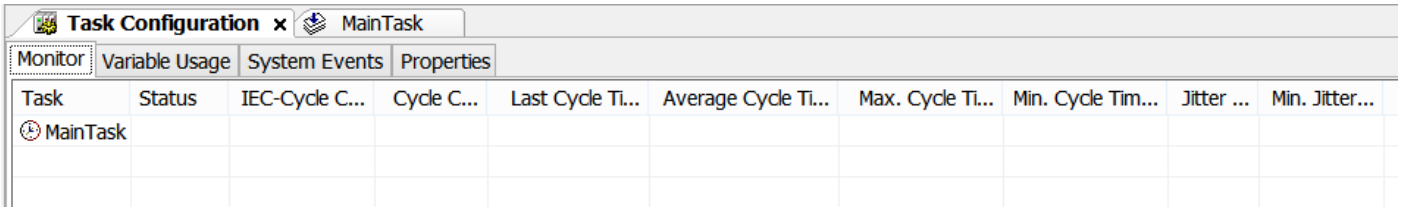
Восприимчивость используется для определения количества исключений сторожевого таймера задачи, которые должны произойти, прежде чем контроллер обнаружит ошибку приложения. Значение по умолчанию равно 1. См. следующую таблицу.

Восприимчивость	Превышение установленного времени
0,1	1
2	2
.....	.....
n	n

Окончательное время срабатывания сторожевого таймера = время × восприимчивость. Если фактическое время выполнения программы превышает время срабатывания сторожевого таймера, сторожевой таймер активируется. Например, если время составляет 10 мс, а восприимчивость установлена на 5, время срабатывания сторожевого таймера равно 50 мс. Пока время выполнения задачи превышает 50 мс, сторожевой таймер активен, задача прервана.

7) Мониторинг состояния выполнения задачи

Каждая задача может быть непосредственно включена или отключена, и система автоматически настроит монитор задачи. После перехода в онлайн-режим пользователь может использовать средство мониторинга для отслеживания параметров, связанных с выполнением задачи, таких как среднее / максимальное / минимальное время цикла задачи. Как показано на следующем рисунке:



На начальном этапе проекта можно протестировать максимальное / минимальное / среднее время цикла, что может быть использовано для измерения стабильности программы и оптимизации времени цикла задачи, установленного программой. Конкретные определения каждого параметра в окне мониторинга приведены в следующей таблице:

Параметр	Описание	Параметр	Описание
Задача	Имя задачи, определенное в конфигурации задачи	Среднее время цикла (мкс)	Среднее время выполнения задачи, единица измерения: мкс
Статус	Существует несколько статусов: Не создано (Not created): Это состояние может возникнуть, если задача не выполнялась Создано (Created): задание было создано, но не было запущено Действует (Valid): задание выполняется Исключение: в задании произошло исключение.	Макс/мин время цикла (мкс)	Максимальное/минимальное время выполнения задачи, единица измерения: мкс
Счетчик МЭК-циклов	Накопленный счетчик циклов с момента начала работы программы. '0' означает, что целевая система не вызывалась.	Джиттер (мкс)	Значение джиттера, измеренное в последнем цикле, единица измерения: мкс
Счетчик циклов (мкс)	Количество выполненных циклов. В этом случае, даже если приложение не запущено (режим “СТОП”), цикл также будет подсчитан.	Мин/макс джиттер (мкс)	Измеренное максимальное/минимальное время джиттера, единицы измерения: мкс
Посл. (Время последнего цикла) (мкс)	Время выполнения задачи в крайнем цикле, единица измерения: мкс		

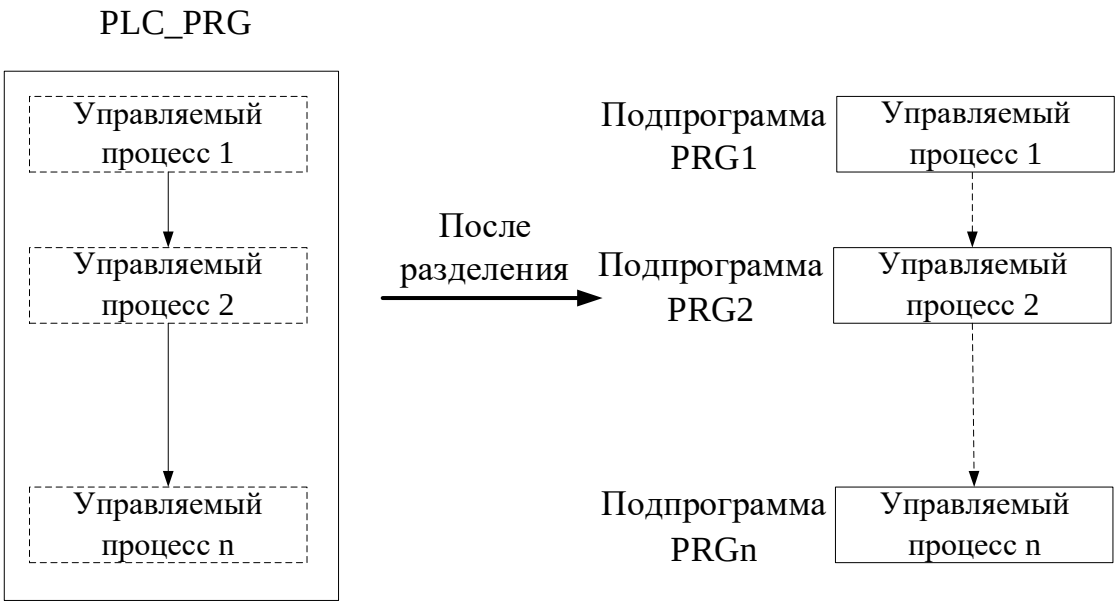
Настройки, цикл выполнения задач программы и время работы сторожевого таймера могут быть оптимизированы для обеспечения стабильности программы и ее работы в реальном времени.

Время срабатывания сторожевого таймера > фиксированное время цикла > максимальное время цикла программы

Если время цикла превышает фиксированное время цикла, ПЛК обнаружит, что программа превысила значение счетчика. Производительность программы в реальном времени будет нарушена. Если время цикла программы больше, чем установленное время сторожевого таймера, ПЛК обнаружит ошибку сторожевого таймера и остановит выполнение программы.

8) Выполнение нескольких подпрограмм

В реальных инженерных проектах программа обычно может быть разделена на множество подпрограмм в соответствии с потоком исполнения программы или в соответствии с условиями выполняемой задачи. Поэтому разработчики могут программировать ПЛК в соответствии с каждым блоком обработки. Основная программа разделена на несколько подпрограмм с различными процессами в соответствии с процессом управления. Цель разбиения - сделать основную программу более понятной и облегчить отладку в будущем.



В правой половине рисунка показаны подпрограммы PRG1, PRG2...PRGn, классифицированные по процессам, в левой половине рисунка - основная программа PLC_ PRG, PRG1...PRGn могут быть вызваны соответственно в основной программе.

Существует два способа запуска нескольких подпрограмм. Первый - добавить подпрограммы в конфигурацию задачи. Второй способ - вызывать подпрограммы в основной программе, что также является распространенным способом.

2.3.2 Файлы библиотеки

Файлы библиотек используются для хранения организационных единиц программы (POU), которые могут быть многократно использованы в Codesys. Пользователи могут создать собственную библиотеку на основе базовой библиотеки и ссылаться на нее в программе.

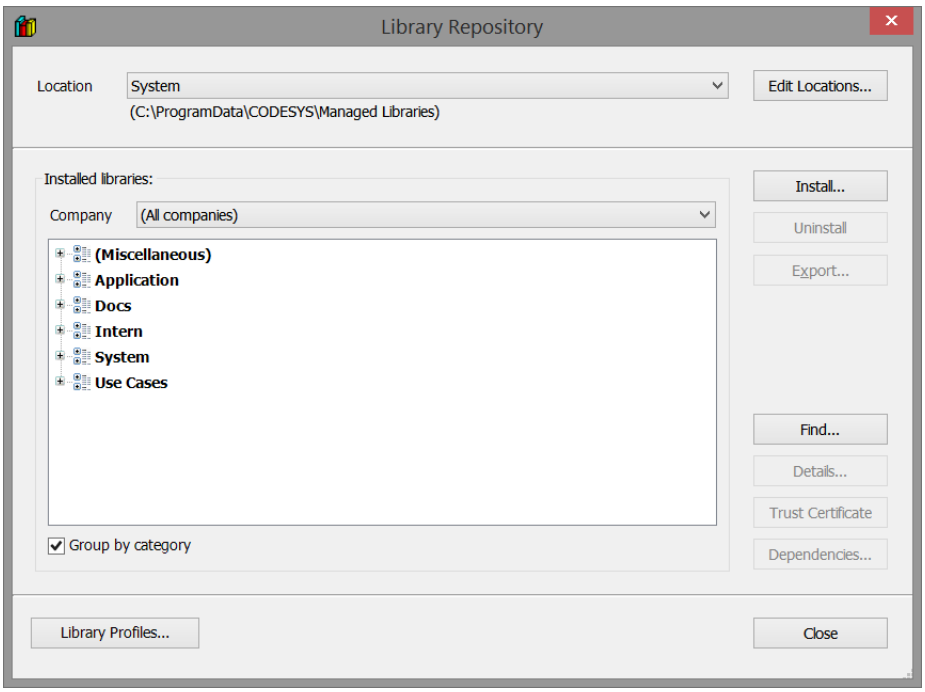
Библиотечный файл - это набор функций, функциональных блоков и программ, который также содержит некоторые специально определенные структуры и т.д.

По назначению библиотечные файлы можно разделить на системные библиотечные файлы, библиотечные файлы приложений и библиотечные файлы, определенные производителем. Среди них системный библиотечный файл - это файл, поддерживающий Codesys, включая поддержку структуры и синтаксиса программ, а также поддержку стандартного ввода-вывода. Файл библиотеки приложений - это библиотека файлов, поддерживающая основные приложения, включая функции работы с данными, таймеры, счетчики, обнаружение фронта и т. д. Файл библиотеки, определяемый производителем, - это специально созданный файл библиотеки соответствующий характеристикам продуктов различных производителей.

1) Управление библиотечными файлами

Менеджер библиотек отображает все библиотеки, относящиеся к текущему проекту. POU, тип данных и глобальные переменные библиотеки могут быть пользовательскими POU и классам данных. Менеджер библиотек открывается командой library manager, соответствующая информация о библиотеке сохраняется вместе с проектом.

Если вам необходимо установить файл библиотеки на компьютер или вызвать файл библиотеки, предоставленный поставщиком, необходимо воспользоваться управлением файлами библиотеки. Управление файлами библиотеки задается с помощью команды меню "Инструменты" > "Репозиторий библиотеки". На следующем рисунке показан вид управления библиотечными файлами.



Категории отображаемых библиотечных файлов включают приложения, коммуникации, контроллеры, устройства и т. д.

Процесс использования библиотечных файлов выглядит следующим образом:

(1) Установка библиотечных файлов.

Перед использованием файла библиотеки необходимо сначала "Установить" его в диалоговом окне “Репозиторий библиотек”. После установки библиотеку можно вызывать в проекте.

(2) Вызов библиотечного файла.

После установки файла библиотеки необходимо добавить его через менеджер библиотек, чтобы реализовать обращение проекта к файлу библиотеки.

2) Свойства библиотечных файлов

Библиотечный файл должен обеспечивать уникальность и безопасность доступа.

(1) Уникальность доступа.

Если несколько модулей или переменных в проекте имеют одинаковые имена, пути доступа к переменным с одинаковыми именами должны быть разными (то есть "уникальный доступ"), иначе возникнут ошибки компиляции. Это правило относится к локальным проектам, библиотекам, а также к модулям или переменным в библиотеках, на которые ссылаются другие библиотеки. Пользователи могут добиться уникального доступа, добавив пространство имен (namespace) перед именем модуля или переменной.

(2) Безопасность доступа.

Codesys предоставляет функцию шифрования библиотечных файлов для защиты исходного кода файлов библиотек разработчика. Добавив информацию о правах доступа к файлу библиотеки в настройках проекта и сохранив его как "скомпилированную библиотеку функций", пользователь должен войти в систему с паролем, чтобы открыть файл библиотеки в следующий раз. Если пароль неверен, файл библиотеки не может быть использован и открыт, и срабатывает сигнал тревоги.

2.3.3 Путь доступа

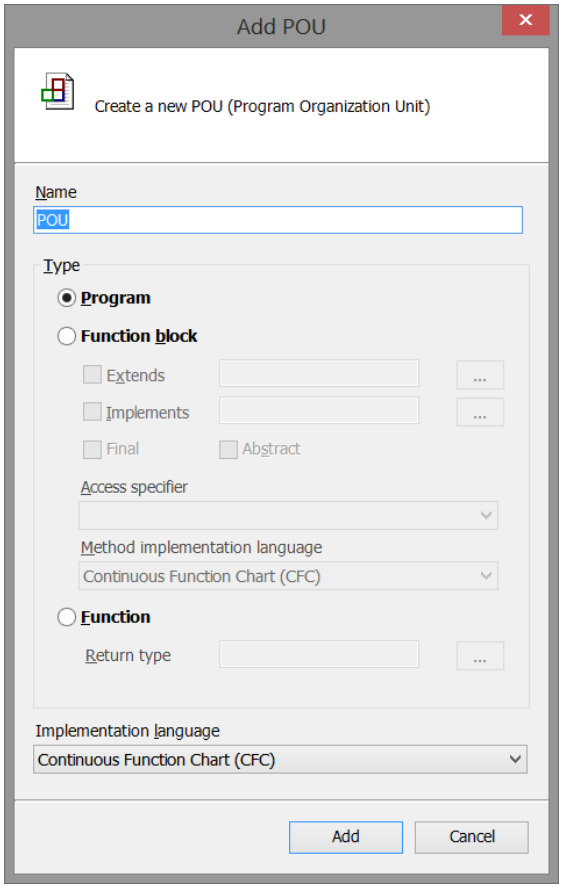
Путь доступа используется для соединения глобальных переменных, переменных прямого представления, ввода/вывода функциональных блоков и локальных переменных для хранения информации. Путь доступа обеспечивает обмен данными и информацией между различными конфигурациями. Ко многим переменным с заданными именами в каждой конфигурации можно получить доступ через другие удаленные конфигурации.

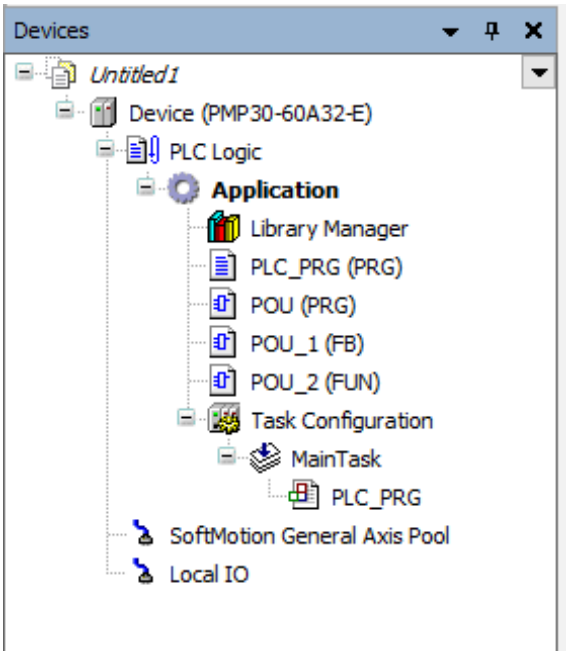
Функция пути доступа интегрирована в Codesys. Пользователям не нужно управлять ею. Все операции доступа будут выполняться автоматически в фоновом режиме Codesys.

2.4 POU

Единица организации программы (POU) - это наименьшая программная единица пользовательской программы, которая состоит из области объявления и программы. POU можно разделить на функции (FUN), функциональные блоки (FB) и программы (PRG).

Щелкните правой кнопкой мыши "Приложение"(Application), нажмите "Добавить объект..." --- "POU", в результате чего появится рисунок ниже. В диалоговом окне пользователи могут выбрать добавление программ, функциональных блоков или функций, а соответствующий язык программирования можно выбрать в выпадающем меню. После добавления можно просмотреть соответствующие атрибуты в скобках POU в дереве устройств проекта слева. FB - это функциональный блок, FUN - функция, а PRG - программа.



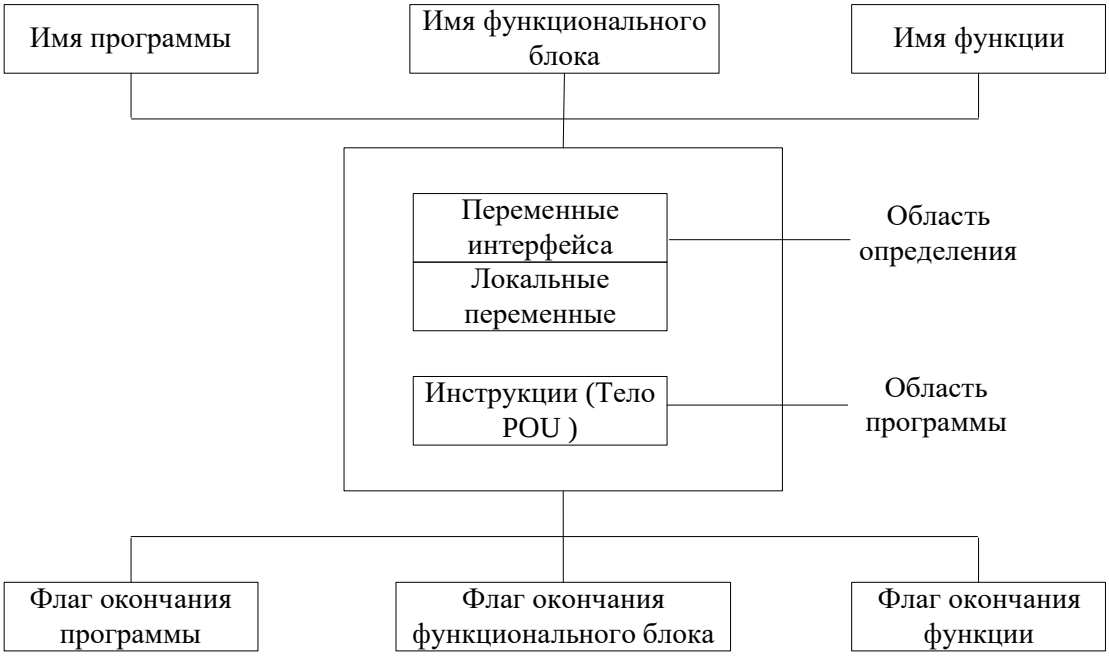


POU имеет следующие характеристики:

- Пользовательская библиотека функциональных блоков может быть создана для каждой области применения.
- Функциональные блоки можно тестировать и записывать
- POU может обеспечить функцию поиска в глобальной библиотеке
- POU можно использовать многократно, причем количество раз использования не ограничено

2.4.1 Структура POU

Полный POU состоит из трех частей: Тип и именование POU, часть объявления переменных и часть программы (тело POU). Структурная схема выглядит следующим образом:



На рисунке выше, с точки зрения конкретных функций, можно выделить программу (PRG) слева, промежуточный функциональный блок (FB) и функцию (FUN) справа соответственно. По структуре каждая функция может быть разделена на часть определения и часть программы.

Все переменные, объявленные пользователем, в конечном итоге используются POU. Интерфейсные и локальные переменные могут быть объявлены в объявлении переменных.

1) Область определения

Область определения переменных используется для указания имени, типа и начального значения переменных.

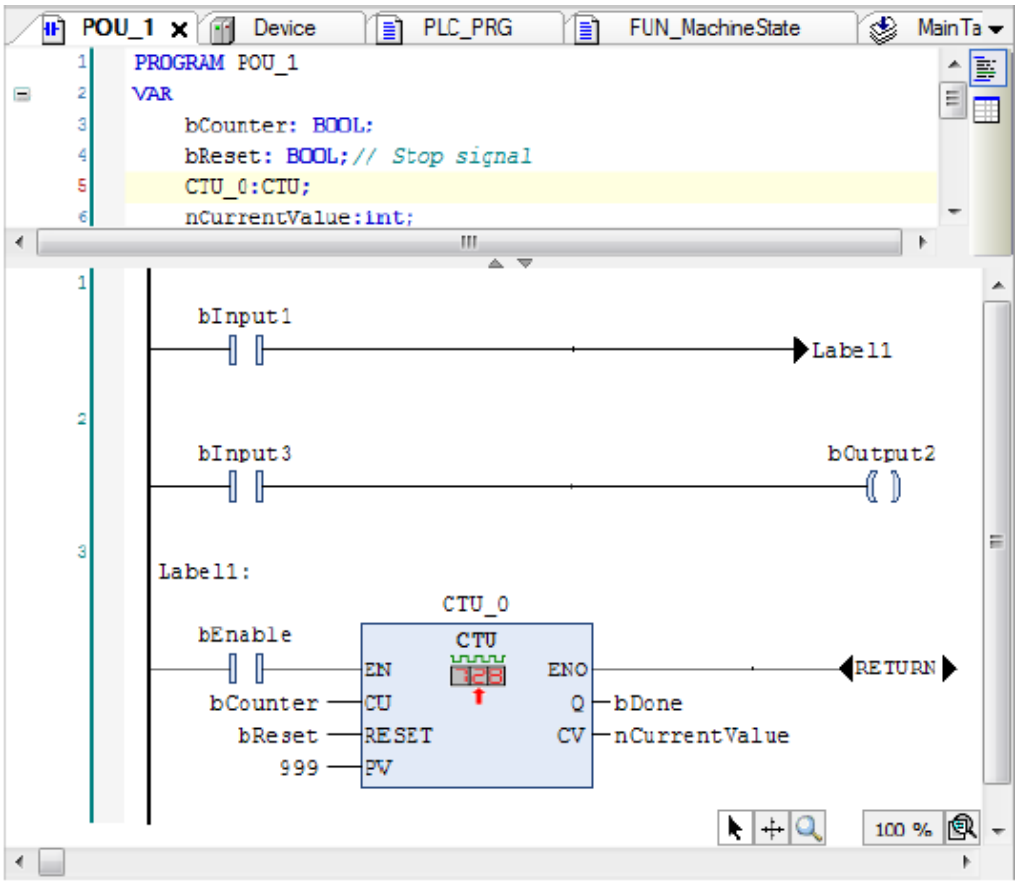
Редактор переменных используется для объявления переменных и типов данных POU. Все переменные, которые будут использоваться в POU, объявляются в части объявления POU, включая входные переменные, выходные переменные, входные / выходные переменные, локальные переменные, добавленные переменные и константы. Формат основан на стандарте IEC61131-3. Объявление переменных имеет следующий формат:

<идентификатор>{AT<Address>}: <тип данных>{: =<инициализация >}:

Часть {} является необязательной.

2) Область программы

Область программы Codesys поддерживает два текстовых языка: список инструкций (IL) и структурированный текст (ST). Четыре графических языка: функциональная блок-схема (FBD), лестничная диаграмма (LD), последовательная функциональная диаграмма (SFC) и непрерывная функциональная диаграмма (SFC). Пользователи могут выбрать один или несколько языков для программирования в основной части. На рисунке ниже показан интерфейс главного редактора, в котором используется язык программирования лестничных диаграмм (LD).



2.4.2 Функция

Применительно к языку программирования ПЛК функция (FUN) также определяется как единица организации программы POU. Функция — это единица организации программы, которой могут быть присвоены параметры, но у которой нет статических переменных. То есть при вызове функции с одними и теми же входными параметрами функция всегда возвращает один и тот же результат в виде значения функции (возвращаемое значение).

Функция (FUN) — это базовая единица алгоритма, не имеющая внутренней памяти (без выделения памяти во время выполнения). Другими словами, при оди-

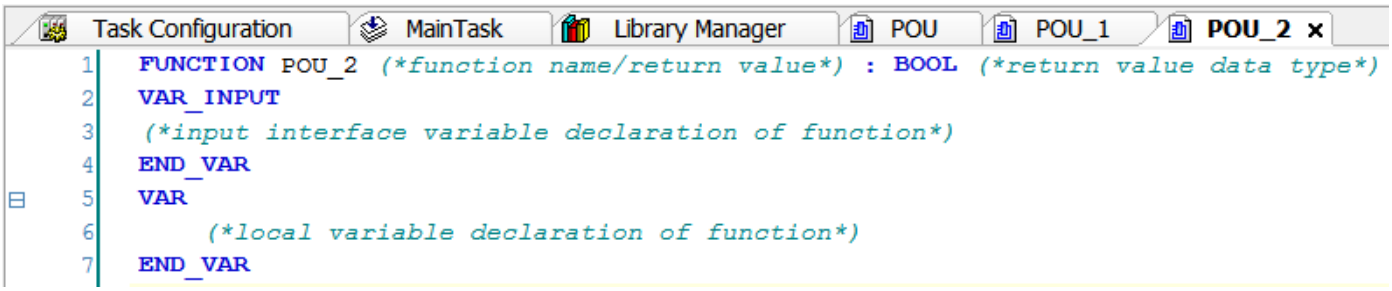
наковых входных параметрах вызываемая функция должна получить тот же результат операции. Типичными типами функций являются различные математические операции, например sin (x), sqrt (x) и т.д.

Функция — это базовая единица алгоритма, имеющая как минимум одну входную переменную, не содержащая инкапсулированных данных и возвращающая только одно значение. Стандартные функции находятся в стандартной библиотеке Codesys. Функции могут вызывать функции, функциональные блоки и программы.

1) Объявление функций

(1) Представление пользовательских функций

Внутренняя логическая часть функции может использовать любой из шести языков программирования. Имя функции — это возвращаемое значение функции, которое также можно понимать как выходное значение функции, как показано на следующем рисунке:



(2) Объявление переменных в функциях

При настройке функций пользователям следует обратить внимание на следующие моменты:

- Функция может иметь много входных переменных, но только одно возвращаемое значение (выходная переменная). Однако нет ограничений на тип данных возвращаемого значения, поэтому в качестве возвращаемого значения может выступать структура.
- Важной особенностью функций является то, что они не могут хранить значения во внутренних переменных, чем отличаются от функциональных блоков.
- Функция не имеет распределения памяти.
- Функции могут вызывать только функции, но не функциональные блоки.
- Аргумент, заданный в VAR_INPUT, может быть пустым, константой, переменной или вызовом функции. При вызове функции в качестве фактического аргумента вызывается функция.

2) Стандартные функции

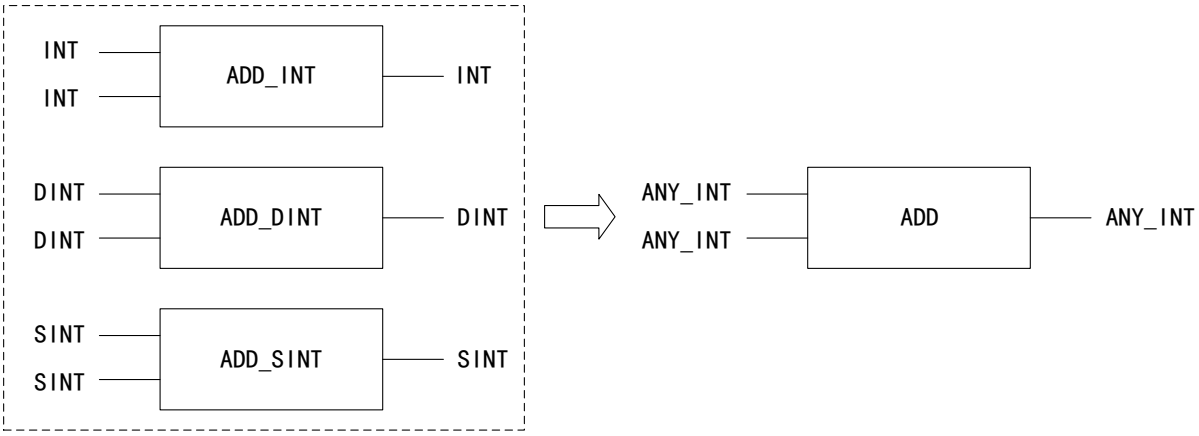
Codesys поддерживает все стандартные функции IEC класса 8. Кроме того, можно использовать следующие функции, не указанные в стандартах IEC: ANDN, ORN, XORN, INDEXOF, SIZEOF, ADR, BITADR и др. Codesys поддерживает следующие 11 типов функций. Использование и описание конкретных функций будет подробно рассмотрено в главе 6.

3) Свойства функции

(1) Свойство перегрузки

Функция, вход которой описывается общим типом данных, называется перегруженной. Это означает, что вход этой функции не ограничен одним типом данных, а может быть использован для различных типов данных. Все стандартные функции Codesys обладают свойствами перегрузки, которые можно применять к различным типам данных. Если функция применима только к определенному типу данных, это должно быть объявлено в имени функции, что называется строгой типизацией функции.

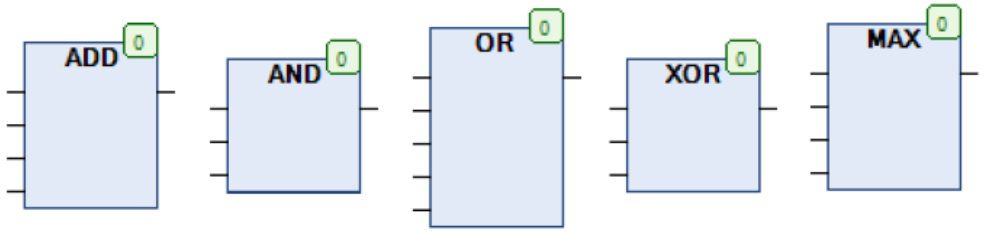
Например, если ПЛК распознает INT, DINT и SINT, он поддерживает функцию перегрузки ADD для общего типа данных ANY_INT (включая BYTE, WORD, DWORD, SINT, USINT, REAL и т.д.). Например, ADD_INT - это функция сложения INT, ограниченная типами данных. Это типизированная функция. Таким образом, функция перегрузки не зависит от типа. Описание перегруженных функций показано на следующем рисунке:



При использовании перегруженных функций система автоматически выбирает соответствующий тип данных. Например, если тип данных вызываемого аргумента ADD - DINT, система вызовет стандартные функции ADD_DINT.

(2) Масштабируемость

Свойство, согласно которому количество входных переменных функции может быть увеличено, называется расширяемым свойством функции. Например, входных переменных у функции ADD может быть больше двух. Она может реализовать сложение нескольких входных переменных. Поэтому можно сказать, что функция add обладает расширяемыми свойствами. Не все стандартные функции обладают расширяемыми свойствами. Предел расширения этой функции зависит от верхнего предела, установленного ПЛК, предела высоты поля в графическом языке программирования или предела определения самой функции. Например, функция DIV имеет этот атрибут. Функции с расширяемыми свойствами могут упростить программу и сократить требуемый объем памяти. На следующем рисунке приведен пример некоторых функций с расширяемыми свойствами.



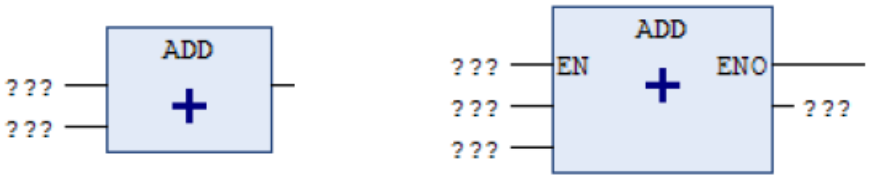
(3) EN и ENO

Этот атрибут действителен только в языках программирования лестничных и функциональных блок-схем. EN и ENO - это разрешение входа и разрешение выхода функции соответственно. Все функции могут включать или выключать это свойство.

Принципы применения разрешающего входа и разрешающего выхода следующие:

- Если при вызове входной функции значение EN равно false, то операция, заданная телом функции, не будет выполнена программой, а значение ENO равно false.
- Если значение EN равно true, функция вызывается, выполняется операция, определенная телом функции, и значение ENO равно true.
- Атрибуты EN и ENO - это дополнительные атрибуты, которые могут быть включены или выключены в соответствии с реальными потребностями.

На следующем рисунке функция ADD с EN/ENO сравнивается с обычной функцией ADD.



2.4.3 Функциональный блок

Функциональный блок — это преобразование некоторых программных блоков, которые используются многократно, в общий компонент. Он может вызываться любым языком программирования в программе и использоваться многократно, что не только повышает эффективность разработки программы, но и уменьшает количество ошибок при программировании, тем самым повышая качество программы.

Блок организации программы, который может генерировать одно или несколько значений при выполнении функционального блока. Функциональный блок сохраняет свои собственные специальные внутренние переменные, и целевая система исполнения контроллера должна выделить память для внутренних переменных состояния функционального блока, которые представляют собой его собственные характеристики состояния.

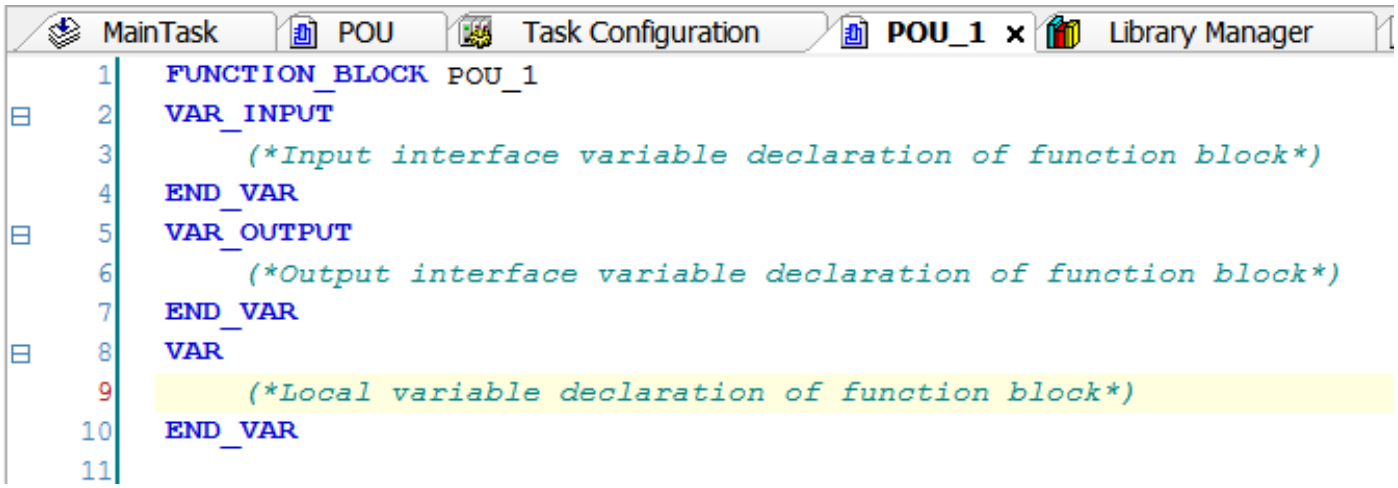
Логика выполнения функционального блока представляет собой собственную характеристику поведения объекта. Поэтому для входного значения одного и того же параметра могут существовать различные внутренние переменные состояния, поэтому могут быть получены различные результаты вычислений. В системе управления функциональный блок может представлять собой несколько видов алгоритмов управления, например, функциональный блок PID используется для управления в замкнутом контуре, а другие функциональные блоки могут использоваться для счетчиков, наклонов, фильтров и т.д.

