ООО «Индас холдинг»

INDAS PRO

Программное обеспечение

для программирования ПЛК ATEKON

РУКОВОДСТВО ПО ПРОГРАММИРОВАНИЮ

NA2000

NA300

NA400

www.indas.ru

Новокузнецк

2024 год

Предупреждения

В данном руководстве содержатся предупреждения нескольких уровней, отмеченные следующими обозначениями:

Предупреждение – означает, что нарушение мер безопасности может вызвать ущерб.

Важно - указывает на информацию, которую необходимо учитывать.

Работа с продуктом или системой, описанными в этой документации, разрешена только квалифицированному персоналу, допущенному к выполнению задач и соблюдающему указания, включая меры безопасности. Такой персонал обладает необходимыми знаниями и опытом, чтобы распознавать риски и предотвращать возможные угрозы.

Использование ПЛК или системы допускается строго в рамках, указанных в данном руководстве. Для надежной и безошибочной работы необходимо обеспечить правильную транспортировку, хранение, установку, монтаж, ввод в эксплуатацию, обслуживание и своевременное поддержание работоспособности. Следуйте рекомендациям, указанным в документации.

Отказ от ответственности

Содержимое данного руководства проверено на соответствие описанному программному и аппаратному обеспечению. Однако не исключены разночтения или отклонения, поэтому полное соответствие не гарантируется. Руководство регулярно обновляется, а изменения вносятся в последующие редакции.

Содержание

Введение

Данное руководство пользователя было написано, чтобы помочь вам лучше создавать пользовательские программы, редактировать и отлаживать программы с помощью программного обеспечения INDAS PRO.

Программное обеспечение INDAS PRO – это интегрированная среда разработки, созданная для взаимодействия с программируемыми логическими контроллерами компанией Atekon. Данное ПО включает в себя редактор, компилятор, отладчик, симулятор и инструменты графического пользовательского интерфейса для конфигурации оборудования, конфигурации регистров измерения, программирования программного обеспечения, эмуляции, отладки и загрузки. Программное обеспечение предоставляет инженерам набор простых и практичных инструментов для программирования программного обеспечения и отладки в режиме онлайн, а также ряд функций, которые могут помочь достичь более высокой производительности и лучшей совместимости программного обеспечения. Программное обеспечение INDAS PRO может оптимизировать инвестиции клиента в программное обеспечение, сократить расходы на обучение и предоставить потенциал для разработки и совместимости за счет снижения стоимости разработки и оптимизации работы.

INDAS PRO предоставляет языки программирования релейных диаграмм (LD), функциональных блок-схем (FBD), списков инструкций (IL) и структурированного текста (ST) в соответствии со стандартом IEC61131-3, а также предоставляет уникальный язык программирования последовательных схем управления (SFC) и язык программирования диаграмм последовательности (SCC).

Особенности системы

Стиль Windows

В настоящее время для некоторых типов задач использование графических пользовательских интерфейсов стало основным требованием. По этой причине INDAS PRO разработан как прикладная программа MS Windows. INDAS PRO может работать в Windows XP, Windows Vista, Windows 10 и Windows 11. Данные операционные системы популярны во всем мире, и это является их преимуществом. Стиль INDAS PRO такой же, как у Windows, со стандартным меню, сочетанием клавиш, панелью инструментов и двойным нажатием мыши, что упрощает использование и сокращает время обучения и значение программирования.

Международный стандарт - МЭК 61131-3

Из-за различий в системе инструкций производителей ПЛК и требований к методам программирования МЭК разработала стандарт языка программирования МЭК 61131-3 на основе стандарта языка программирования Windows. Он установил пять языков программирования: IL, LD, SFC, FBD и ST, включая текстовое программирование (IL, ST) и графическое программирование (LD, FBD) и SFC, доступные в обоих типах языков программирования. Это стандартизированный документ, который направляет программируемые логические контроллеры к большей открытости на высоком уровне, основанном на цифровых технологиях, что является основной тенденцией в развитии ПЛК.

Программное обеспечение INDAS PRO обеспечивает унифицированные и эффективные среды конфигурации системы в соответствии с международным стандартом МЭК 61131-3. Инженеры могут изучить один раз и использовать везде.

Управление проектом - структура управления деревом

Программное обеспечение INDAS PRO использует концепцию управления проектами, отображает древовидную структуру в интегрированной среде разработки и наглядно отображает содержимое программы в многодокументном виде, благодаря чему соответствующий рабочий контент понятен, будь то разработка или обслуживание программы.

Язык программирования - язык программирования, совместимый с МЭК61131-3

Как решение промышленной автоматизации, INDAS PRO предоставляет языки программирования, совместимые с IEC61131-3: LD, IL, ST, FBD, SFC и новый язык программирования SCC. Программы, написанные на разных языках, могут вызываться друг у друга, что делает написание программ более гибким и удобным и может соответствовать требованиям различных сложных условий. LBD, LD, SCC используют графическое программирование, гибкое и удобное. Различные языки поддерживают вырезание, копирование, вставку, удаление, отмену, восстановление, поиск, замену и другие функции быстрого редактирования.

Режим программирования - вызов программных блоков

Программа управления (MAIN) состоит из программ с логической структурой. Используйте только один язык программирования в программе. Эти программы объединяются в полную программу управления для управления процессом. Различные программные блоки на языке IEC (LD, FBD, IL, ST, SFC, SCC) могут вызывать друг друга внутри программы.

Функционал операций - широкий функционал управления операциями

Программное обеспечение INDAS PRO имеет множество встроенных стандартных операторов, модулей функций управления и стандартных функций, а также обеспечивает импульсный ввод и вывод, переключение ведущего и ведомого устройств, интернет-связь и связь через последовательные порты, а также другие практичные функциональные модули, которые позволяют инженерам легко справляться с требованиями к управлению в сложных процессах и сокращать период разработки проекта.

Мониторинг - функция онлайн-мониторинга

В режиме «Онлайн» вы можете контролировать выполнение лестничной диаграммы, соединение красным цветом указывает на поток, а зеленым — на препятствие. Диаграмма управления последовательностью может не только контролировать, выполняется ли программа, но и контролировать выполнение программы шаг за шагом, используя метод контроля выполнения. В то же время она также может выполнять такие

операции, как синхронизация, сброс, переключение Master/Slave, и инженеры могут легко реализовывать различные функции.

Функция модификации - полная функция онлайн-модификации

В режиме «Онлайн» вы можете изменять параметры функциональных модулей и добавлять, удалять или перемещать функциональные модули. Изменения передаются непосредственно в ПЛК во время рабочего процесса без прерывания программ. Таким образом, изменение может вступить в силу в течение одного периода сканирования.

Отладка - функция онлайн-отладки

Релейная диаграмма (LD), таблица инструкций (IL), структурированный текст (ST), функциональная блок-схема (FBD) — все они поддерживают установку точек останова, работу в пошаговом режиме и другие функции онлайн-отладки.

Диаграммы управления последовательностью имеют функции автоматического выполнения, мониторинга выполнения и отладки. В онлайн-отладке три цвета используются для различения различных условий выполнения: те, которые не выполняются, отображаются серым цветом, те, которые выполняются, отображаются красным цветом, а те, которые выполнены, отображаются синим цветом. Инженеры могут отлаживать программы и легко находить ошибки, устанавливая точки останова, выполняя программу шаг за шагом, останавливая и перезапуская программы в любое время.

Наблюдение - функция онлайн-наблюдения в реальном времени

В режиме «Онлайн» всеми переключаемыми регистрами можно оперировать через таблицу регистров памяти (форсирование, присваивание, наблюдение), все переменные можно наблюдать через значения переменных, все события SOE можно проверить через таблицу событий SOE, всю информацию об авариях можно найти через окно «Авария», и все ошибки можно проверить через окно «Отладка». Данные можно отображать в десятичном, двоичном или шестнадцатеричном формате.

Симулятор - полная функция моделирования без аппаратного обеспечения

С помощью программной имитации ПЛК, аппаратные функции могут быть смоделированы, разработаны и отлажены идеально без ПЛК. Поведение целевой программы может быть воспроизведено точно, что значительно сокращает период разработки.

Инструменты диагностики - эффективные инструменты диагностики

INDAS PRO имеет комплексную функцию, применяемую при диагностике прикладных программ. Окно отладки отображает все сбои систем и программных блоков. В окне можно ввести место, где есть ошибки, всего лишь дважды щелкнув мышью.

Режим передачи - стандартный режим передачи файлов

Результаты программирования сохраняются, загружаются и скачиваются в виде файлов, что позволяет гарантировать единообразие всех настроек в программе

Поддержка программирования на английском языке

В INDAS PRO английский язык полностью поддерживается. В программе пользователи могут называть переменные на английском языке, а также комментировать и описывать их на английском языке

Функция печати - печать открытой области проекта

Программа позволяет распечатывать открытую в данный момент область проекта - всю созданную конфигурацию ПЛК, информацию о точках переключения, релейную диаграмму, функциональную блок-схему, схему управления последовательностью, список инструкций и структурированный текст, что удобно для архивирования.

Пользовательский интерфейс

INDAS PRO в полной мере использует преимущества графического и контекстно-зависимого интерфейсов Windows, обеспечивая максимальное удобство использования за счет оптимизированного использования экранного пространства, прямого доступа к инструментам и информации, а также аннотаций на русском, и английском языках

Требования к системе

Операционная система

INDAS PRO поддерживает распространенные операционные системы Windows, операционные системы Kirin и Linux, такие как Ubuntu, а также различные встраиваемые операционные системы Linux.

Аппаратное обеспечение

Рекомендуемая конфигурация:

- Процессор: 2 ГГц или больше
- Оперативная память: 4 ГБ или больше
- Жесткий диск: 32 ГБ или больше
- Видеокарта: DirectX 9 или выше
- Монитор: Разрешение 1024 x 768 и выше
- Сетевая карта: 10/100/1000 Мб.

Эксплуатация среды разработки

INDAS PRO включает в себя полный набор конфигураций ПЛК, а система разработки прикладного программного обеспечения соответствует стилю работы системы Windows, что удобно для изучения и использования.

Рабочие окна

Рабочий интерфейс

После запуска программного обеспечения INDAS PRO интерфейс отображается как на рисунке ниже. Среда разработки включает в себя следующие части: панель меню, панель инструментов, дерево проекта, таблицу выходной информации, строку состояния и окно программы. Положение каждой части показано как на рисунке 2.1

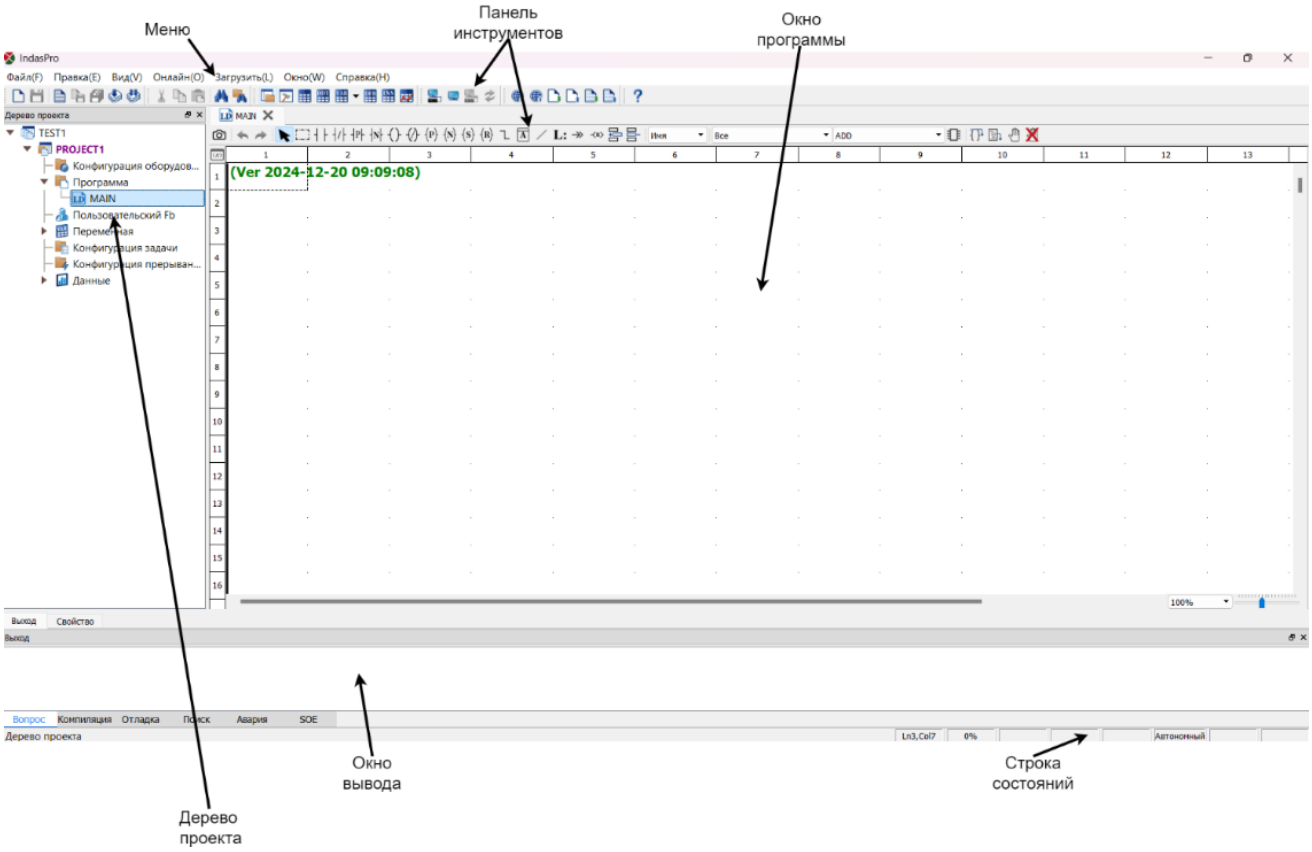


Рисунок 2.1 Интерфейс программы

Основная функция каждого окна

Строка меню: реализация основных функций программного обеспечения.

- **Панель инструментов**: создание, открытие, сохранение и другие файловые операции, вход в систему, загрузка, скачивание и другие онлайн-операции, а также панель инструментов каждого языка программирования.

- **Строка состояния**: строка состояния расположена в нижней части экрана. Справа от строки состояния находятся координаты программы, маркеры онлайн/офлайн, симуляция и обязательная информация о состоянии. В левой части строки состояния отображается каждое действие.
- **Окно программы**: отображение конфигурации системы, редактирования программы и отладки.
- **Дерево проекта**: модуль управления проектом и его структурой.
- **Таблица выходной информации**: отображение результатов поиска, компиляции и отладки программного обеспечения.

Главное меню или раскрывающийся список

Главное меню, реализующее основные функции программного обеспечения для программирования, включает в себя пункты Файл, Правка, Вид, Онлайн, Загрузить, Окно и Справка, как показано на рисунке 2.2.