1) Представление и объявление функциональных блоков

(1) Представление пользовательских функциональных блоков

Как и функции, внутренняя логическая часть функциональных блоков может использовать любой из шести языков программирования. Имя функции - это возвращаемое значение функции, которое также можно понимать как выходное значение функции.

На следующем рисунке представлено выражение функционального блока.



(2) Объявление переменных в функциональных блоках

Объявления переменных в функциональных блоках аналогичны объявлениям в функциях. При написании текста следует обратить внимание на следующие моменты:

- Внутренние и выходные переменные функционального блока могут использовать атрибут RETAIN, чтобы указать, что переменная имеет функцию удержания значения. Входные переменные могут быть объявлены со свойствами retain только во время вызова.
- Присваивать значения входным переменным функционального блока обычно не разрешается. Только если вход является вызывающей частью функционального блока, разрешается присваивать значение входной переменной функционального блока.
- Поскольку функциональные блоки могут вызывать функции и функциональные блоки, вы также можете вызывать экземпляры функциональных блоков как переменные экземпляров других функциональных блоков. Например, DB_ FF(S1:=DB_ON.Q, R:=DB_OFF.Q).
- Входы функциональных блоков не назначаются, что означает сохранение их начальных значений.
- Чтобы функциональный блок не зависел от оборудования, адресные переменные с фиксированными адресами (например, %IX1.1, %QD12) не разрешается использовать в качестве локальных переменных в объявлении переменных функционального блока, но им можно присваивать значения при вызове.
- Использование VAR_INPUT и VAR_OUTPUT может занять много памяти. Поэтому при программировании функциональных блоков можно максимально использовать VAR_IN_OUT, чтобы уменьшить занимаемый объем памяти.

2) Стандартный функциональный блок

В стандартную библиотеку включены детектор фронта, таймеры и другие функциональные блоки.

3) Атрибуты функциональных блоков

(1) Инстанция

Согласно стандарту IEC61131-3, тип функционального блока — это определение типа абстрактной структуры. Если он не определен и не инстанцирован, он не может быть вызван и выполнен программой. Поэтому функциональные блоки должны быть инстанцированы до того, как их можно будет использовать.

Инстанцированный функциональный блок — это независимая структурная переменная, которая обладает данными, может выполнять определенные функции в соответствии с установленной логикой и полностью инкапсулирована. Таким образом, предыдущее определение абстрактного типа преобразуется в данные.

(2) Масштабируемость

Codesys поддерживает объектно-ориентированное программирование, поэтому функциональные блоки также могут создавать "дочерние" функциональные блоки. Таким образом, "дочерний" функциональный блок обладает атрибутами "родительского" функционального блока и может иметь свои собственные дополнительные характеристики. Визуально можно считать, что "дочерний" функциональный блок является расширением "родительского" функционального блока. Поэтому мы называем это "расширением функционального блока".

Добавьте ключевое слово "extends" при объявлении функционального блока, чтобы использовать расширенную функцию. Вы также можете расширить функцию, выбрав опцию "extends" при добавлении функционального блока в диалоговом окне "Добавить объект".

(3) EN и ENO

Функциональные блоки имеют вспомогательные атрибуты EN и ENO, которые аналогичны использованию EN и ENO в функциях.

(4) Различия между функциональными блоками

Подводя итог, можно сказать, что очевидные различия между функциями и функциональными блоками сведены в следующую таблицу:

	Функция (FUN)	Функциональный блок (FB)
Память распределение	Не указан адрес выделения памяти	Все данные, выделенные по адресу памяти
Входные/выходные переменные	Допускается только одна выходная переменная	Несколько выходных переменных или отсутствие выходных переменных
Звонок на отношения	Функции можно вызывать, но функциональные блоки вызывать нельзя	Вызываемый функциональный блок или функция

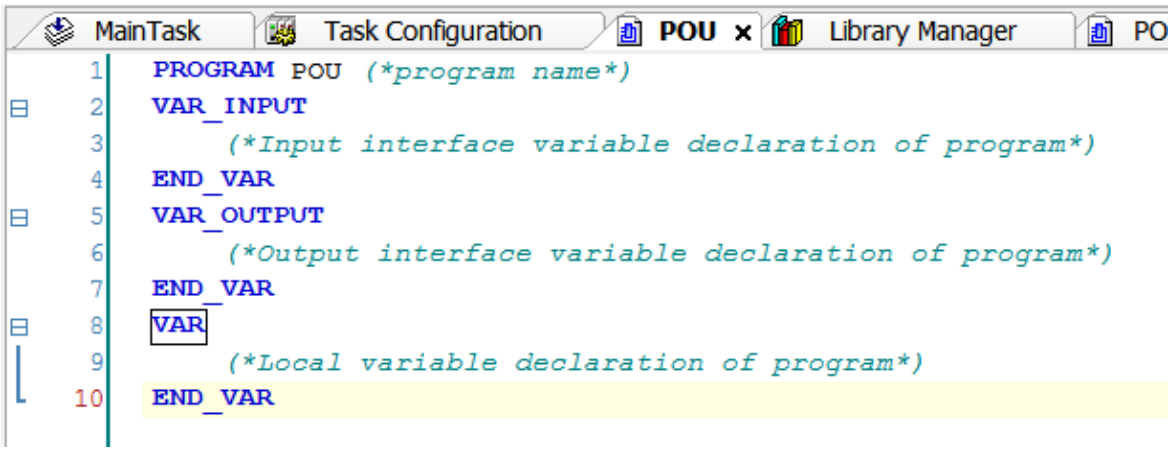
2.4.4 Программа

Программа - это основное ядро планирования задачи. Программа обладает наибольшим правом вызова и может вызывать функциональные блоки и функции. В общем случае она делится на основную программу и подпрограмму. В широком смысле она также включает в себя конфигурацию оборудования, конфигурацию задачи, конфигурацию связи и информацию о целевых настройках.

Как правило, в программе определяются общие глобальные переменные, глобальные переменные с сопоставленным аппаратным адресом и локальные переменные. Прикладная логика реализуется посредством вызовов между программами.

1) Представление и декларирование программы

Программа выражается следующим синтаксическим выражением, а логическая часть программы может использовать любой из шести языков программирования.



2) Выполнение программы

(1) Программа может содержать конфигурацию адресов.

Разрешается объявлять переменные прямого представления, которые хранят физический адрес ПЛК, а конфигурация адреса прямого представления используется только для объявления внутренних переменных в программе. Переменные прямого представления позволяют описывать иерархический режим адресации, например, следующие представления.

Вы можете заполнить декларацию программы в следующем формате.

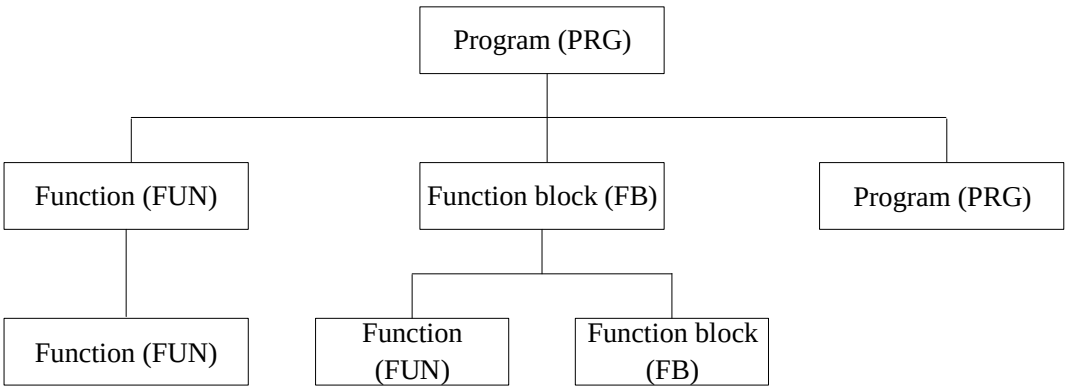
bTest AT %QX10.3:BOOL:=TRUE;

(2) Единица организации программы не может вызывать саму себя прямо или косвенно, то есть единица организации программы не может вызывать экземпляр единицы организации программы с тем же типом и именем.

(3) Программы инстанцируются только в ресурсах. Объявляются в ресурсе. Экземпляр программы должен только объединять программу с задачей, иначе он не будет выполнен. Функциональные блоки могут быть инстанцированы только в программы или другие функциональные блоки.

3) Отношения вызова программы

В программе разрешается вызывать экземпляры функциональных блоков, функции и другие программы, как показано на следующем рисунке:



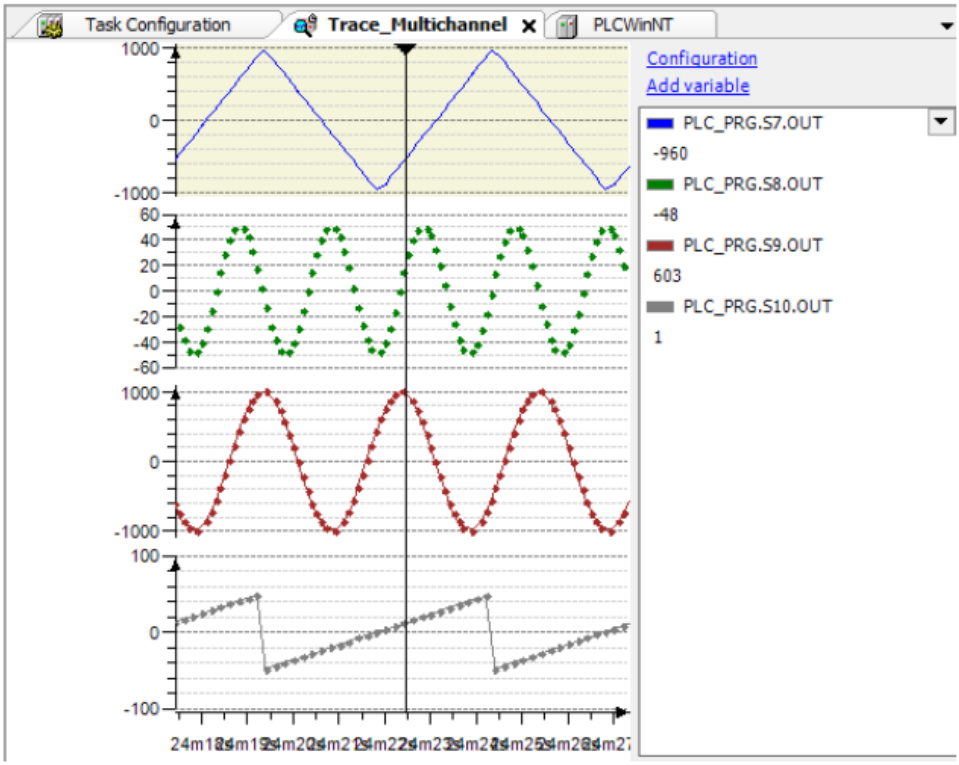
Согласно приведенному рисунку, функции и функциональные блоки используются для формирования подпрограмм, а программы - для формирования основных программ пользователя. Поэтому программы считаются глобальными. Программа - это самая большая форма организационной единицы программы, которая может вызывать функции, функциональные блоки и программы.

Функциональные блоки могут вызывать другие функциональные блоки и функции. Поскольку в функции нет приватных переменных, функция может вызывать только другие функции, а не экземпляры функциональных блоков.

2.5 Объект приложения

2.5.1 Отслеживание образцов

Функция выборки и отслеживания заключается в мониторинге и отслеживании истории значений переменных на контроллере. Режим работы выборочного слежения аналогичен режиму работы цифрового выборочного осциллографа. Это очень практичный и эффективный инструмент отладки в процессе отладки и диагностики программы. Пользователь может добавить "объект отслеживания" и установить "конфигурацию отслеживания" в менеджере отслеживания для записи командного слова, слова состояния, скорости двигателя, положения и других параметров, используемых при выполнении программы. Наблюдая за этими параметрами, пользователь может понять весь процесс выполнения программы в системе управления. Эта функция показана на следующем рисунке:



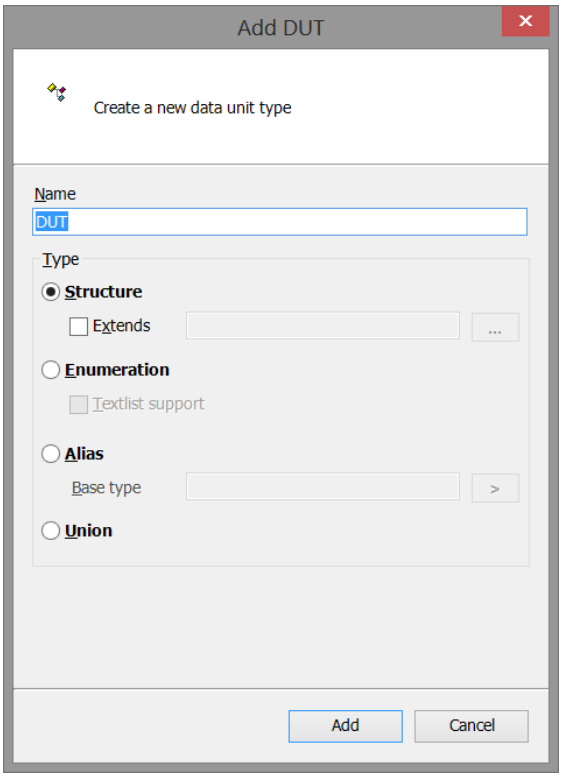
2.5.2 Постоянная переменная

Функция постоянной переменной заключается в сохранении данных в запоминающем устройстве после выключения системы или аномального прерывания, а также в их вызове после повторного включения питания, и они могут продолжать использоваться программой. Чтобы адаптироваться к условиям работы на объекте, при проектировании системы управления ПЛК необходимо учитывать возможность сохранения и восстановления данных после отключения питания или аномального прерывания. Пользователи могут зарегистрировать данные,

которые необходимо сохранять при отключении питания, в списке, добавив список persistentvars, что позволяет реализовать функцию непрерывной переменной.

2.5.3 Тип единицы данных

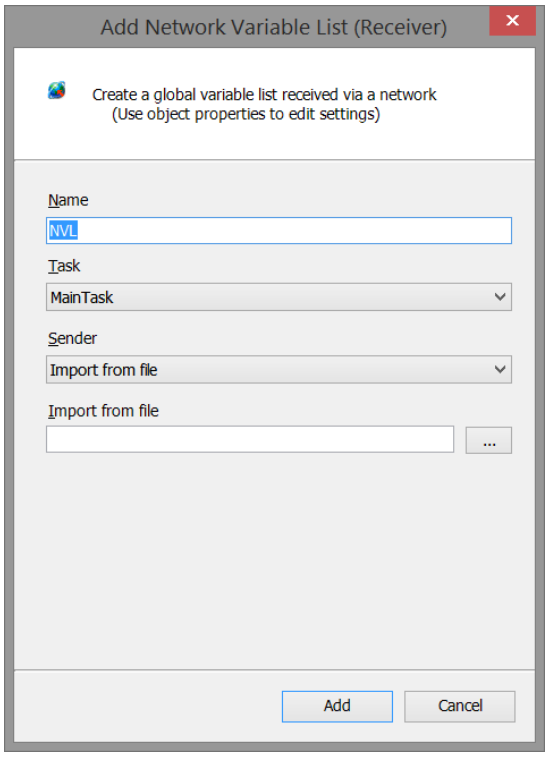
Функция типа единицы данных (DUT) заключается в предоставлении пользователям определенного пользователем типа данных, включая структуру, перечисление, псевдоним и объединение, как показано на следующем рисунке. Использование типа единицы данных играет роль в стандартизации процесса программирования, повышении эффективности программирования, оптимизации формата программирования и реализации объектно-ориентированного программирования.



2.5.4 Глобальные сетевые переменные

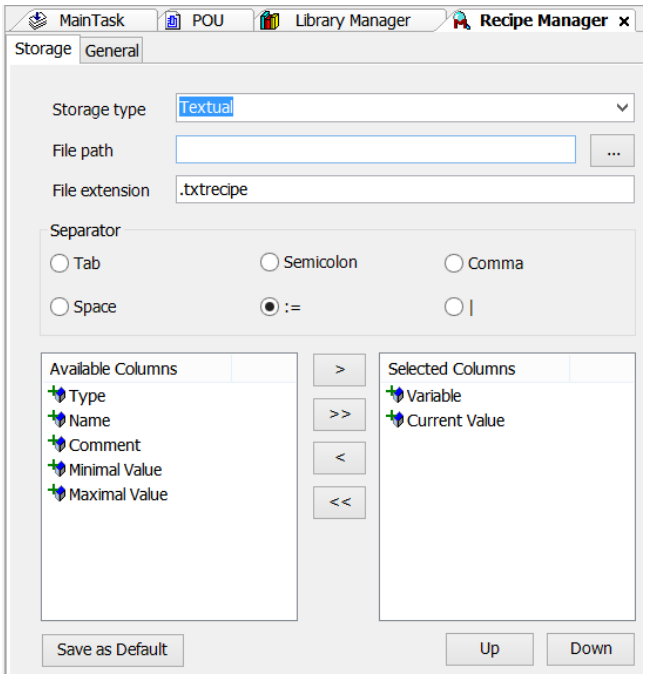
Глобальный список сетевых переменных (GNVL) делится на две формы: отправитель и получатель. Функция отправителя заключается в объявлении и перечислении глобальных переменных списка сетевых переменных (приемника), которые должны быть отправлены другим устройствам или элементам сети. Функция получателя заключается в составлении списка полученных сетевых переменных и отображении информации (сеть, информация о передаче, отправитель и т. д.). Пользователи могут добавить список глобальных сетевых переменных отправителя и получателя в дерево устройств, настроив редактор

глобальных сетевых переменных, чтобы реализовать взаимодействие глобальных переменных в сети.



2.5.5 Менеджер по рецептам

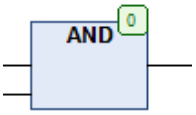
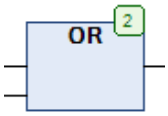
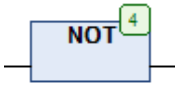
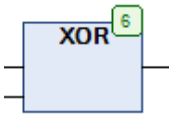
Функция менеджера рецептов заключается в предоставлении списка определяемых пользователем переменных (определений рецептов). Пользователи могут настроить место хранения, способ хранения и категорию хранения с помощью менеджера рецептов, как показано на следующем рисунке. После успешной настройки менеджера рецептов пользователи могут загружать и скачивать определения рецептов.



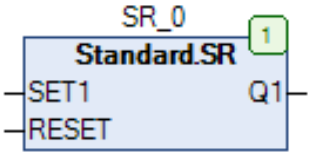
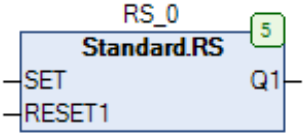
3 Основные инструкции

3.1 Битовые логические инструкции

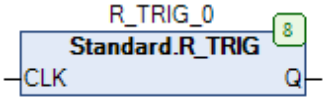
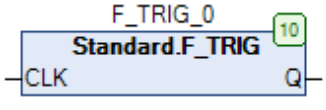
3.1.1 Основные логические инструкции

Инструкция	Значок команды	Функция
AND		Оператор AND
OR		Оператор OR
NOT		Оператор NOT
XOR		Оператор XOR

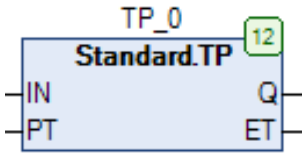
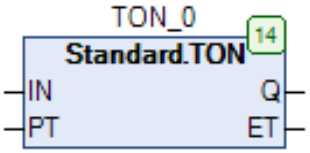
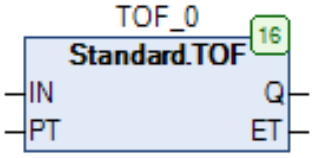
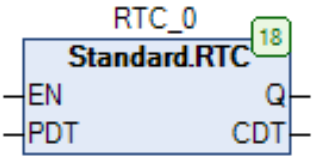
3.1.2 Инструкции по установке приоритета и сбросу приоритета триггера

Инструкция	Значок команды	Функция
SR		Установка SR-триггера
RS		Установка RS-триггера

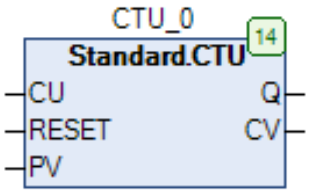
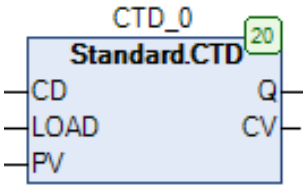
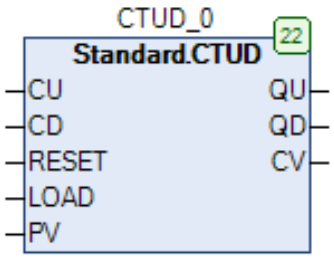
3.1.3 Тип единицы данных

Инструкция	Значок команды	Функция
R_TRIG		Триггер по нарастающему фронту
F_TRIG		Триггер по падающему фронту

3.2 Инструкции к таймеру

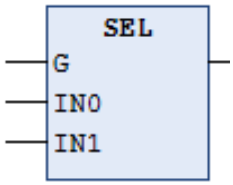
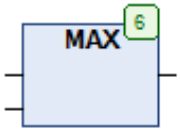
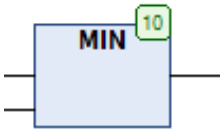
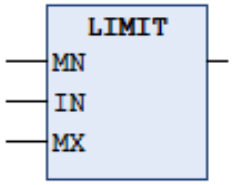
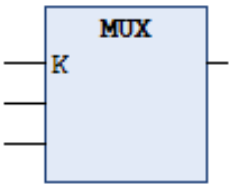
Инструкция	Значок команды	Функция
TP		Импульсный таймер: как только IN станет TRUE, Q станет истинным, и время начнет отсчитываться в миллисекундах в ET, пока его значение не станет равным PT, тогда Q станет FALSE
TON		Таймер задержки включения питания: как только IN станет TRUE, время начнет отсчитываться в миллисекундах в ET, пока его значение не станет равным PT, тогда Q станет TRUE
TOF		Таймер задержки выключения питания: если IN - FALSE и ET равен PT, Q - FALSE. В противном случае - TRUE
RTC		Часы реального времени: запускаются в заданное время и возвращают дату и время

3.3 Инструкции для счетчиков

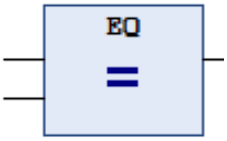
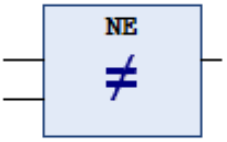
Инструкция	Значок команды	Функция
CTU		Инкр. счетчик: если RESET равен TRUE, инициализация равна 0. Восходящий фронт CU всегда увеличивает на 1. Как только CV > = PV, Q будет установлен в TRUE
CTD		Дикр. счетчик: если LOAD равен TRUE, CV будет установлен на начальное значение, заданное PV. Восходящий фронт CD всегда увеличивает на 1. Как только CV уменьшится до 0, Q устанавливается в TRUE
CTUD		Двунаправленный счетчик вверх/вниз

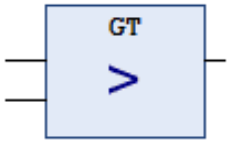
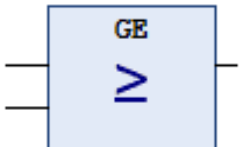
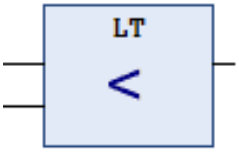
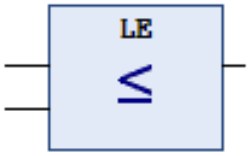
3.4 Инструкции по обработке данных

3.4.1 Выберите инструкцию по эксплуатации

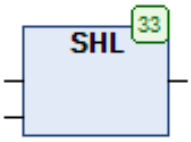
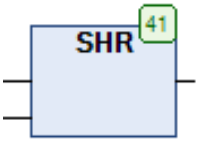
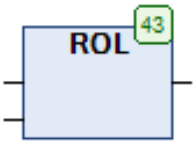
Инструкция	Значок команды	Функция
SEL		Инструкция "один из двух": когда переключатель выбора G в FALSE, на выходе зн-е IN1; когда переключатель выбора TRUE, на выходе зн-е IN2
MAX		Поиск максимального значения
MIN		Поиск минимального значения
LIMIT		Предельное значение: если значение IN больше верхнего предела Max, LIMIT генерирует Max. Если значение IN меньше нижнего предела Min, результатом будет Min
MUX		Выбор одного из многих: MUX выбирает K-е значение из группы значений. Первым значением является K = 0. Если K больше, чем количество других входов (n), Codesys передает последнее значение

3.4.2 Сравните инструкции

Инструкция	Значок команды	Функция
EQ		Наравне с
NE		Не равно

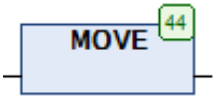
Инструкция	Значок команды	Функция
GT		Больше, чем
GE		Больше или равно
LT		Меньше, чем
LE		Меньше или равно

3.4.3 Инструкция по переключению

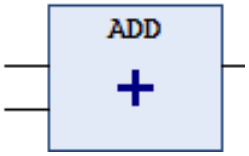
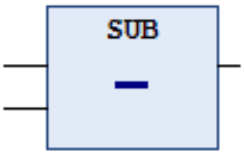
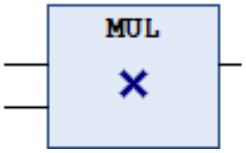
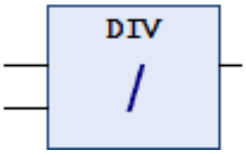
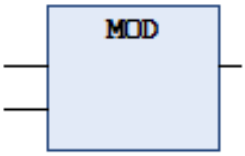
Инструкция	Значок команды	Функция
SHL		Сдвиг влево на бит
SHR		Сдвиг вправо на бит
ROL		Повернуть влево
ROR		Повернуть вправо

3.5 Инструкция по эксплуатации

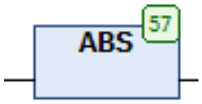
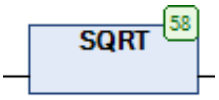
3.5.1 Инструкция по выполнению заданий

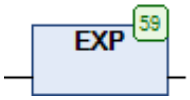
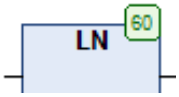
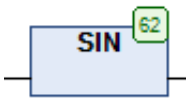
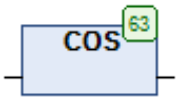
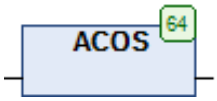
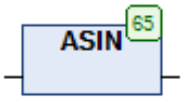
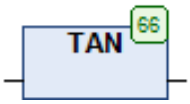
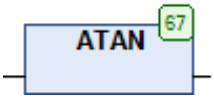
Инструкция	Значок команды	Функция
MOVE		Задание

3.5.2 Арифметическая операция

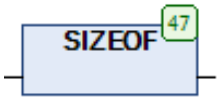
Инструкция	Значок команды	Функция
ADD		Дополнение
SUB		Вычитание
MUL		Умножение
DIV		Деление
MOD		Остаток

3.5.3 Инструкция по выполнению математических операций

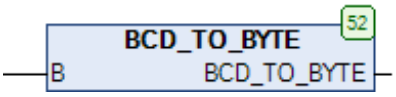
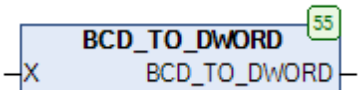
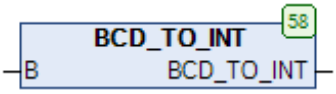
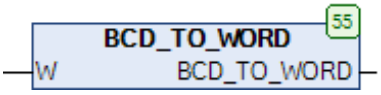
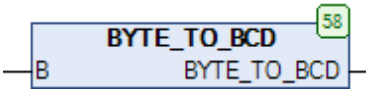
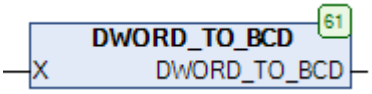
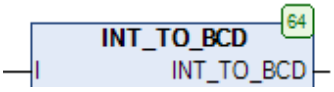
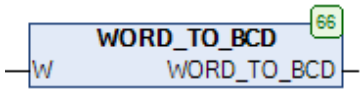
Инструкция	Значок команды	Функция
ABS		Инструкция абсолютного значения
SQRT		Указание квадратного корня

Инструкция	Значок команды	Функция
EXP		Инструкция экспоненты
LN		Инструкция натурального логарифма
LOG		Обычная логарифмическая инструкция
SIN		Инструкция синуса
COS		Инструкция косинуса
ACOS		Инструкция арккосинуса
ASIN		Инструкция арксинуса
TAN		Инструкция тангенса
ATAH		Инструкция арктангенса

3.5.4 Инструкция по работе с адресами

Инструкция	Значок команды	Функция
SIZEOF		Размер типа данных
ADR		Оператор адреса
BITADR		Оператор битового адреса

3.5.5 Инструкция по преобразованию данных

Инструкция	Значок команды	Функция
BCD_TO_BYTE		Преобразование BCD в BYTE
BCD_TO_DWORD		Преобразование BCD в DWORD
BCD_TO_INT		Преобразование BCD в INT
BCD_TO_WORD		Преобразование BCD в WORD
BYTE_TO_BCD		Преобразование BYTE в BCD
DWORD_TO_BCD		Преобразование DWORD в BCD
INT_TO_BCD		Преобразование INT в BCD
WORD_TO_BCD		Преобразование WORD в BCD

4 Специальные функции

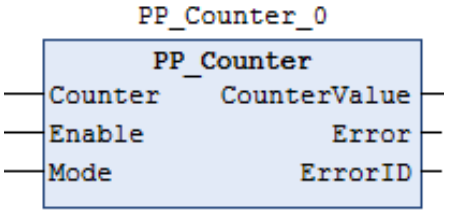
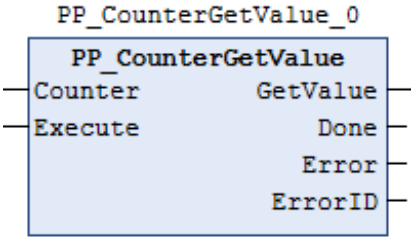
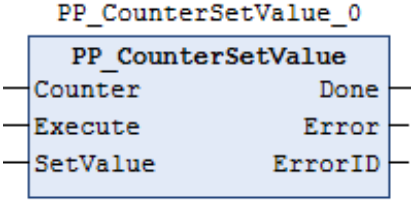
4.1 Высокоскоростной счёт

4.1.1 Обзор функций

ПЛК серии PMP30 имеет функцию высокоскоростного счета. Выбирая различные счетчики, он может измерять высокоскоростные входные сигналы, такие как измерительные датчики и поворотные энкодеры, а его максимальная частота измерений может достигать 200 кГц. Максимальная частота измерений ПЛК серии PMP301 может достигать до 80 кГц.

4.1.2 Введение функционального блока

1) Формат команды

Команда	Имя	Графическое представление	Производительность ST
PP_Counter	Высокоскоростной счетчик		PP_Counter(Counter:= , Enable:= , Mode:= , CounterValue=> , Error=> , ErrorID=>);
PP_CounterGet-Value	Значение высокоскоростного счетчика		PP_CounterGetValue(Counter:= , Enable:= , Mode:= , CounterValue=> , Error=> , ErrorID=>);
PP_CounterSet-Value	Запись высокоскоростного счетчика		PP_CounterSetValue(Counter:= , Enable:= , SetValue=> , Выполнено=> , Ошибка=> , ErrorID=>);

2) Связанные переменные

【PP_Counter】

(1) Входные переменные

Вход переменные	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
Counter	Счетчик	COUN-TER_REF	-	-	Высокоскоростной счет-чик, задающий вход для высокоскоростного счета и начальное значение
Execute	Включить	BOOL	ИСТИНА, ЛОЖЬ	FALSE	Нормально разомкнутый разрешающий счетчик
Mode	Режим подсчета	Mode	AB_Mode, Sin-gle_Mode	FALSE	Режим высокоскорост-ного счета: MODE=PP.AB_Mode, это высокоскоростной под-счет фаз AB MODE=PP.Single_Mode - однофазный высокоско-ростной счетчик.

(2) Выходные переменные

Выход переменные	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
CounterValue	Значение счетчика	DINT	Тип данных	0	Значение высокоскоростного счетчика
Error	Флаг ошибки	BOOL	ИСТИНА, ЛОЖЬ	FALSE	
ErrorID	Тип ошибки	UINT	-	0	

【PP_CounterGetValue】

(1) Входные переменные

Вход переменные	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
Counter	Счетчик	COUN-TER_REF	-	-	Высокоскоростной счет-чик, задающий вход для высокоскоростного счета и начальное значение
Execute	Включить	BOOL	ИСТИНА, ЛОЖЬ	FALSE	Срабатывание по нарас-тающему фронту для

					считывания текущего значения высокоскорост-ного счета

(2) Выходные переменные

Выход переменные	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
GetValue	Значение чтения	DINT	Диапазон данных	0	Приведенная стоимость счетчика
Done	Флаг за-верше-ния	BOOL	ИСТИНА, ЛОЖЬ	FALSE	После считывания бит флага равен TRUE
Error	Флаг ошибки	BOOL	ИСТИНА, ЛОЖЬ	FALSE	
ErrorID	Тип ошибки	UINT	-	0	

【PP_CounterSetValue】

(1) Входные переменные

Вход переменные	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
Counter	Счетчик	COUN-TER_REF	-	-	Высокоскоростной счет-чик, задающий вход для высокоскоростного счета и начальное значение
Execute	Включить	BOOL	ИСТИНА, ЛОЖЬ	FALSE	Срабатывание по нарас-тающему фронту, запись значения высокоскорост-ного счета, запись значе-ния SetValue в Counter-Value
SetValue	Запись значения	DINT	Диапазон данных	0	Запись значения настройки высокоско-ростного счета

(2) Выходные переменные

Выход переменные	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
Done	Полный флаг	BOOL	ИСТИНА, ЛОЖЬ	FALSE	После записи бит флага равен TRUE

Error	Флаг ошибки	BOOL	ИСТИНА, ЛОЖЬ	FALSE	
ErrorID	Тип ошибки	UINT	-	0	

Примечание: если отображаемое значение ErrorID равно 2, это связано с тем, что диапазон CounterID не 0-3.

3) Описание функций

(1) Функция высокоскоростного подсчета имеет три функциональных блока: блок функции высокоскоростного подсчета, блок функции высокоскоростного считывания и блок функции высокоскоростного подсчета записи. Высокоскоростной вход серии PMP3xx предназначен для приема сигнала с открытым коллектором (OC).