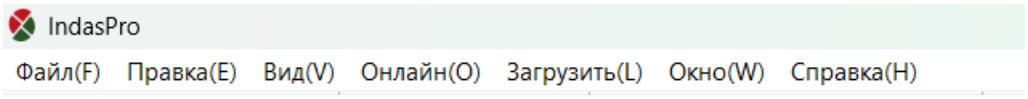


Рисунок 2.2 – Схема главного меню

В раскрывающемся меню щелкните мышью по любому элементу главного меню, значок элемента станет нажатым, и одновременно появится раскрывающееся меню. Когда мышь наводится на любую команду меню раскрывающегося списка, элемент становится синим, указывая на то, что операция была выбрана. Щелкните в любом месте за пределами меню или нажмите клавишу ESC, чтобы закрыть меню, как показано на рисунке 2.3:

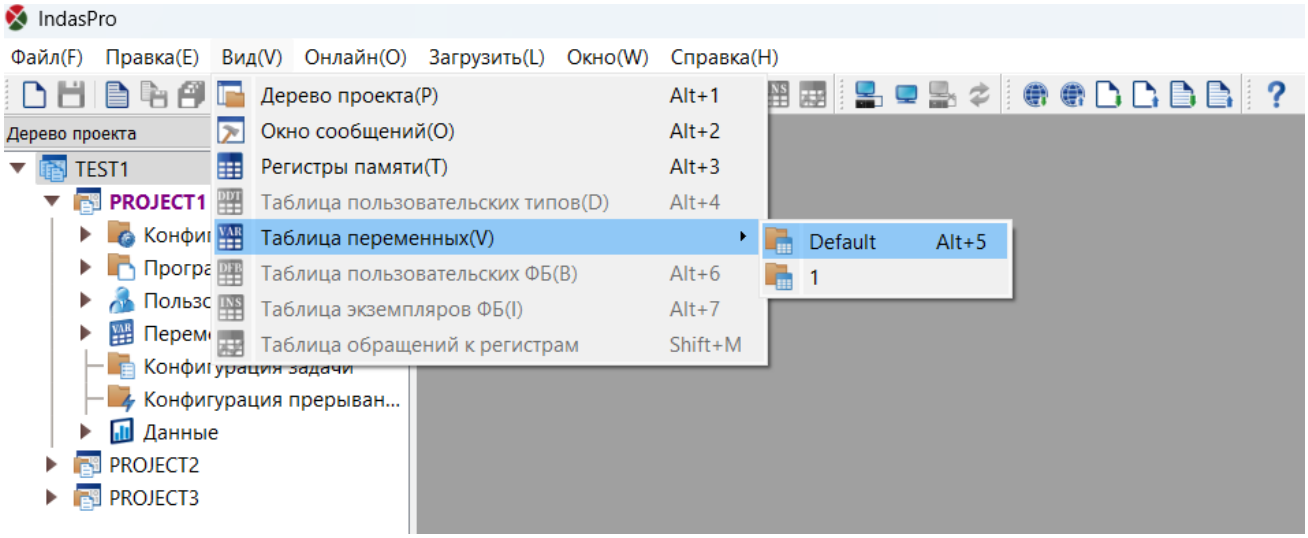


Рисунок 2.3 Подменю

Контекстное меню или всплывающее меню

Один раз щелкните элемент (правая кнопка мыши), чтобы открыть контекстное меню. Если вы выбрали более одного элемента, контекстное меню также может быть вызвано. В этой ситуации меню содержит только команды, которые эфективны для всех выбранных элементов. Щелкните в любом месте за пределами меню или нажмите клавишу ESC, чтобы закрыть меню, как показано на рисунке 2.4.

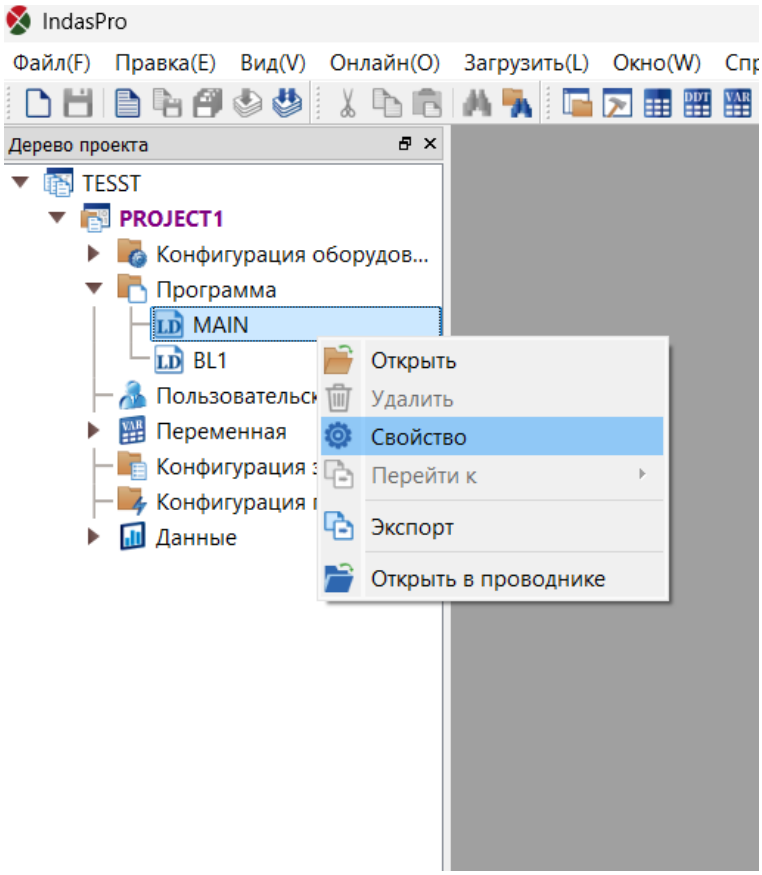


Рисунок 2.4 Контекстное меню

Разделы меню

Файл

Раздел меню «Файл» используется для управления файлами, ее раскрывающийся список в основном включает в себя «Новый», «Открыть», «Сохранить», «Закрыть», «Новый проект», «Сохранить проект», «Новая программа», «Сохранить программу», «Сохранить все программы», «Компилировать программу», «Компилировать все программы», «Пароль», «Число обращений», «Точка экспорта», «Точка импорта», «Печать», «Предварительный просмотр печати», «Выход», как показано на рисунке 2.5:

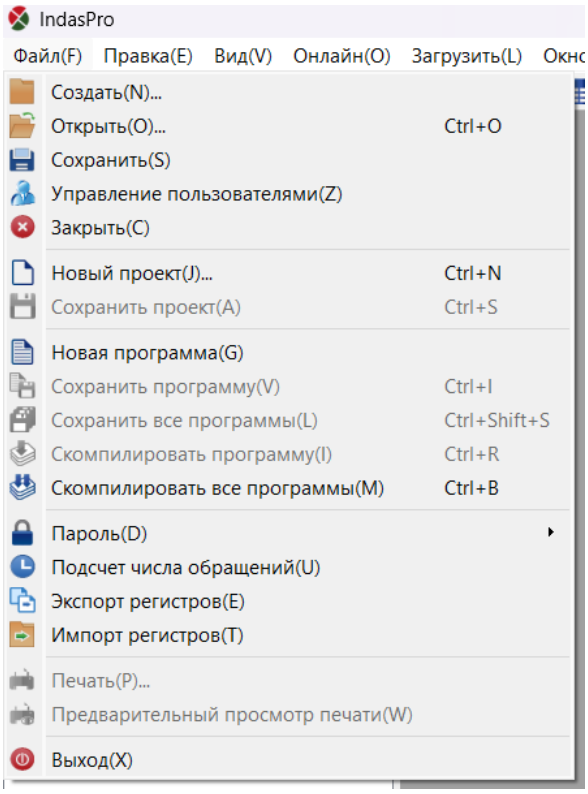


Рисунок 2.5 Функции раздела меню «Файл»

Функции:

[Новый]: Создание нового рабочего пространства. Оно может включать несколько файлов проекта, включая базу данных, лестничную диаграмму, функциональную блок-схему, диаграмму управления последовательностью, список инструкций, структурированный текст и т. д.

[Открыть]: Открытие существующего рабочего пространства. Нажмите «Открыть», и откроется диалоговое окно. Тип файла — INDAS PRO Workspace Files, суфикс — .wsp.

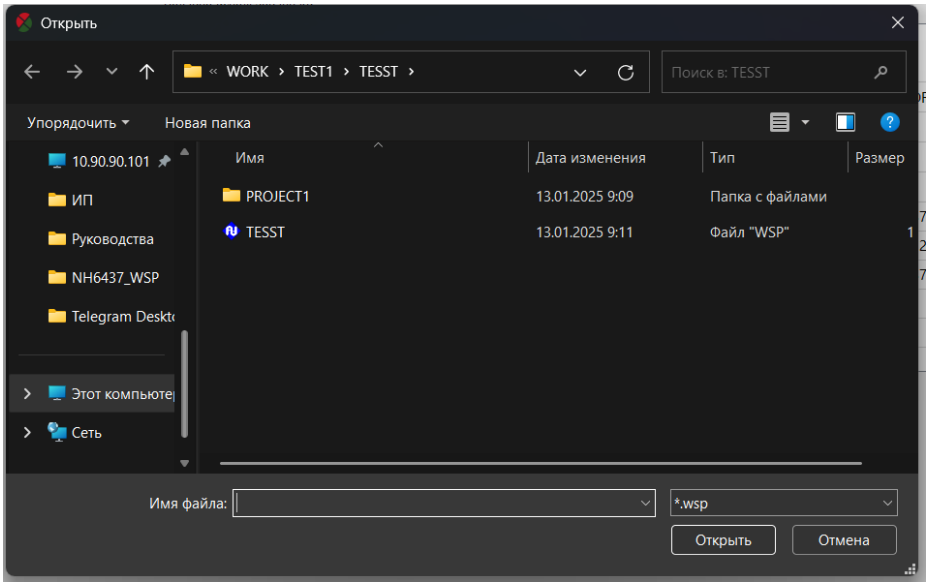


Рисунок 2.6 Открытие файла проекта

Если проект защищен паролем, также откроется диалоговое окно ввода пароля, показанное на рисунке 2.7.

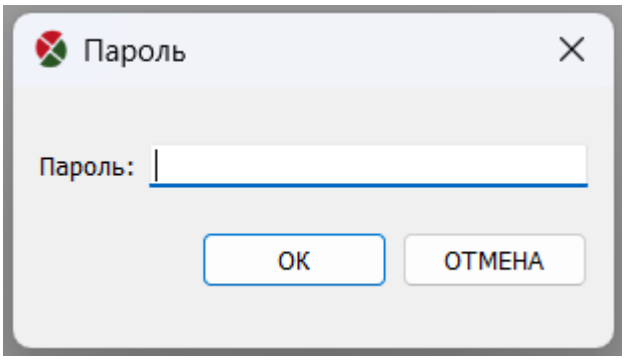


Рисунок 2.7 Ввод пароля

[**Сохранить**]: Сохранение файла. Если редактируемый вами файл уже существует, исходный файл будет перезаписан напрямую.

> **Предупреждение**
> Изменение файлов проекта может повлиять на все программы, поэтому обязательно скомпилируйте все программы перед загрузкой.

[**Закрыть**]: Закрытие текущего редактируемого файла проекта.

[**Новый проект**]: Добавление нового файла проекта ПЛК в текущее рабочее пространство

[**Сохранить проект**]: Сохранение активного файла проекта в текущем проекте.

[**Новая программа**]: Создание новой программы, включая лестничную диаграмму, схему управления последовательностью, список инструкций, структурированный текст и т. д. Новая программа должна быть названа, и ее также можно описать, а содержимое описания отображается в поле редактора программ, как показано на рисунке 2.8.

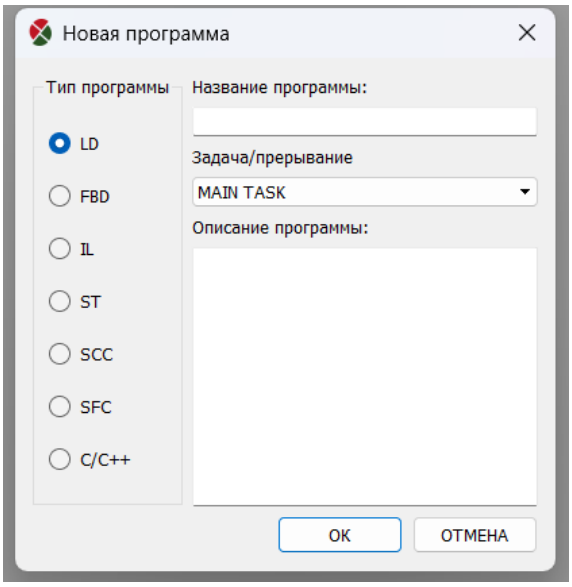


Рисунок 2.8 Создание новой программы

[**Сохранить программу**]: Сохранение редактируемой программы. Если выбрано [Сохранить программу], то независимо от того, существует программа или нет, программное обеспечение для программирования сохранит ее автоматически.

[**Сохранить все программы**]: Сохранение всех открытых программ.

[**Скомпилировать программу**]: Компиляция текущей программы. Если программа была изменена, программное обеспечение для программирования выведет диалоговое окно сохранения при компиляции (рисунок 2.9). Затем программа автоматически проверит ошибки в программе. Если есть ошибки, компиляция завершится неудачей, а места, типы и количество ошибок будут показаны в окне «Компиляция», как показано на рисунке 2.10.

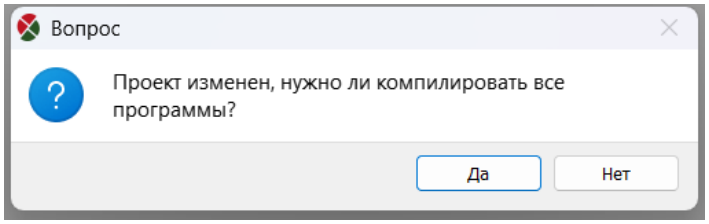


Рисунок 2.9 Окно предупреждения

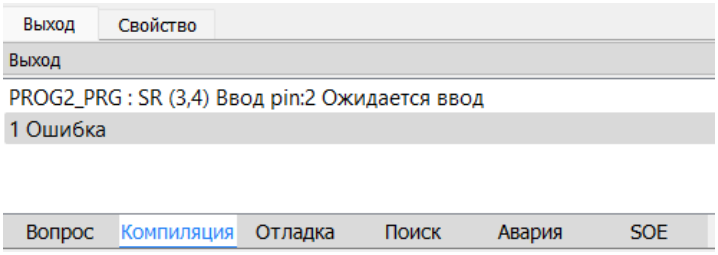


Рисунок 2.10 Ошибка при компиляции программы

[**Скомпилировать все программы**]: Компиляция всех программ. Так же, как и при компиляции одной программы, программное обеспечение для программирования выведет диалоговое окно сохранения при компиляции. Затем программа автоматически проверит ошибки в программе. Если есть ошибки, компиляция завершится неудачей, а места, типы и количество ошибок будут показаны в окне «Компиляция».

[**Пароль**]: Изменение пароля проекта. Новый проект не имеет пароля файла, поэтому поле «Текущий пароль» нужно оставить пустым, как показано на рисунке 2.11:

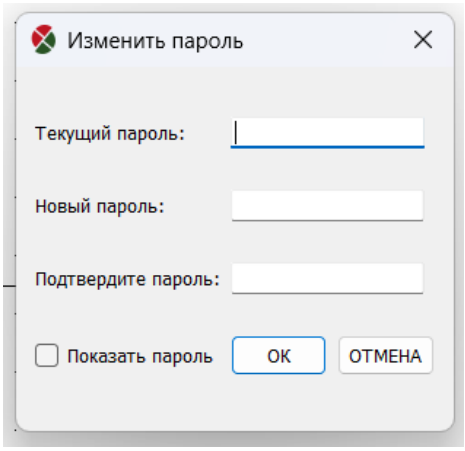


Рисунок 2.11 Смена пароля проекта

Важно
Пожалуйста, запомните пароль после его изменения, в противном случае его невозможно будет восстановить и доступ к проекту будет закрыт.

[**Число обращений**]: Подсчет количества использования каждой точки, как показано на рисунке 2.12.

Регистры памяти

Регистр	Название	Описание	Число обращений	Значение
%I00001			1	
%I00002			1	
%I00003			0	
%I00004			0	
%I00005			0	
%I00006			0	
%I00007			0	
%I00008			0	
%I00009			0	
%I00010			0	
%I00011			0	
%I00012			0	
%I00013			0	
%I00014			0	
%I00015			0	
%I00016			0	
%I00017				
%I00018				
%I00019				

Рисунок 2.12 Число обращений

[**Экспорт регистров**]: Экспорт регистров, их названия и описания в файл Excel.

[**Импорт регистров**]: Импорт регистров из Excel.

[**Печать**]: Печать конфигурации ПЛК, информации о точках, LD, FBD, SCC, IL, ST. Когда пользователи выбирают Печать, всплывает диалоговое окно печати, и пользователи могут выбрать принтер и диапазоны печати, копии и т. д.

[**Предварительный просмотр**]: Предварительный просмотр результатов печати конфигурации ПЛК, информации о точках, LD, FBD, SCC, IL и ST.

[**Выход**]: выхода из программного обеспечения INDAS PRO.

Правка

Раздел «Правка» имеет необходимые функции для редактирования программ, такие как «Вырезать», «Копировать», «Вставить», «Удалить», «Выбрать все», «Найти», «Заменить» и «Глобальный поиск», как показано на рисунке 2.13.

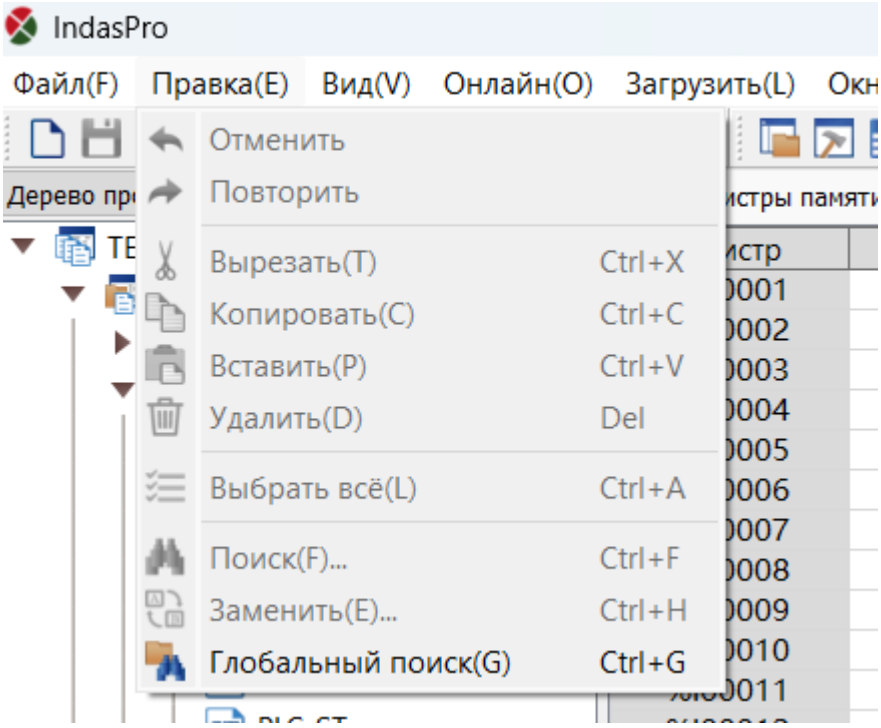


Рисунок 2.13 Функции раздела «Правка»

Функция панели редактирования

[**Отменить**]: Отмена последнего выполненного действия.

[**Повторить**]: Возврат программы к состоянию до выполнения команды «Отмена».

[**Вырезать**]: Удаление выбранного содержимого и его перенос в буфер обмена, содержимое которого можно вставить. Выбранное содержимое может быть функциональным модулем (контакт, катушка и специальный функциональный модуль), функциональным блоком в схеме управления последовательностью, инструкцией в списке инструкций,

оператором в тексте структуры или содержимым области, выбранной блочной операцией.

[**Копировать**]: Перенос выбранного содержимого в буфер обмена, без удаления.

[**Вставить**]: Перенос содержимого из буфера обмена в то место, где щелкает мышь.

[**Удалить**]: Удаление выбранного контента.

Примечание: Вырезать, Копировать, Вставить и Удалить также можно, щелкнув правой кнопкой мыши в области редактирования и вставив в пустое место.

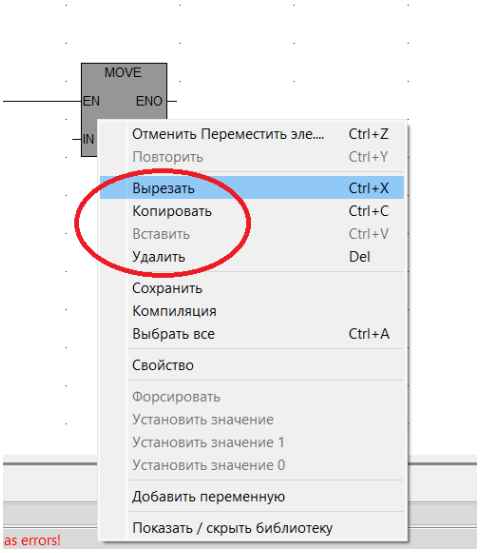


Рисунок 2.14 Вырезать, Копировать, Вставить, Удалить

[**Выбрать все**]: Выделение всего содержимого в области редактирования (рисунок 2.15).

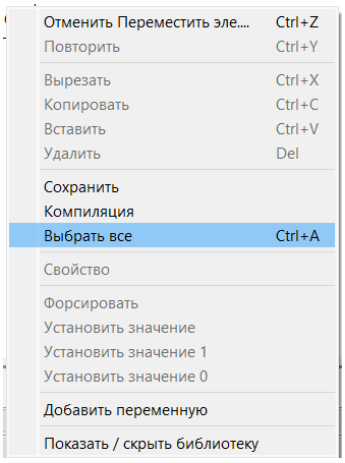


Рисунок 2.15 Команда «Выбрать все»

[**Поиск**]: Поиск функционального модуля, блока функций, инструкции или оператора, которые соответствуют вашим требованиям. Поиск ограничен текущей рабочей областью.

Если текущая рабочая область — LD, поиск ограничен LD. Например, если вы хотите найти параметр %M1, выберите [Изменить]/[Найти], после чего появится диалоговое окно поиска. Введите содержимое %M1 и нажмите [Найти]. Он автоматически перейдет к первому функциональному модулю, который соответствует требованиям, и функциональный модуль будет отображен оранжевым цветом. Если вы хотите найти другой соответствующий функциональный модуль, просто нажмите [Найти], и он перейдет к %M1 один за другим, как показано на рисунке 2.16.

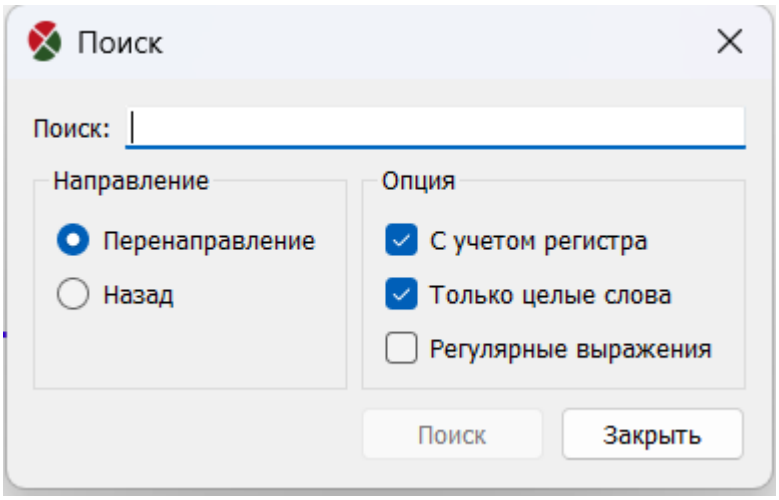


Рисунок 2.16 Поиск

Если текущая рабочая область представляет собой диаграмму управления последовательностью, она находится в текущей диаграмме управления последовательностью. Функция поиска диаграммы управления последовательностью находится не только в операторе выполнения функционального блока диаграммы управления последовательностью, но и в ее описании, как показано на рисунке 2.17.

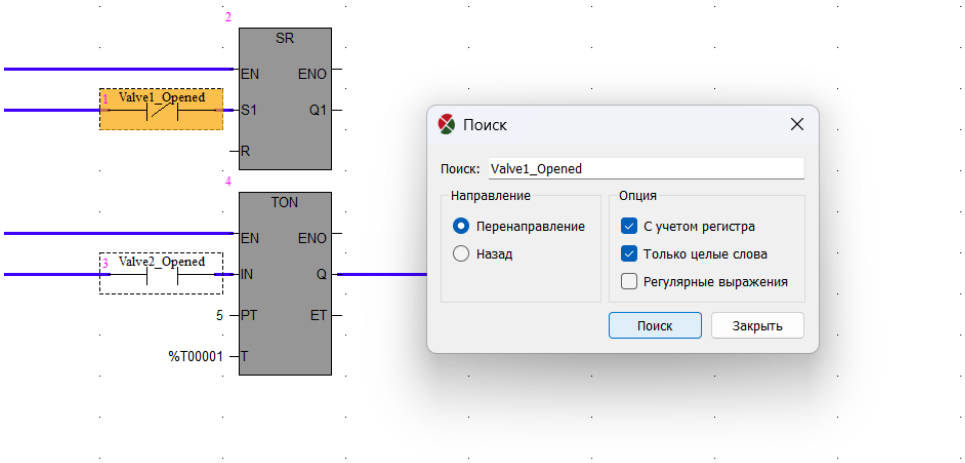


Рисунок 2.17 Поиск определенного объекта

Если текущая рабочая область представляет собой список инструкций, поиск выполняется в текущем списке инструкций.

Примечание: Если вы хотите выполнить поиск R1 и ввести «r1» в содержимом поиска, параметр с учетом регистра выбрать нельзя, в противном случае поиск невозможен, как показано на рисунке 2.18, что соответствует поиску в ST.

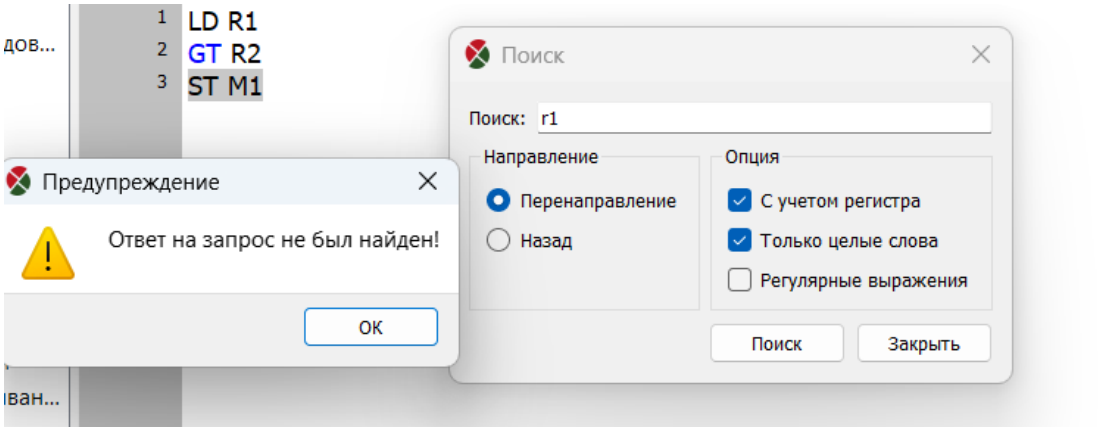


Рисунок 2.18 Учет регистра в команде «Поиск»

[**Заменить**]: Поиск функционального модуля, функционального блока, инструкцию или оператор, которые соответствуют требованиям в текущей программе, и заменить соответствующие параметры. Функция замены похожа на функцию поиска, за исключением того, что есть функция замены, о которой также можно сказать, что [Заменить] включает функцию поиска. Эта функция может заменять выборочно или заменять все соответствующие модули. Замена ограничена в текущей рабочей области.

[**Глобальный поиск**]: Поиск функционального модуля, блока функций, инструкции или оператора, которые соответствуют вашим требованиям. Эта функция выполняется во всех программах (LD, FBD, SCC, IL, ST и т. д.). После завершения глобального поиска результаты отображаются в строке информации о поиске, включая местоположение и номер.

Вид

Меню в основном включает в себя дерево проекта, вывод, регистры памяти, DDT, VAR, DFB и INS, как показано на рисунке 2.19.

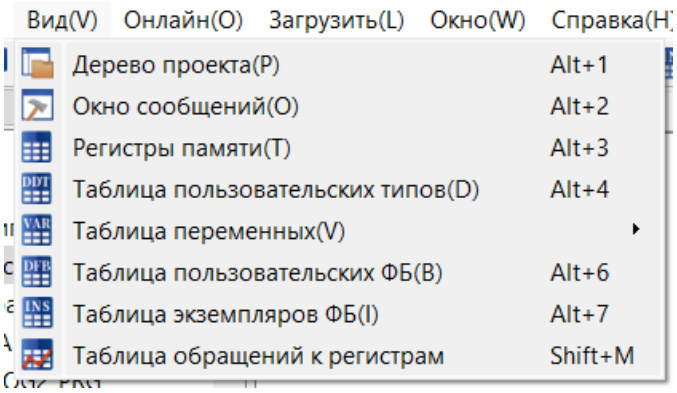


Рисунок 2.19 Функции раздела «Вид»

Функции раздела «Вид»

[**Дерево проекта**]: Отобразить или скрыть дерево проектов.

[**Сообщение**]: Отобразить или скрыть окно сообщений.

[**Таблица регистров**]: Отобразить или скрыть регистры памяти.

[**Таблица пользовательских типов**]: Отобразить или скрыть таблицу пользовательских типов.

[**Таблица переменных**]: Отобразить или скрыть группы переменных.

[**Таблица пользовательских FB**]: Отобразить или скрыть таблицу пользовательских функциональных блоков.

[**Таблица экземпляров FB**]: Отобразить или скрыть таблицу экземпляров функциональных блоков.

[**Таблица обращений к регистрам**]: Открыть таблицу обращений к регистрам

Онлайн

Раздел «Онлайн» включает в себя команды «Подключить», «Отключить», «Формат отображения», «Обновить программу», «Отменить принудительный режим», «Сброс», «Установить время», «Переключение главный/подчиненный», как показано на рисунке 2.20.

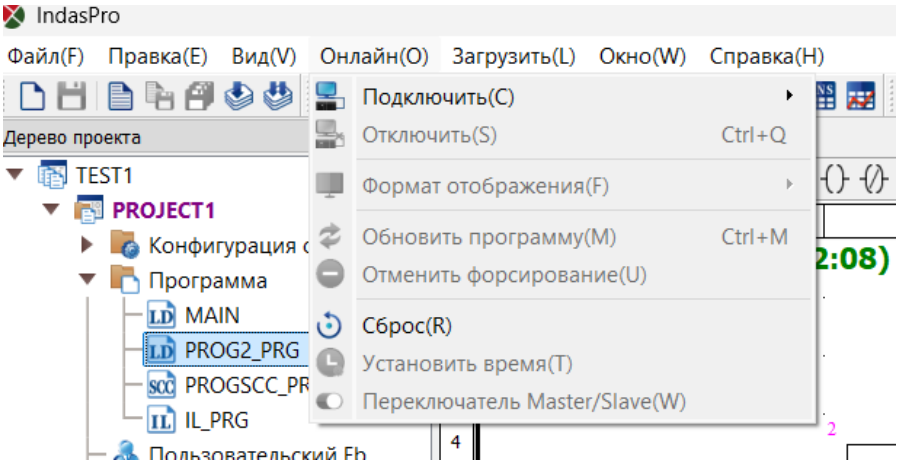


Рисунок 2.20 Функции раздела «Онлайн»

Функции раздела «Онлайн»:

[**Подключить**] / [**Подключить ПЛК**]: Подключите текущий используемый отладочный компьютер к ПЛК. Перед подключением к сети убедитесь, что сеть физически подключена; в противном случае программное обеспечение для программирования отобразит диалоговое окно. Как показано на рисунке 2.21:

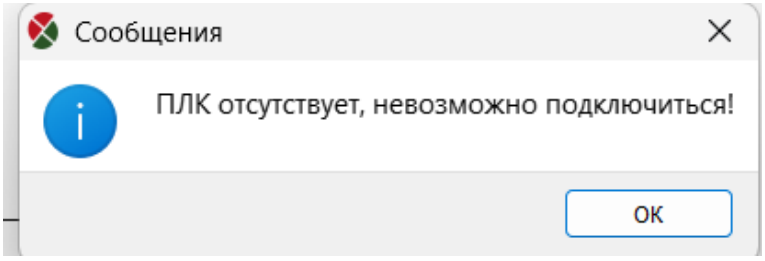


Рисунок 2.21 Отсутствие соединения с ПЛК

Необходимость подключения ПЛК:

IP-адрес отладочного компьютера должен находиться в том же домене, что и IP-адрес ПЛК, то есть первые три сегмента IP-адреса должны быть одинаковыми, в противном случае подключение невозможно. Например, если IP-адрес ПЛК 192.168.1.100, IP-адрес отладочного компьютера должен быть 192.168.1.***.

Программное обеспечение автоматически ищет ПЛК в сети в соответствии с IP-адресом Ethernet модуля ЦП в конфигурации ПЛК, и никаких других настроек не требуется.