(2) Счетчик имеет тип данных COUNTER_REF:

Конкретное описание COUNTER_REF выглядит следующим образом:

Переменная	Имя	Тип данных	Эффективный диапазон	Начальное значение	Описание
CounterID	Порт счетчика	INT	0,1,2,3	0	Выбор порта ввода высокоскоростного счетчика
Counter Value	Начальное значение счетчика	DINT	Диапазон данных	0	Установка начального значения счетчика

(3) Функция высокоскоростного счета серии PMP3xx имеет два режима: однофазный режим увеличения и AB-фазный режим соответственно.

a) Инкрементный режим (Mode= Single_Mode)
В этом режиме подсчитывается входной импульсный сигнал, и значение подсчета увеличивается с нарастанием фронта каждого импульсного сигнала.

b) Режим фазы AB (Mode=AB_Mode)
В этом режиме значение высокоскоростного счета увеличивается или уменьшается в соответствии с импульсным сигналом (фаза A и фаза B) с разницей фаз 90°, а режим счета по умолчанию - 4-кратная частота.

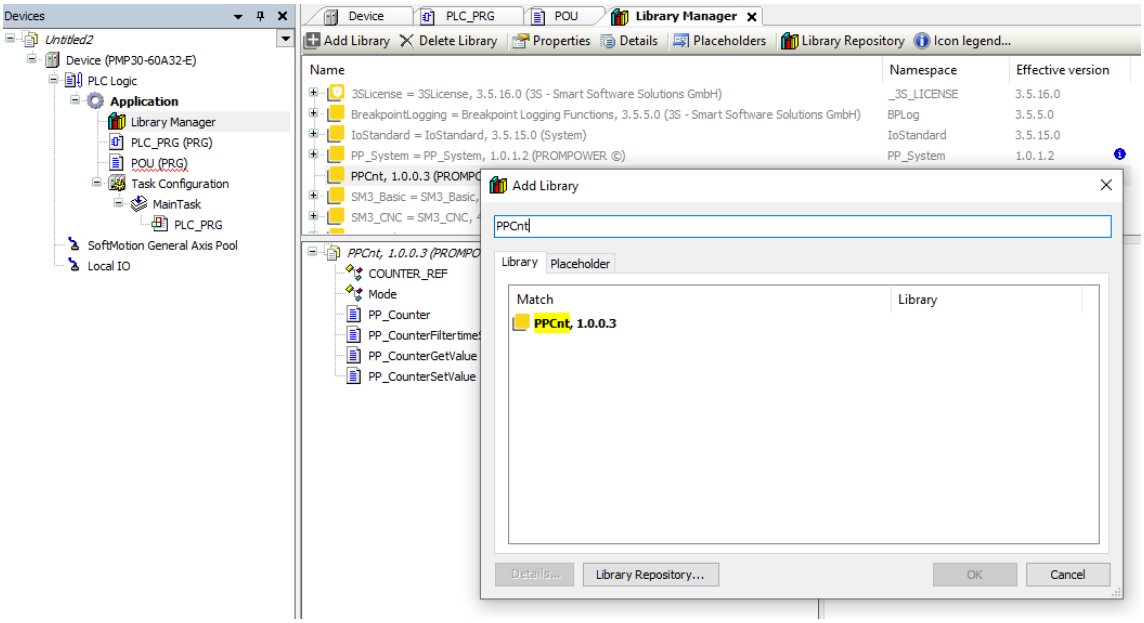
PMP30-60A32-E								
	Однофазный режим				Двухфазный режим			
HSC	0	1	2	3	0	1	2	3
Макс частота	200k	200k	200k	200k	100k	100k	100k	100k
X0	U				A			
X1					B			
X2								
X3		U				A		
X4						B		
X5								
X6			U				A	
X7							B	
X10								
X11				U				A
X12								B
X13								

PMP301-24A8								
	Однофазный режим				Двухфазный режим			
HSC	0	1	2	3	0	1	2	3
Макс частота	80k	80k	80k	80k	50k	50k	50k	50k
Удвоение частоты					2/4	2/4	2/4	2/4
Прерывание счётчика	√	√	√	√	√	√	√	√
X0	U				A			
X1					B			
X2								
X3		U				A		
X4						B		
X5								
X6			U				A	
X7							B	
X10								
X11				U				A

X12								B

4.1.3 Настройка параметров

Добавьте файл библиотеки:
Добавьте "PPCnt" в диспетчер библиотек. После добавления можно использовать функцию высокоскоростного подсчета.



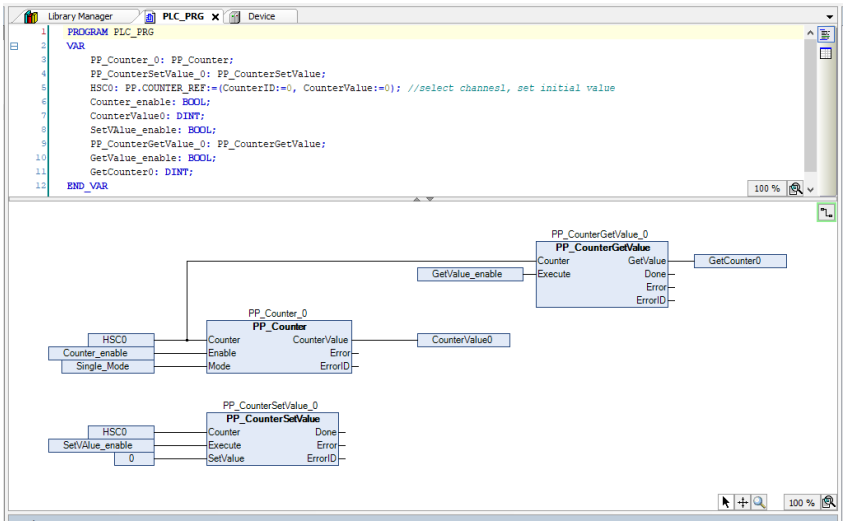
4.1.4 Пример применения

Пример 1: используйте первый канал высокоскоростного счетчика, считайте текущее значение счета в счетчике и измените текущее значение высокоскоростного счета.

Работа программы:

(1) Установите используемую библиотеку в соответствии с шагами, описанными в разделе 4-1-3.
(2) Напишите программу высокоскоростного подсчета.

Программирование: используйте функциональные блоки "PP_Counter", "PP_CounterGetValue", "PP_CounterSetValue". Установите в программе порт высокоскоростного подсчета, режим высокоскоростного подсчета и значение высокоскоростного подсчета.



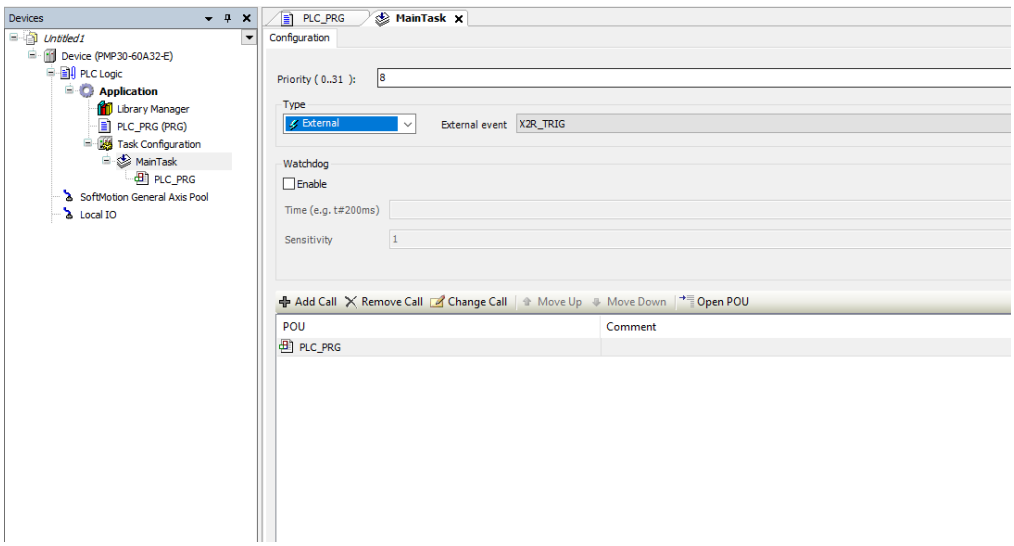
4.2 Внешнее прерывание

4.2.1 Обзор функций

ПЛК серии PMP3xx поддерживает прерывание по X-терминалу, и этот же терминал поддерживает прерывание по нарастающему и спадающему фронту. В Codesys прерывание используется в виде внешних событий в типе задачи. Например, X2R_TRIG означает прерывание по нарастающему фронту X2, а X2F_TRIG - прерывание по спадающему фронту. Количество и тип прерываний, поддерживаемых каждой моделью, см. в параметре "внешнее событие".

4.2.2 Пример применения

Дважды щелкните на "MainTask" и установите для него внешнее событие "external" во всплывающем интерфейсе - внешнее прерывание использует терминал X, и вы также можете установить приоритет событий внешнего прерывания.



4.3 PLC SHELL

4.3.1 Обзор функций

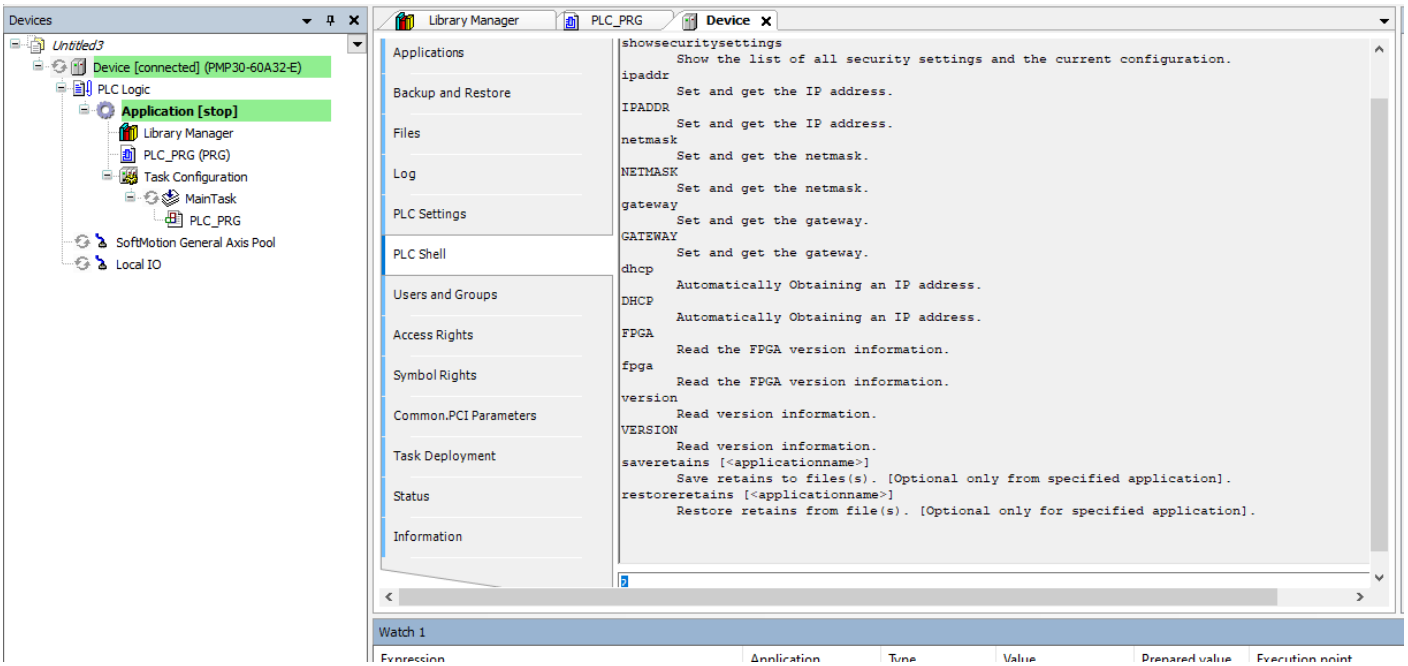
Функция оболочки ПЛК - это текстовый монитор управления, который можно использовать для запроса специфической информации контроллера, ввода заданной команды в окно ввода и получения ответа от контроллера в окне результатов.

4.3.2 Список команд

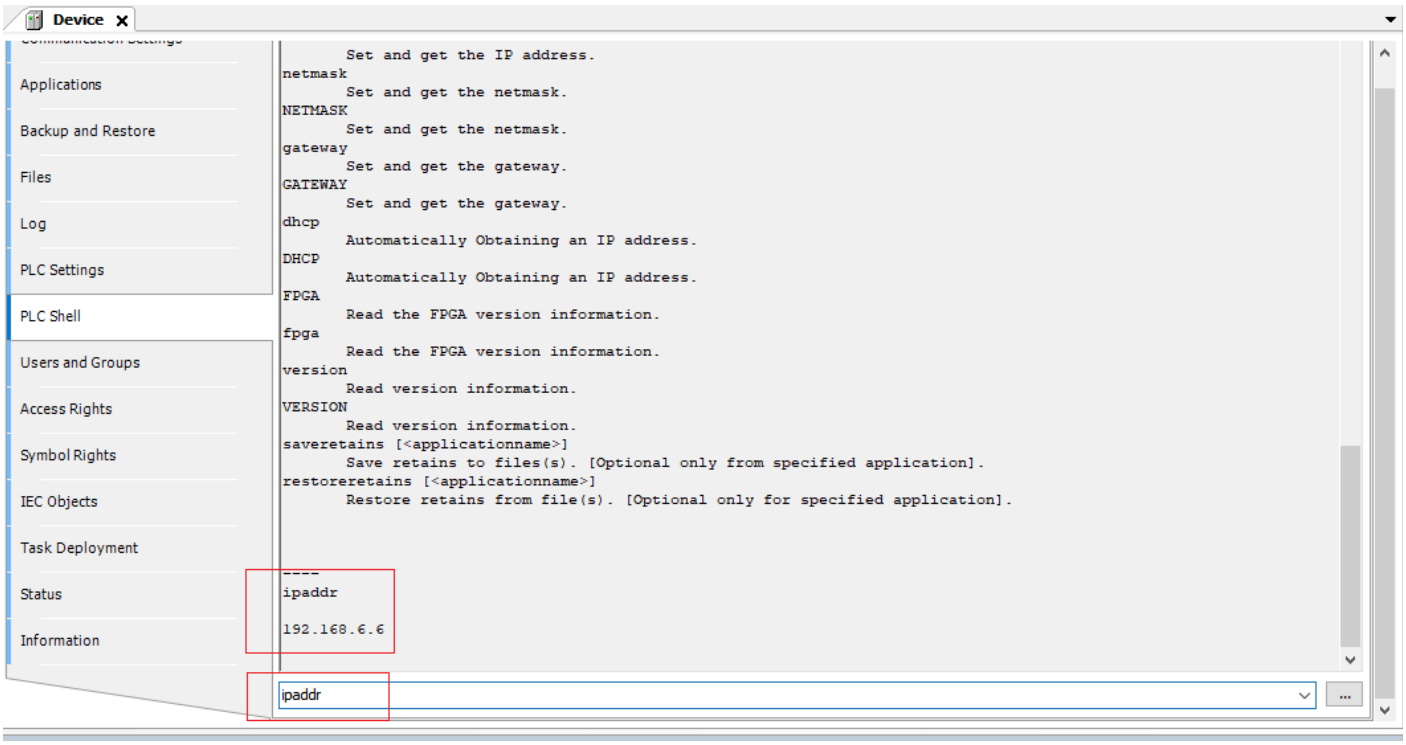
Название команды	Функция
ipaddr / IPADDR	Получение/установка IP-адреса ПЛК
netmask / NETMASK	Получение/установка маски подсети ПЛК
gateway / GATEWAY	Получение/установка шлюза ПЛК
dhcp / DHCP	Установите IP в режим автоматического приобретения
fpga / FPGA	Получение версии FPGA ПЛК
версия / VERSION	Получение версии микропрограммы ПЛК
rtc-get / RTC-GET	Получение текущего времени по Гринвичу
rtc-set / RTC-SET	Установите время UTC

4.3.3 Пример применения

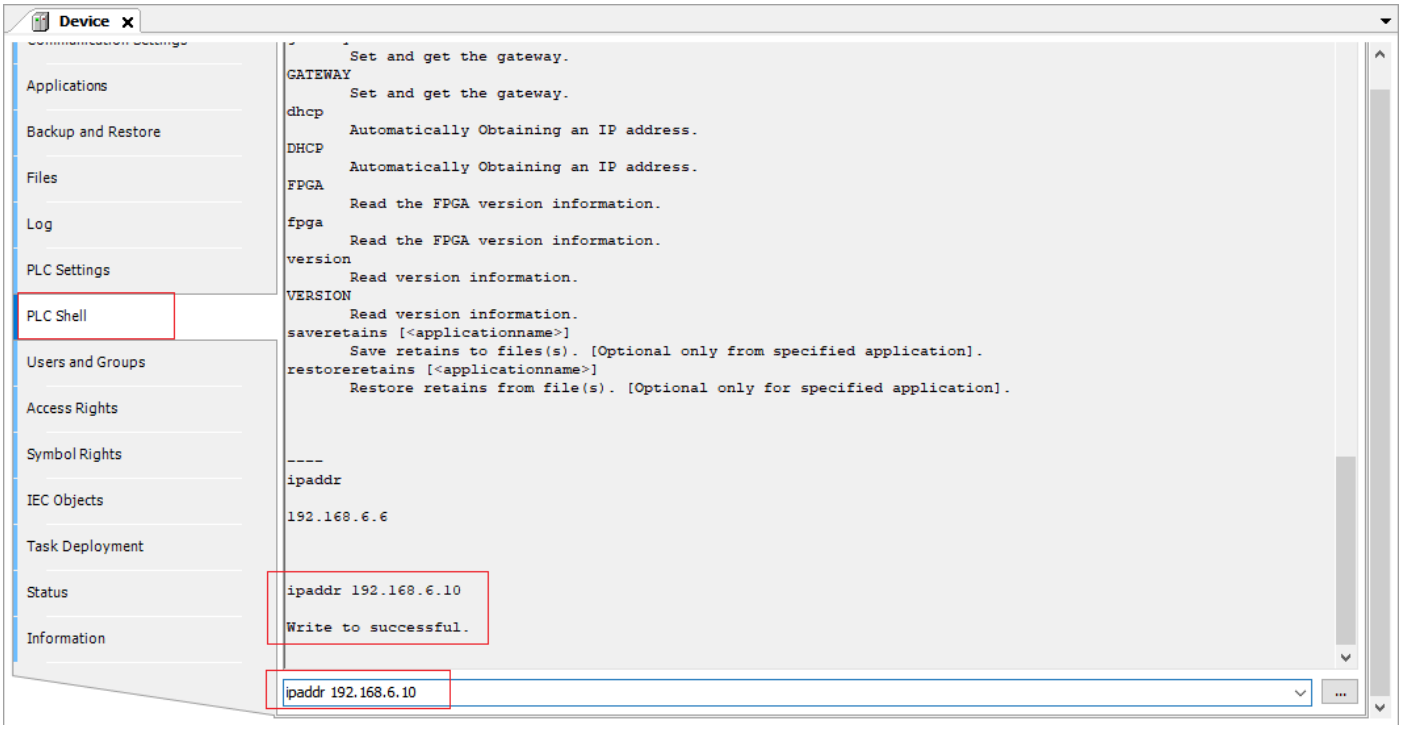
Дважды щелкните на "Device", введите "?" в "PLC Shell", и на экране появятся все функции. Здесь вы можете изменить IP, узнать версию прошивки, установить / считать информацию о часах и т.д.



Например, введите "ipaddr", чтобы получить текущий IP-адрес ПЛК.

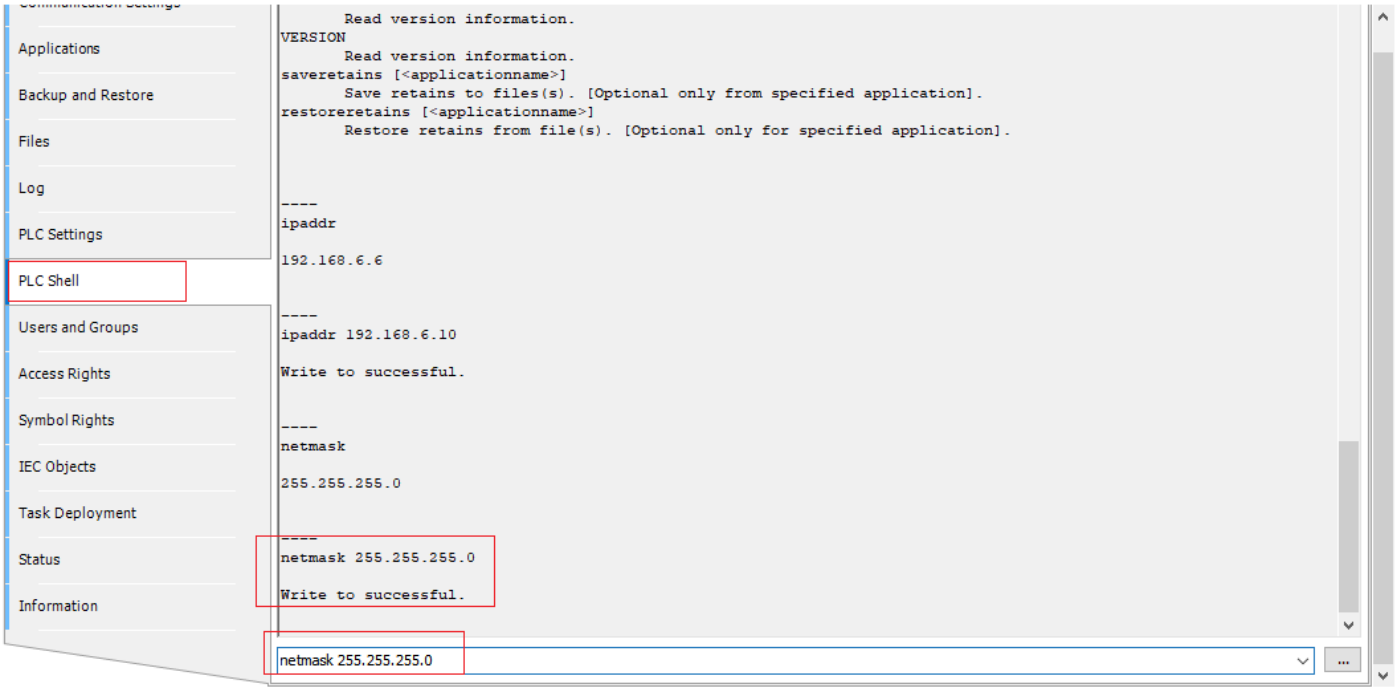


Введите "ipaddr 192.168.61.196", установите IP-адрес ПЛК. Если отображается "write to successful", запись прошла успешно.

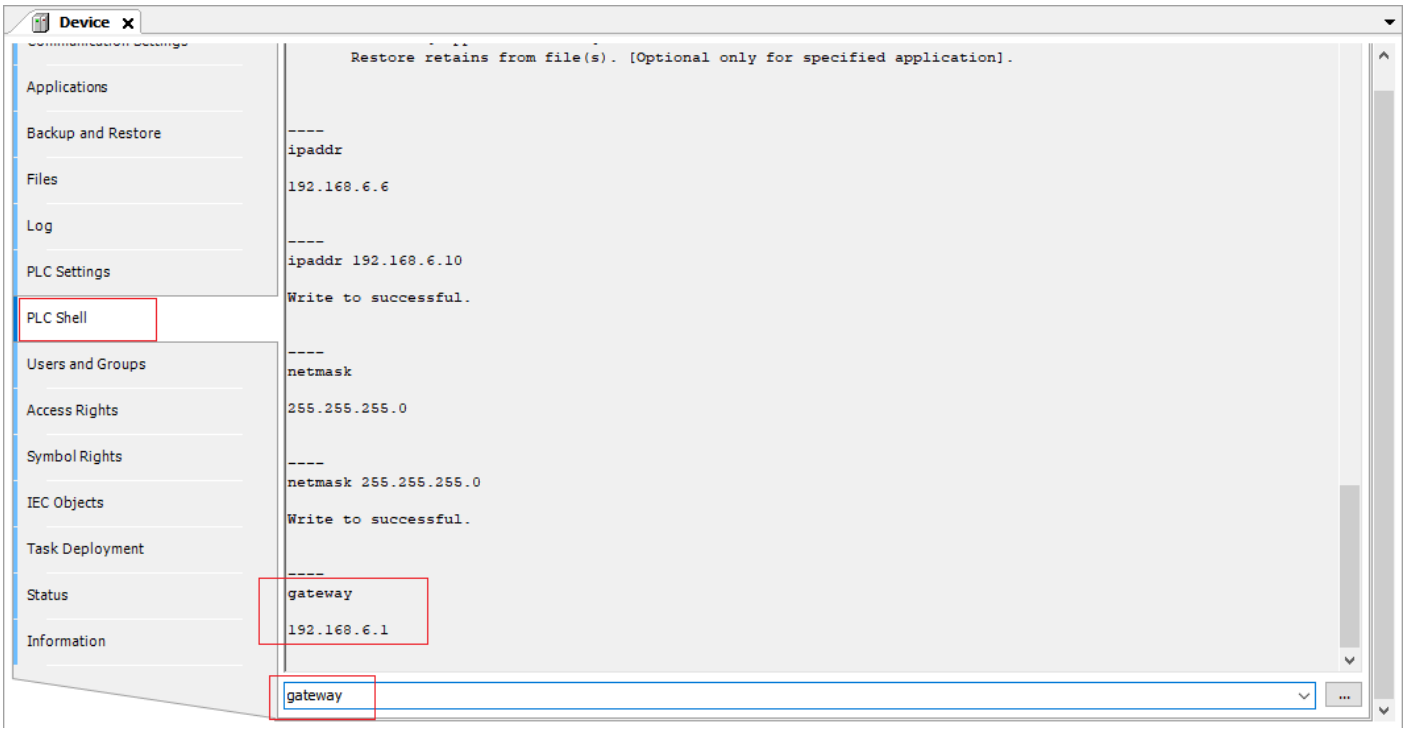


Введя "netmask", можно получить текущую маску подсети ПЛК.

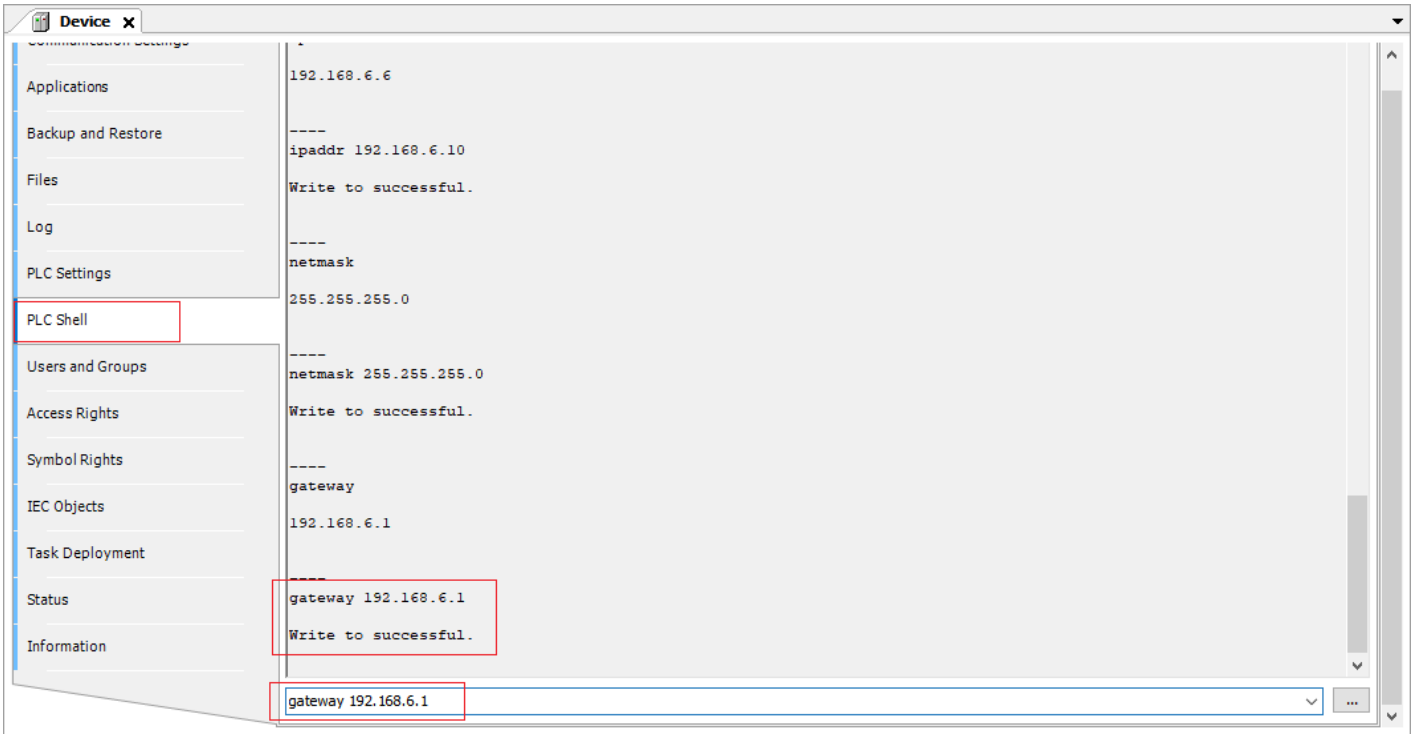
Введите "netmask 255.255.254.0", чтобы установить маску подсети ПЛК. Если отобразится сообщение "write to successful", значит, запись прошла успешно.



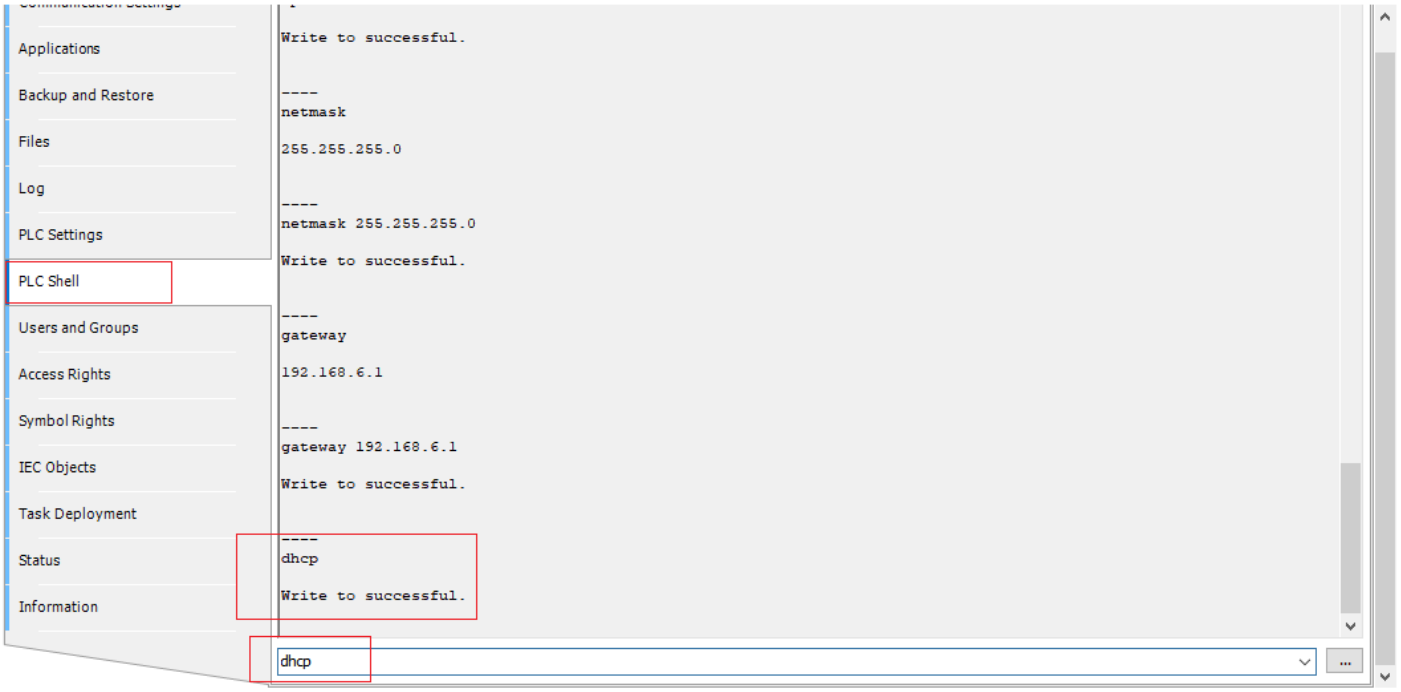
Введите "gateway", чтобы получить текущий шлюз по умолчанию ПЛК.



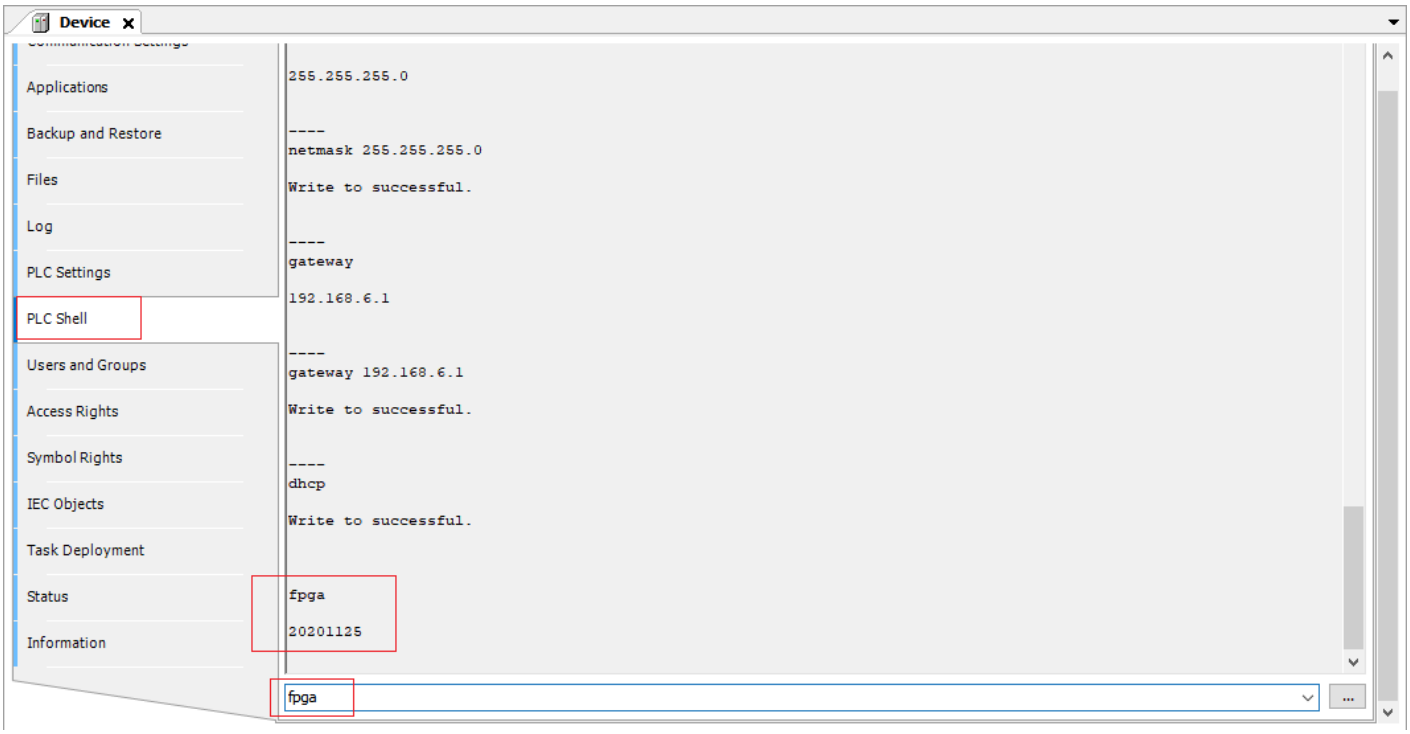
Введите "gateway 192.168.60.1", чтобы установить шлюз ПЛК, если отображается "write to successful", запись прошла успешно.