Описание после успешного подключения

После успешного подключения фоновый цвет области редактирования программы станет светло-фиолетовым. Информация о данных в ПЛК будет передана в программное обеспечение программирования через Интернет. Значение параметра равно 1, а

соединение в проводящем состоянии показано красным цветом. Значение параметра равно 0, а соединение в непроводящем состоянии показано зеленым цветом, как показано на рисунке 2.22:

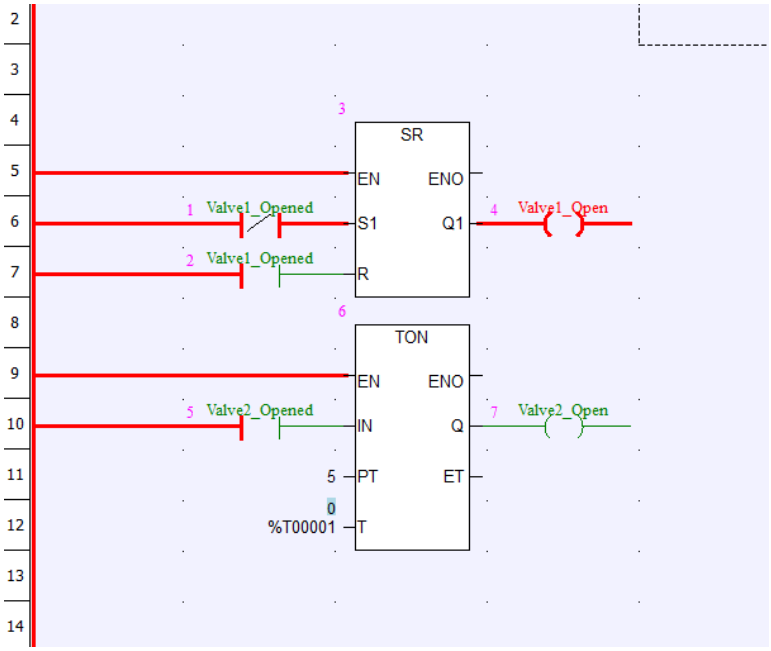


Рисунок 2.22 Программа на LD в режиме «Онлайн»

[**Подключить**]/[**Подключить Симулятор**]: Имитация онлайн-состояния ПЛК. Пользователь может принудительно задать фактическую точку и значение переменной, имитировать онлайн-состояние для отладки программы. Функции сетевой связи, запланированные прерывания и запланированные задачи не могут быть смоделированы.

[**Отключить**]: Отключение отладочного компьютера от устройства ПЛК или выход из режима моделирования.

[**Формат отображения**]: Формат отображения целых данных (не bool) в режиме онлайн. Можно выбрать три формата отображения: десятичный, шестнадцатеричный или двоичный. Перед выбранным режимом стоит знак √.

[**Обновить программу**]: Удобно для отладки онлайн-программ. В режиме онлайн, если пользователь добавляет, удаляет или перемещает функциональный блок или изменяет параметры функционального блока, справа от имени программы появится *. В это время пользователю необходимо обновить программу, чтобы загрузить изменяемую часть в ПЛК. После изменения, перед выходом из программного обеспечения для программирования, обязательно сохраните измененную программу; в противном случае это приведет к несоответствию между верхней и нижней программами. Перед обновлением программное обеспечение для программирования автоматически скомпилирует программу, если есть ошибка, программа не будет обновлена, и ошибка будет

указана. После обновления программы ПЛК может быть напрямую выполнен в соответствии с загруженной программой, и системе не нужно перезапускаться. Поскольку команда обновления изменяет только исполняемую программу, а исходная программа, сохраненная в ПЛК, не изменяется, если вы загружаете программу в это время, она все еще остается старой программой, поэтому рекомендуется сохранить измененную программу после онлайн-изменения и полностью загрузить проект в ПЛК после определения программы.

[**Отменить присваивание**]: В режиме соединения сканируемый статус сигнала DI, DO, AI, AO, который принудительно установлен, не будет отправлен в соответствующую область памяти. Пользователи могут устанавливать значения в соответствии с потребностями отладки независимо от фактического статуса поля. Unforce означает, что все форсированные переменные выходят из принудительного статуса и возобновляют сканирование.

[**Сброс**]: Сбросьте модуль ЦП ПЛК через Интернет. Для двухпроцессорной системы сбросьте оба ЦП одновременно.

[**Установить время**]: Установите время для ПЛК через интернет в режиме онлайн. Для двухпроцессорной системы установите время для двух ЦП одновременно. Но установленное время — это время отладочного компьютера, а не стандартных часов.

[**Переключатель Master/Slave**]: Действует только для двухпроцессорной системы. Когда система настроена с двумя процессорами, один работает как ведущий, а другой как ведомый. Когда пользователи выбирают [Переключатель Master/Slave], ведущий становится ведомым, а ведомый работает как ведущий.

Загрузить

См. соответствующее содержание загрузки и выгрузки программного файла и файла проекта в разделе «Загрузка и выгрузка» главы 3 «Управление проектом».

Окно

Раздел «Окно» включает в себя библиотеку модулей, библиотеку функциональных блоков, раздел окна вывода информации «Свойство», несколько вариантов расположения открытых окон («Каскад», «Горизонтально», «Вертикально»), а также может сохранить макет или загрузить макет (рисунок 2.23)

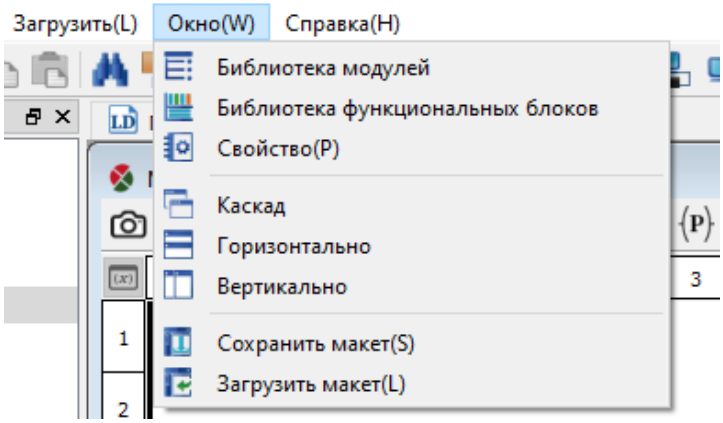


Рисунок 2.23 Функции раздела «Окно»

Справка

Раздел «Справка» включает в себя разделы «Содержание», «Указатель», «Поиск», «Язык», «О программе», как показано на рисунке 2.24.

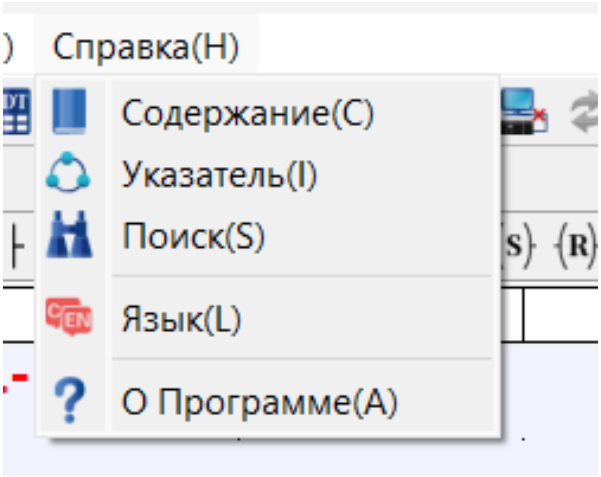


Рисунок 2.24 Функции раздела «Справка»

Функция панели помощи

[**Содержание**]: Отображение содержимого файла справки.

[**Указатель**]: Отображение поисковой строки файла справки.

[**Поиск**]: Отображение поиска по файлу справки.

[**Язык**]: Отображение языка справки, чтобы изменить язык программного обеспечения, как показано на рисунке 2.25.

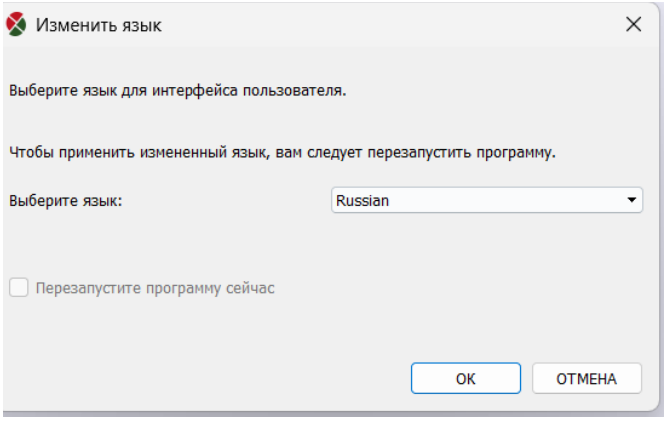


Рисунок 2.25 Смена языка программы

[О программе]: Отображение версии программного обеспечения и авторских прав, как показано на рисунке 2.26.

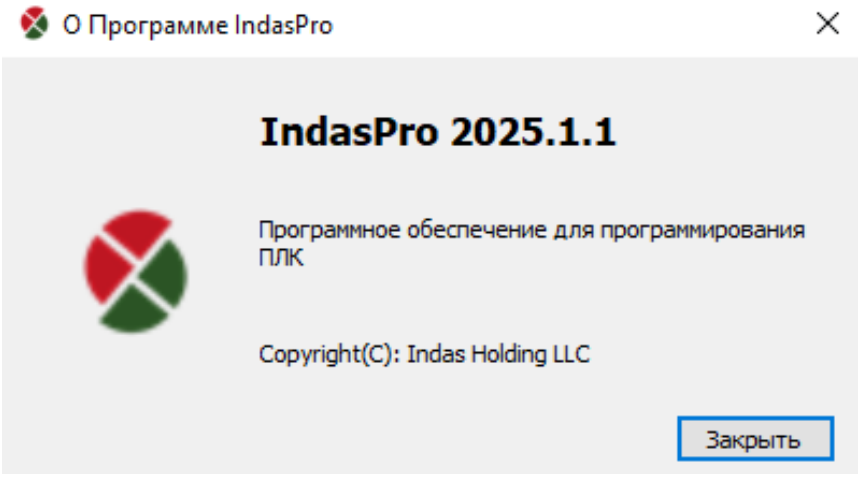


Рисунок 2.26 Версия программного обеспечения

[О программе]: Отображение версии программного обеспечения и авторских прав, как показано на рисунке 2.26.

Системная панель инструментов

Панель инструментов системы предназначена для размещения иконок некоторых часто используемых функций над областью редактирования, что облегчает работу пользователей. Все функции, которых они достигают, могут быть реализованы через строку меню, поэтому просто представьте соответствующую операцию меню, а подробную функцию можно найти во введении к строке меню.

	Новый проект, соответствующий [Файл] / [Новый проект].
	Сохранить проект, соответствующий [Файл] / [Сохранить проект].
	Новая программа, соответствующая [Файл] / [Новая программа].
	Сохранить программу, соответствует [Файл] / [Сохранить программу].
	Сохранить все программы, что соответствует [Файл] / [Сохранить все программы].
	Компилировать программу, соответствующую [Файл] / [Компилировать программу].
	Компилировать все программы, что соответствует [Файл] / [Компилировать все программы].
	Вырезать, соответствует [Редактировать] / [Вырезать].
	Копировать, соответствует [Изменить] / [Копировать].
	Вставить, соответствует [Изменить] / [Вставить].
	Найти, соответствует [Изменить] / [Найти].
	Глобальный поиск, соответствующий [Изменить] / [Глобальный поиск].
	Дерево проекта, соответствующее [Просмотр] / [Дерево проекта].
	Вывод, соответствующий [Просмотр] / [Вывод].
	Открыть регистры памяти переключения, соответствующую [Вид] / [Таблица регистров переключения].
	Открыть таблицу DDT, соответствующую [Просмотр] / [DDT].
	Открыть таблицу VAR, соответствующую [View] / [VAR].

	Открыть таблицу DFB, соответствующую [Просмотр] / [DFB].
	Открыть таблицу INS, соответствующую [Просмотр] / [INS].
	Подключение ПЛК, соответствующее [Онлайн] / [Подключиться] / [Подключить ПЛК].
	Simulator Connect, соответствующий [Online] / [Connect] / [Simulator Connect].
	Отключить, что соответствует [В сети] / [Отключить].
	Обновить программу, что соответствует [Онлайн] / [Обновить программу].
	Загрузить все, что соответствует [Загрузить] / [Загрузить все].
	Загрузить все, что соответствует [Загрузить] / [Загрузить все].
	Загрузить проект, соответствующий [Загрузить] / [Загрузить проект].
	Загрузить проект, соответствующий [Загрузить] / [Загрузить проект].
	Загрузка программы, соответствующая [Загрузить] / [Загрузить программу].
	Программа загрузки, соответствующая [Загрузить] / [Загрузить программу].
	О программе, соответствует [Справка] / [О программе].

Панель инструментов функциональных блоков

Поскольку существуют различные функциональные блоки, они выбираются с помощью двух раскрывающихся списков. Первый для выбора типа, а второй для выбора конкретного функционального блока. Например, текущий тип на рисунке — Move, тогда все функциональные блоки перемещения находятся во вторых списках для выбора. При нажатии на значок выбранный функциональный блок можно поместить в область редактирования напрямую, как показано на рисунке 2.27.

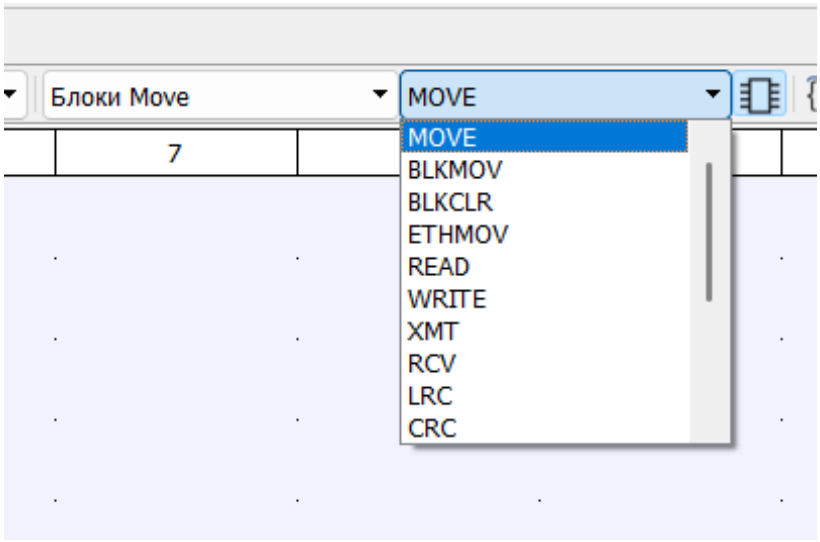


Рисунок 2.27 Выбор функционального блока

В интерфейсе редактирования LD пользователи могут ввести имя функционального блока или первую букву для размещения функционального блока, как показано на рисунке 2.28.

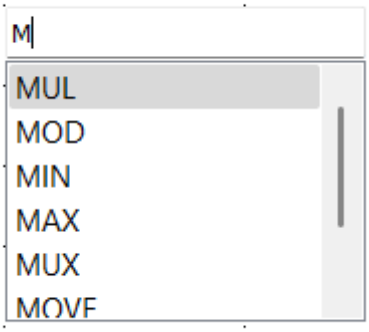


Рисунок 2.28 Ручной ввод имени функционального блока

Панель инструментов LD

Панель инструментов LD включает в себя значки некоторых операций и функциональных блоков для удобства работы с программой.

	Переместить. Если не выбран ни один функциональный блок, область редактирования всегда находится в состоянии перемещения. Положение выбранного функционального модуля можно изменить движением мыши. При перемещении функционального модуля удерживайте левую кнопку мыши, переместитесь в указанное положение, а затем отпустите.
	Блок. Операция блока используется для выбора всех линий и функциональных блоков в определенной области. Используйте мышь, чтобы обвести нужную область, и все элементы будут выбраны. Перемещение, вырезание, копирование, удаление и некоторые другие операции для нескольких элементов должны выполняться операцией блока.
	Ссылка. Ссылка используется для размещения пути тока между выводами двух функциональных модулей. Переместите указатель мыши на первый вывод параметра, который необходимо подключить, щелкните левой кнопкой мыши, чтобы выбрать первый вывод параметра; Затем переместите указатель мыши на второй вывод параметра, щелкните левой кнопкой мыши, и между двумя выводами параметра появится линия. Если выбранные два вывода параметра не соответствуют правилам подключения, появится всплывающее окно с подсказкой.
	Обратный. Когда выбранный функциональный модуль является контактом, обратная функция может циклически переключаться между нормально открытыми, нормально закрытыми, положительными и отрицательными контактами. Если выбранным функциональным модулем является катушка, то обратная функция может использоваться для циклического преобразования между нормально открытыми, нормально закрытыми, положительными и отрицательными катушками, а также катушками установки и сброса.
	Нормально разомкнутый контакт. Клавиша быстрого доступа: F2.
	Нормально замкнутый контакт. Клавиша быстрого доступа: F3.
	Контакт с положительным переключением. Клавиша быстрого доступа: F4.

	Отрицательный переходный чувствительный контакт. Клавиша быстрого доступа: F5.
	Катушка. Клавиша быстрого доступа: F6.
	Катушка с отрицательным переходом. Клавиша быстрого доступа: F7.
	Катушка с положительным переходом. Клавиша быстрого доступа: F8.
	Катушка, реагирующая на отрицательный переход. Клавиша быстрого доступа: F9.
	Установить катушку. Клавиша быстрого доступа: F10.
	Катушка сброса. Клавиша быстрого доступа: F11.
	Метка.
	Перейти.
	Вернуться.
	Комментарий.
	Вставить одну строку.
	Удалить одну строку.
	Шаг.
	Продолжить.
	Вставить/удалить точку останова.
	Удалить все точки останова.

Панель инструментов FBD

Панель инструментов FBD включает в себя значки некоторых операций и функциональных блоков для удобства работы с программой.

	Двигаться.
	Блокировать.
	Связь.
	Отрицать.
	Комментарий.
	Вставить одну строку.
	Удалить одну строку.
	Шаг.
	Продолжить.
	Вставить/удалить точку останова.
	Удалить все точки останова.

Панель инструментов IL

Панель инструментов IL включает в себя значки некоторых операций и функциональных блоков для удобства работы с программой.

LD	Инструкция LD.
LDN	Инструкция LDN.
ST	Инструкция ST.
S	Инструкция S.
R	Инструкция R.
AND	И инструкция.
OR	ИЛИ инструкция.
XOR	Инструкция XOR.
GT	Инструкция GT.
GE	Инструкция GE.
LT	Инструкция LT.
LE	Инструкция LE.
EQ	Инструкция по эквалайзеру.
NE	Инструкция NE.
JMP	Инструкция JMP.
CAL	Инструкция CAL.
RET	Инструкция RET.
	Шаг.
	Продолжить.
	Вставить/удалить точку останова.
	Удалить все точки останова.

Панель инструментов ST

Панель инструментов ST включает в себя значки некоторых операций и функциональных блоков для удобства работы с программой.

:=	Присвойте значение.
IF	Оператор ЕСЛИ.
CASE	Оператор CASE.
FOR	Заявление «ЗА».
WHILE	Оператор WHILE.
REPEAT	Повторите оператор.
EXIT	Оператор EXIT.
RETURN	Оператор RETURN.
	Шаг.
	Продолжить.
	Вставить/удалить точку останова.
	Удалить все точки останова.

Панель инструментов SCC

Панель инструментов SCC включает в себя значки некоторых операций и функциональных блоков для удобства работы с программой.

	Двигаться.
	Блокировать.
	Стартовая коробка.
	Конечная коробка.
	Казненное поле.
	Состояние коробки.
	Ограниченное по времени условие.
	Разъем 1.
	Разъем 2.
	Связь.
	Комментарий.
	Автоматический.
	Смотрю.
	Отладка.
	Останавливаться.
	Замок.
	Разблокировать.
	Перезапуск.
	Шаг.
	Продолжить.
	Вставить/удалить точку останова.
	Удалить все точки останова.

Окно выводимой информации

Таблица выходной информации в основном включает вопросы, результаты компиляции, состояния отладки, результаты поиска, сигналы тревоги и информацию о событиях SOE, что обеспечивает удобство компиляции, отладки и мониторинга состояния программ.

[**Проблемы**]: Окно вывода информации о вопросе. В этом окне выводится информация о версии программного обеспечения и проблемах при выполнении программ.

[**Компиляция**]: Окно вывода информации о результатах компиляции. Дважды щелкните информацию об ошибке, чтобы перейти к месту ошибки, как показано на рисунке 2.29.

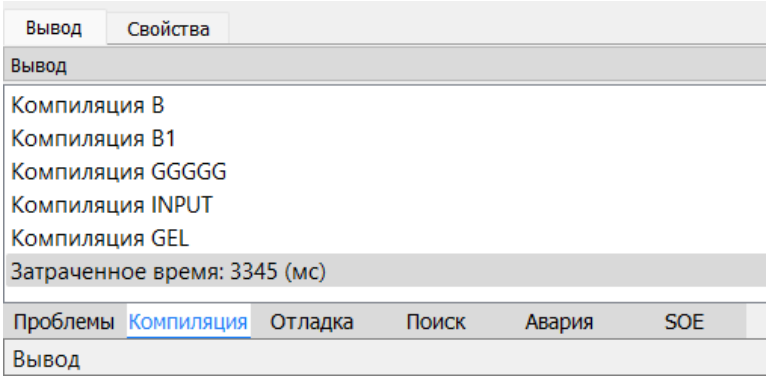


Рисунок 2.29 Раздел информационного окна «Компиляция»

[**Отладка**]: Окно вывода информации о процессе отладки. В состоянии отладки в режиме онлайн в этом окне выводится версия программы и информация об ошибках в процессе выполнения программы.

[**Поиск**]: Результаты Глобального поиска выводятся в этом окне. При использовании Глобального поиска конкретные позиции и общее количество найденного контента выводятся в этом окне, как показано на рисунке 2.30.

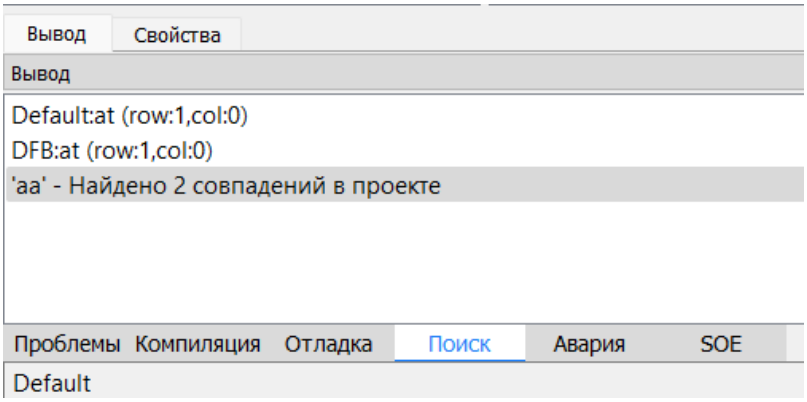


Рисунок 2.30 Схема поиска информации

[**Авария**]: Окно вывода информации о тревогах программы. При использовании редактора последовательных функциональных диаграмм для написания программы, информация о тревогах, заполненная в [блоке выполнения функции] / функции [тревога], а также информация о статусе запуска/выхода из программы будет выводиться в этом окне во время выполнения программы.

[**SOE**]: Окно вывода информации о событиях SOE. Если ПЛК сконфигурирован с модулем SOE, вся информация о событиях SOE будет выводиться в этом окне, включая серийный номер переменной, изменение 0->1 или 1->0 и информацию о времени изменения и т.д.

Списки сочетаний клавиш

[Переключение выводов функционального блока]: Выберите функциональный модуль и нажмите Enter, чтобы переключить вводимый контакт, как показано на рисунке 2.31.

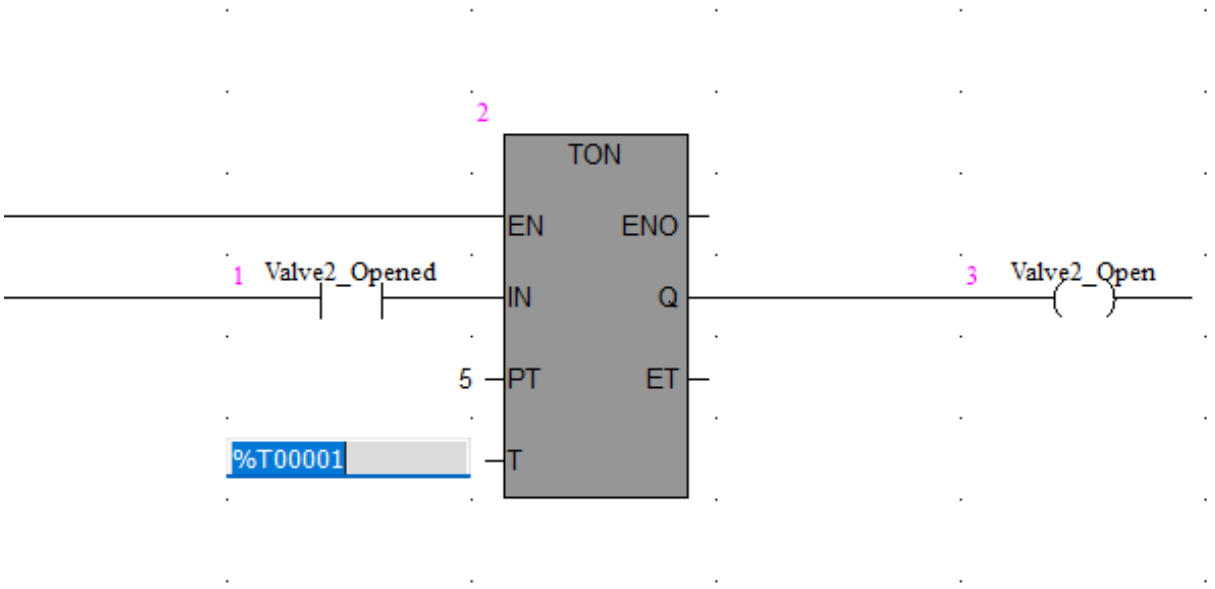


Рисунок 2.31 Переключение вводимого контакта

[Используйте клавиатуру для перемещения курсора]:Вверх (↑), Вниз (↓), влево (←), вправо (→), Page Up, Page Down, влево (Home), вправо (End), вверх (Ctrl + Home), End (Ctrl + end).

[Используйте клавиатуру для перемещения функциональных блоков]: Переместить блок вверх (Ctrl + ↑), переместить блок вниз (Ctrl + ↓), переместить блок влево (Ctrl + ←), переместить блок вправо (Ctrl + →).

[Поместите контакт и катушку]: Нажмите F2~F11, чтобы поместить контакт и катушку в пустые места, конкретное соответствие показано на рисунке 2.32.

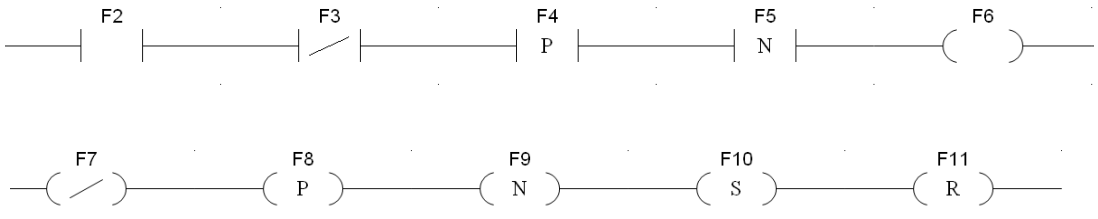


Рисунок 2.32 Схема вызова контактов горячими клавишами

[Размещение функционального блока]: Введите первую букву функционального блока в пустые поля, и появится список выбора функциональных блоков, как показано на рисунке 2.33.

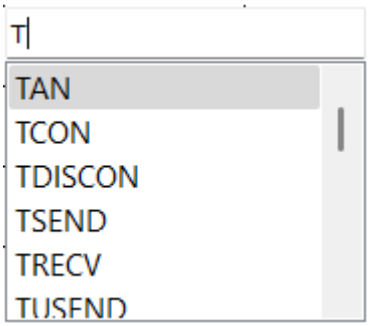


Рисунок 2.33 Поиск функционального блока по первой букве

[Другие операции]:

Файл	Новый проект	Ctrl+N
	Новая программа	Ctrl+E
	Сохранить проект	Ctrl+C
	Сохранить программу	Ctrl+I
	Скомпилировать всю программу	Ctrl+B
	Скомпилировать программу	Ctrl+R
	Открыть проект	Ctrl+O
	Печать	Ctrl+P
Правка	Отменить	Ctrl+Z
	Повторить	Ctrl + Y
	Вырезать	Ctrl+X
	Копировать	Ctrl+C
	Вставить	Ctrl+V
	Удалить	Delete
	Выбрать Все	Ctrl + A
	Поиск	Ctrl+Ф
	Заменить	Ctrl+H
	Глобальный поиск	Ctrl+G
	Вставить одну строку	Ctrl + J
	Удалить одну строку	Ctrl+К
Онлайн	Подключить / ПЛК	Ctrl+W
	Подключить /Симулятор	Ctrl+U
	Отключить	Ctrl+Q
Загрузить	Загрузить все	Ctrl + L
	Загрузить программу	Ctrl+T

Управление проектом

Когда вы входите в среду разработки INDAS PRO, вы сталкиваетесь с задачей превращения фактического проекта в код, который может быть запущен в ПЛК через INDAS PRO. В среде разработки проект управляет всеми элементами, такими как программы, данные, ресурсы и т. д. Для технического персонала управление проектом состоит из двух основных категорий: файл конфигурации проекта и файл программы проекта. Файл конфигурации проекта содержит набор информации о конфигурации оборудования, управлении программой, конфигурации различных параметров ПЛК и т. д. Программные файлы проекта содержат набор программ, которые определяют процесс, который должен быть реализован, включая программы релейной логики (LD), функциональных блоков (FBD), списка инструкций (IL), структурированного текста (ST), программы диаграммы управления последовательностью и т. д.

Дерево проекта

Используя дерево проекта, вы можете отображать содержимое проекта INDAS PRO и переключаться между различными элементами проекта, например, между программами, данными, ресурсами.

Дерево проекта отображается в виде дерева каталогов и обеспечивает прямую навигацию по: программам — лестничным диаграммам, схемам функциональных блоков, таблицам инструкций, структурированному тексту, схемам управления последовательностью и т. д.; данным — таблицам регистров измерения, переменным, самостоятельно выбранным точкам измерения и т. д.; ресурсам — конфигурации ПЛК, конфигурации задач и т. д. Как показано на рисунке 3.1.

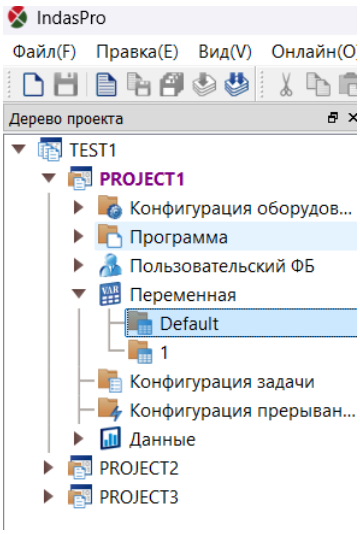


Рисунок 3.1 Дерево проекта

По умолчанию дерево каталогов отображается до уровня 3, а соответствующее содержимое можно открыть, дважды щелкнув по узлу.

Создать новый проект

Новые проекты создаются на следующих этапах.

Шаги	Операция
1	Создание нового рабочего пространства. См. пункт «Создание нового рабочего пространства».
2	Создание нового проекта. См. пункт «Создание нового проекта»
3	Настройте ПЛК и задайте конфигурацию оборудования. См. пункт «Конфигурация оборудования ПЛК».
4	Создание пользовательских программ. См. пункт «Управление программой».
5	Компиляция программы.
6	Загрузка проектов и программ. См. пункт «Загрузка и выгрузка файла проекта и пункт «Загрузка и выгрузка программы».

Создание нового рабочего пространства

Сначала создайте пустой проект для этого проекта. Откройте программное обеспечение INDAS PRO и нажмите [Файл] / [Создать] в главном меню, как показано на рисунке 3.2.