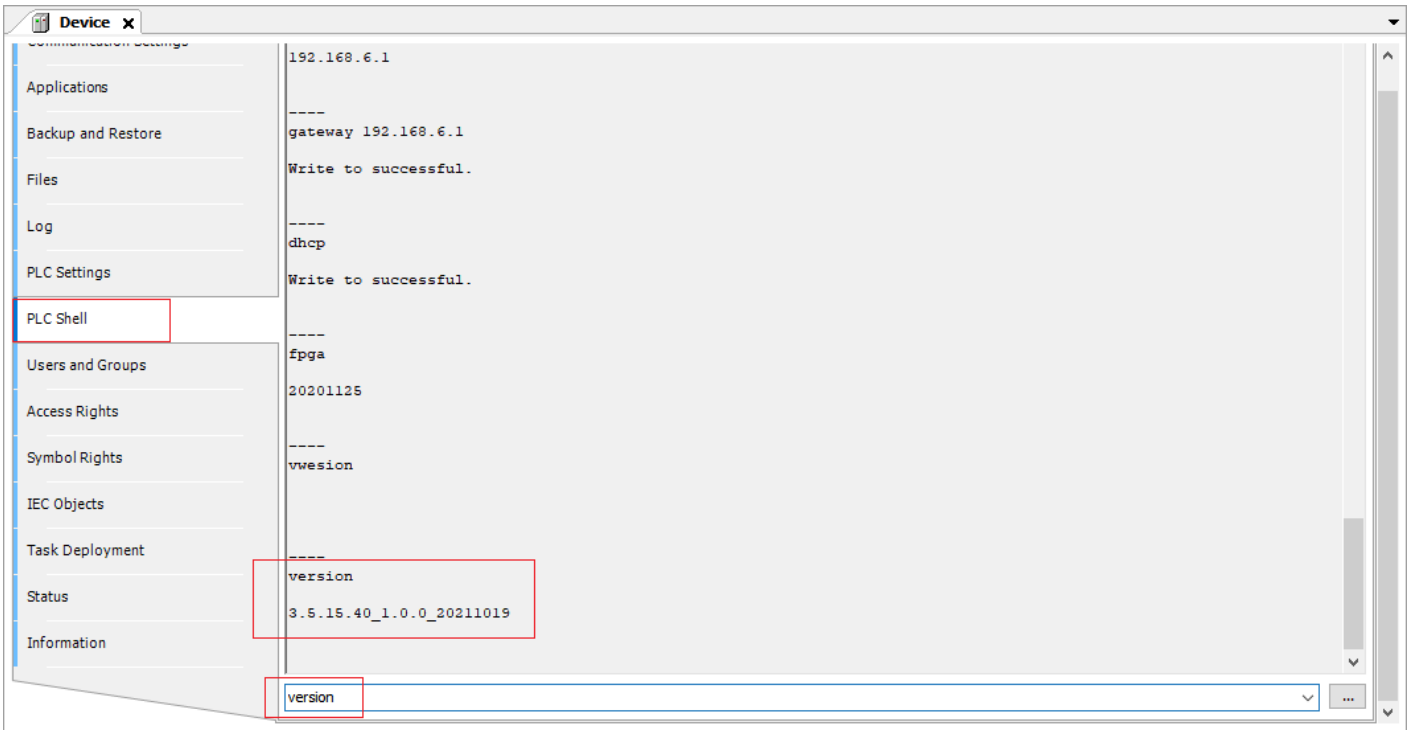
Введите "dhcp" и установите метод получения IP-адресов ПЛК на автоматическое получение. Если отображается сообщение "write to successful", запись выполнена успешно. При автоматическом способе получения IP-адреса необходимо обеспечить хорошее сетевое окружение.



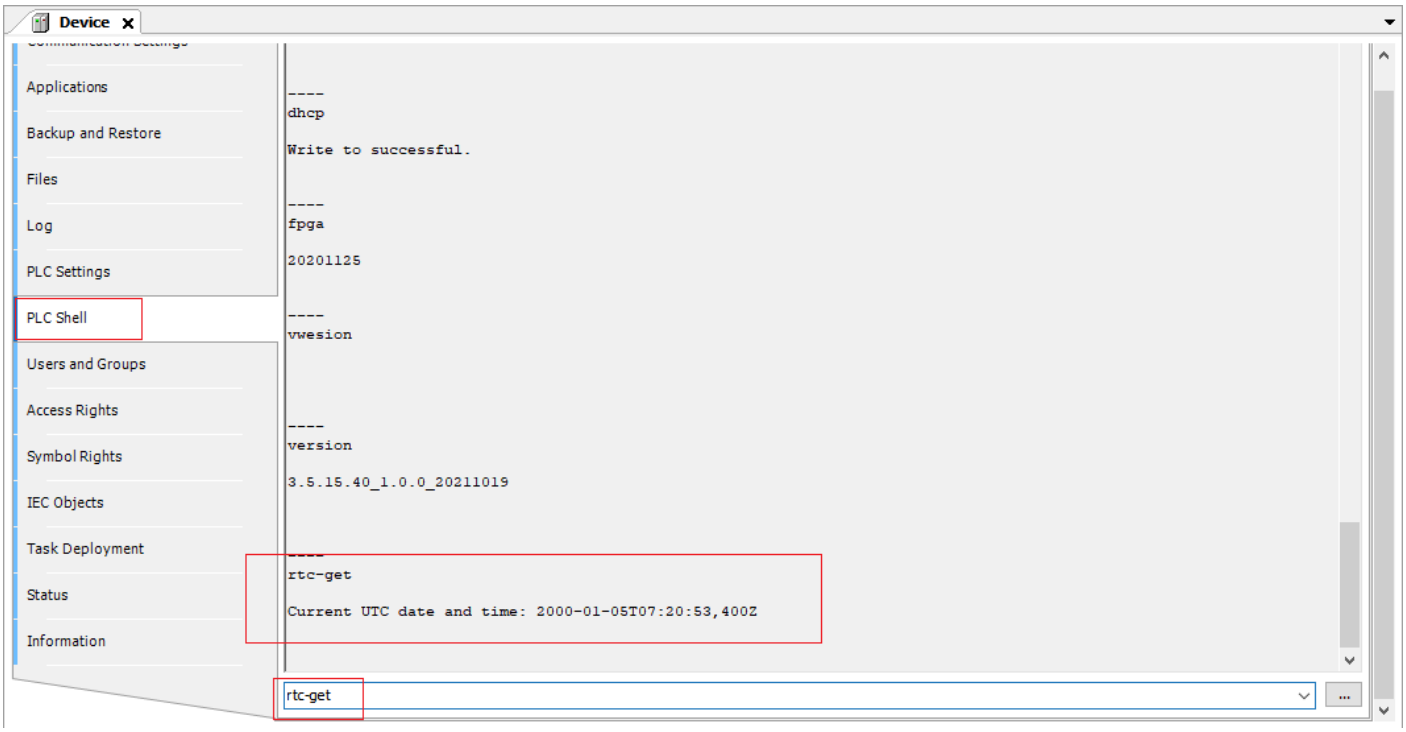
Введите "FPGA", чтобы узнать текущую версию FPGA ПЛК.



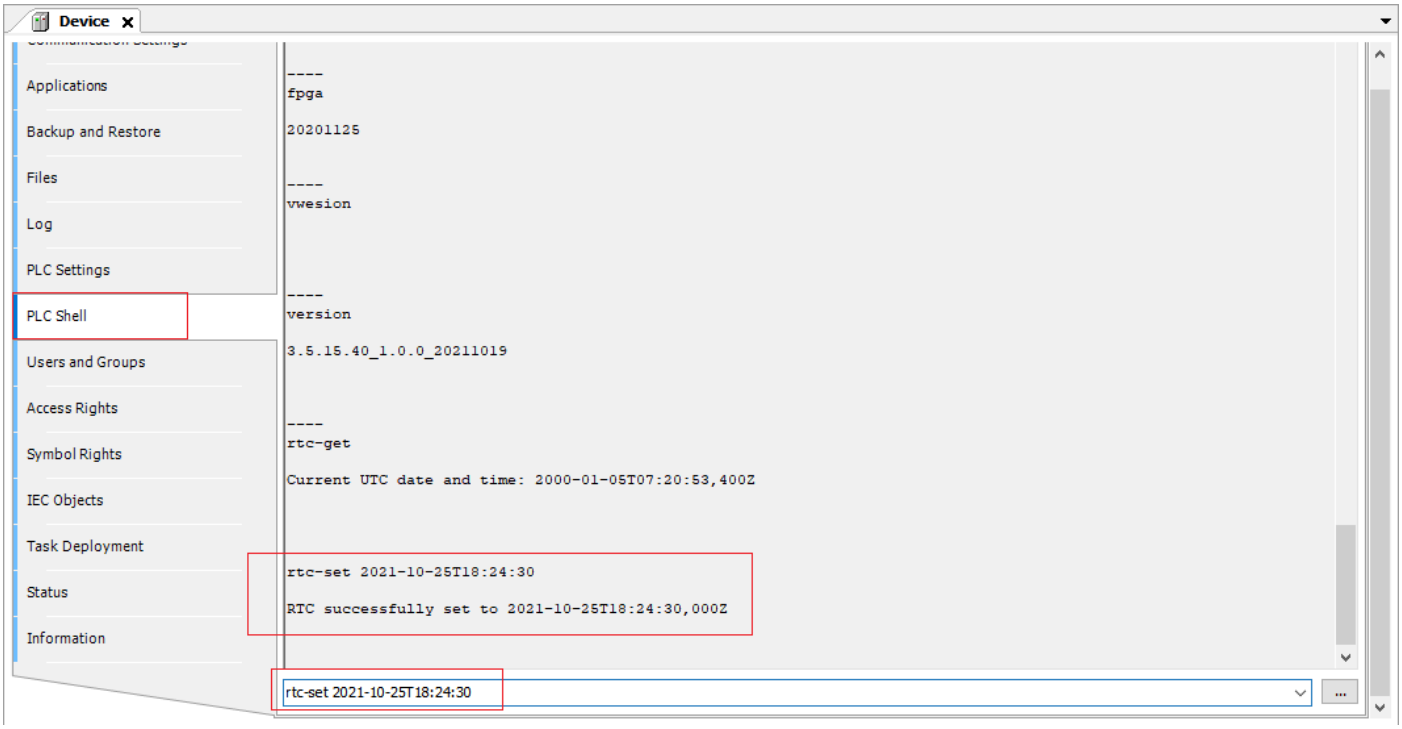
Введите "version", чтобы получить текущую версию прошивки ПЛК.



Введите "rtc-get", чтобы получить текущее время по Гринвичу.



Введите "rtc-set 2021-10-25T18:24:30", чтобы установить время UTC. Если отображается "RTC successfully set to 2021-10-25T18:24:30,000Z", запись прошла успешно. Содержимое дисплея "000Z" не фиксируется.



4.4 Часы

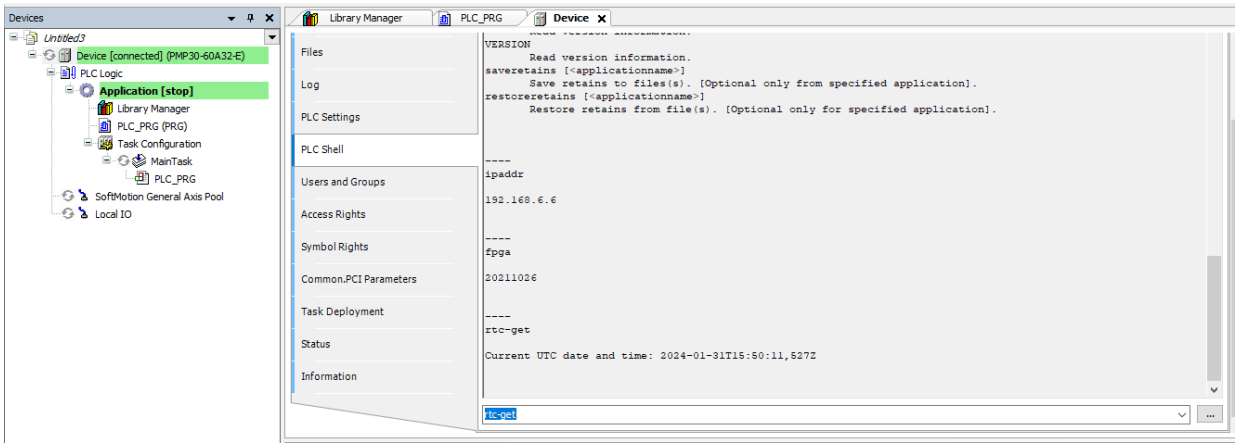
4.4.1 Обзор функций

В ПЛК серии PMP3xx встроен RTC, который используется для записи текущего системного времени. Часы питаются от батареи, что позволяет обеспечить точность времени. В то же время, они также поддерживают возможность изменения времени RTC вручную.

4.4.2 Пример применения

Как получить события:

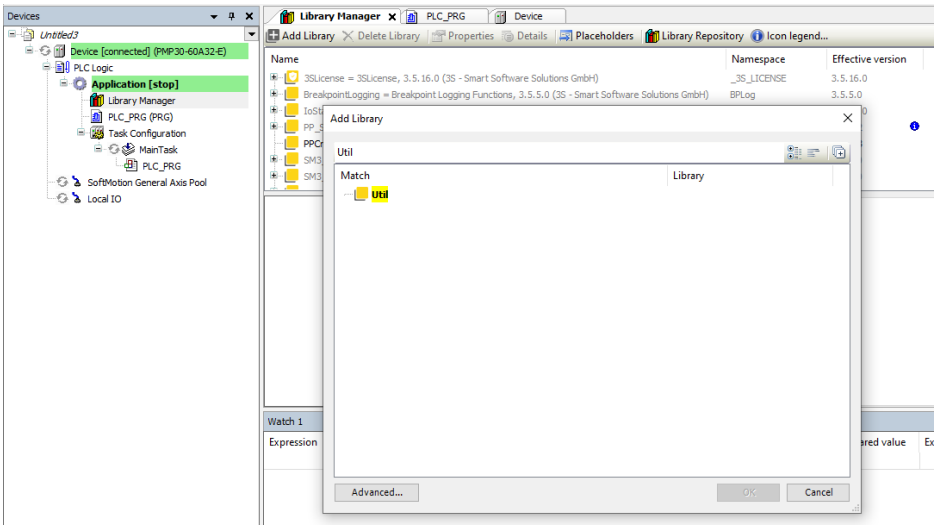
1. Дважды щелкните "Device", введите "rtc-get" в "PLC Shell", чтобы получить текущее время.



2. Использование тактовой инструкции

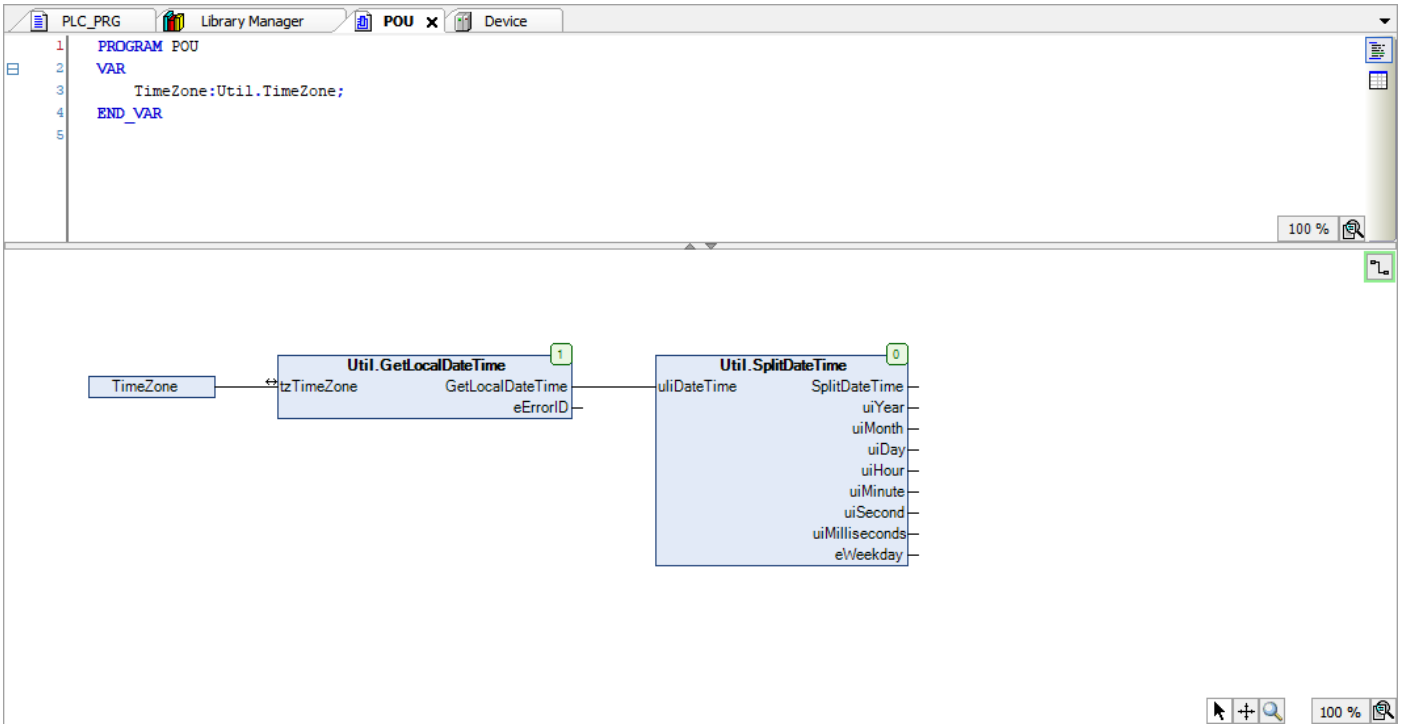
(1) Добавьте файл связанной библиотеки

Добавьте "Util" в "Менеджер библиотек". После добавления вы сможете использовать функцию часов.



(2) Составьте программу часов

Получение текущего времени с помощью функциональных блоков "Util.GetLocalDateTime", "Util.SplitDateTime". В этой библиотеке есть и другие функциональные блоки о часах, которые можно посмотреть в библиотеке "Util".



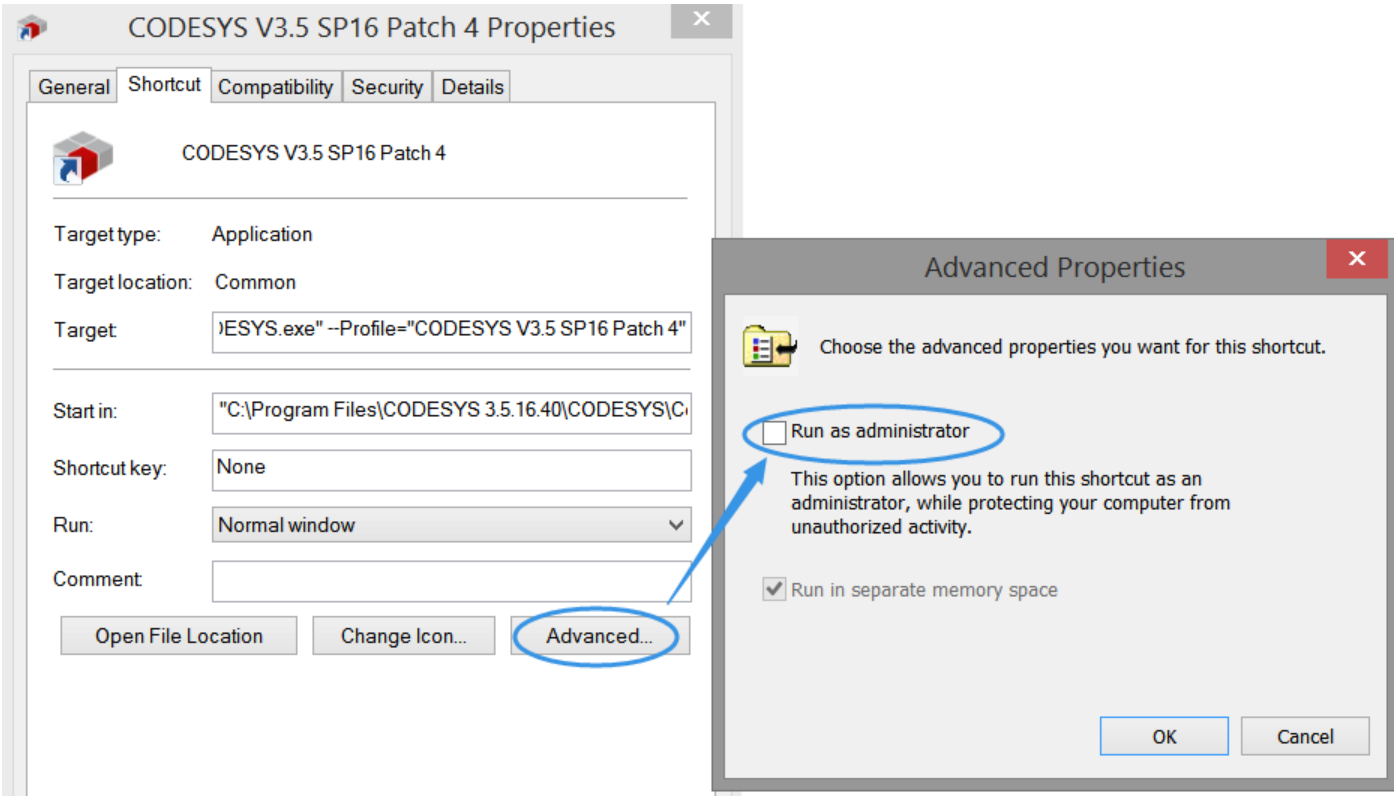
5 Примеры проектов Codesys

5.1 Основные операции программирования

1) Запуск Codesys

(1) Установите права администратора

Щелкните правой кнопкой мыши программу Codesys v3.5, нажмите свойства, выберите "Запуск от имени администратора".



(2) Запуск Codesys



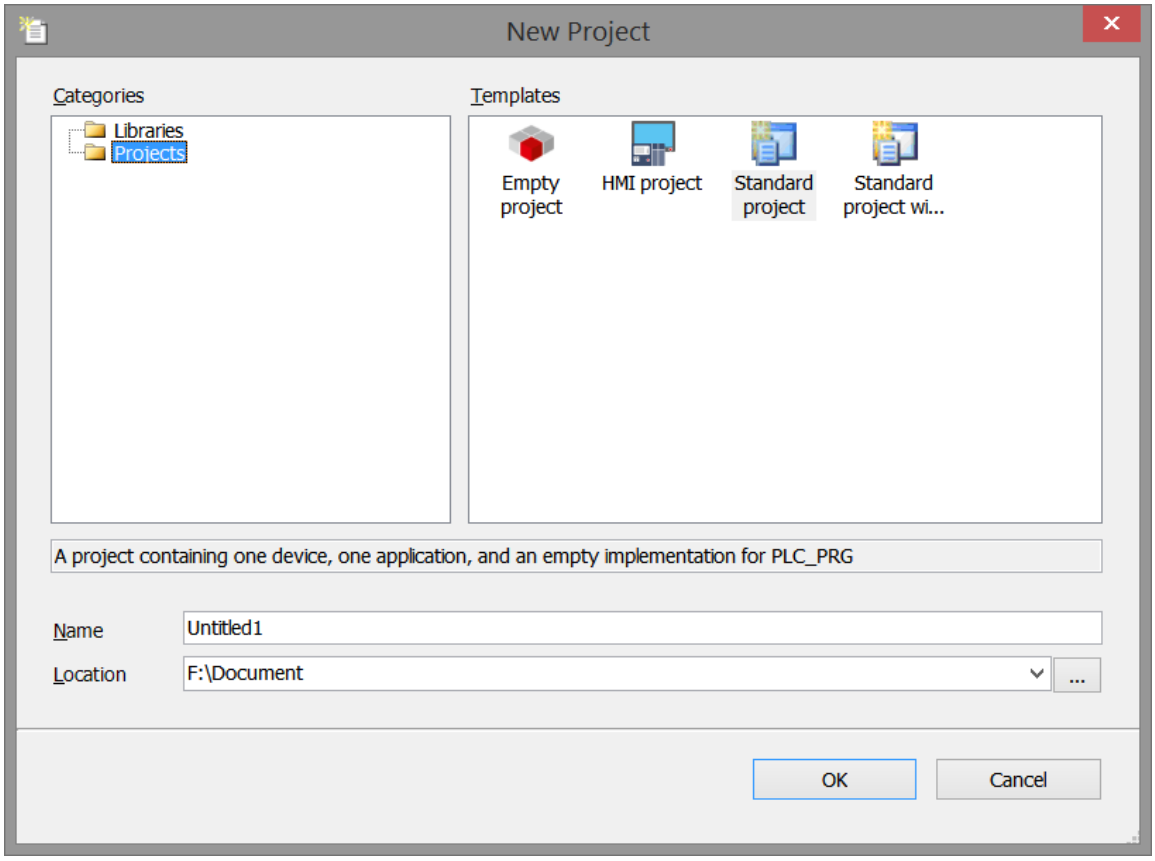
Дважды щелкните программное обеспечение Codesys на рабочем столе.

(3) Создать проект

Выберите новый проект в меню файл, чтобы создать новый проект, как показано на рисунке.

(4) Выберите проект

Пользователь может создать пустой или стандартный проект. Введите имя и путь к файлу проекта, затем нажмите OK.



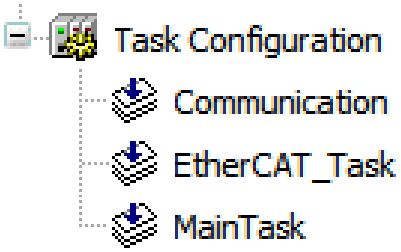
2) Создание файла программы ПЛК

Создание программного файла ПЛК - это не только установление последовательности операций в структуре работы, но и установление режима программирования, и даже сегментация области данных. Перед созданием программного файла необходимо детально разделить структуру операций, определить непрерывные, периодические и запускаемые событиями задачи, а также установить приоритет периодических и запускаемых событиями задач. После создания проекта Codesys автоматически генерируется непрерывная задача по умолчанию, под которой имеется программа по умолчанию и PLC_ PRG.

(1) Создать задачу

Прежде всего, управляйте задачами в "конфигурации задач". В общих проектных приложениях ее можно разделить на основные логические задачи и

задачи коммуникации. Коммуникационные задачи будут иметь более высокий приоритет и более короткое время цикла, поскольку им необходимо обновлять источник данных. Кроме того, если в проекте задействовано управление движением, оно также будет выделено в отдельную задачу и поставлено на самый высокий приоритет, как показано на рисунке:

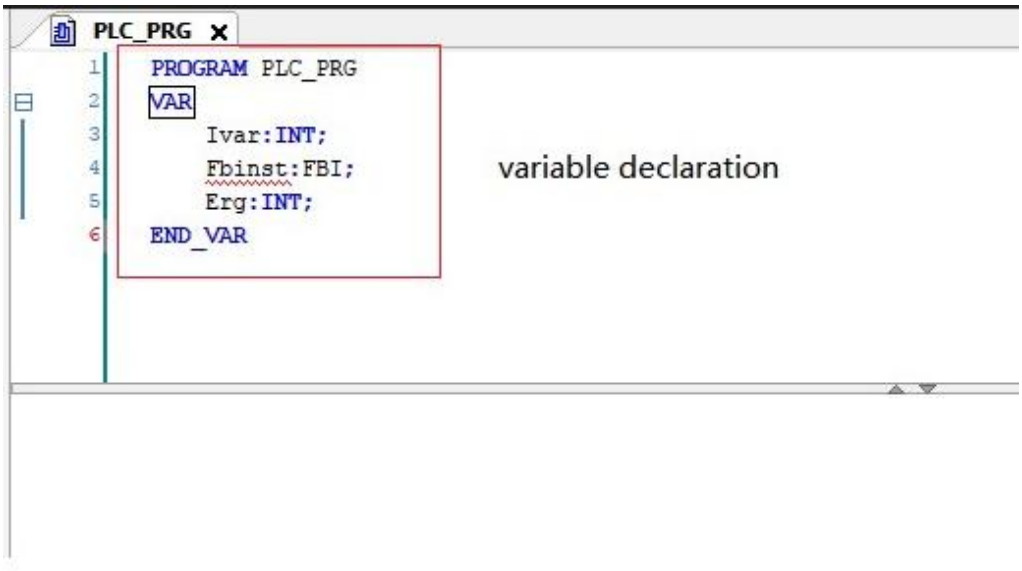


(2) Создать POU

Нажмите "Приложение" > правой кнопкой мыши "Добавить объект", выберите POU.

a) Объявление переменной

В окне устройства по умолчанию используется POU "PLC_PRG". Дважды щелкните "PLC_PRG" в дереве устройств, чтобы автоматически открыть его в редакторе языка ST в центре пользовательского интерфейса Codesys. Редактор языка состоит из части декларации (верхняя часть) и части реализации (нижняя часть), разделенных настраиваемой разделительной линией. В части объявления отображаются: номер строки, тип и имя POU (например, "PROGRAM PLC_PRG"), а также объявление переменной между ключевыми словами "VAR" и "END_VAR", как показано на следующем рисунке. В разделе объявления редактора переместите курсор после VAR и нажмите Enter. Вставьте новую пустую строку, объявите переменную INT "Ivar", переменную INT "Erg", переменную FB1 "Fbinst".



b) Введите программу

Введите следующий код в область редактирования программы под областью объявления:

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

c) Пользовательская функция / функциональный блок

В области объявления переменных вы видите, что вызывается функциональный блок "FB1", но "FB1" - это не стандартный функциональный блок, поэтому вам нужно настроить его. Выберите команду "Добавить объект" в меню "Проект". Выберите "POU" в левой части диалогового окна "Добавить объект", введите имя: FB1 и активируйте опцию "функциональный блок (b)" в опции типа. В качестве языка реализации выберите "структурированный текст (ST)". Нажмите кнопку "открыть", чтобы подтвердить настройку объекта.

Откроется окно редактирования нового функционального блока FB1. Как и в объявлении переменных PLC_PRG, здесь объявляются следующие переменные:

```
FUNCTION_BLOCK FB1
VAR_INPUT
    in:INT;
END_VAR
VAR_OUTPUT
    out:INT;
END_VAR
VAR
    ivar:INT:=2;
END_VAR
```

В разделе реализации редактора программ введите следующее:

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

Функция заключается в том, чтобы добавить "2" к входной переменной "in" и присвоить ее выходу "out".

5.2 Отображение ввода/вывода

При использовании Codesys, когда требуется сопоставление переменных с модулем ввода/вывода программируемого логического контроллера или сетевая связь с внешним оборудованием, можно использовать два метода:

(1) Привяжите параметры, определенные в POU, к переменным

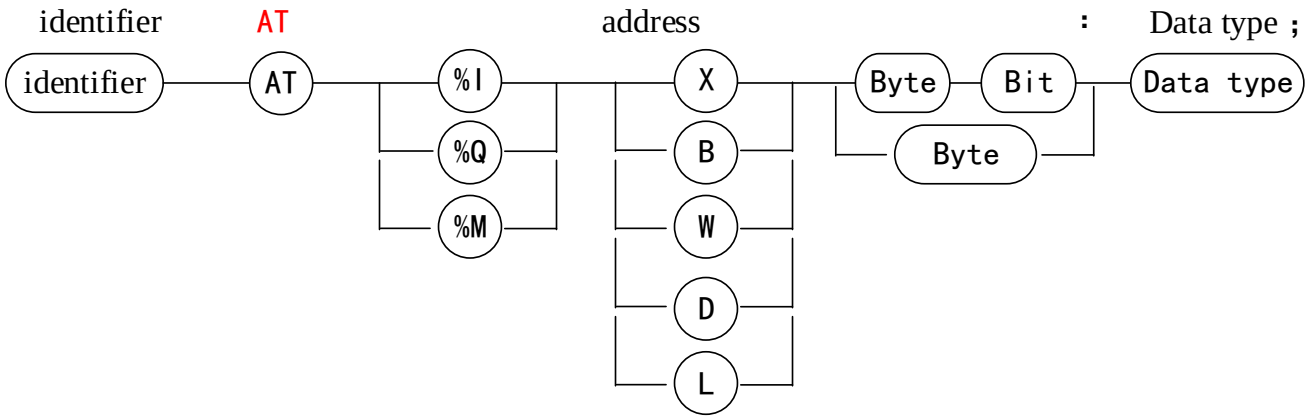
(2) Используйте ключевое слово AT, чтобы напрямую связать переменную с определенным адресом. Прямая переменная должна соответствовать следующим правилам:

AT<адрес>:

<идентификатор> AT<адрес>:<тип данных>{:=<инициализирующее значение>};

{} - необязательная часть.