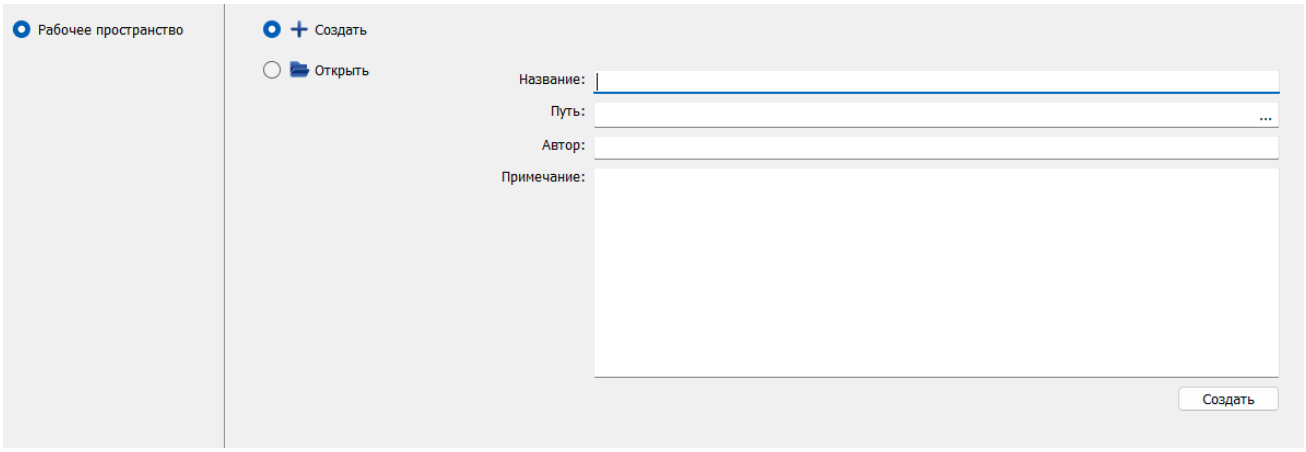


Рисунок 3.2 Создание нового рабочего пространства

Если рабочее пространство создается, когда открыто другое рабочее пространство, то появится всплывающее окно создания нового рабочего пространства (рисунок 3.3).

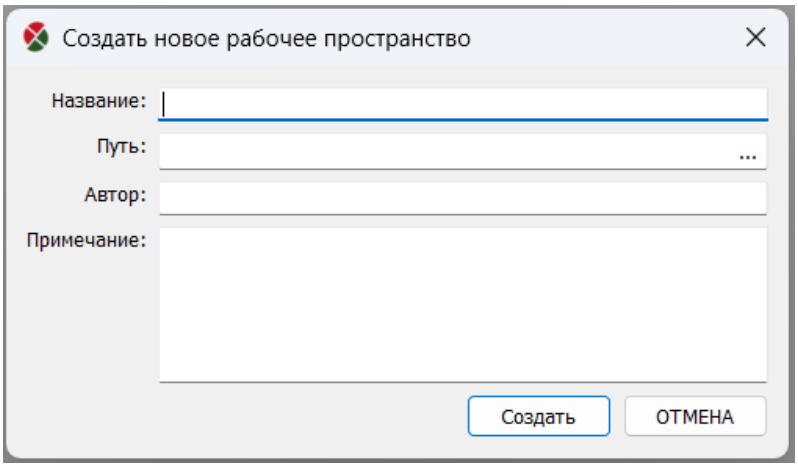


Рисунок 3.3 Создание нового рабочего пространства через всплывающее окно

Создание нового проекта

Щелкните правой кнопкой мыши [NAPLC] / [Новый проект] или значок на панели инструментов введите имя проекта во всплывающем диалоговом окне и нажмите [OK], как показано на рисунке 3.4.

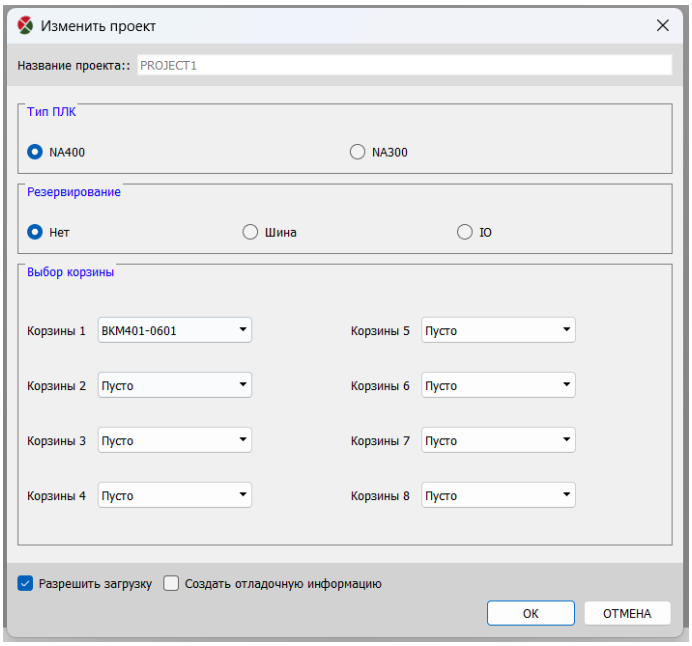


Рисунок 3.4 Создание нового проекта

Примечание
Перед программированием необходимо завершить настройку оборудования ПЛК. Если конфигурация ввода/вывода отсутствует, информация о точке не будет отображаться, и компиляция завершится неудачей. Конфигурация должна совпадать с типами модулей, установленных на объединительной плате, в противном случае модуль ввода/вывода не будет распознан.

Конфигурация оборудования ПЛК

В дереве проекта дважды щелкните «Конфигурация оборудования», чтобы открыть диалоговое окно «Конфигурация корзины», как показано на рисунке 3.5.

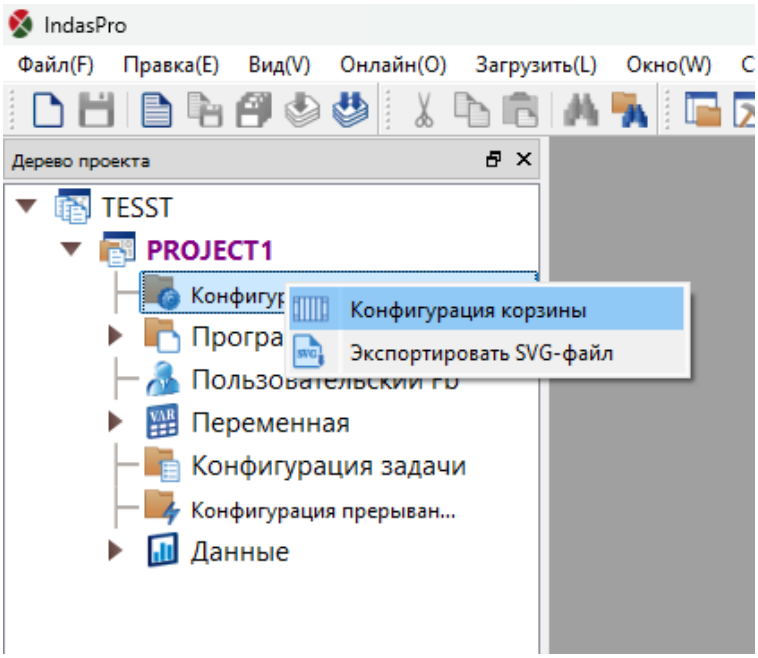


Рисунок 3.5 Открытие раздела «Конфигурация корзины»

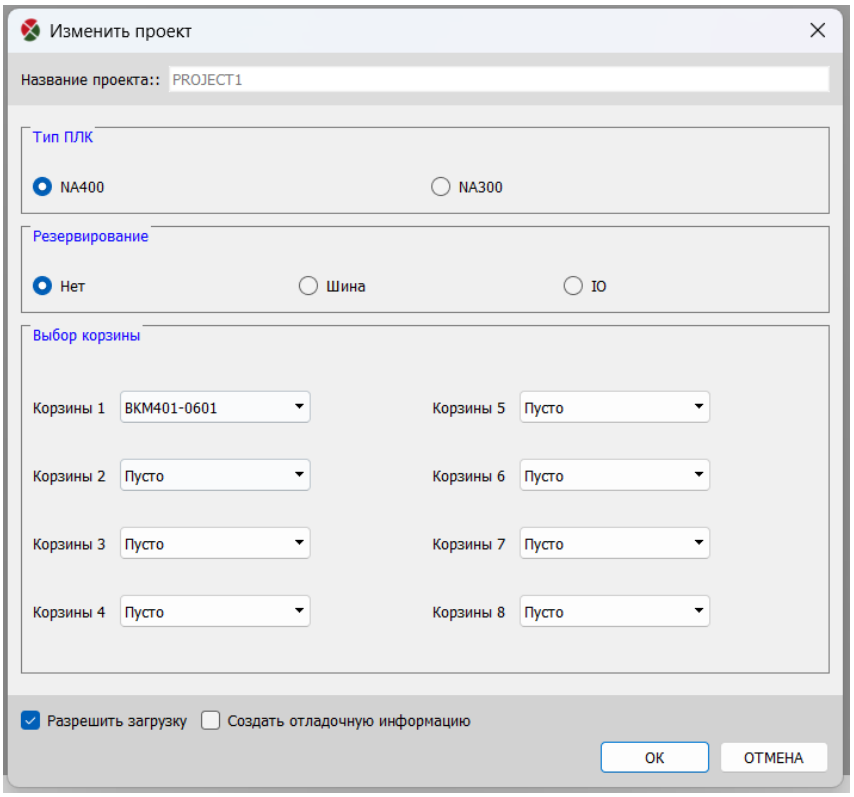


Рисунок 3.6 Конфигурация корзины проекта

Флажок «Разрешить загрузку» используется для выбора разрешения на сохранение проекта и исходного файла программы в ПЛК. Если флажок установлен, во время

загрузки исполняемый файл и исходный файл вместе будут загружены в ПЛК для последующей загрузки; если флажок не установлен, будет загружен только исполняемый файл, и загрузка не будет выполнена.

Флажок «Создать отладочную информацию» используется для онлайн-отладки программы пользователя. Пользователи могут устанавливать точки останова и пошаговое выполнение, которые могут быть не выбраны в целом.

Флажок «IO» предназначен для CPU401-0511, чтобы реализовать избыточность любого канала модуля ввода/вывода. Не выбирайте эту опцию для других моделей CPU.

В соответствии с требованиями проекта к конфигурации корзины, корзина 1 по умолчанию является основной, другое - корзина расширения, каждая корзина может быть выбрана в соответствии с потребностями проекта (ПЛК NA400 имеет выбор корзины из 6, 9, 12 и 15 слотов). Нажмите кнопку «ОК» после конфигурации, как показано на рисунке 3.7.

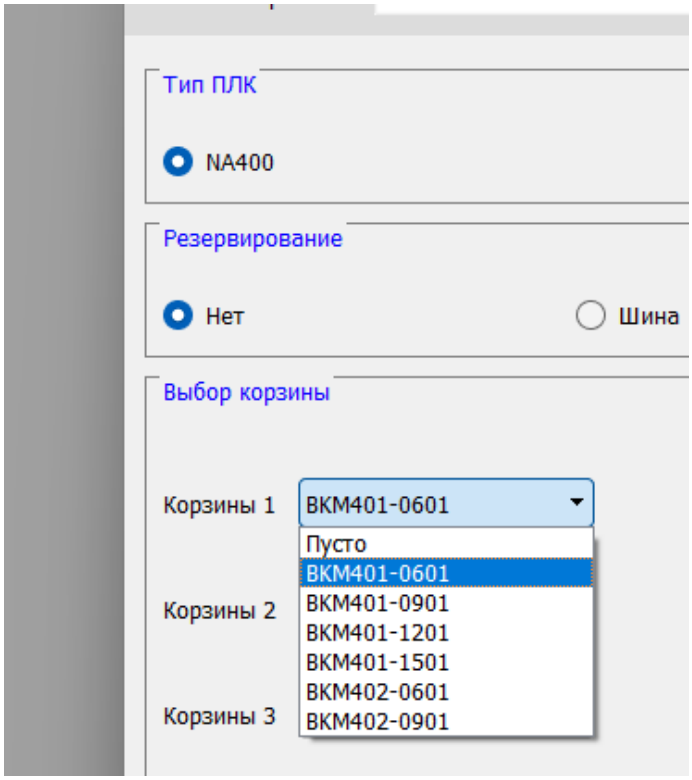


Рисунок 3.7 Выбор корзины

Структура оборудования в поле каталога после подтверждения показана на рисунке 3.8.

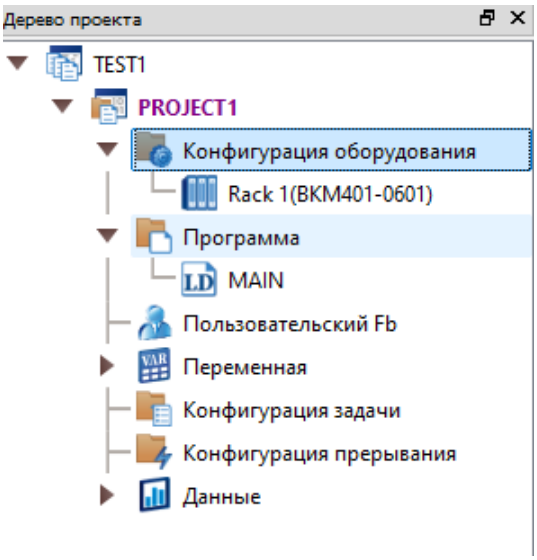


Рисунок 3.8 Отображение созданной корзины в дереве проекта

Теперь вы можете начать выбирать модули для каждого шасси. Дважды щелкните на «Rack 1» в дерева проекта, и конфигурация модуля для корзины отобразится в области редактирования справа. Это показано на рисунке 3.9.

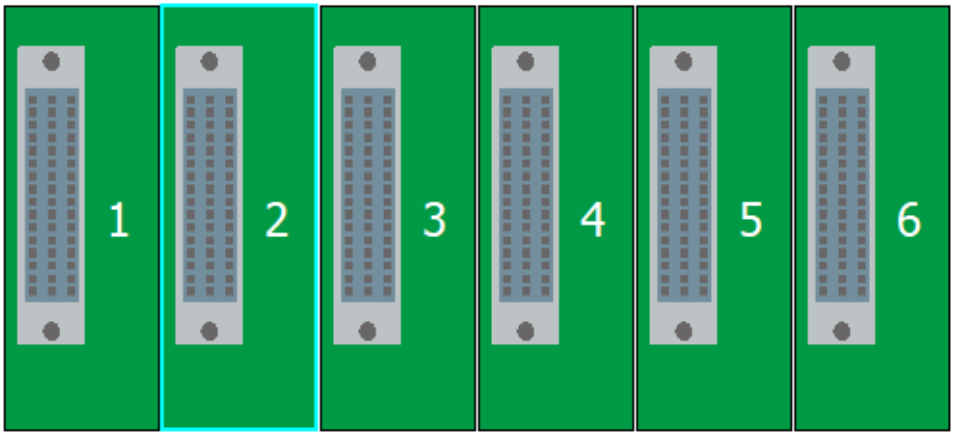


Рисунок 3.9 Монтажная плата корзины

Нет никаких специальных требований к типам модулей в каждом слоте стойки ПЛК серии NA, т. е. все модули могут быть сконфигурированы в любом слоте. Пожалуйста, выберите тот же тип, что и фактический модуль ввода/вывода при настройке, в противном случае ЦП не распознает информацию о модуле и чтение данных не удастся.

Двойной щелчок по модулю или пустому слоту откроет библиотеку модулей для выбора типа и модели модуля, как показано на рисунке 3.10.