Начните с символа "%", за которым следует символ префикса позиции и символ префикса размера. Если есть оценка, используйте целое число для представления оценки и символ десятичной точки ".", например %IX0.0, %QW0. Особый формат прямого объявления переменной показан на следующем рисунке:

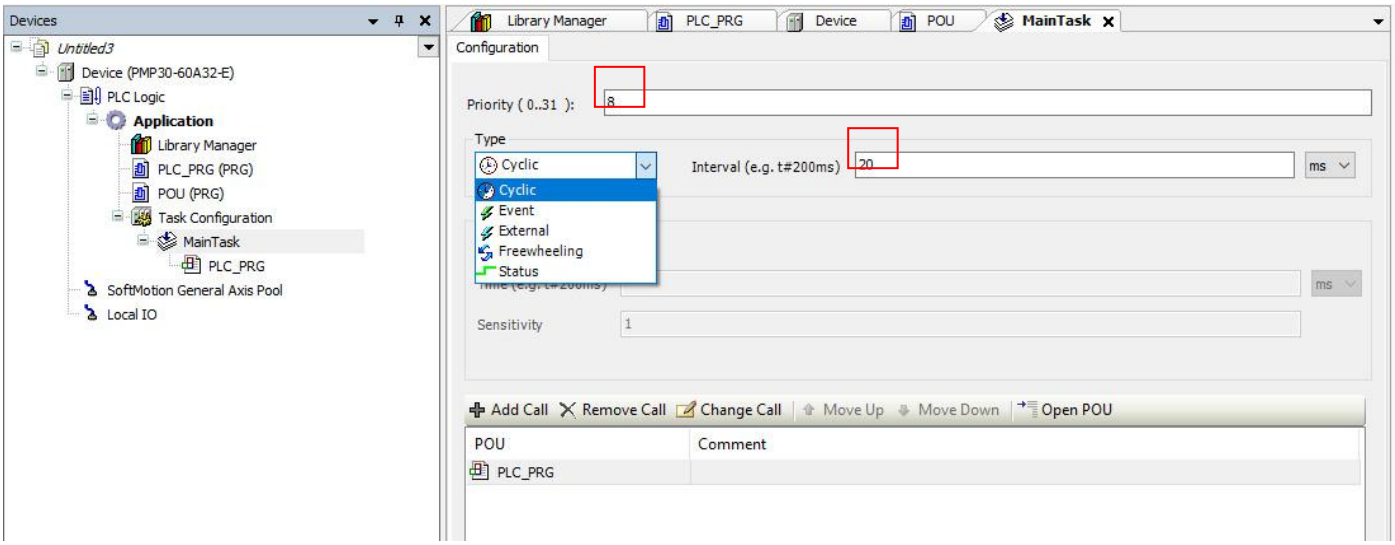


5.3 Конфигурация задачи

1) Обзор

Программа может быть написана на различных языках программирования. Типичная программа состоит из множества взаимосвязанных функциональных блоков, которые могут обмениваться данными друг с другом. Выполнение различных частей программы контролируется "задачами". После настройки "задачи" серия программ или функциональных блоков может выполняться периодически или запускаться по определенному событию для начала выполнения программы. В дереве устройств есть вкладка диспетчера задач, с помощью которой можно объявить конкретную PLC_PRG и управлять выполнением других подпрограмм в проекте. Задача используется для задания атрибутов организационной единицы программы во время выполнения. Это элемент управления выполнением с возможностью вызова. В конфигурации задачи может быть установлено несколько задач, а в задаче может быть вызвано несколько программных организационных единиц. Как только задача установлена, она может управлять выполнением программного цикла или запускать выполнение по определенным событиям.

В конфигурации задачи задается имя, приоритет и тип запуска задачи. Этот тип запуска может быть задан по времени (периодический, случайный) или по времени внутреннего или внешнего триггера задачи, например, по нарастающему фронту глобальной булевой переменной или по определенному событию в системе. Для каждой задачи можно задать серию программ, запускаемых этой задачей. Если эта задача выполняется в текущем цикле, то эти программы будут обработаны в течение одного цикла. Сочетание приоритета и условия определяет последовательность выполнения задачи. Интерфейс установки задачи показан на следующем рисунке:



Поскольку Codesys v3.x имеет следующие атрибуты при конфигурировании задач, программисты должны следовать следующим правилам:

- Максимальное количество заданий цикла - 100.
- Максимальное количество свободно выполняемых заданий - 100.
- Максимальное количество задач, запускаемых событиями, - 100.
- В соответствии с целевой системой, PLC_ PRG может выполняться как свободная программа в любом случае без вставки в конфигурацию задачи.
- Программа обработки и вызова выполняется в соответствии с нисходящей последовательностью в редакторе задач.

2) Приоритет задачи

В Codesys можно установить приоритет задач. Существует 32 уровня (число от 0 до 31, 0 - самый высокий приоритет, 31 - самый низкий). Во время выполнения программы задача с высоким приоритетом имеет приоритет над задачей с низким приоритетом. Задача с высоким приоритетом 0 может прервать выполнение программы с более низким приоритетом в том же ресурсе, так что выполнение программы с низким приоритетом замедлится.

Примечание: при назначении уровня приоритета задачи не назначайте задачи с одинаковым приоритетом. Если перед задачами с таким же приоритетом есть другие представления задач, результаты могут быть неопределенными и непредсказуемыми.

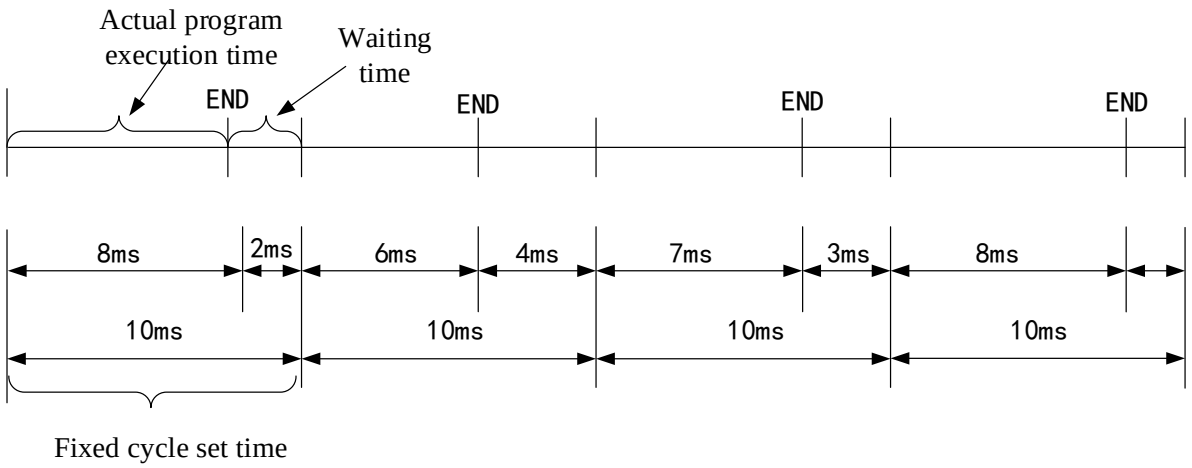
3) Тип выполнения задачи

Для каждой независимой задачи можно редактировать и настраивать тип. В том числе фиксированный цикл, триггер события, внешний триггер, свободный ход и триггер состояния.

(1) Цикл

В зависимости от того, выполняются или нет инструкции, используемые в программе, время обработки программы будет различным, поэтому фактическое время выполнения будет изменяться по-разному в каждом цикле сканирования, и время выполнения будет разным. При использовании режима фиксированного цикла программа может выполняться многократно с определенным временем цикла. Даже если время выполнения программы меняется, можно поддерживать определенный интервал обновления. Здесь мы также рекомендуем предпочесть режим запуска задач с фиксированным циклом.

Например, если задача, соответствующая программе, установлена в режим фиксированного цикла, а время интервала установлено на 10 мс, то схема последовательности выполнения программы показана на следующем рисунке.

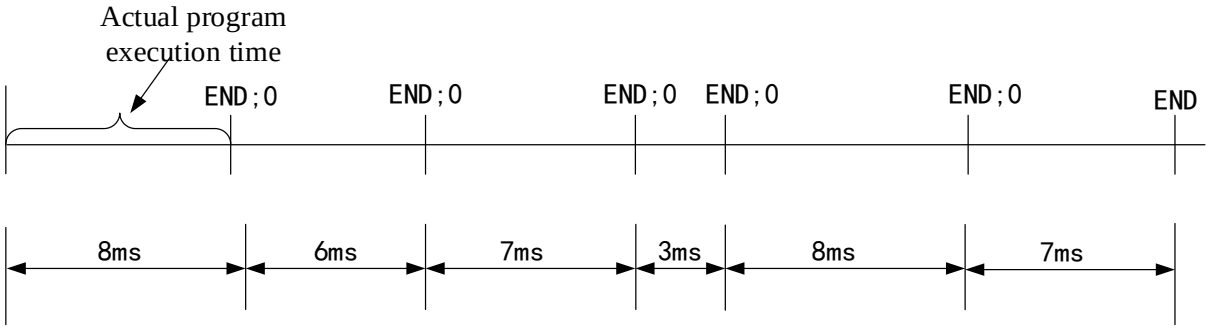


Если фактическое время выполнения программы завершается в течение указанного фиксированного времени установки цикла, то оставшееся время используется как время ожидания. Если в приложении есть задачи с более низким приоритетом, которые не выполняются, оставшееся время ожидания используется для выполнения задач с более низким приоритетом.

(2) Свободное движение

Задание будет обработано, как только программа начнет выполняться. По окончании одного цикла работы задание будет автоматически перезапущено в следующем цикле.

На него не влияет цикл сканирования программы (время интервала). То есть каждый раз следите за тем, чтобы следующий цикл выполнялся после выполнения последней инструкции программы. В противном случае цикл программы не будет завершен.



Поскольку в этом режиме выполнения нет фиксированного времени выполнения задачи, время выполнения каждый раз может быть разным. Поэтому производительность программы в реальном времени не может быть гарантирована, и этот метод редко используется в практических приложениях.

(3) Событие

Если переменная в области событий получает нарастающий фронт, задача запускается.

(4) Статус

Если переменная в области событий равна true, задание начинается.

На следующем рисунке сравниваются триггер события и триггер состояния соответственно. Зеленая сплошная линия - это булева переменная status, выбранная двумя методами триггера. Результаты сравнения приведены в следующей таблице.



Входной сигнал запуска задачи

Режим триггера состояния похож на функцию триггера события, разница в том, что программа будет выполняться до тех пор, пока триггерная переменная триггера состояния истинна, и не будет выполняться, если она ложна. В то время как событийный триггер собирает только эффективный сигнал нарастающего фронта триггерной переменной.

Разные типы задач показали разные ответы в точках выборки 1-4 (фиолетовый). Это конкретное событие должно быть истинным, чтобы выполнить условие задачи, управляемой состоянием. Однако задача, управляемая событиями, требует, чтобы событие изменилось с ложного на истинное. Если частота выборки плана задачи слишком мала, нарастающий фронт события может быть не обнаружен.

Место выполнения	1	2	3	4
Событие	Не выполнять	Выполнить	Выполнить	Выполнить
Статус	Не выполнять	Выполнить	Не выполнять	Не выполнять

(5) Внешнее прерывание

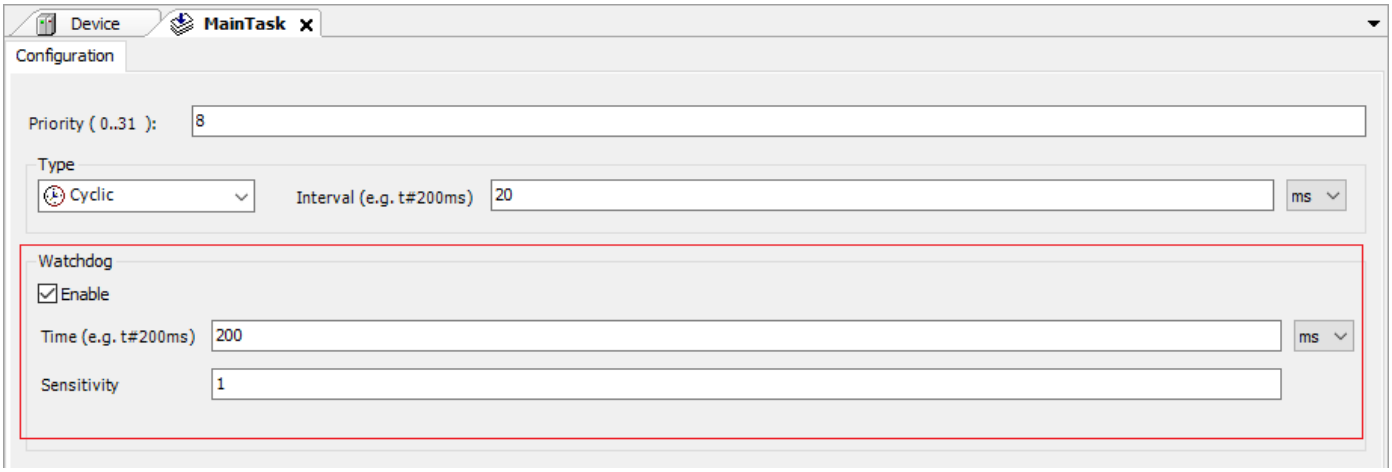
Если переменная в области событий получает нарастающий или спадающий фронт внешнего сигнала прерывания X, задача запускается.

Входная клемма X может использоваться в качестве входа внешнего прерывания. Каждая входная клемма соответствует внешнему прерыванию, и нарастающий или спадающий фронт входного сигнала может вызвать прерывание.

(6) Сторожевой пес

Сторожевой таймер - это устройство синхронизации аппаратного типа контроллера. Его можно включить с помощью "конфигурации задачи" в Codesys. По умолчанию функция сторожевого таймера не используется.

Основная функция сторожевого таймера заключается в отслеживании аномалий во время выполнения программы или сбоя внутренних часов. Если в системе произойдет сбой или программа войдет в мертвый цикл, сторожевой таймер пошлет сигнал сброса в систему или остановит программу, выполняемую ПЛК в данный момент. Мы можем наглядно представить себе, как щенок нуждается в том, чтобы хозяин регулярно его кормил. Если его не покормить в течение указанного времени, он немедленно проголодается. Чтобы настроить сторожевой таймер, необходимо задать два параметра - время и чувствительность. Конфигурация сторожевого пса показана на рисунке.



① Время

Codesys может настроить независимый сторожевой таймер для каждой задачи. Если целевое оборудование поддерживает установку длительного времени сторожевого таймера, можно установить верхний и нижний пределы. По умолчанию единицей времени работы сторожевого таймера являются миллисекунды (мс). Если цикл выполнения программы превышает время срабатывания сторожевого таймера, активируется функция сторожевого таймера и текущая задача прерывается.

② Чувствительность

Чувствительность используется для определения количества исключений сторожевого таймера задачи, которые должны произойти, прежде чем контроллер обнаружит ошибку приложения. По умолчанию используется значение 1.

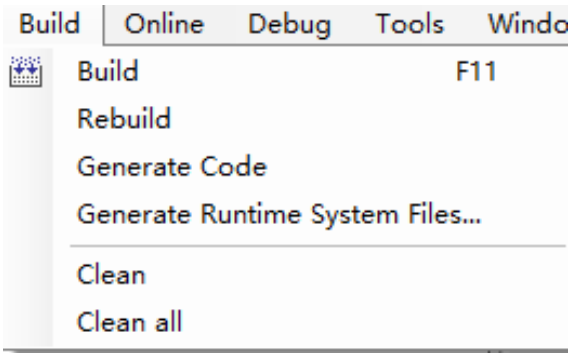
Конечное время срабатывания сторожевого таймера = время × чувствительность. Если фактическое время выполнения программы превышает время срабатывания сторожевого таймера, то сторожевой таймер активируется. Например, если время составляет 10 мс, а чувствительность установлена на 5, время срабатывания сторожевого таймера равно 50 мс. Пока время выполнения задачи превышает 50 мс, сторожевой таймер будет активирован немедленно, и выполнение задачи будет приостановлено.

5.4 Загрузка/чтение программ

5.4.1 Компиляция

После того как программа написана, ее необходимо скомпилировать перед загрузкой. Команда компиляции проверяет синтаксис написанной программы и компилирует только программы, добавленные в задачу. Если созданный POU не добавлен в задачу, команда компиляции не проверяет синтаксис POU.

Команда компиляции не генерирует никакого кода, а только проверяет синтаксис POU. Когда команда входа в устройство выполняется напрямую, система по умолчанию также выполнит команду компиляции (что эквивалентно выполнению команды компиляции сначала вручную), а затем выполнит команду входа в соединение после компиляции и проверки отсутствия синтаксических ошибок. Аналогично, проверка синтаксиса не выполняется для POU, которые не добавлены в задачу во время компиляции. Выполнение команды входа в систему одновременно генерирует код.



(1) Компилировать: компиляция текущего приложения.

(2) Перекомпиляция: если вам нужно перекомпилировать скомпилированное приложение, вы можете перекомпилировать его.

(3) Сгенерировать код: после выполнения этой команды генерируется машинный код текущего приложения. При выполнении команды login сгенерированный код выполняется по умолчанию.

(4) Очистить: удаляет информацию о компиляции текущего приложения. При повторном входе в устройство информацию о компиляции необходимо создать заново.

(5) Очистить все: удаляет всю информацию о компиляции в проекте.

После выполнения команды компиляции видно, что добавленный в задачу " PLC_PRG " отображается синим цветом, а не добавленный в задачу " PLC_PRG " отображается серым цветом. Команда компиляции не проверяет синтаксис се-

рого POU, поскольку программный блок не находится в активном состоянии. Инструкция компиляции проверяет только синтаксис POU в активном состоянии. Если программный блок, который необходимо запустить, отображается серым цветом в процессе компиляции, вы можете проверить, был ли этот программный блок успешно добавлен в задачу, которую необходимо запустить.

После выполнения команды компиляции вы можете увидеть информацию, полученную в результате компиляции, в строке сообщений, где можно узнать, есть ли в скомпилированной программе ошибки или предупреждения, а также количество ошибок и предупреждений. Если ошибки и предупреждения есть, вы можете просмотреть и найти их в окне сообщений и изменить программу в соответствии с полученной информацией.

5.4.2 Вход в систему загрузки

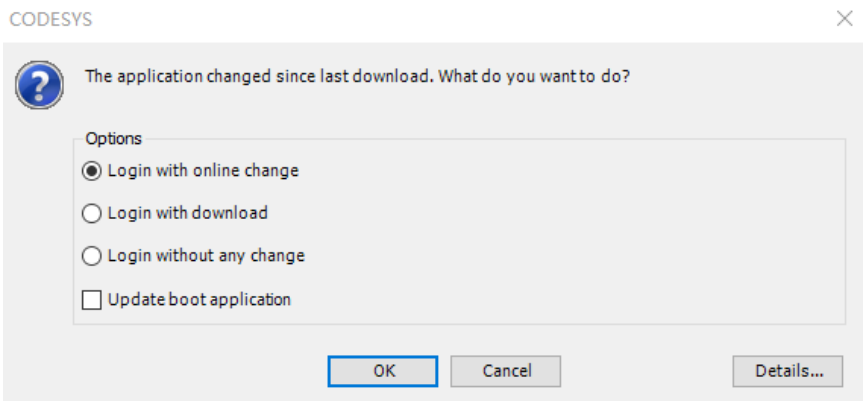
1) Вход в систему

Вход в систему позволяет приложению установить соединение с целевым устройством и перейти в состояние онлайн. Необходимым условием для корректного входа в систему является правильная настройка коммуникационных параметров устройства, а приложение не должно содержать ошибок при компиляции.

Для входа в систему с текущим активным приложением сгенерированный код не должен содержать ошибок, а параметры связи с устройством должны быть настроены правильно. После входа в систему система автоматически выберет загрузку программы.

2) Скачать

Команда Download действует в онлайн-режиме. Она включает в себя компиляцию текущего приложения и генерацию объектного кода. Помимо проверки синтаксиса (обработка компиляции), генерируется объектный код приложения и загружается в ПЛК.



(1) Вход в систему с помощью онлайн-изменений

Когда пользователь выбирает эту опцию, измененная часть проекта загружается в контроллер. Чтобы предотвратить переход контроллера в состояние останова, используйте операцию "вход в систему с изменением в режиме онлайн".

Примечание:

1) Пользователь выполнил хотя бы одну полную загрузку.
2) Данные указателя будут обновлять значение последнего цикла. Если тип данных исходной переменной изменен, точность данных не может быть обеспечена. В это время данные указателя должны быть перераспределены.

(2) Вход в систему с загрузкой

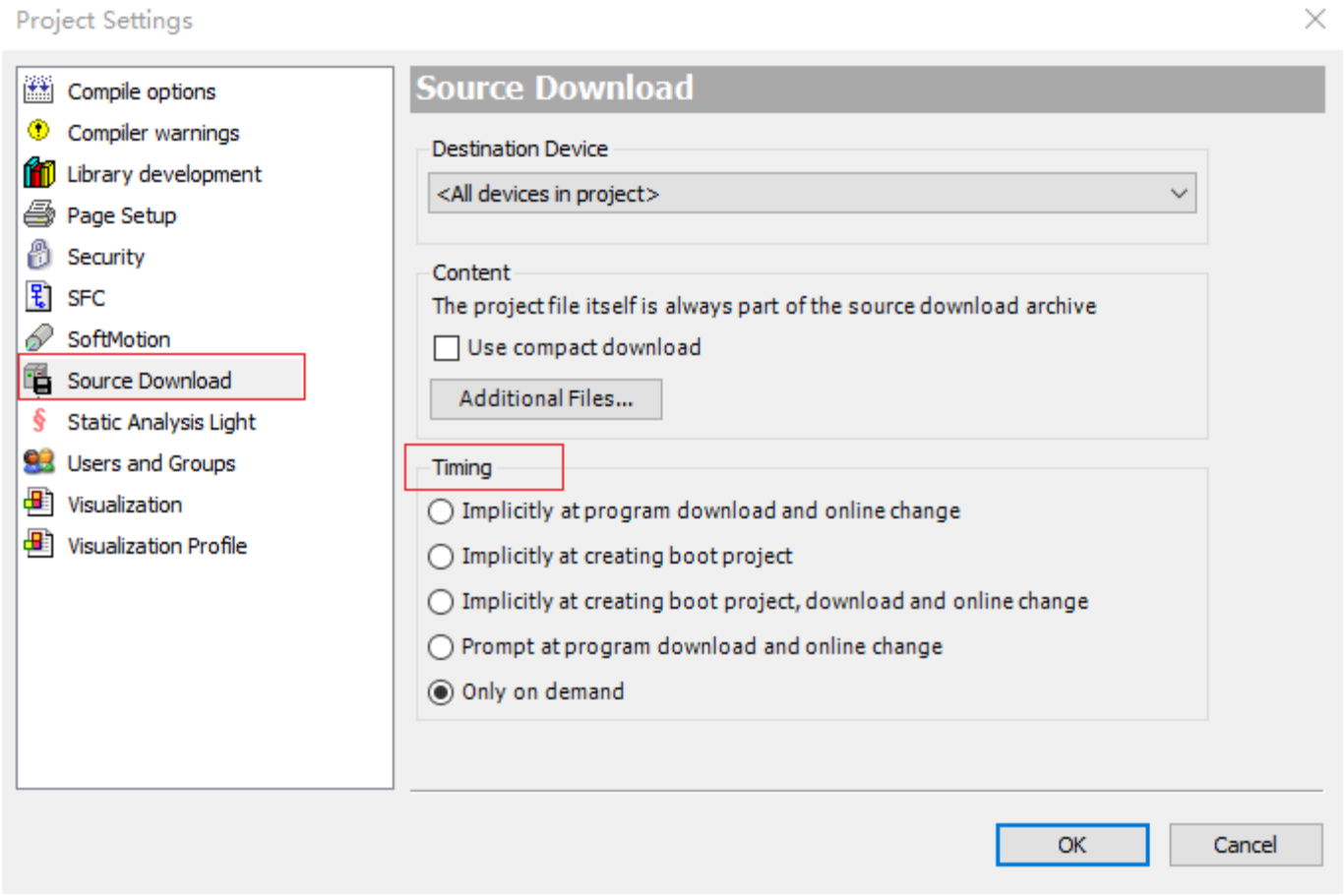
После выбора входа и загрузки перезагрузите весь проект в контроллер. Самое большое отличие от "входа в систему с изменением онлайн" заключается в том, что после загрузки контроллер останется в режиме останова и будет ждать, пока пользователь отправит команду run или перезапустит контроллер, чтобы запустить программу.

(3) Вход в систему без изменений

При входе в систему вы не изменяете программу, которая была загружена в контроллер в последний раз.

5.4.3 Загрузка исходного кода

Codesys не загружает исходный код автоматически по умолчанию для защиты исходного кода программиста. Если вам нужно загрузить исходный код, вам нужно установить его вручную. Нажмите "онлайн" > "загрузка исходного кода на подключенное устройство". Вы также можете установить этот атрибут в меню "Проект" > "Настройки проекта" > "Загрузка исходного кода" > "Тайминг".



5.4.4 Программа для чтения

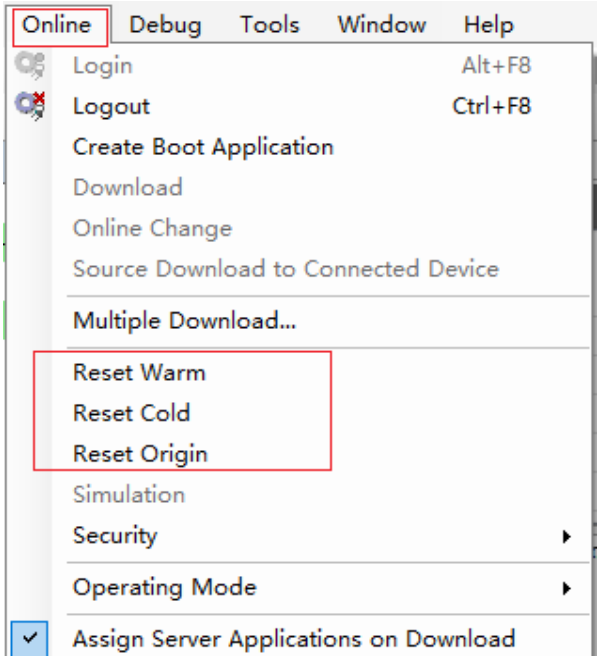
Откройте диалоговое окно выбора устройства в меню "Файл > Загрузка источника". Пользователь выбирает сетевой путь, подключенный к ПЛК, и нажимает "OK". Если архивный файл уже существует по выбранному пользователем пути, будет выдан запрос о необходимости его перезаписи.

Следует отметить, что перед считыванием программы необходимо убедиться, что в предыдущем процессе загрузки была выполнена "загрузка источника на подключенное устройство". В противном случае данные в контроллере не смогут быть прочитаны.

5.5 Отладка программы

5.5.1 Сброс

Существует три способа сброса программы Codesys, которые можно выбрать в меню "онлайн".



1. Горячий сброс

После горячего сброса, за исключением переменных типа удержания (переменные PERSISTENT и RETAIN), другие применяемые в данный момент переменные инициализируются заново. Если установлены переменные с начальными значениями, то после горячего сброса значения этих переменных будут восстановлены до заданных начальных значений. В противном случае переменные будут установлены в стандартное начальное значение 0.

2. Холодный сброс

В отличие от "горячего сброса", команда "холодного сброса" не только устанавливает значение общей переменной в начальное значение активного в данный момент приложения, но и устанавливает значение переменной удержания (переменная RETAIN) в начальное значение 0. Постоянная переменная остается неизменной.

3. Сброс в исходное состояние

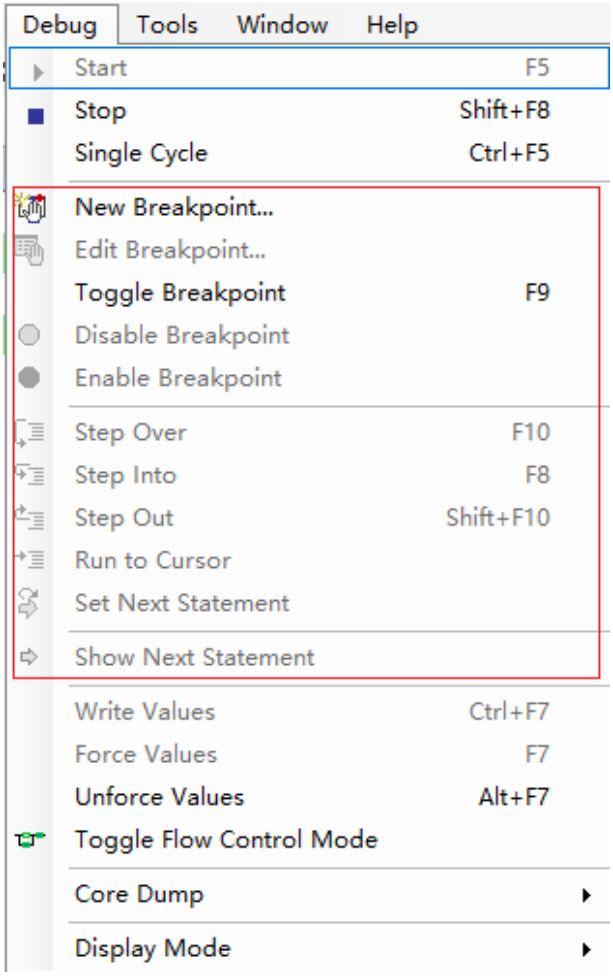
Когда программируемое устройство выбрано в дереве устройств, эта команда может быть использована как в автономном, так и в онлайн-режиме. Использование этой команды приведет к сбросу устройства в исходное состояние, то есть

все приложения, проекты загрузки и оставшиеся переменные в устройстве будут очищены.

Поскольку вся информация о проекте была удалена, необходимо снова "загрузить" программу и "запустить" ее после повторного входа в систему.

5.5.2 Отладка программы

Вид меню "Отладка" в Codesys выглядит так, как показано на рисунке. Основные операции включают установку точки останова и одиночный цикл.



1. Точка останова

Точка останова - это функция остановки обработки в программе. Когда программа останавливается, сотрудники отдела исследований и разработок программы могут наблюдать за содержимым ее переменных, вводом/выводом и другими связанными переменными, когда программа достигает позиции точки останова, что помогает глубоко понять механизм работы программы, найти и устранить ошибки программы.

Точки останова можно устанавливать на всех языках программирования в Codesys. В текстовом редакторе языка ST точка останова устанавливается на

строке. В редакторе FBD и LD точка останова устанавливается на номере сети. В то время как в SFC она устанавливается на шаге.

2. Шаг

После установки точки останова программа может выполняться шаг за шагом. Эта функция позволяет выполнять программу шаг за шагом, что удобно для программистов при отладке и проверке логических ошибок в программе.

(1) Переступите порог

Эта команда выполнит текущую инструкцию в программе и остановится после ее выполнения. Когда POU не вызывается, команды step over и step into имеют одинаковый эффект. Однако, если POU вызван, step over не будет входить в POU. Вместо этого вызов POU рассматривается как полный шаг и выполняется за один раз. Шаг в будет входить в POU. Если используется язык SFC, шаг over будет рассматривать действие как полный шаг и выполнять его за один раз. Если вы хотите перейти к вызываемому POU для одношаговой отладки, вы должны использовать step into.

(2) Шаг в

После выполнения текущая позиция инструкции обозначается желтой стрелкой. Если текущая инструкция не вызывает POU, использование этой команды имеет тот же эффект, что и использование команды skip.

(3) Выходите

Когда в POU выполняется одношаговая отладка, оставшиеся инструкции POU будут выполнены за один раз с помощью step out, а затем будет возвращена следующая инструкция в месте вызова POU. Таким образом, если POU вызывается вниз слой за слоем, step out будет возвращаться вверх слой за слоем, по одному слою за раз. Если в программе нет ни одного вызова POU, то step out не сможет вернуться на верхний слой и вернется в начало программы.

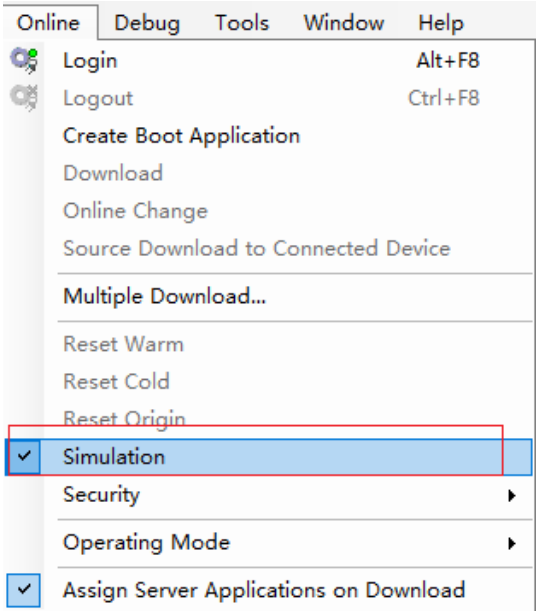
(4) Одиночный цикл

Выберите "одиночный цикл" в "отладке", чтобы программа выполнялась пошагово. То есть нажмите один раз для запуска, и программа остановится после одного цикла и будет ждать следующей команды запуска.

5.6 Моделирование

Автономное моделирование

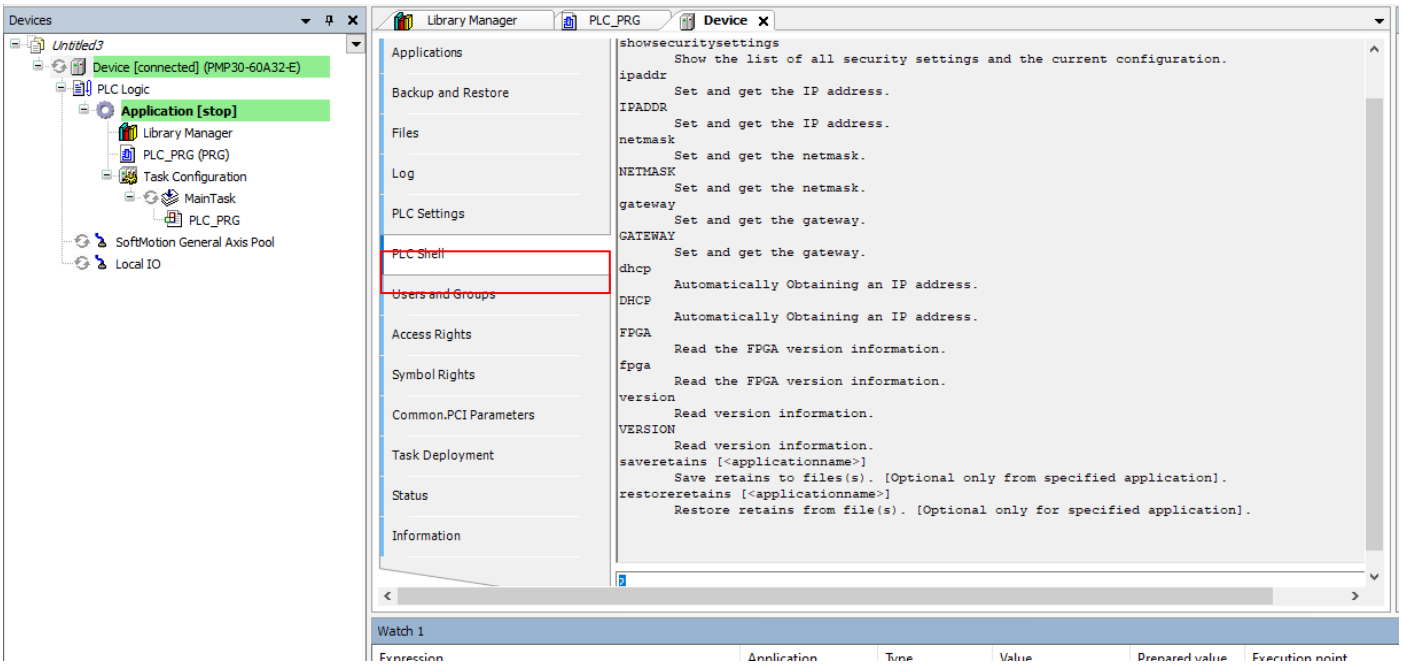
Выберите опцию "эмуляция" в меню "онлайн", чтобы войти в процесс выполнения программы в режиме эмуляции. Убедившись, что опция "эмуляция" отмечена, скомпилируйте программу и войдите в режим эмуляции после отсутствия ошибок.