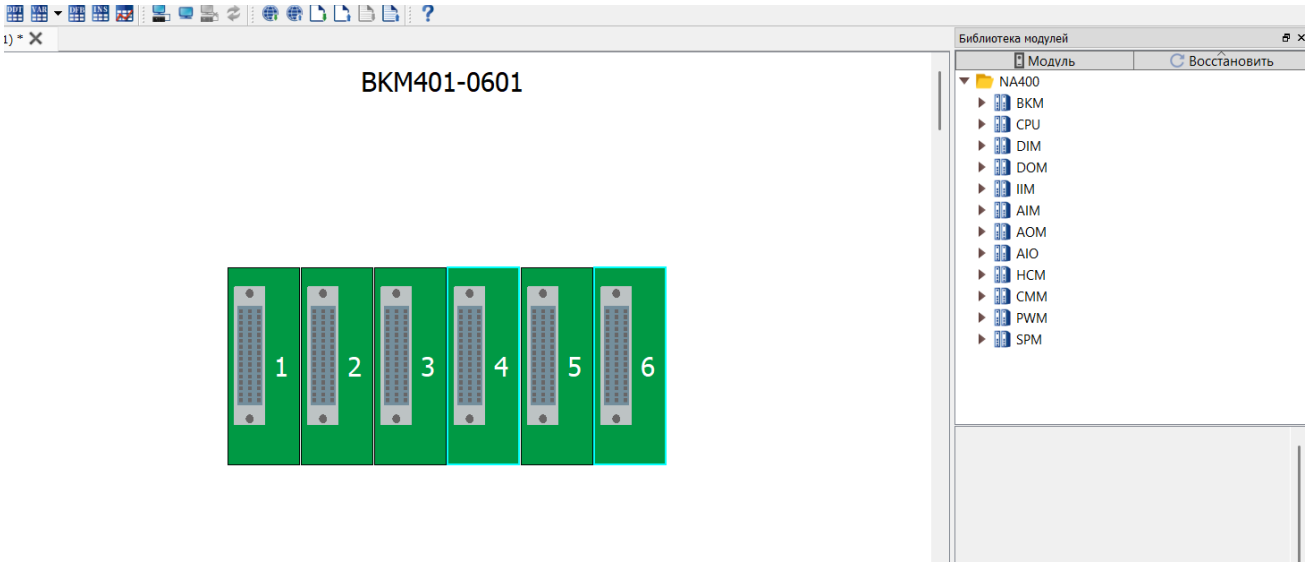


Рисунок 3.10 Библиотека модулей

Типы модулей: ЦП, Цифровой вход, Цифровой выход, Цифровой вход/выход, Регистрация событий, Аналоговый вход, Аналоговый выход, Аналоговый вход/выход, Высокоскоростной счетчик, Связь, Специальная функция, Источник питания. Выберите модуль и нажмите кнопку «Свойства», чтобы настроить свойства модуля. Это показано на рисунке 3.11.

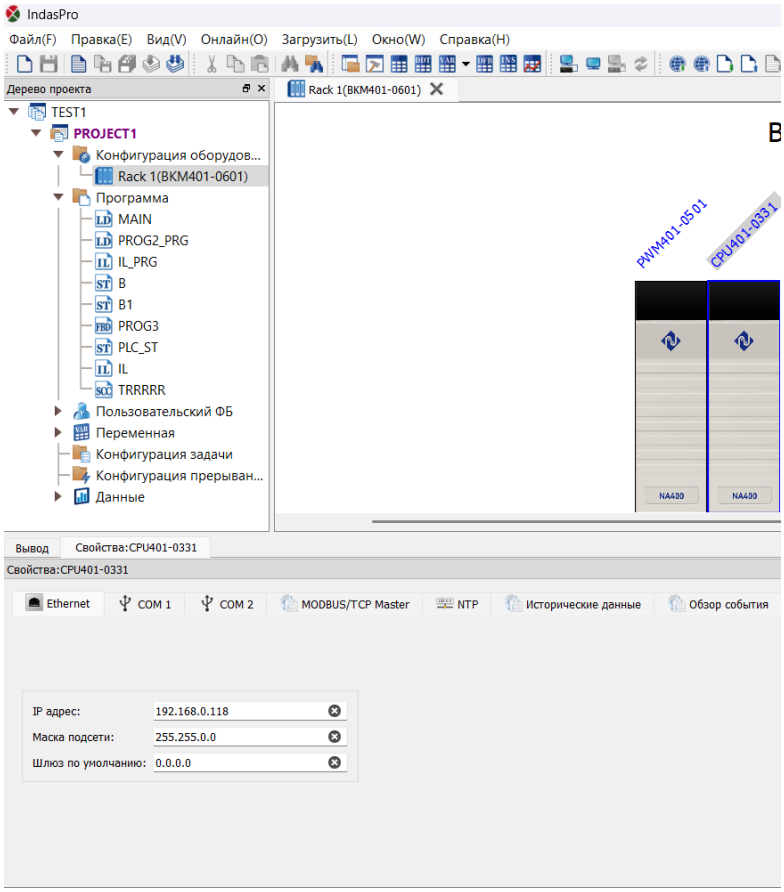


Рисунок 3.11 Свойства модуля

Модуль ЦП

Модули ЦП включают стандартные модули ЦП, высокопроизводительные модули ЦП, высокопроизводительные избыточные модули ЦП и т. д. Каждый тип модуля делится на различные модули в зависимости от количества последовательных портов, количества портов Ethernet и размера программного пространства. Высокопроизводительные избыточные модули ЦП могут использоваться только в двойных блоках. Конфигурация двух последовательных портов одинакова для каждого модуля ЦП. Скорость передачи данных, биты данных, стоповые биты и четность выбираются с помощью раскрывающегося списка.

Существуют некоторые специальные требования к конфигурации адресов Ethernet ЦП с двойной сетью. IP-адрес Ethernet состоит из четырех частей. IP-адрес двойной сети требует, чтобы первая, вторая и четвертая части двух сетей имели одинаковые адреса, но третья часть отличалась. Например, IP-адреса двойной сети могут быть 192.168.200.100 и 192.168.201.100. Как показано на рисунке 3.12.

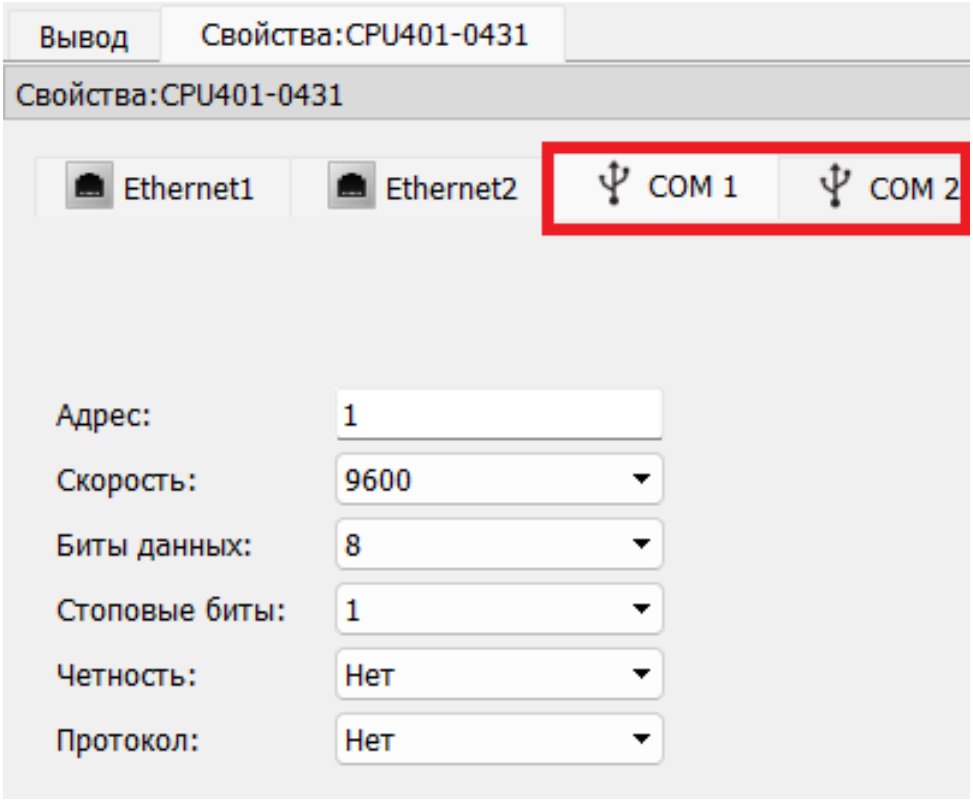


Рисунок 3.12 Параметры процессорного модуля

Модуль цифрового ввода

Существует четыре типа цифровых входных модулей, но конфигурация та же самая, нужно задать только начальный номер. После того, как начальный номер задан, остальные точки перечислены в порядке убывания. Это показано на рисунке 3.13.

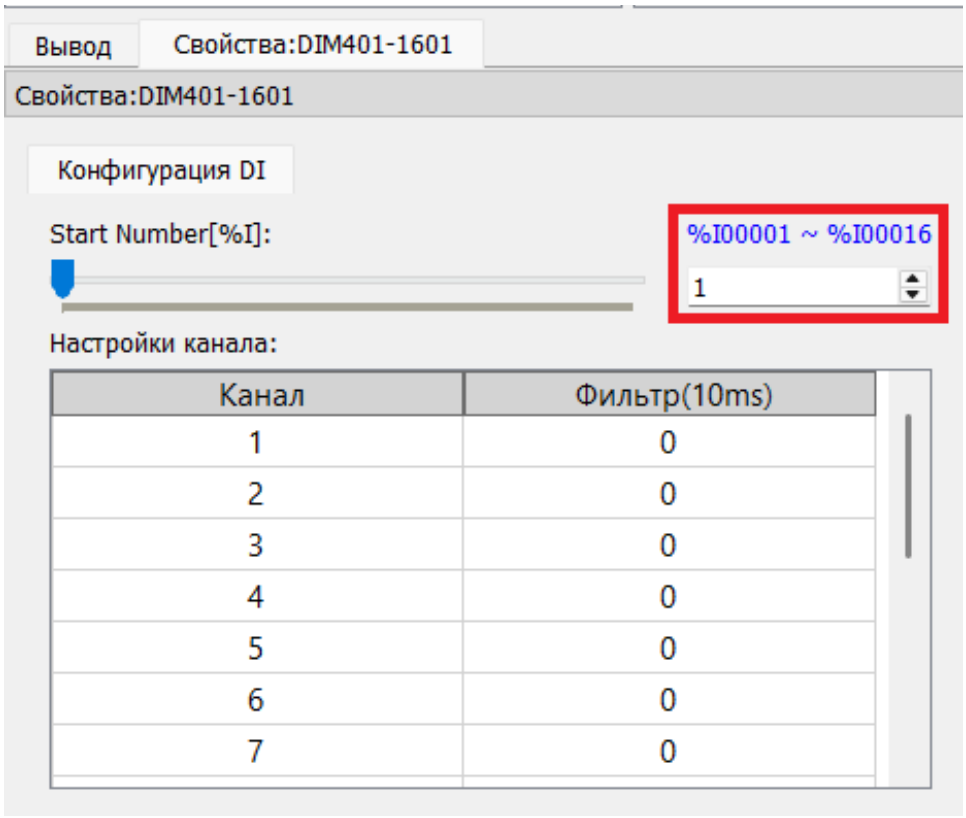


Рисунок 3.13 Параметры цифрового входного модуля

Модуль цифрового вывода

Существует четыре типа цифровых выходных модулей, но конфигурация та же самая, нужно задать только начальный серийный номер. После того, как начальный серийный номер задан, серийные номера других регистров располагаются в порядке убывания. Это показано на рисунке 3.14.

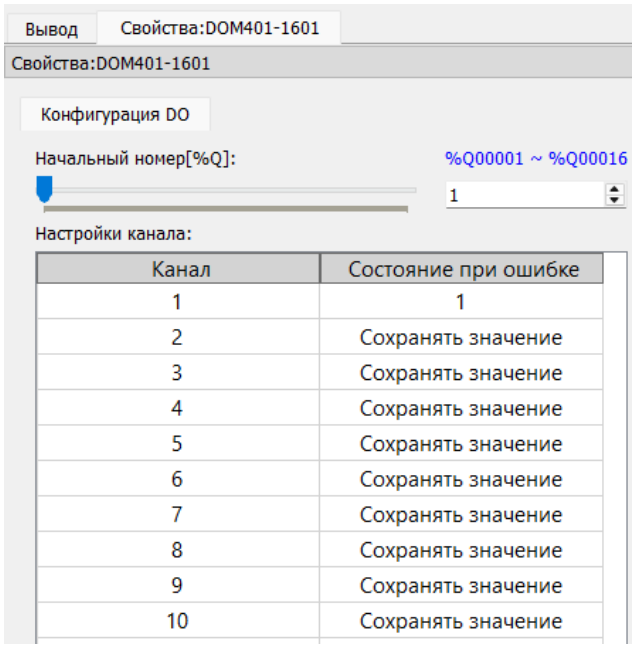


Рисунок 3.14 Параметры цифрового выходного модуля

Модуль последовательности событий

Модуль последовательности событий является одним из видов цифрового входного модуля, занимает адрес цифрового входного модуля, конфигурация такая же, как у цифрового входного модуля, нужно только задать начальный номер. После того, как начальный номер установлен, другие серийные номера будут расположены последовательно. Это показано на рисунке 3.15.

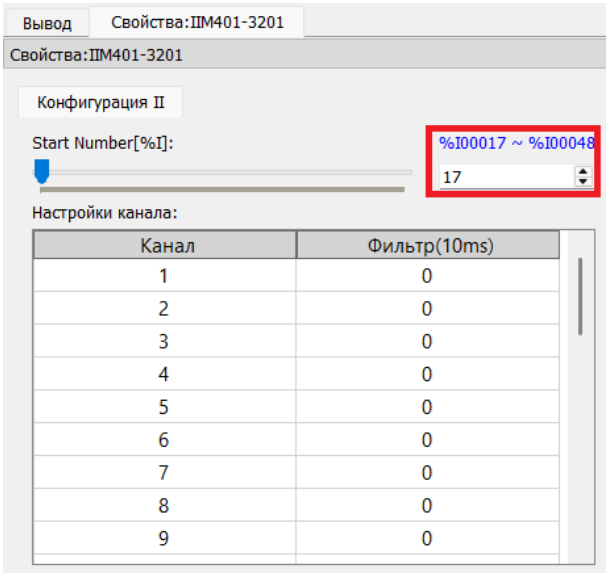


Рисунок 3.15 Параметры модуля SOE

Модуль аналогового ввода

Существует восемь типов модулей аналогового ввода, но конфигурация та же самая, нужно задать только начальный номер. После того, как начальный номер задан, серийные номера других регистров располагаются в порядке убывания. Это показано на рисунке 3.16.

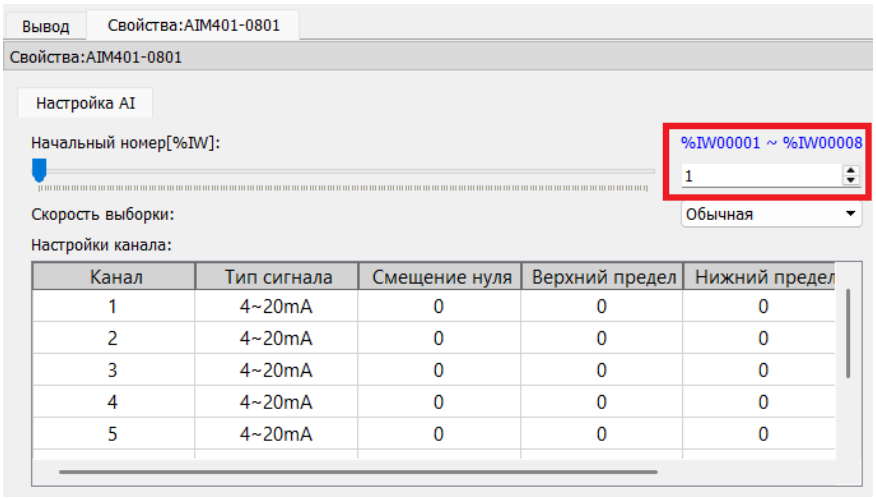


Рисунок 3.16 Параметры аналогового входного модуля

«Нормальная» скорость сбора данных: 500 мс на модуль для аналоговых показаний для ЦП.

«Быстрая» скорость сбора данных: каждый аналоговый модуль считывает данные в ЦП один раз за цикл сканирования.

Модуль аналогового вывода

Существует два типа модулей аналогового вывода, но конфигурация одинакова, необходимо задать только начальный серийный номер. После того, как начальный серийный номер задан, серийные номера других регистров располагаются в порядке убывания. Это показано на рисунке 3.17.