5.7 Функция сценария ПЛК

Сценарий ПЛК - это текстовый монитор управления (терминал). Команда с конкретной информацией, полученной от контроллера, вводится в строку ввода и передается контроллеру в виде строки. Возвращается результат отображения соответствующей строки в окне просмотра. Эта функция используется для диагностики и отладки.

Дважды щелкните мышью, чтобы выбрать "устройство", найдите "оболочку ПЛК" в правом представлении и введите соответствующую команду в поле ввода команд ниже. Введите ?, Нажмите enter, чтобы отобразить все команды, поддерживаемые контроллером. См. раздел 4-3.



Примечание: чтобы использовать функцию сценария, необходимо войти в систему ПЛК перед использованием соответствующей команды.

6 Технология промышленной полевой шины

6.1 Связь по протоколу MODBUS

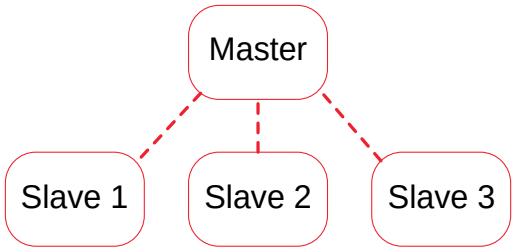
6.1.1 Обзор MODBUS

Корпус программируемого контроллера серии PMP3xx поддерживает обмен данными по протоколу Modbus в виде ведущего и ведомого устройства.

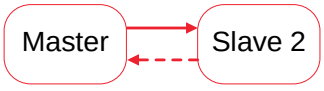
Форма мастер-станции: когда программируемый контроллер используется в качестве устройства мастер-станции, он может обмениваться данными с другими ведомыми устройствами по протоколу Modbus. Обмен данными с другим оборудованием. Пример: ПЛК серии Prompower PMP3xx может управлять частотным преобразователем посредством связи.

Форма ведомой станции: когда программируемый контроллер используется в качестве оборудования ведомой станции, он может только отвечать на требования других ведущих станций.

Концепция "ведущий-ведомый": в сети RS485 в определенный момент времени может быть один ведущий и несколько ведомых (как показано на рисунке ниже). Ведущая станция может читать и записывать данные на любую из ведомых станций, а ведомые станции не могут напрямую обмениваться данными. Для чтения и записи на одну из ведомых станций мастеру необходимо написать программу связи. Ведомой станции не нужно писать программу связи, она должна только отвечать на чтение и запись ведущей станции (режим подключения: все 485 + соединены вместе, а все 485 - соединены вместе).



В сети RS232 (как показано на рисунке ниже) возможен только обмен данными один на один, и одновременно существует только один ведущий и один ведомый.

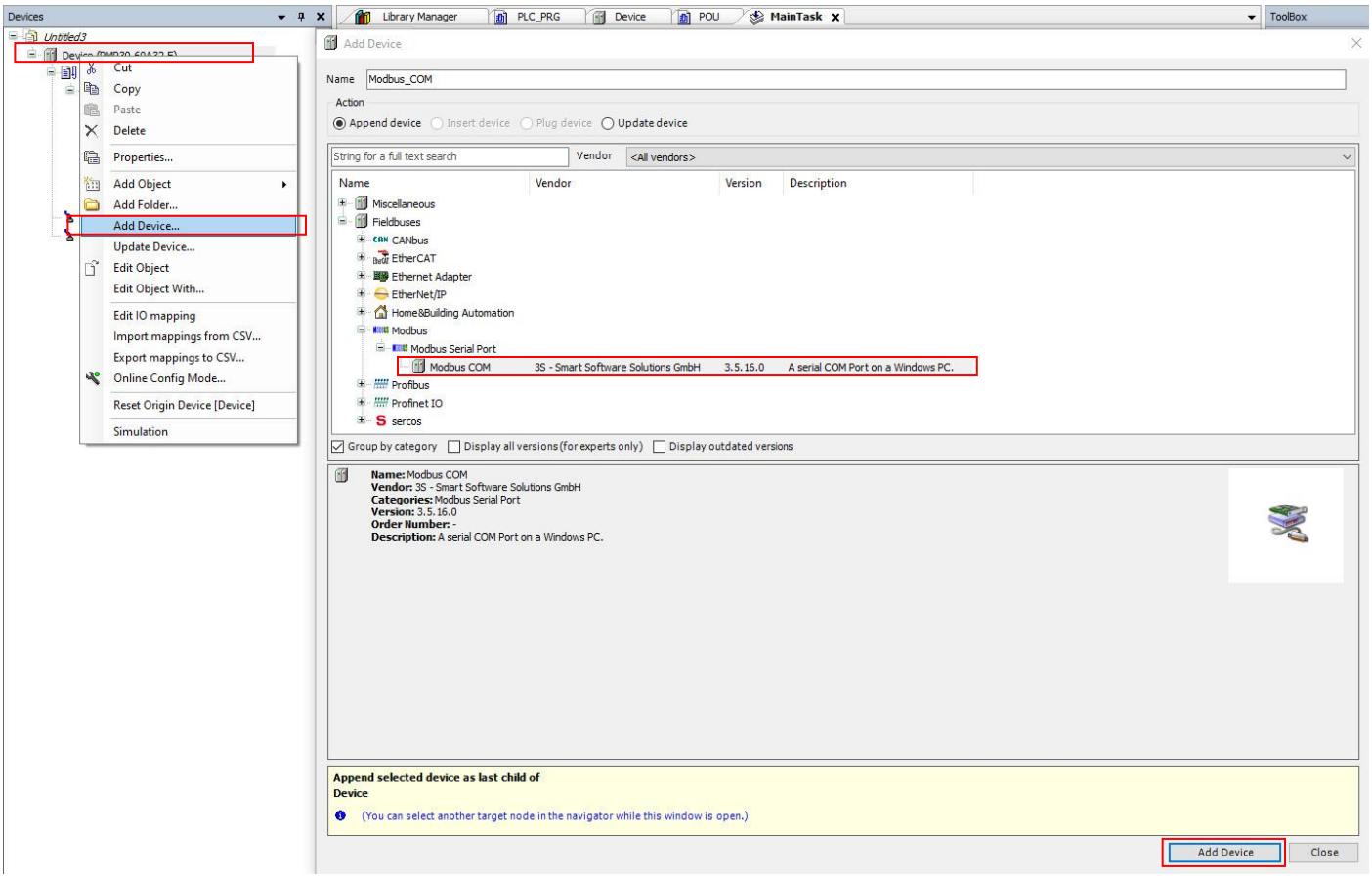


Причина пунктирных стрелок на рисунке (в том числе в сети RS485) заключается в том, что теоретически в этих двух сетях, пока каждый ПЛК не отправляет данные, любой ПЛК в сети может использоваться в качестве ведущей станции, а другие ПЛК - в качестве ведомой станции. Однако, поскольку между несколькими ПЛК нет единого эталона синхронизации, легко отправить данные с нескольких ПЛК одновременно, что приведет к сбою связи. Поэтому не рекомендуется использовать этот метод.

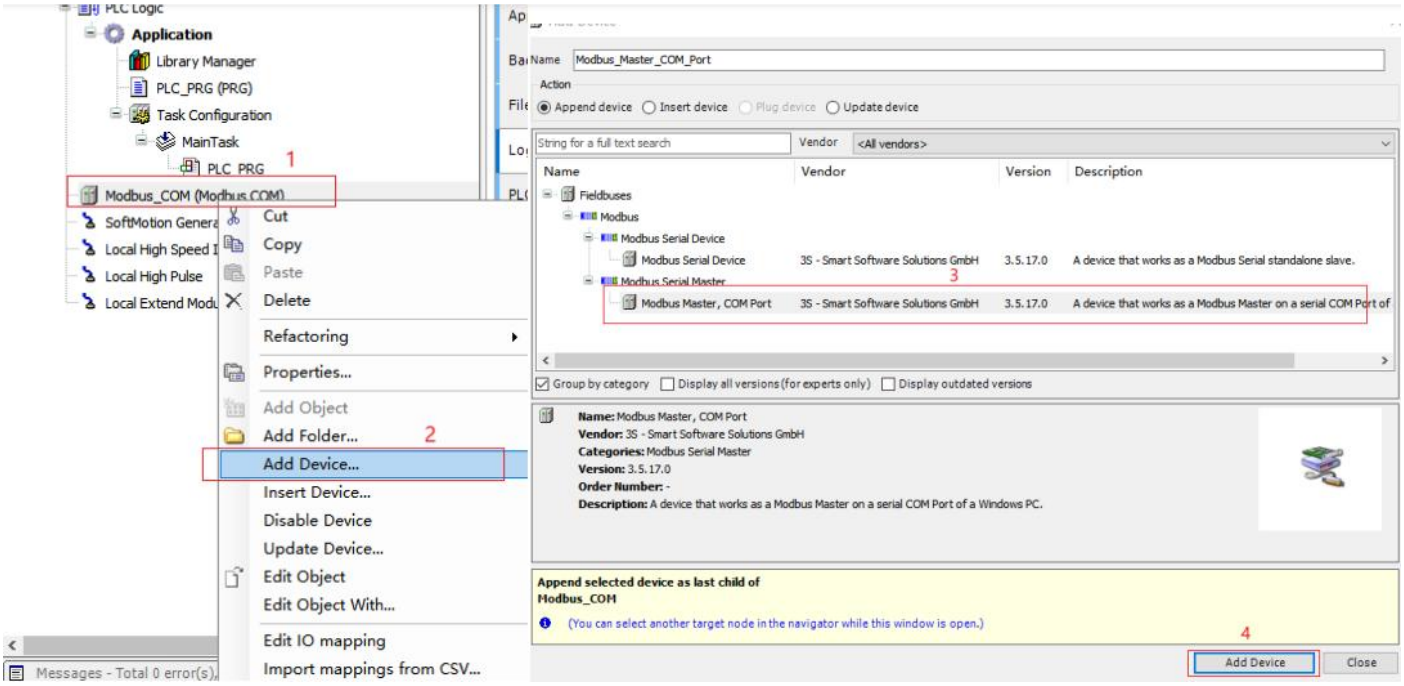
6.1.2 Конфигурация параметров

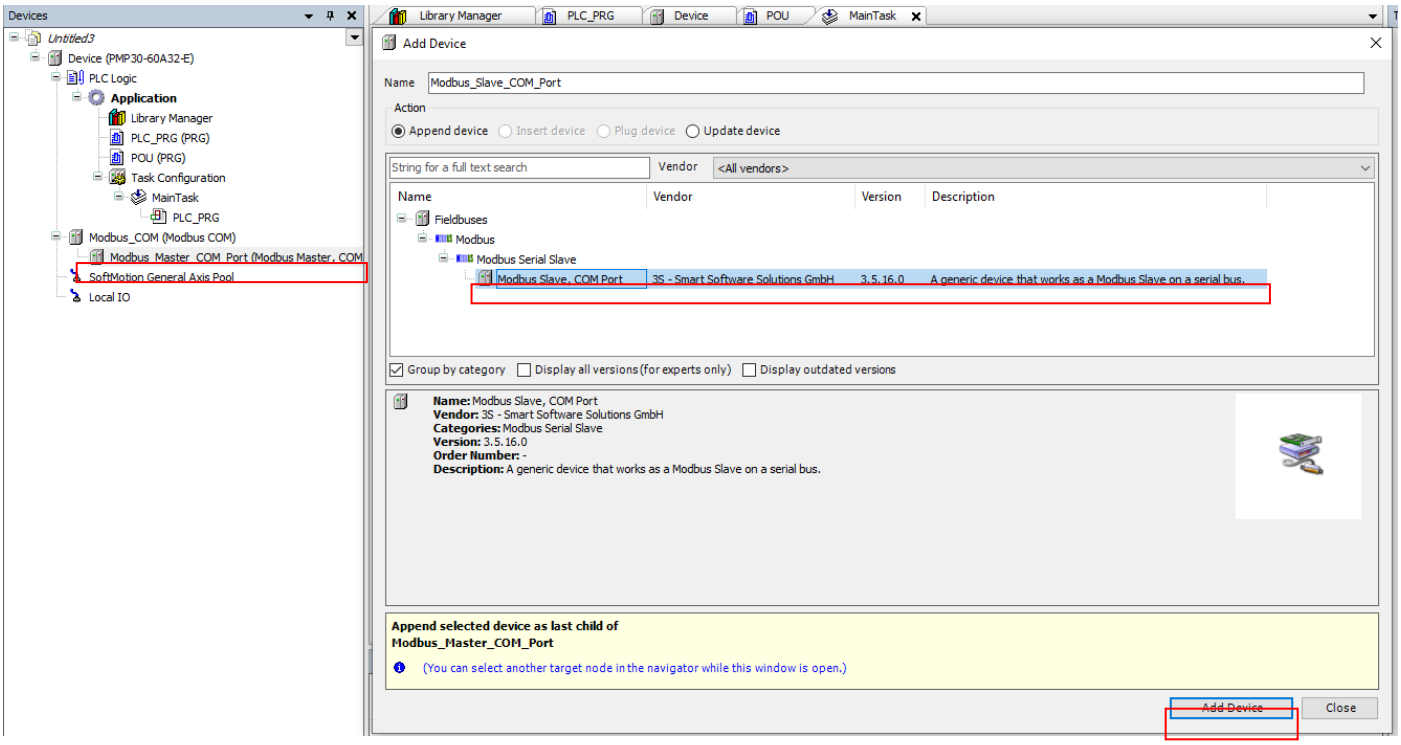
Конфигурация ведущей станции Modbus

(1) В применяемом проекте щелкните правой кнопкой мыши устройство, нажмите "добавить устройство" и выберите "MODBUS com" во всплывающем диалоговом окне для добавления.

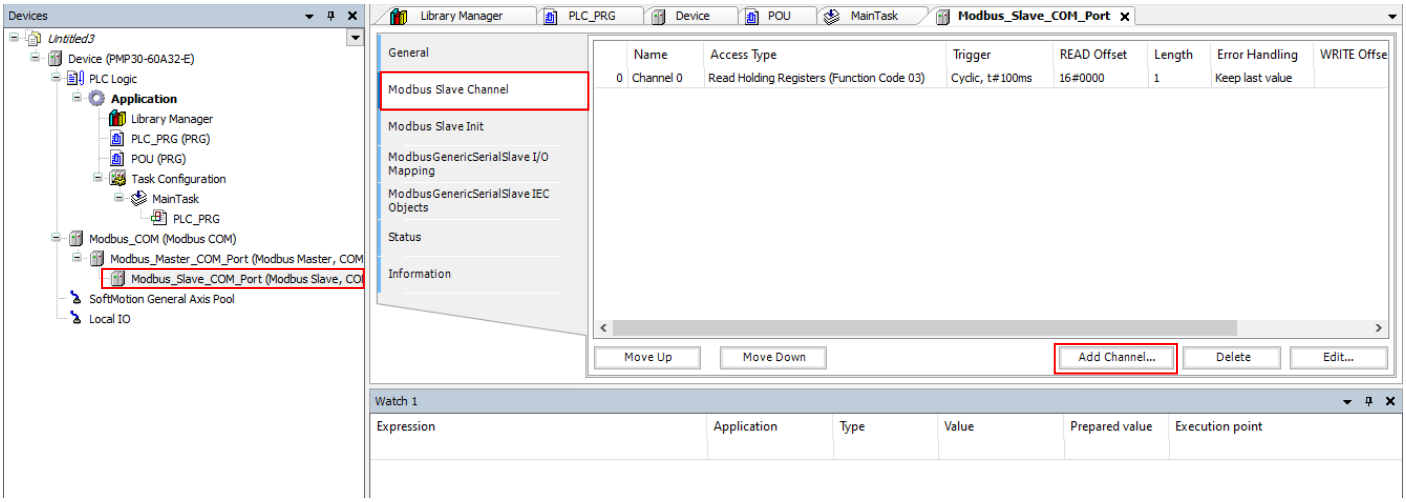


(2) После успешного добавления вы увидите "MODBUS COM" под устройством, нажмите "добавить устройство", выберите "MODBUS master, COM port" во всплывающем диалоговом окне и нажмите "добавить устройство" для добавления. Выберите "MODBUS master, COM port", выберите "MODBUS slave, COM port" во всплывающем диалоговом окне и нажмите "add device" для завершения добавления.



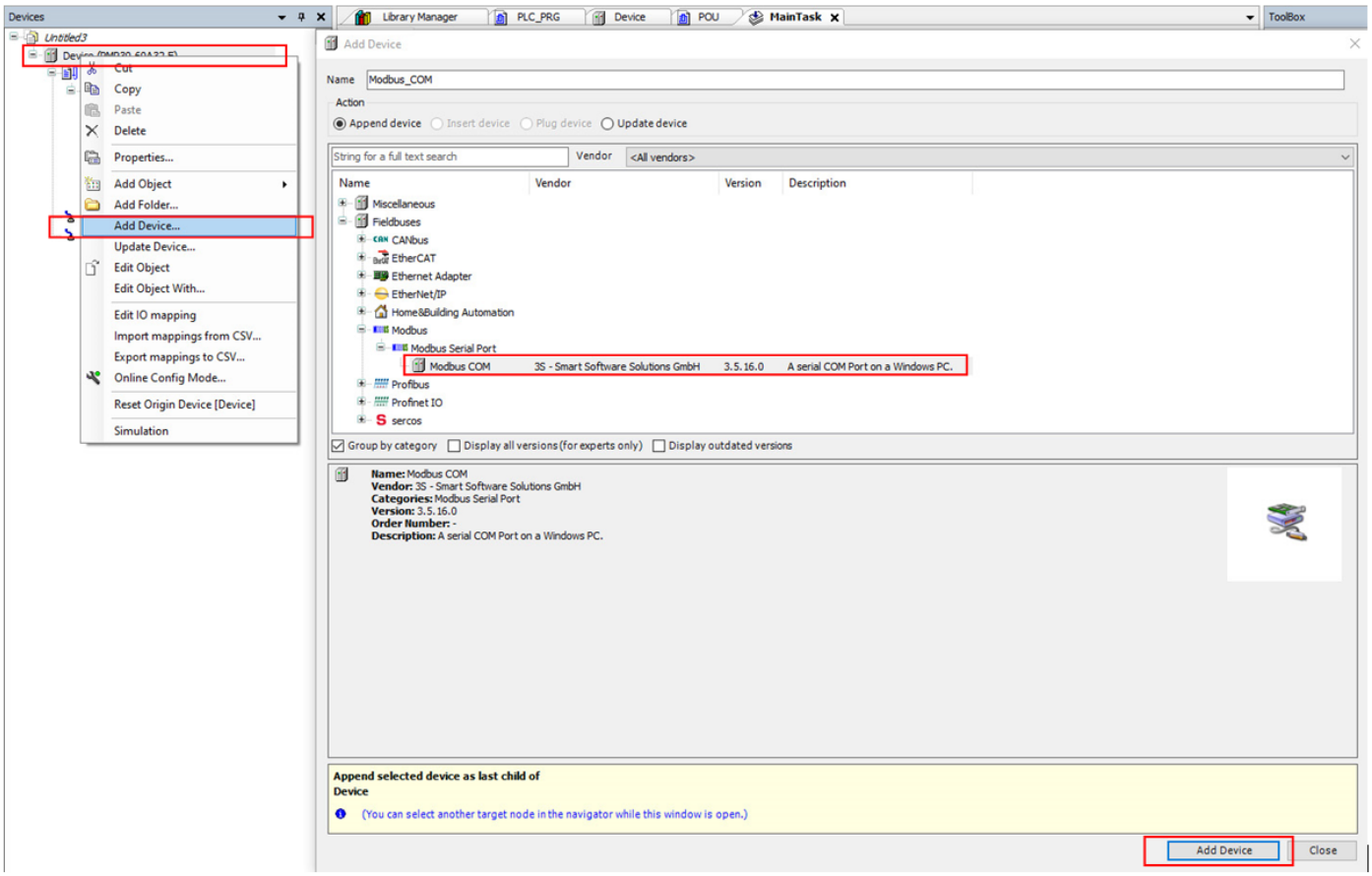


(3) После добавления вы можете увидеть добавление ведущей станции Modbus COM на панели устройств слева. Дважды щелкните на "modbus_slave_com_port", чтобы настроить чтение и запись в "MODBUS slave channel" справа.

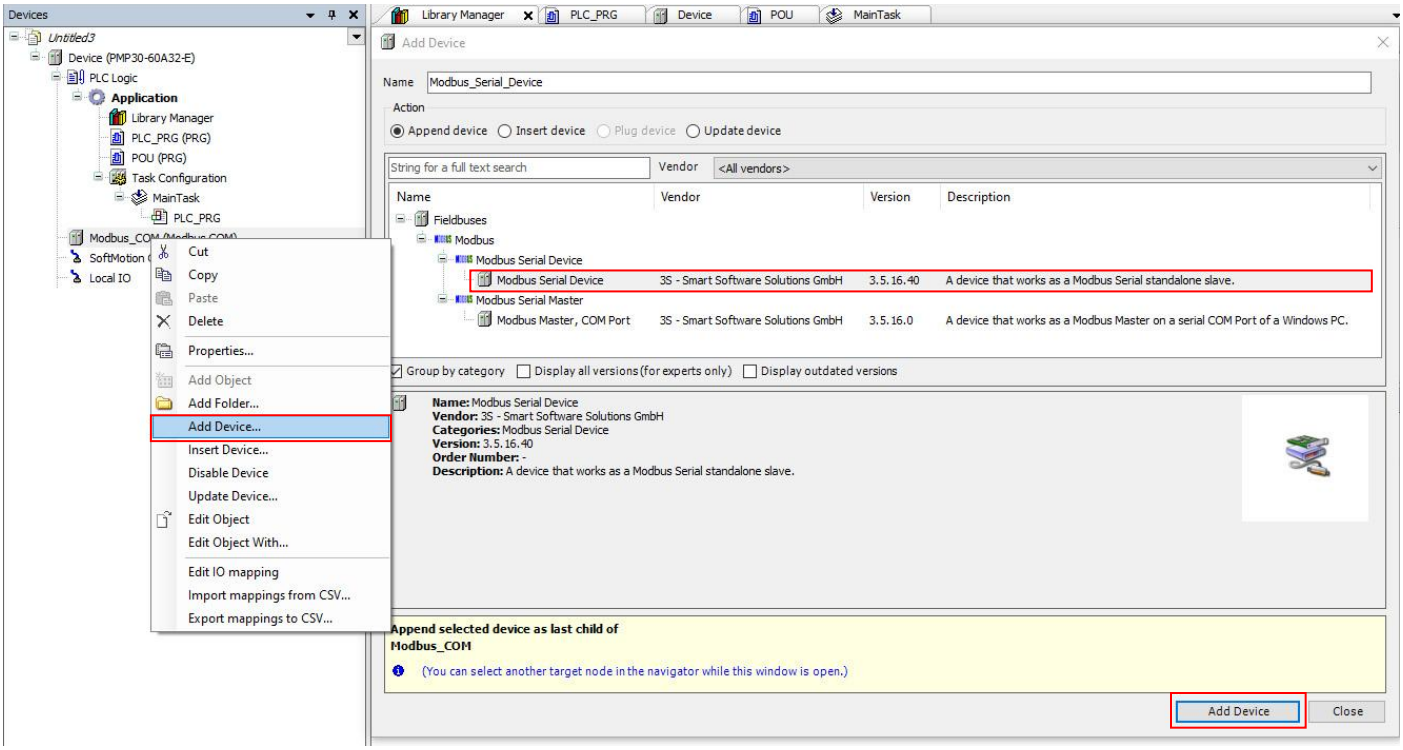


Конфигурация ведомой станции Modbus

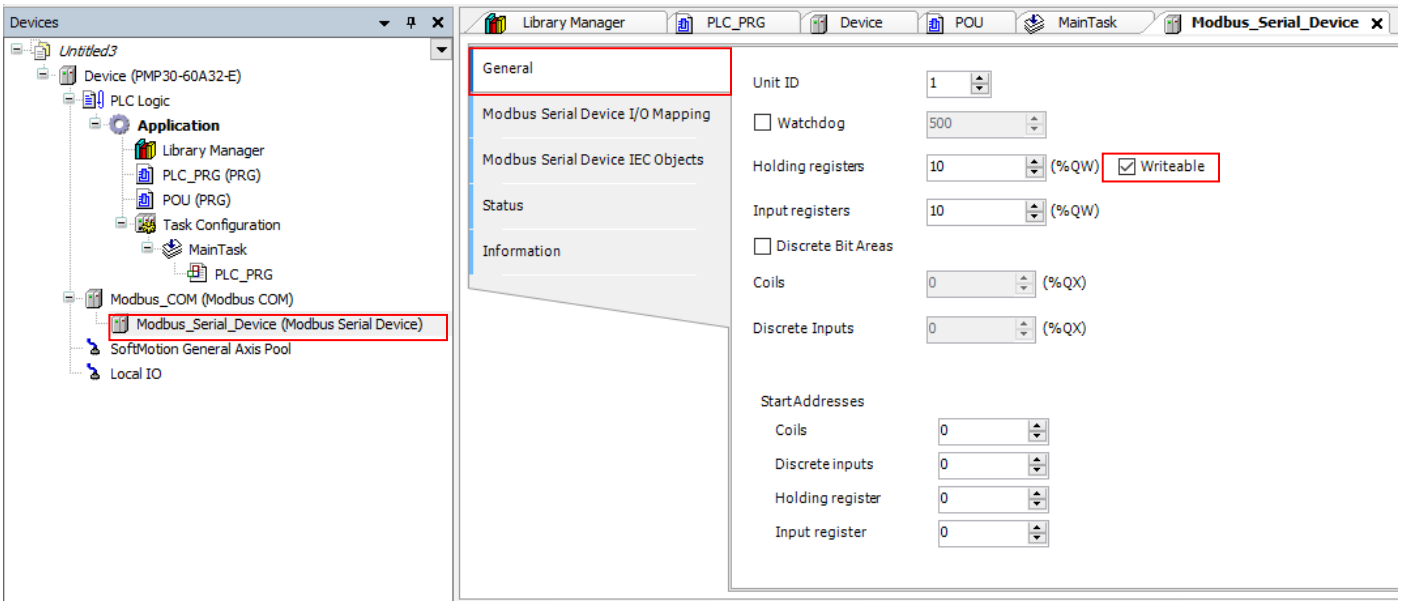
(1) В применяемом проекте щелкните правой кнопкой мыши "устройство", нажмите "добавить устройство" и выберите "MODBUS COM" во всплывающем диалоговом окне для добавления.

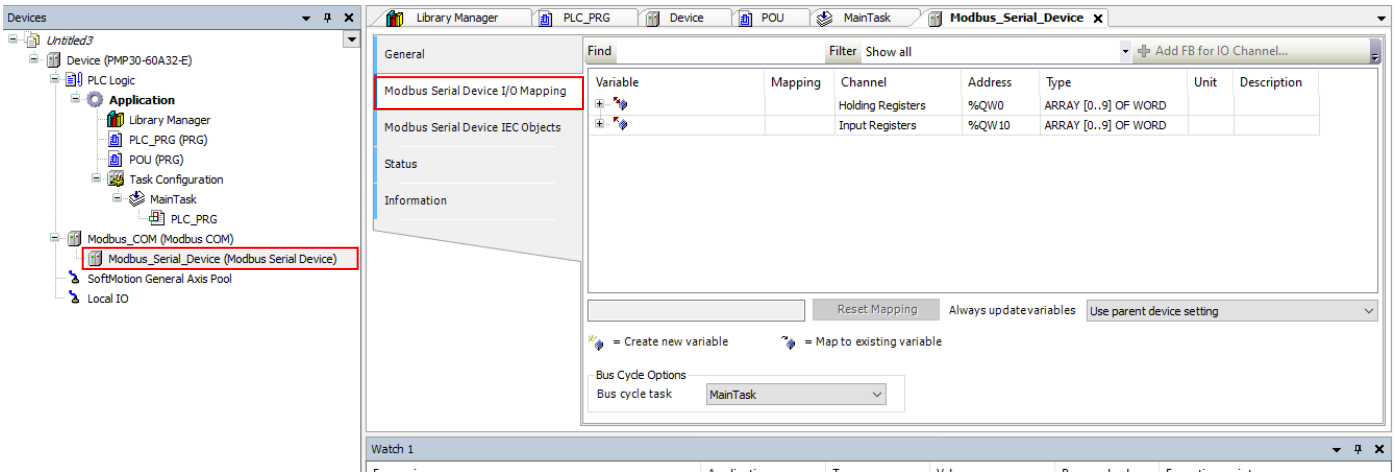


(2) После успешного добавления под устройством появится надпись "MODBUS COM". Щелкните правой кнопкой мыши "Добавить устройство" и выберите "Последовательное устройство MODBUS" во всплывающем диалоговом окне. Как показано на следующем рисунке:



(3) После добавления вы можете увидеть добавление ведомой станции Modbus com на левой панели устройств. Дважды щелкните "Последовательное устройство MODBUS", чтобы настроить регистры и катушки в разделе "Общие". После настройки вы можете отслеживать чтение и запись данных ведущей станции в ведомую станцию PMP30 в "MODBUS serial device I/O mapping".




Devices
Untitled3
Device (PMP30-60A32-E)
PLC Logic
Application
Library Manager
PLC_PRG (PRG)
POU (PRG)
Task Configuration
MainTask
PLC_PRG
Modbus_COM (Modbus COM)
Modbus_Serial_Device (Modbus Serial Device)
SoftMotion General Axis Pool
Local IO
Library Manager
PLC_PRG
Device
POU
MainTask
Modbus_Serial_Device
General
Modbus Serial Device I/O Mapping
Modbus Serial Device IEC Objects
Status
Information
Find
Filter
Show all
Add FB for IO Channel...
Variable
Mapping
Channel
Address
Type
Unit
Description
Holding Registers
%QW0
ARRAY [0..9] OF WORD
Input Registers
%QW10
ARRAY [0..9] OF WORD
Reset Mapping
Always update variables
Use parent device setting
= Create new variable
= Map to existing variable
Bus Cycle Options
Bus cycle task
MainTask
Watch 1

6.2 MODBUS TCP

6.2.1 Обзор MODBUS TCP

Modbus TCP использует протокол TCP/IP для передачи сообщений MODBUS между станциями. Modbus TCP объединяет протокол TCP/IP и протокол Modbus в качестве метода представления данных стандарта прикладного протокола. Коммуникационные сообщения Modbus TCP инкапсулируются в пакеты Ethernet TCP/IP. По сравнению с традиционным режимом последовательного порта, Modbus TCP вставляет стандартное сообщение MODBUS в сообщение TCP без четности данных и адреса.

Программируемый контроллер серии PMP3xx поддерживает обмен данными по протоколу Modbus TCP в виде ведущего и ведомого устройства.

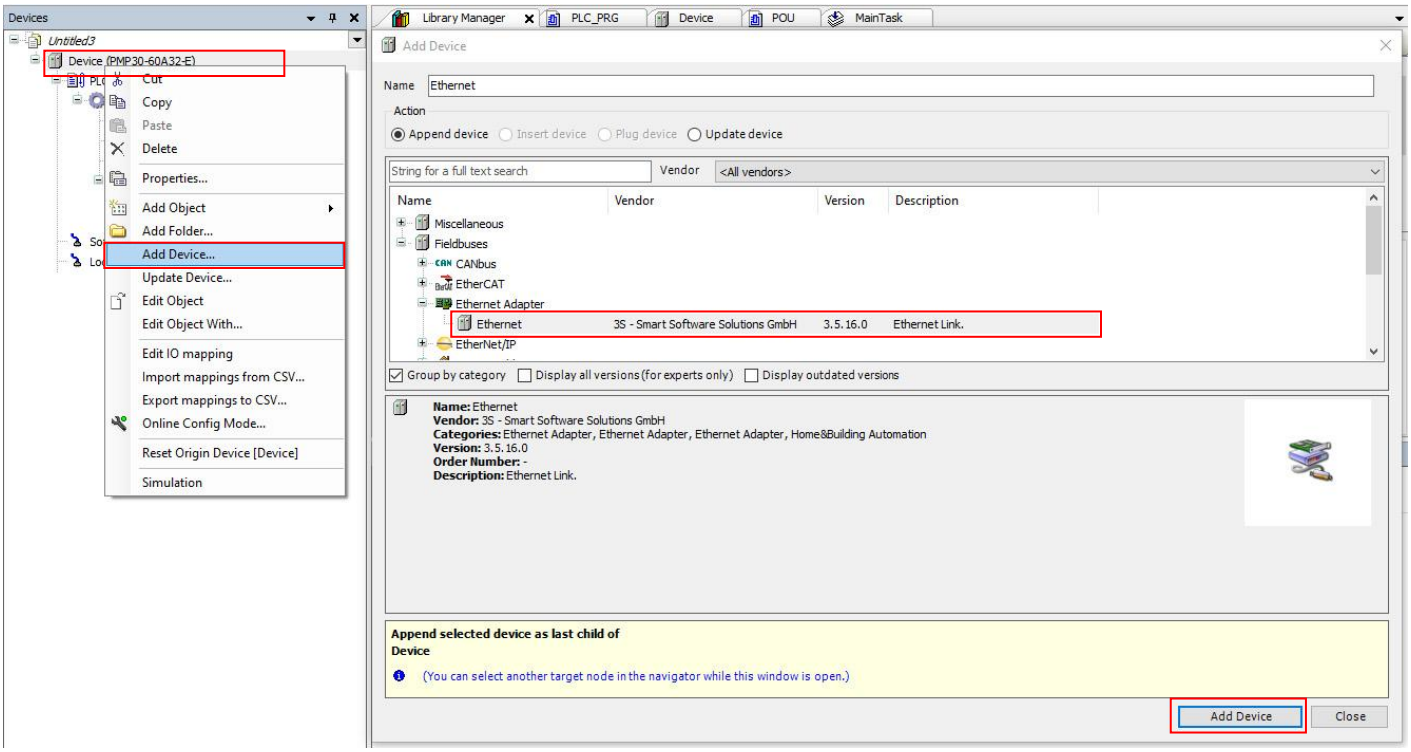
Форма мастер-станции: когда программируемый контроллер является устройством мастер-станции, он может взаимодействовать с другими ведомыми устройствами по протоколу Modbus TCP. К одной мастер-станции может быть подключено до 64 ведомых станций.

Форма ведомой станции: когда программируемый контроллер используется в качестве оборудования ведомой станции, он может только отвечать на требования других ведущих станций.

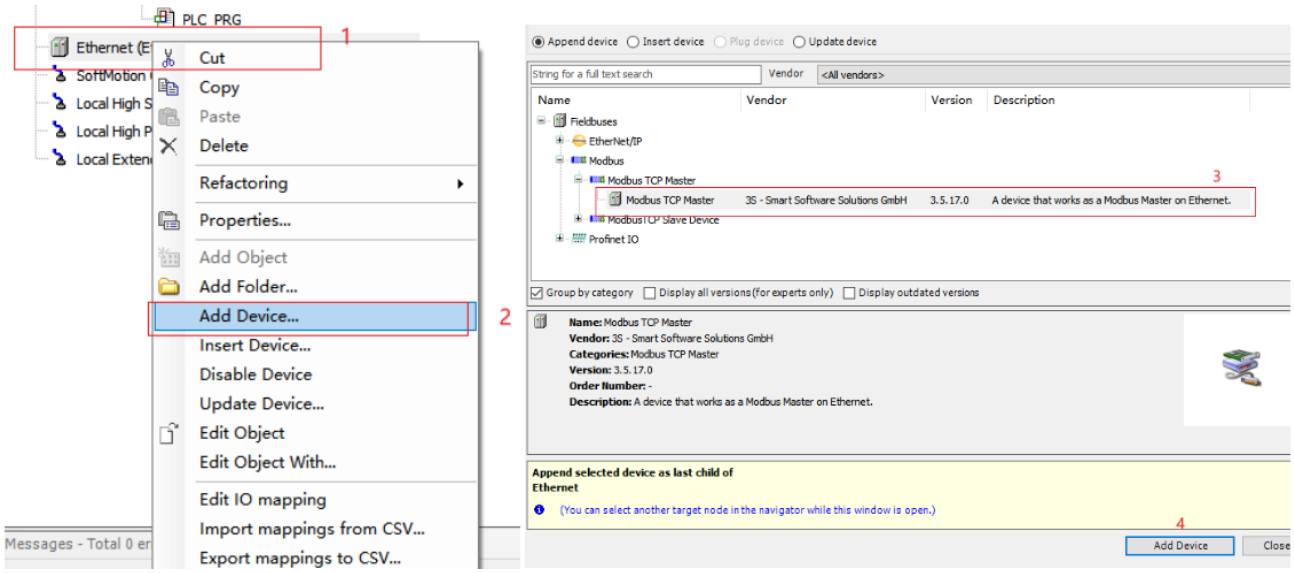
6.2.2 Конфигурация параметров

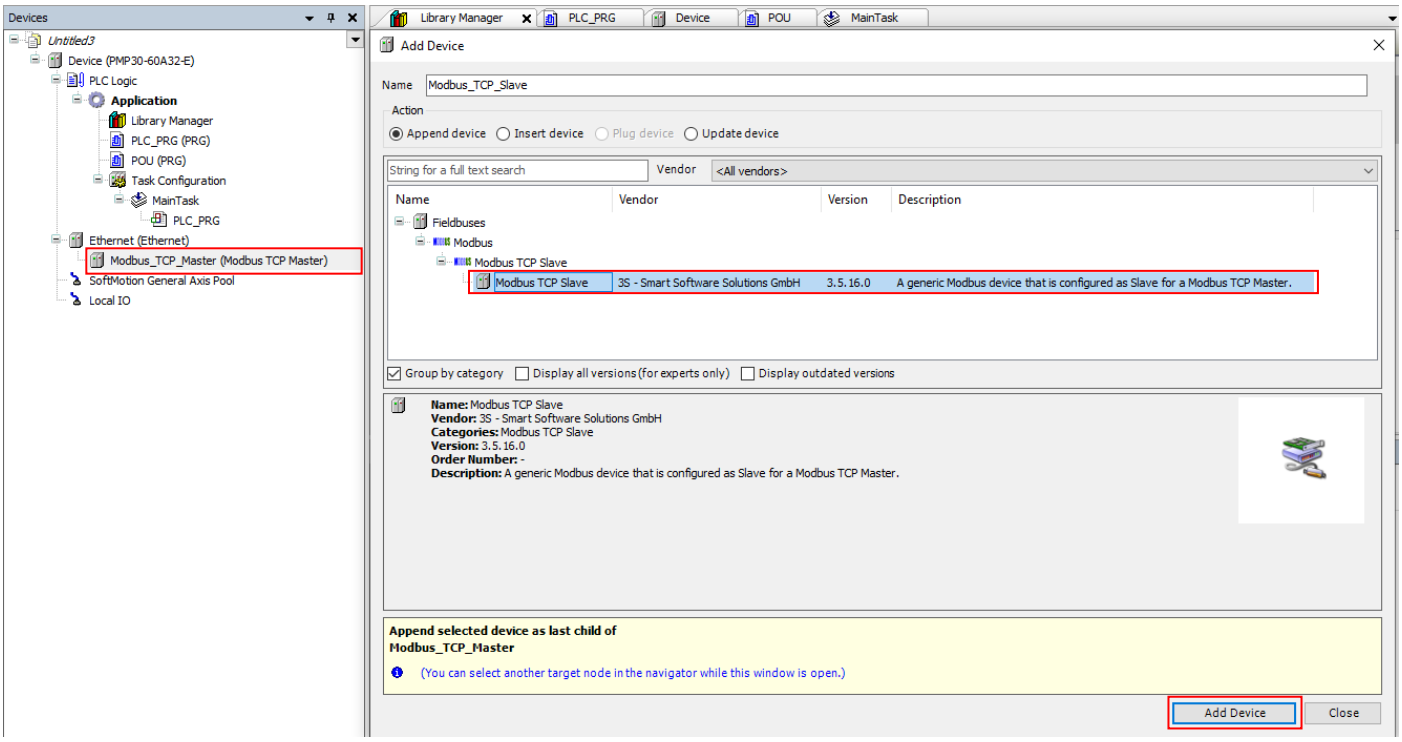
Конфигурация клиента Modbus TCP

(1) В применяемом проекте щелкните правой кнопкой мыши "устройство", нажмите "добавить устройство" и выберите "Ethernet" во всплывающем диалоговом окне, как показано на рисунке:

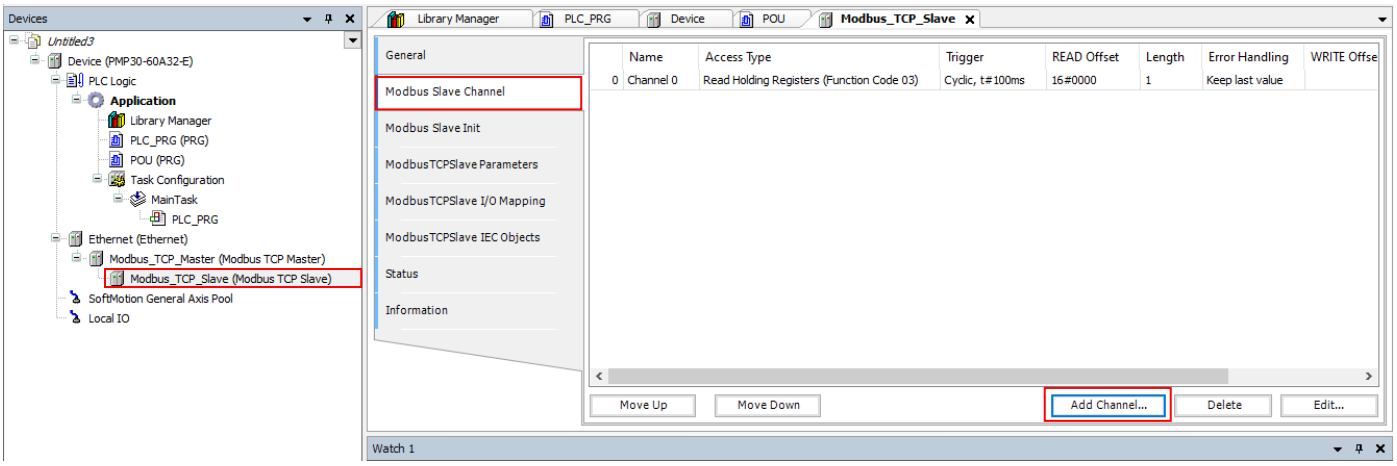


(2) После успешного добавления вы можете увидеть "Ethernet" под устройством. Щелкните правой кнопкой мыши "Ethernet", нажмите "добавить устройство", выберите "Modbus TCP master" во всплывающем диалоговом окне и нажмите "добавить устройство", чтобы добавить. Выберите "Modbus TCP master", выберите "Modbus TCP slave" во всплывающем диалоговом окне и нажмите "add device", чтобы завершить добавление.



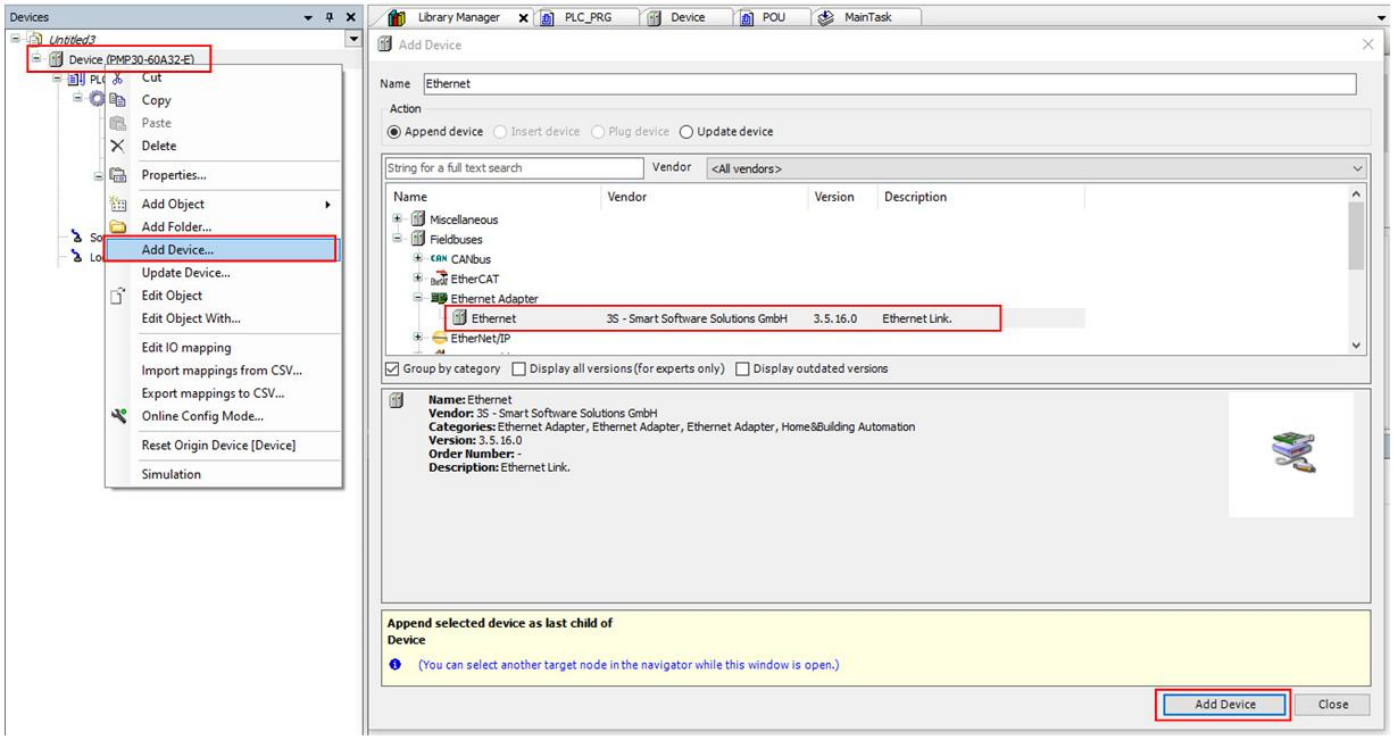


(3) После добавления вы можете увидеть добавление клиента Modbus TCP в левой панели устройств. Дважды щелкните "modbus_tcp_slave", чтобы настроить чтение/запись в правом "MODBUS slave channel".

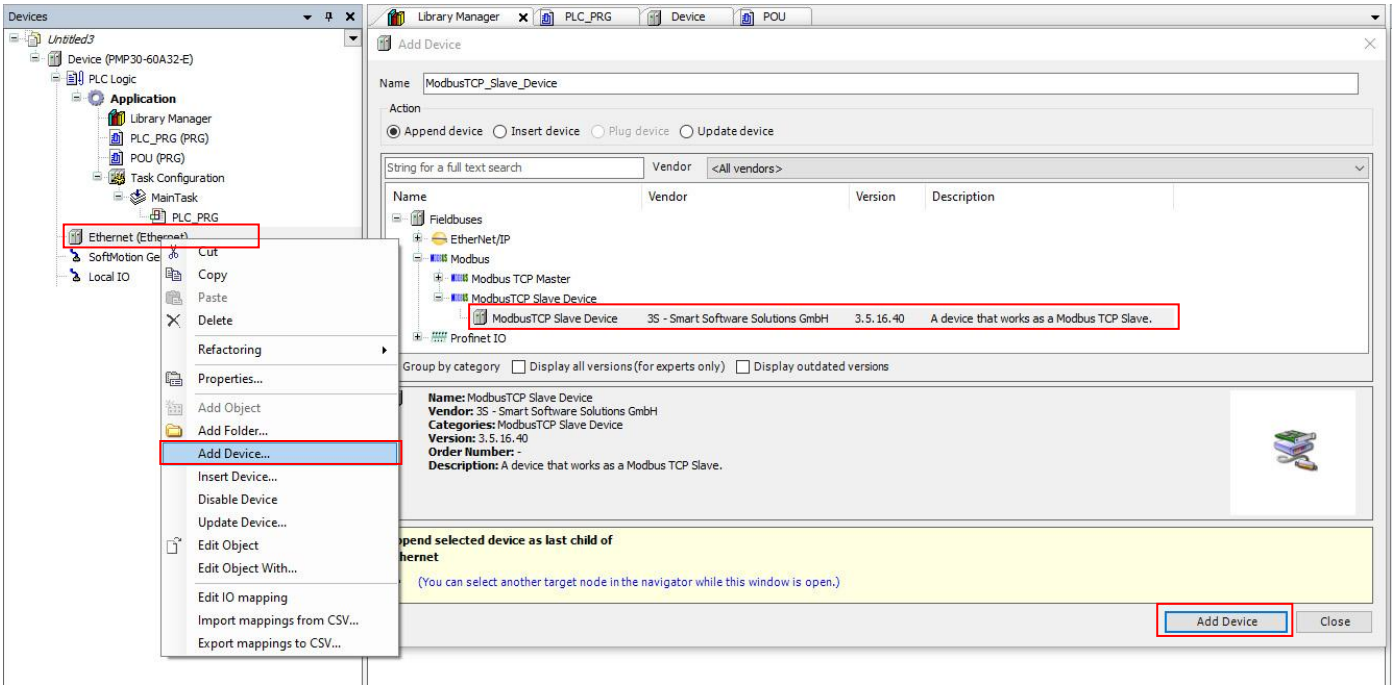


Конфигурация сервера Modbus TCP

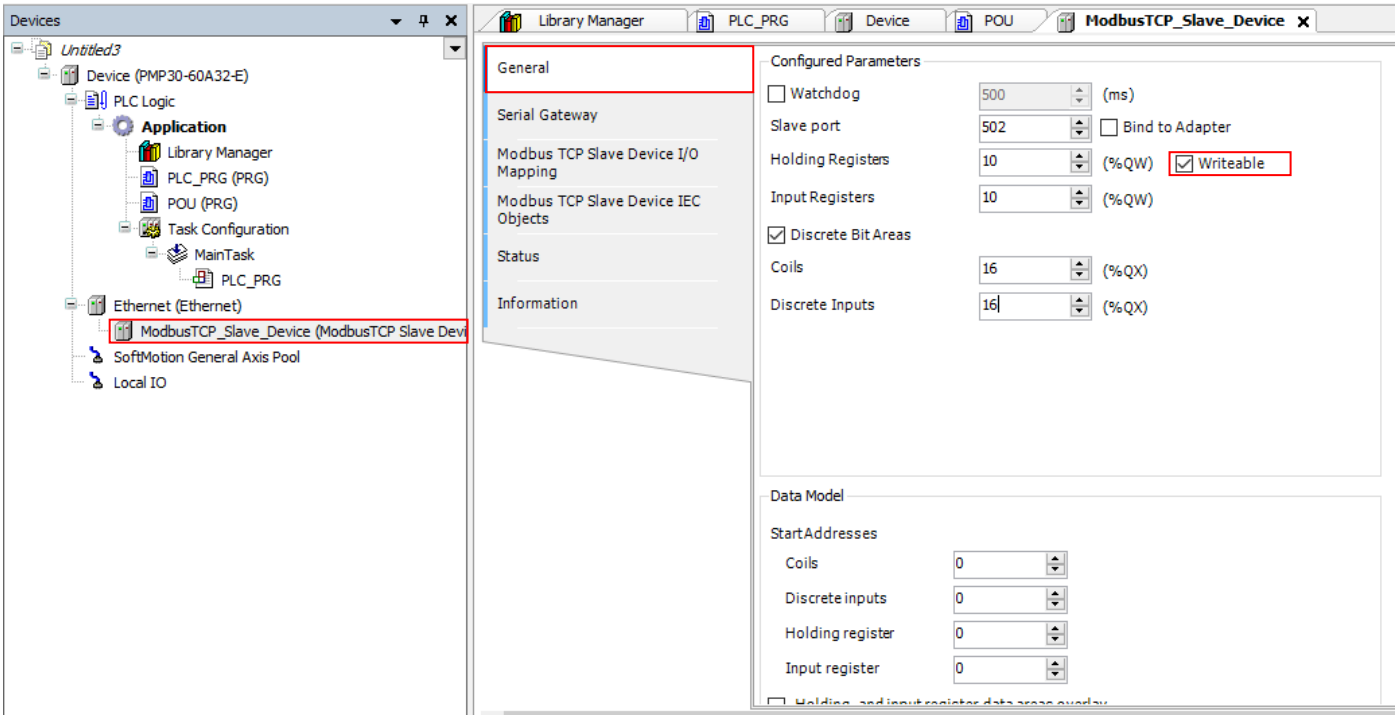
(1) В применяемом проекте щелкните правой кнопкой мыши "устройство", нажмите "добавить устройство" и выберите "Ethernet" во всплывающем диалоговом окне, как показано на рисунке:

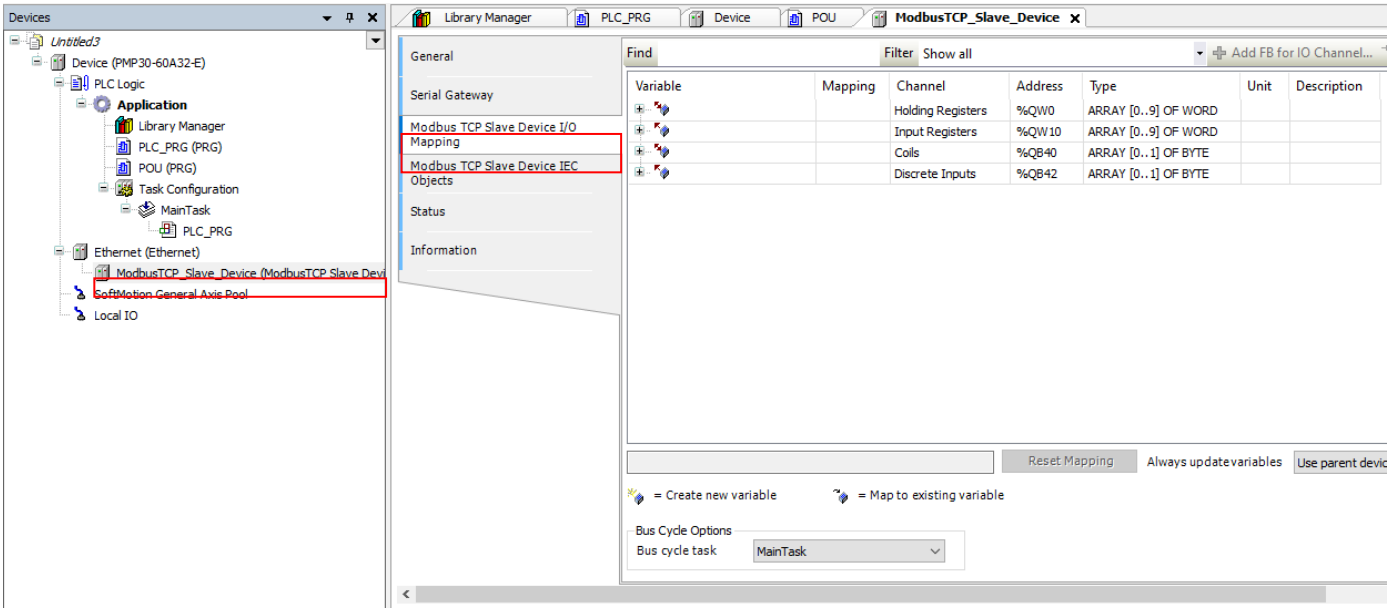


(2) После добавления "Ethernet" нажмите правой кнопкой мыши "добавить устройство" и выберите "ведомое устройство Modbus TCP" во всплывающем диалоговом окне. Как показано на следующем рисунке:



(3) После добавления вы можете увидеть добавление сервера Modbus TCP в левой панели устройств. Дважды щелкните "ModbusTCP_slave_device", чтобы настроить регистры и катушки в разделе "General". После настройки вы можете контролировать чтение и запись данных клиента на сервер PMP3xx в разделе "Modbus TCP slave device I/O mapping".





6.3 OPC UA

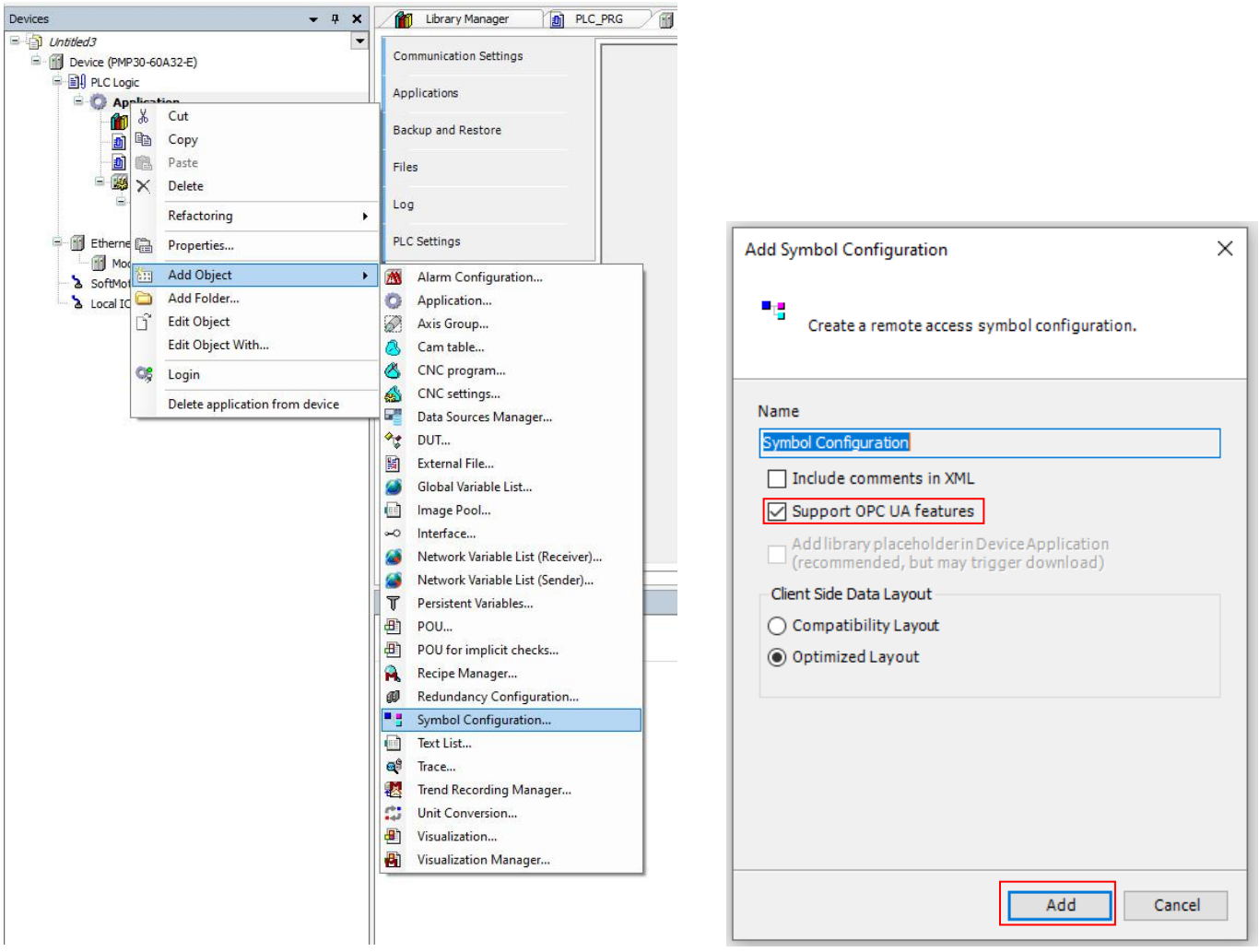
6.3.1 Обзор коммуникаций OPC UA

OPC UA был выпущен в 2008 году. Это независимая от платформы сервис-ориентированная архитектура, которая объединяет все функции различных классических спецификаций OPC в расширяемую структуру.

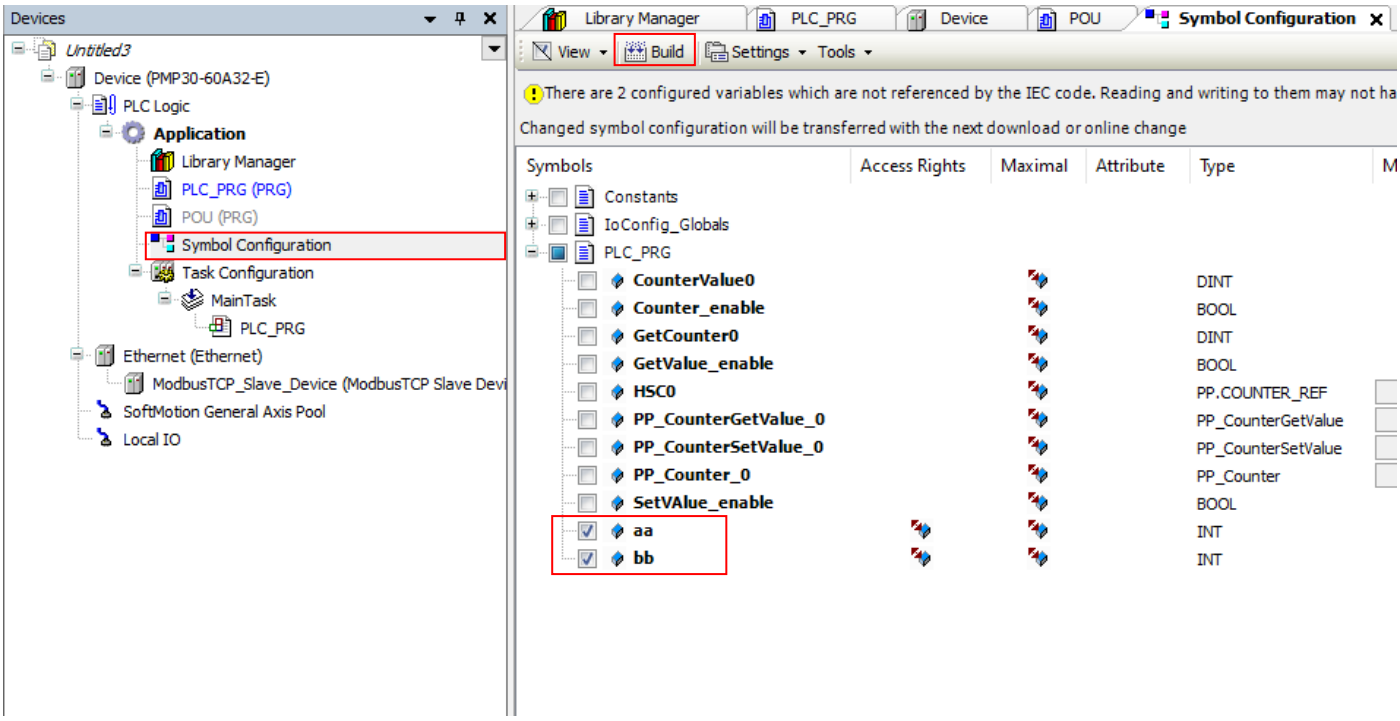
В ПЛК серии PMP3xx интегрирован сервер OPC UA, который может поддерживать доступ пользователей к данным в ПЛК через клиента OPC UA.

6.3.2 Конфигурация параметров

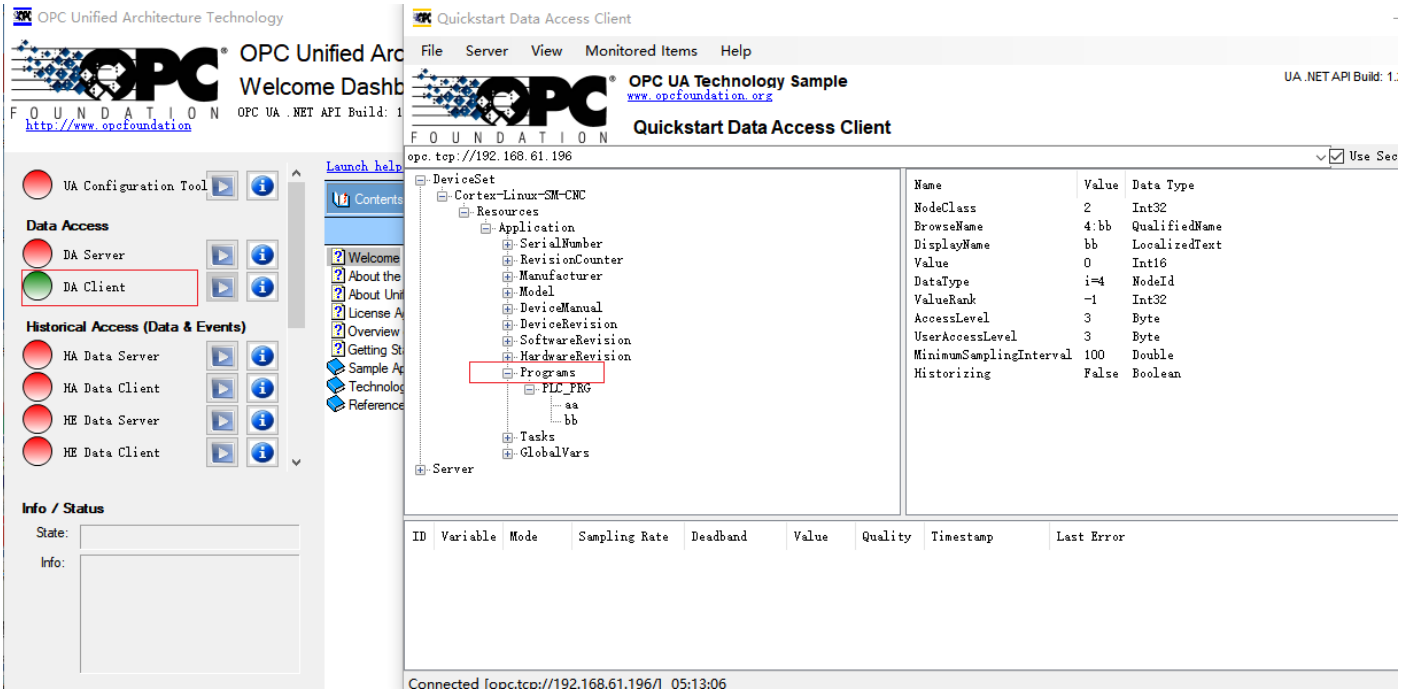
(1) В применяемом проекте щелкните правой кнопкой мыши "приложение", выберите "добавить объект" - "конфигурация символа...", и выберите "поддержка функции OPC UA" во всплывающем диалоговом окне для добавления, после чего функция OPCUA будет включена.

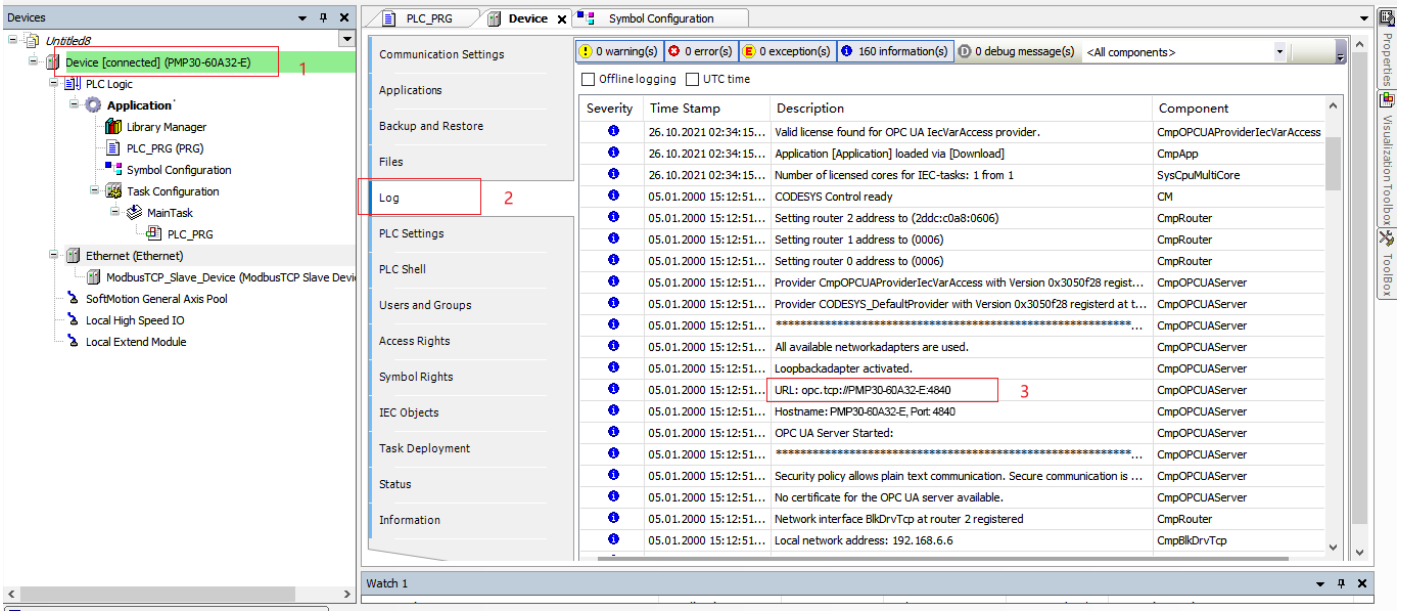


(2) Дважды щелкните "Symbol Configuration", нажмите "build" во всплывающем интерфейсе и выберите параметры для мониторинга.



(3) После загрузки программы откройте программу OPC UA, выберите "DA клиент" и введите "IP" адрес промышленного контроля во всплывающем интерфейсе (например, opc.tcp://192.168.61.196) Или вы можете ввести адрес в "журнал" и найти "программы" в "DeviceSet". Там находятся только что проверенные параметры. Вы можете щелкнуть правой кнопкой мыши на параметре -- monitor -- для чтения и записи параметра.





6.4 Свободный формат

6.4.1 Обзор свободного формата

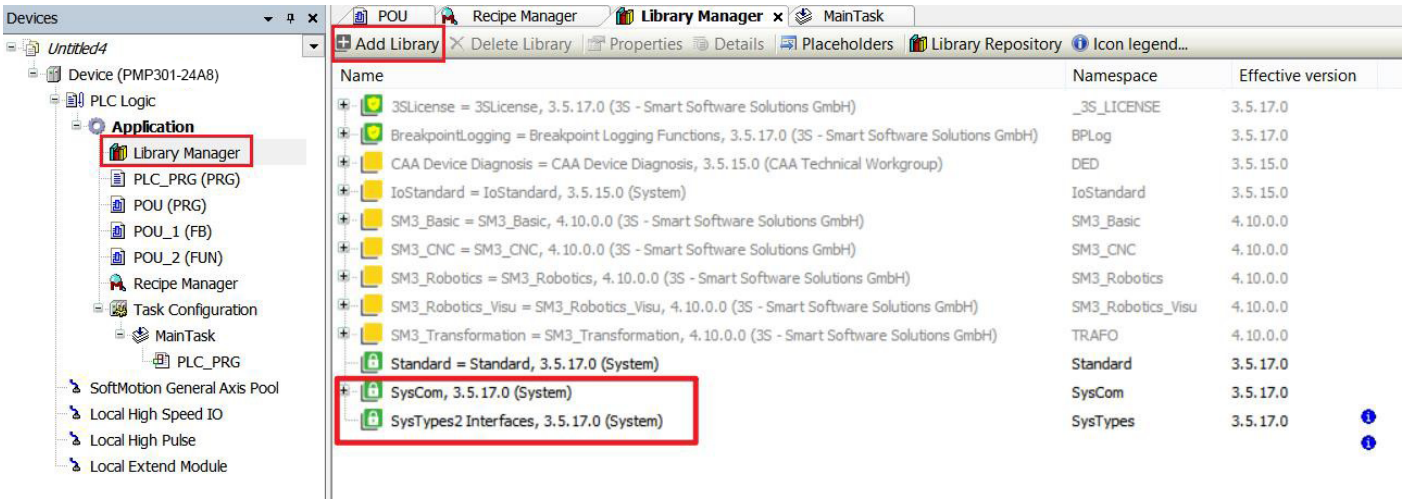
При обмене данными ПЛК PROMPOWER с другим оборудованием, когда ПЛК выступает ведомым устройством, ведущий контроллер должен обмениваться с ним информацией в формате Modbus RTU. Если ПЛК PROMPOWER является ведущим устройством и ведомый узел также поддерживает протокол Modbus RTU, для связи можно непосредственно использовать соответствующие коммуникационные инструкции, что делает написание программы проще и эффективнее. Если же ведомый узел не поддерживает Modbus RTU, применяется режим свободного формата.

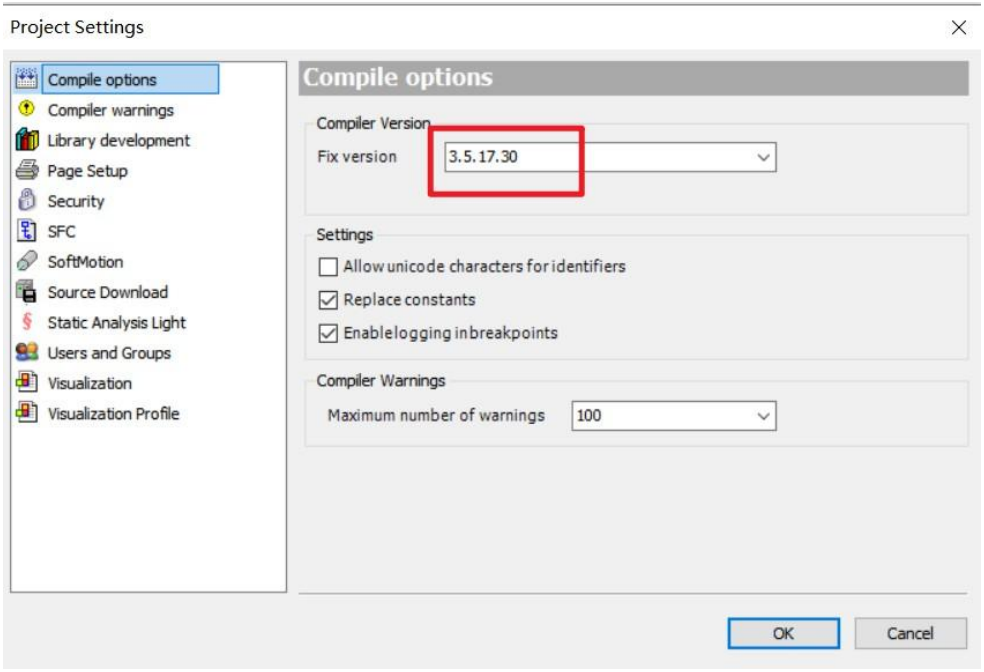
Свободный формат означает, что при несовпадении протоколов ПЛК самостоятельно задаёт структуру передаваемых данных, что позволяет ему взаимодействовать с широким спектром ведомых устройств.

Передача в свободном формате выполняется блоками данных: каждый блок может содержать до 256 байт и, при необходимости, может включать задаваемые пользователем начальный и конечный символы.

6.4.2 Конфигурация параметров

(1) В «Менеджере библиотек» добавьте две библиотеки — SysCom и SysTypes — и установите их версии в соответствии с версией верхнего компьютера (инженерной станции). Версия компилятора также должна соответствовать версии верхнего компьютера.





6.4.3 Применение

Пример 1:

Обмен в свободном формате между двумя ПЛК: передача и приём данных реализуются приведённой ниже программой.

Примечание: при использовании свободного формата для приёма данных необходимо либо увеличить период соответствующей задачи при непрерывном опросе, либо выполнять приём по нарастающему фронту (rising edge).

Порядок работы программы:

(1) Установите необходимые библиотеки согласно шагам, указанным в разделе 6-5-2.

(2) Напишите программу в свободном формате.

```
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
```

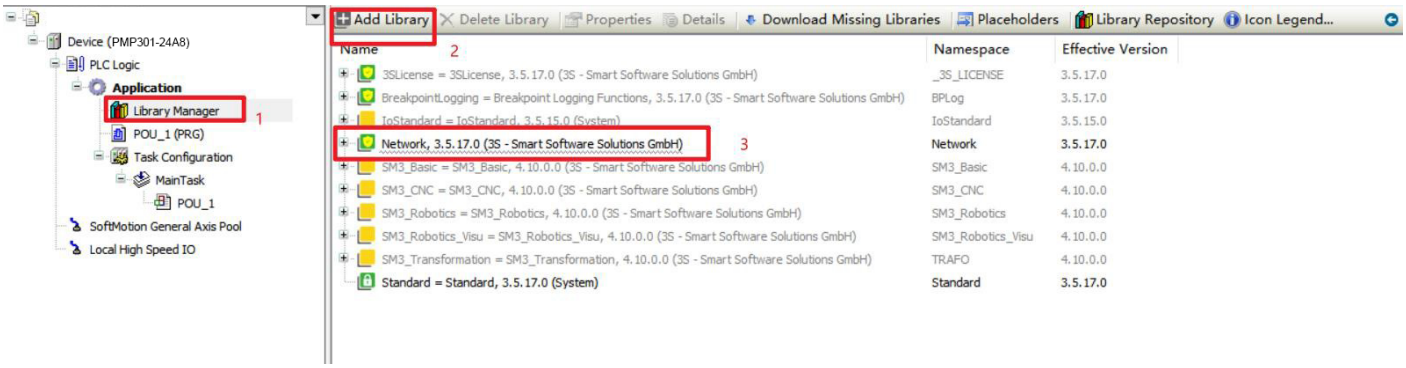
6.5 TCP/IP

6.5.1 Обзор TCP/IP

Протокол TCP/IP — распространённый сетевой протокол для Ethernet. В сравнении с эталонной моделью взаимодействия открытых систем ISO/OSI он использует более открытый подход и широко применяется в практических проектах. Стек TCP/IP может работать поверх различных каналов и базовых протоколов (например, T1, X.25, а также интерфейса RS232). В общем случае под TCP/IP понимают семейство протоколов, включающее TCP, IP, UDP, ICMP и другие.

6.5.2 Конфигурация параметров

Добавьте библиотеку TCP/IP Network в «Менеджер библиотек» и установите версию, соответствующую версии верхнего компьютера (инженерной станции).



Далее определите необходимые переменные в соответствующем POU и напишите программу.

6.5.3 Применение

Пример 1:

Обмен данными по TCP/IP между двумя ПЛК. Связь осуществляется с помощью следующей программы. IP-адрес ПЛК 1 — 192.168.6.17, IP-адрес ПЛК 2 — 192.168.6.18.

Примечание: при использовании свободного формата для приёма данных необходимо либо увеличить период соответствующей задачи при непрерывном опросе, либо выполнять приём по нарастающему фронту (rising edge) сервер должен сначала установить соединение и ожидать подключения клиента. В противном случае соединение по TCP/IP может не установиться.

Порядок работы программы:

(1) Установите необходимую библиотеку в соответствии с шагами, описанными в разделе 6-6-2.

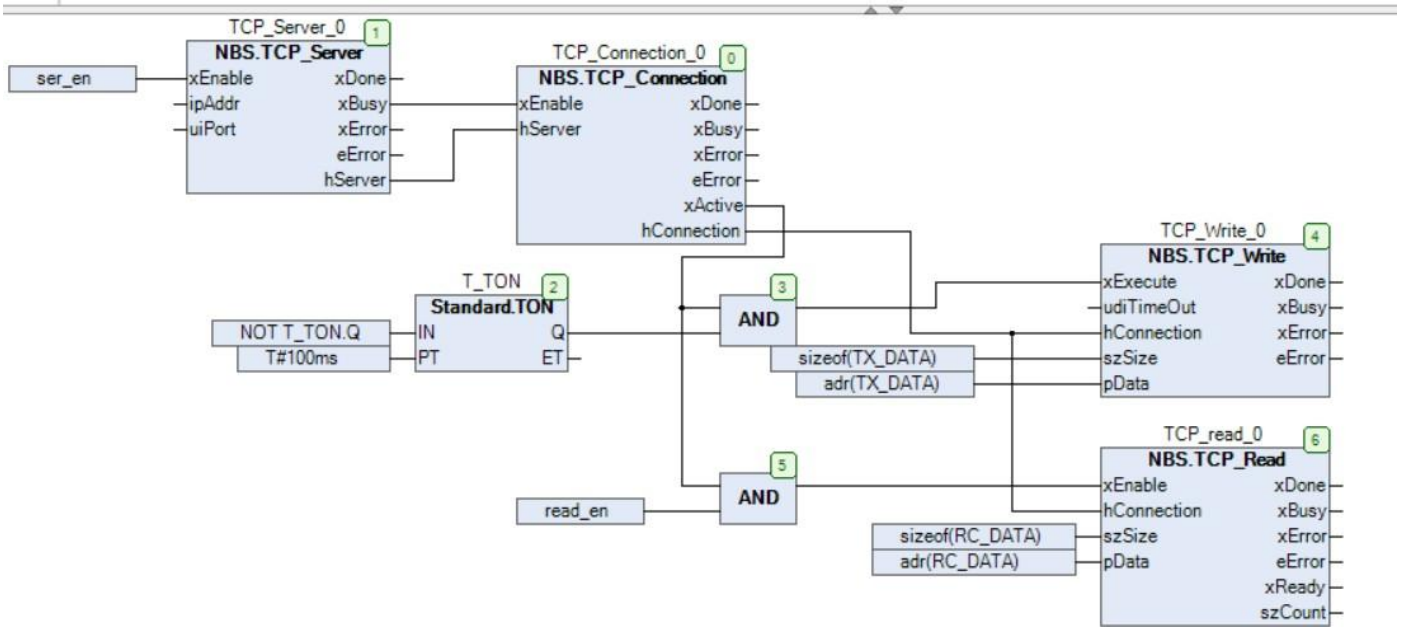
(2) Напишите программу TCP/IP.

Программирование:

Для настройки сервера, клиента, а также параметров отправки и приёма TCP/IP используйте следующие функциональные блоки:

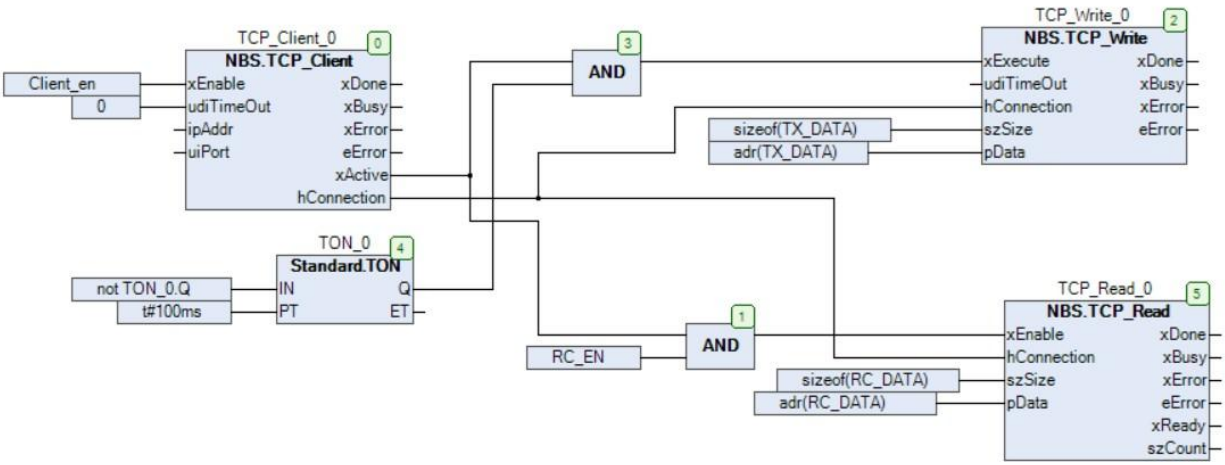
NBS.TCP_Server, NBS.TCP_Client, NBS.TCP_Connection, NBS.TCP_Write, NBS.TCP_Read.

```
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 Общие проблемы и решения

7.1 Таргет-файлы (.package)

7.1.1 Правило именования тагрет-файлов

Формат именования: **PMP30-60A32_3.5.15.40_1.0.0_P1_20211027**
① ② ③ ④ ⑤

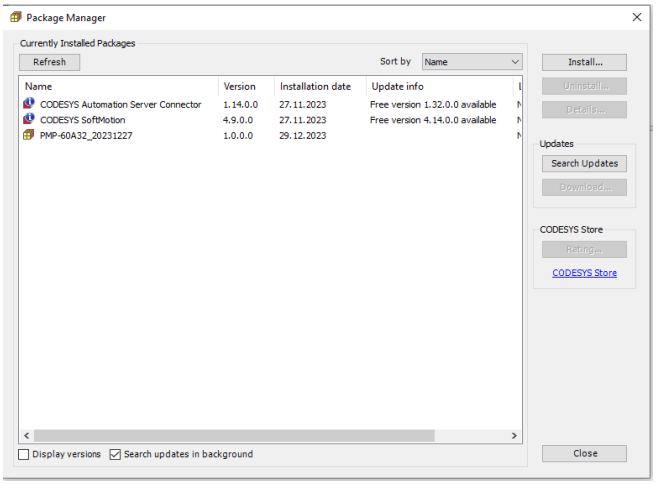
Нет.	Имя	Примечание
①	**PMP30-60A32**	Модель ПЛК
②	**3.5.15.40**	Версия для выполнения
③	**1.0.0**	Производственная версия пакета
④	**P1**	Первый пакет обновления в режиме онлайн после производства
⑤	**20211027**	Дата обновления пакета

7.1.2 Где скачать таргет-файл

Пожалуйста, скачайте таргет-файл в соответствующем разделе ПЛК на сайте prompower.ru или свяжитесь с нами по электронной почте support@prompower.ru.

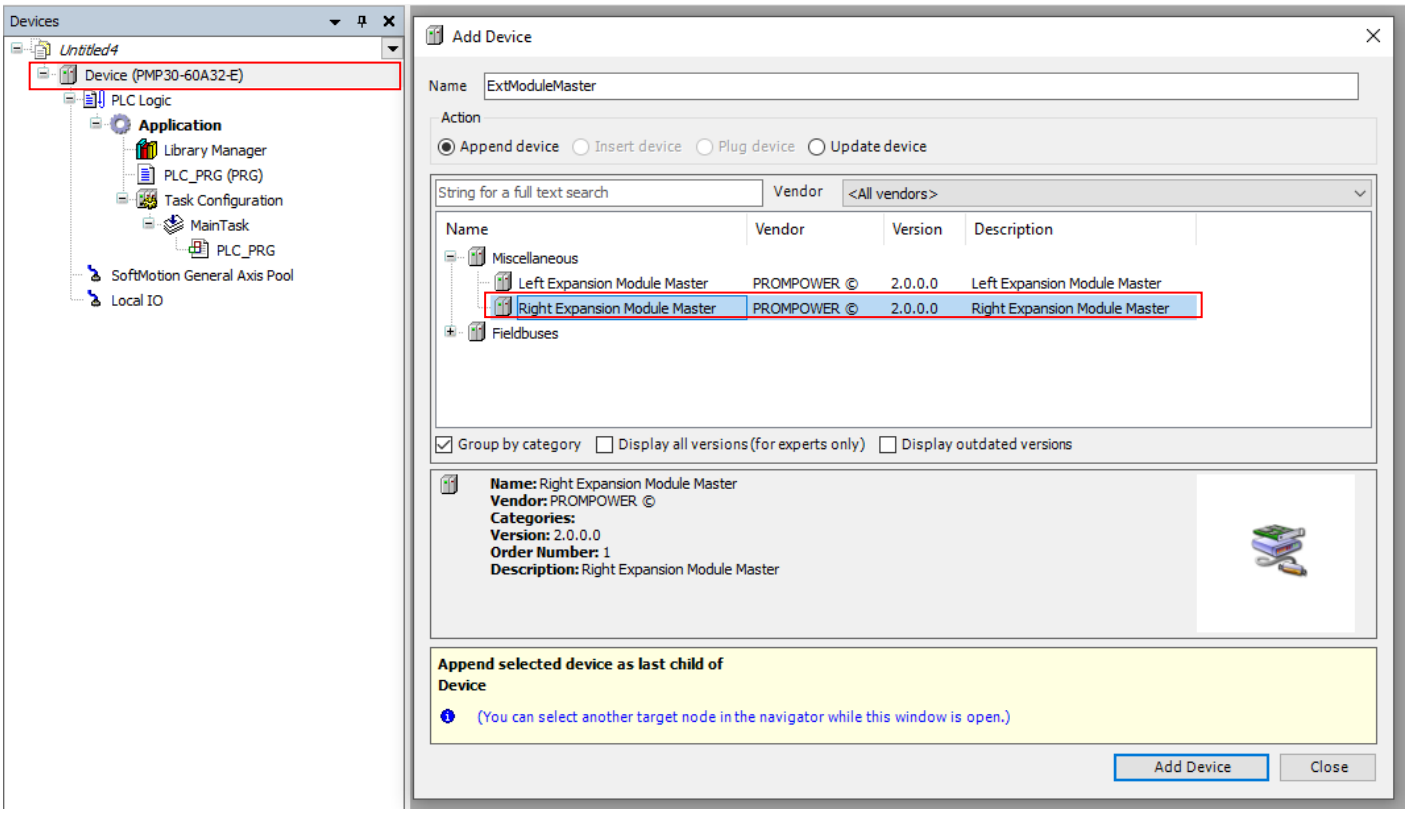
7.1.3 Установка пакета

Выберите "инструменты" - "менеджер пакетов...", установите пакет во всплывающем интерфейсе, выберите "Установить" и найдите местоположение пакета для установки. Например, чтобы установить пакет PMP30-60A32, лучше всего удалить предыдущий пакет перед установкой нового.

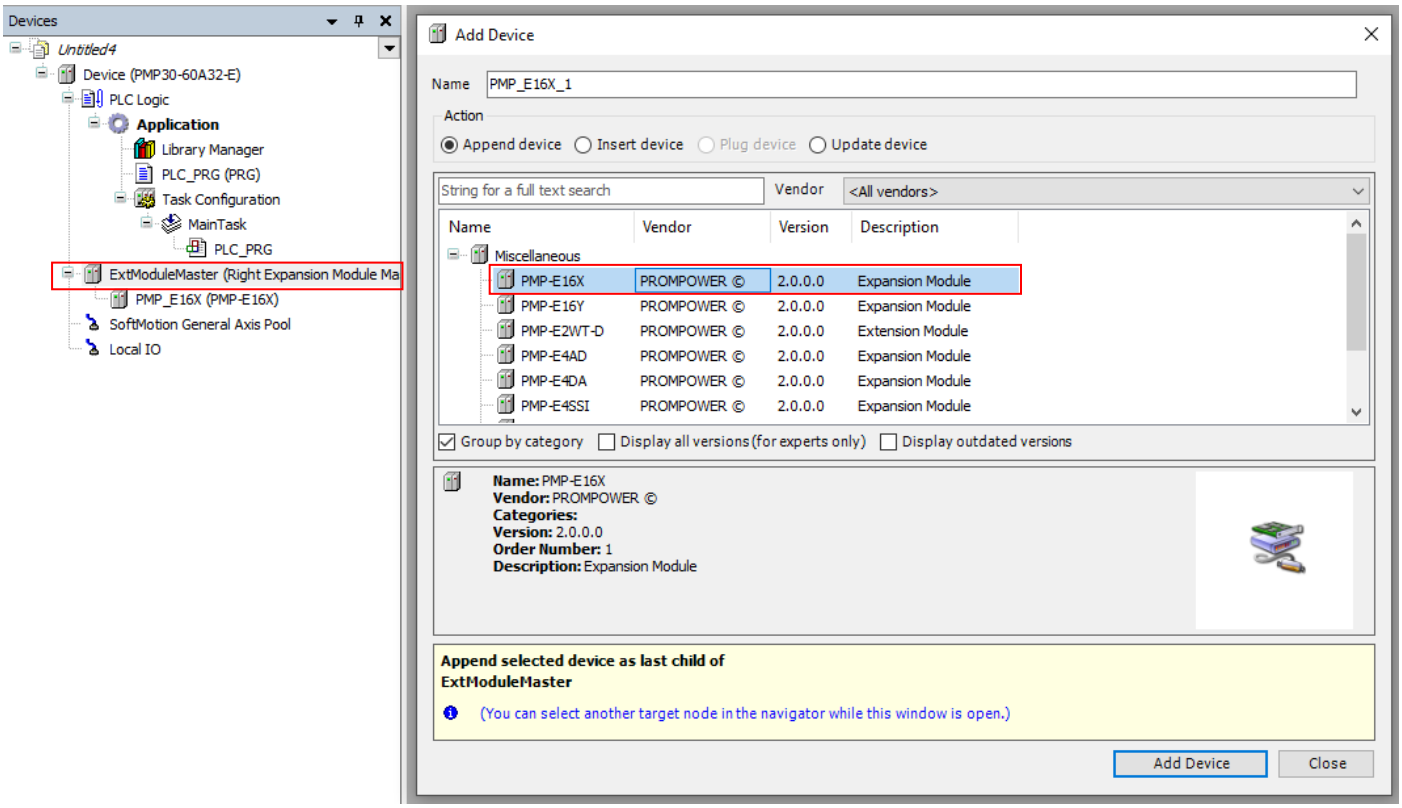


7.2 Модуль локального расширения серии PMP3xx

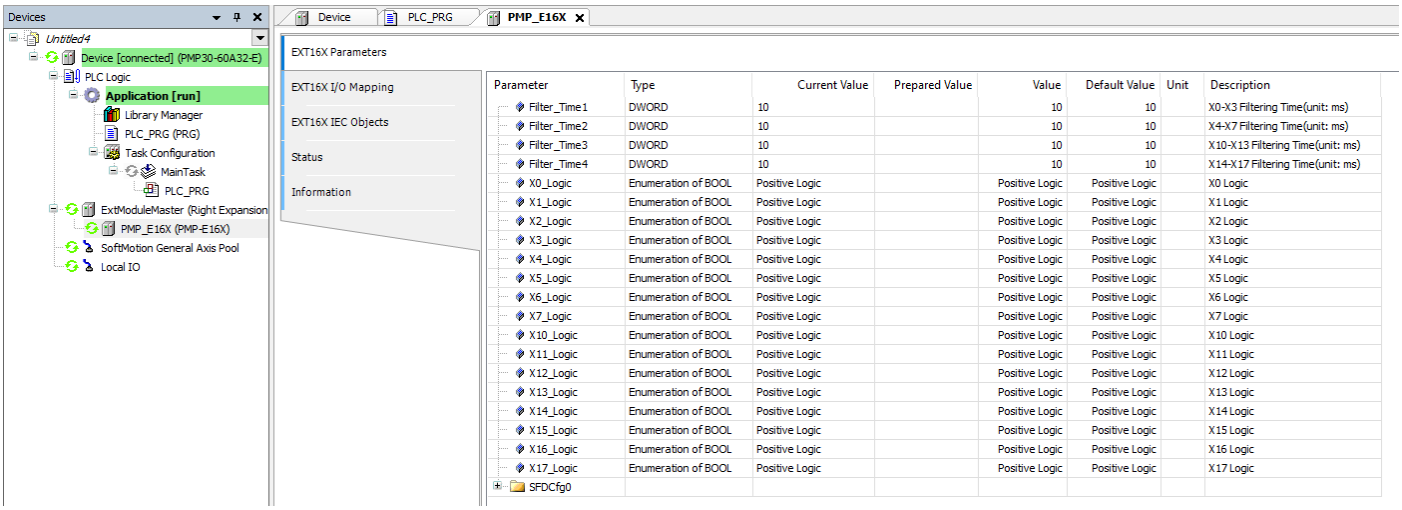
(1) После подключения локального модуля расширения к ПЛК щелкните правой кнопкой мыши устройство > добавить устройство > выберите "Мастер правого модуля расширения".



(2) Модуль расширения можно добавить с помощью сканирования или вручную.



(3) Результат тестирования.

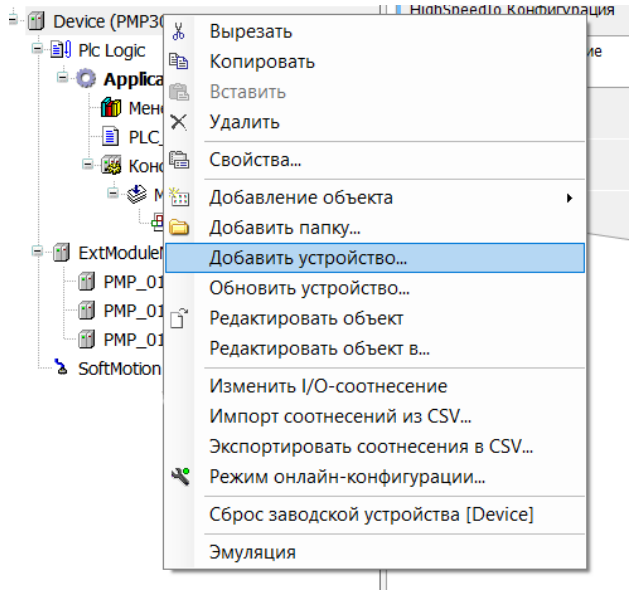


Parameter	Type	Current Value	Prepared Value	Value	Default Value	Unit	Description
Filter_Time1	DWORD	10		10	10		X0-X3 Filtering Time(unit: ms)
Filter_Time2	DWORD	10		10	10		X4-X7 Filtering Time(unit: ms)
Filter_Time3	DWORD	10		10	10		X10-X13 Filtering Time(unit: ms)
Filter_Time4	DWORD	10		10	10		X14-X17 Filtering Time(unit: ms)
X0_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X0 Logic
X1_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X1 Logic
X2_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X2 Logic
X3_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X3 Logic
X4_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X4 Logic
X5_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X5 Logic
X6_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X6 Logic
X7_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X7 Logic
X10_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X10 Logic
X11_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X11 Logic
X12_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X12 Logic
X13_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X13 Logic
X14_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X14 Logic
X15_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X15 Logic
X16_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X16 Logic
X17_Logic	Enumeration of BOOL	Positive Logic		Positive Logic	Positive Logic		X17 Logic
SFDCfg0							

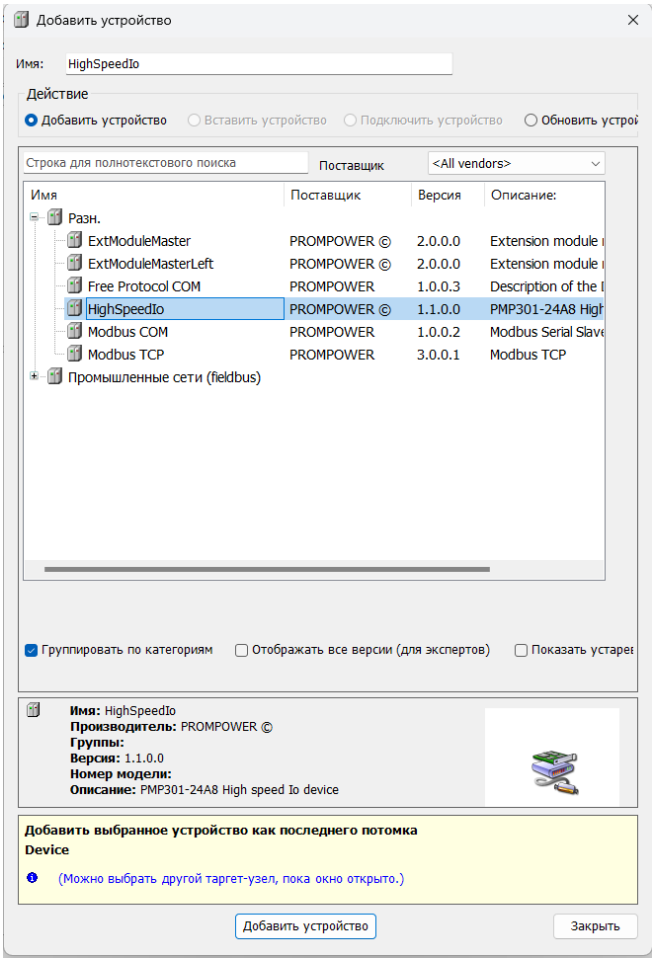
7.3 Для обращения ко входам и выходам ПЛК PMP301

Чтобы обращаться к входам и выходам ПЛК PMP301, нужно добавить модуль HighSpeedIo.

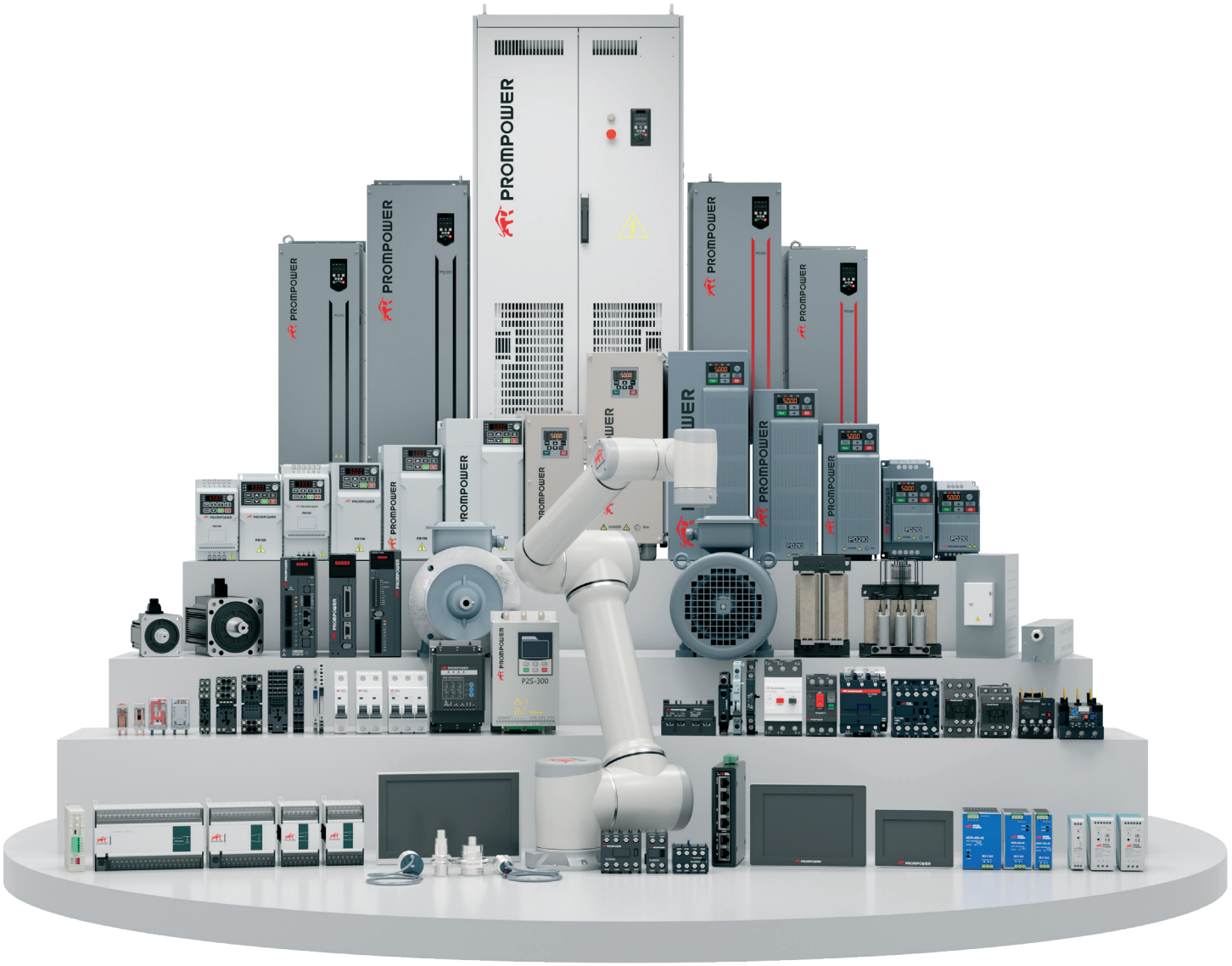
(1) Device -> Добавить устройство.



(2) HighSpeedIo -> Добавить устройство.



Профессиональное оборудование для промышленной автоматизации



Официальный дистрибьютор:

PROMPOWER

www.prompower.ru