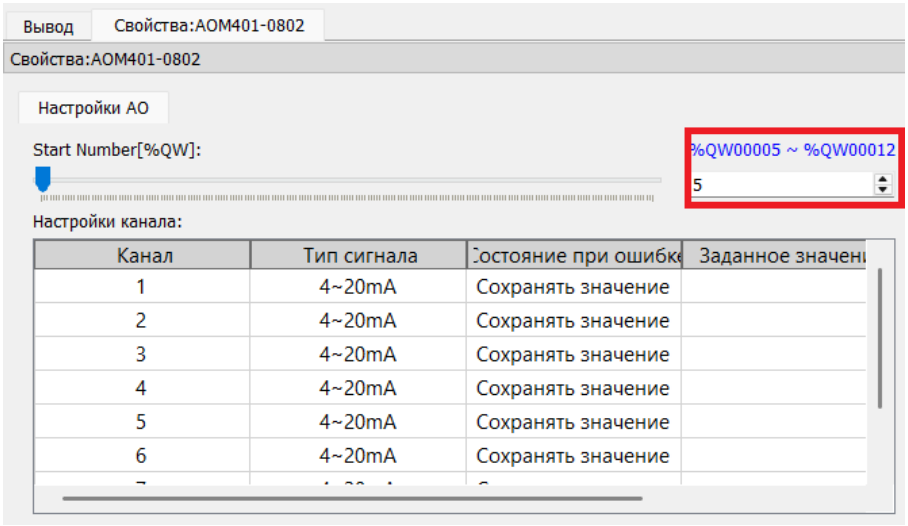


Рисунок 3.17 Параметры аналогового выходного модуля

Импорт проекта

Программное обеспечение INDAS PRO может импортировать несколько проектов в дерево проектов. Щелкните правой кнопкой мыши имя рабочего пространства, на следующем рисунке это [NAPLC], и щелкните [Импортировать проект], чтобы добавить другие проекты, как показано на рисунке 3.18.

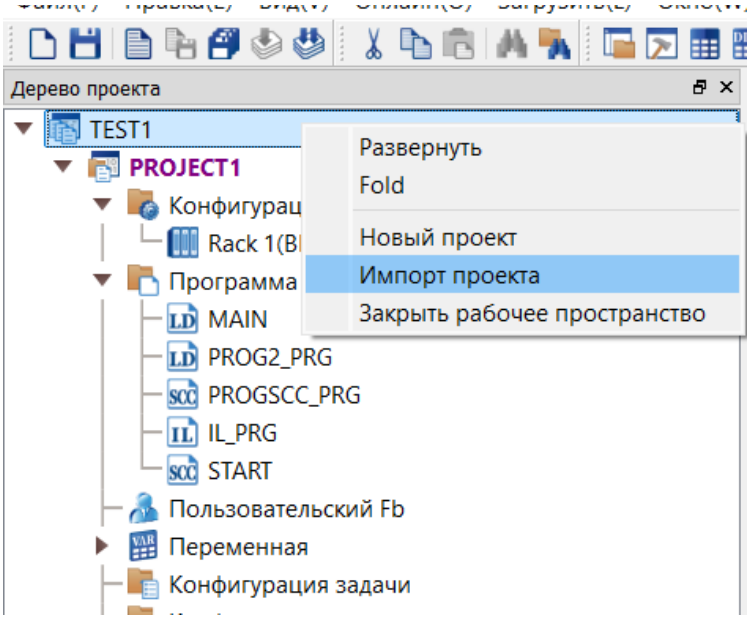


Рисунок 3.18 Импорт проекта

Найдите нужный вам импортированный проект и нажмите [Открыть], как показано на рисунке 3.19.

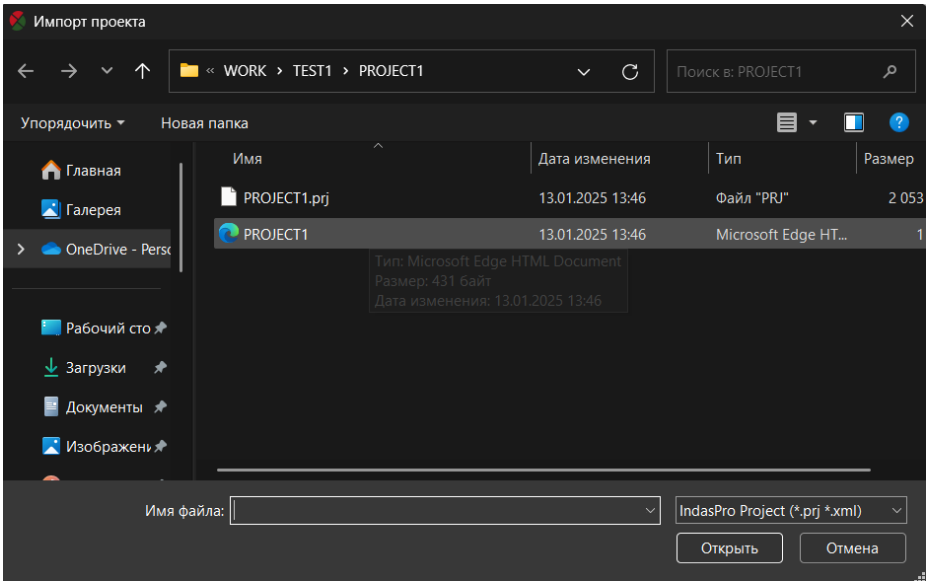


Рисунок 3.19 Выбор проекта для импорта

После импорта проекта его имя становится фиолетовым, и можно выполнить соответствующие настройки оборудования и программного обеспечения, в то время как другие проекты настроить нельзя, как показано на рисунке 3.20.

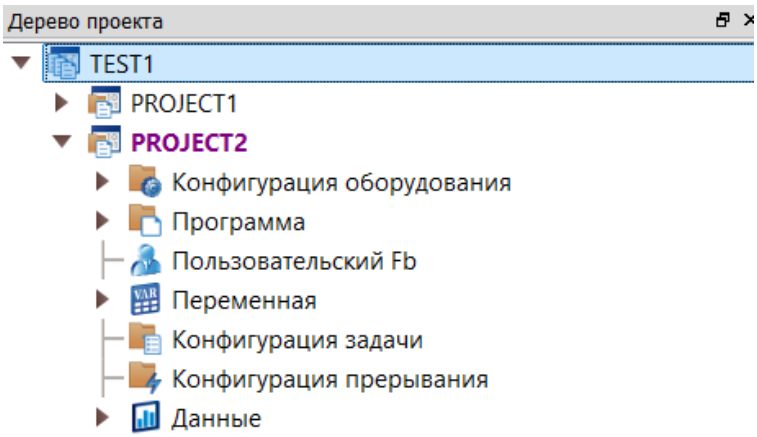


Рисунок 3.20 Импортированный проект в дереве проекта

Активировать проект

Если вы хотите настроить другие элементы, активируйте проект, щелкнув правой кнопкой мыши по имени элемента и нажав [Активировать] во всплывающем диалоговом окне. Как показано на рисунке 3.21.

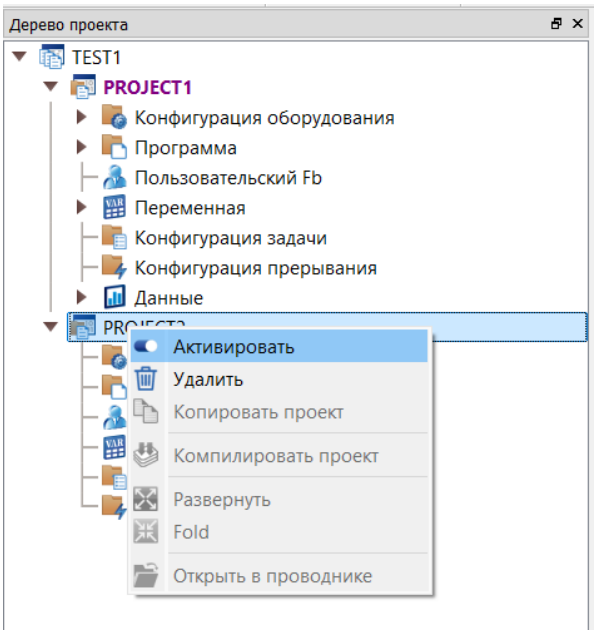


Рисунок 3.21 Активация проекта

Управление программой

Программа состоит как минимум из одной основной программы MAIN, с которой начинается каждый цикл сканирования, а все остальные подпрограммы вызываются из основной программы. Основная программа MAIN и все подпрограммы перечислены в каталоге программ дерева проекта. Программы делятся по типу на лестничные диаграммы, диаграммы функциональных блоков, списки инструкций,

Структурированный текст, диаграммы управления последовательностью и т. д., как показано на рисунке 3.22.

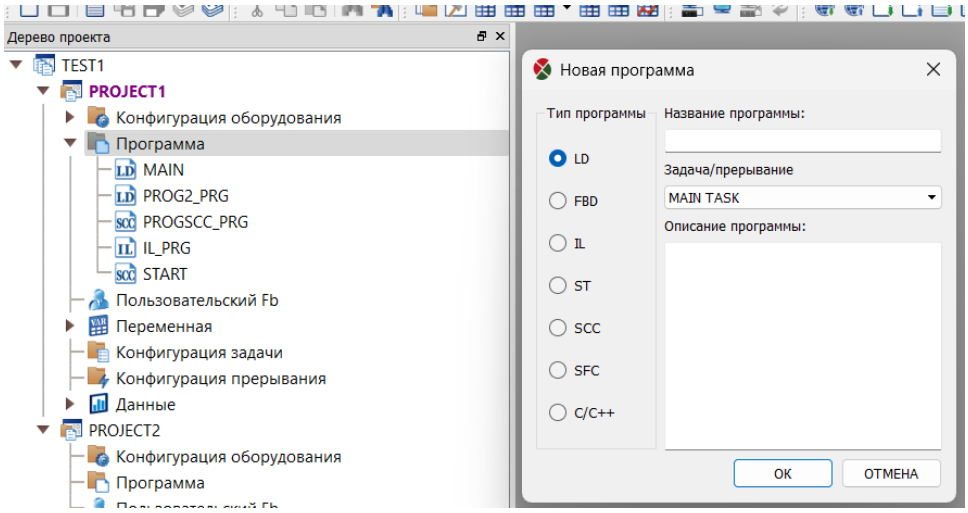


Рисунок 3.22 Структура раздела «Программа»

Схемы управления последовательностью не выполняются сканирующим способом, а последовательно контролируемым способом, т. е. шаг за шагом от начала до конца и завершают выполнение в конце. Программы с диаграммами управления последовательностью не выполняются автоматически, а только инициируются программой MAIN или другими подпрограммами. Выполнение последовательной контрольной схемы не влияет на лестничную схему или другие программы (за исключением программ вызова последовательной контрольной схемы).

Добавление программы

Чтобы добавить программу, воспользуйтесь главным меню [Файл] / [Новая программа] или кнопкой на панели инструментов, затем выберите тип программы и назовите программу. Описание программы помогает пользователю понять назначение программы и может быть заполнено или не заполнено. Программа может быть частью основной задачи (циклическое выполнение), задачи 1-16 (приоритет от низкого до высокого, выполнение по времени) или прерывания (прерывания по времени 1-4, прерывания ввода-вывода, уровень прерывания от низкого до высокого). Как показано на рисунке 3.23.

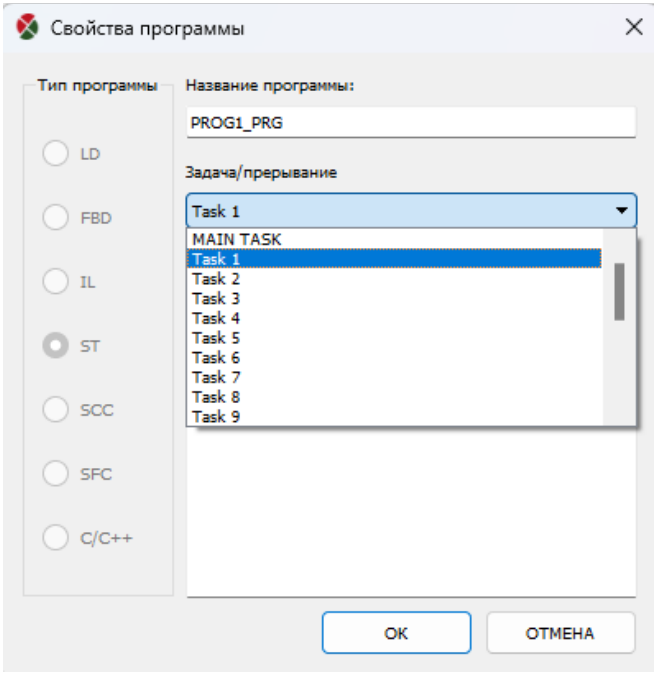


Рисунок 3.23 Выбор приоритета программы

Примечание
Максимальное количество подпрограмм, поддерживаемых каждым ЦП, показано в следующей таблице (каждый язык программирования рассчитывается независимо, а не суммируется).

Тип процессора	LD	FBD	IL	ST	SCC
CPU301-0101	256	256	256	256	256
CPU401-020X	32	32	32	32	32
CPU401-030X	64	64	64	64	64
CPU401-0401	128	128	128	128	128
CPU401-0501	128	128	128	128	128
CPU401-0221	32	32	32	32	32
CPU401-0421	64	64	64	64	64
CPU401-0521	64	64	64	64	64
CPU401-0331	64	64	64	64	64
CPU401-0431	128	128	128	128	128
CPU401-0531	128	128	128	128	128
CPU401-0211	32	32	32	32	32
CPU401-0411	64	64	64	64	64
CPU401-0511	64	64	64	64	64
CPU401-1101 CPU201-1101	32	32	32	32	32

Удалить программу

Чтобы удалить программу, щелкните по ней правой кнопкой мыши и выберите «Удалить» (основную программу MAIN удалить нельзя). Это показано на рисунке 3.24.

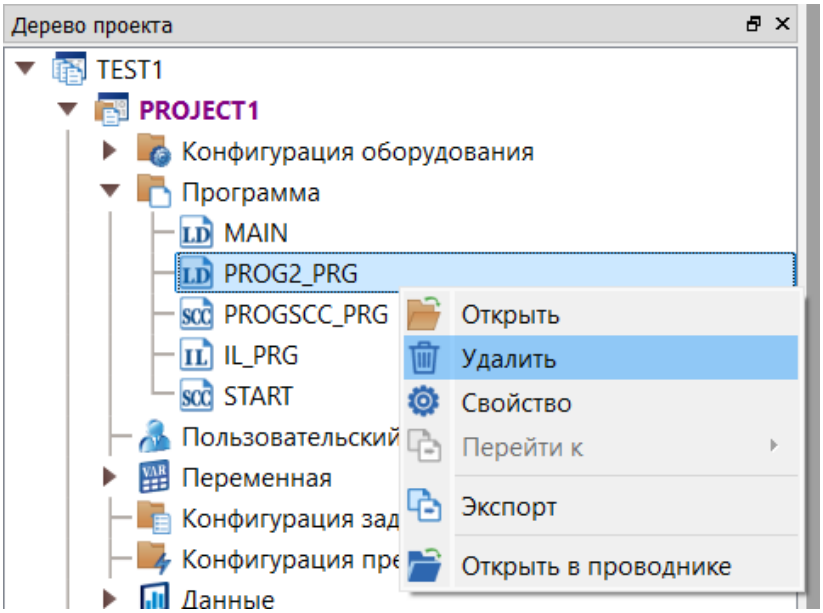


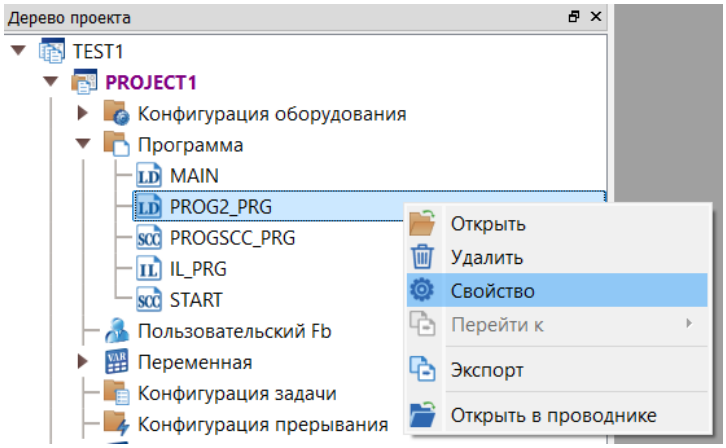
Рисунок 3.24 Удаление программы

Переименовать программу

Чтобы переименовать программу, щелкните правой кнопкой мыши по программе и выберите «Свойство». Измените имя в поле «Название программы».

Описание программы

Чтобы описать программу, щелкните правой кнопкой мыши по программе и выберите «Свойства». Напишите описание программы в поле «Описание программы». Это показано на рисунке 3.25.



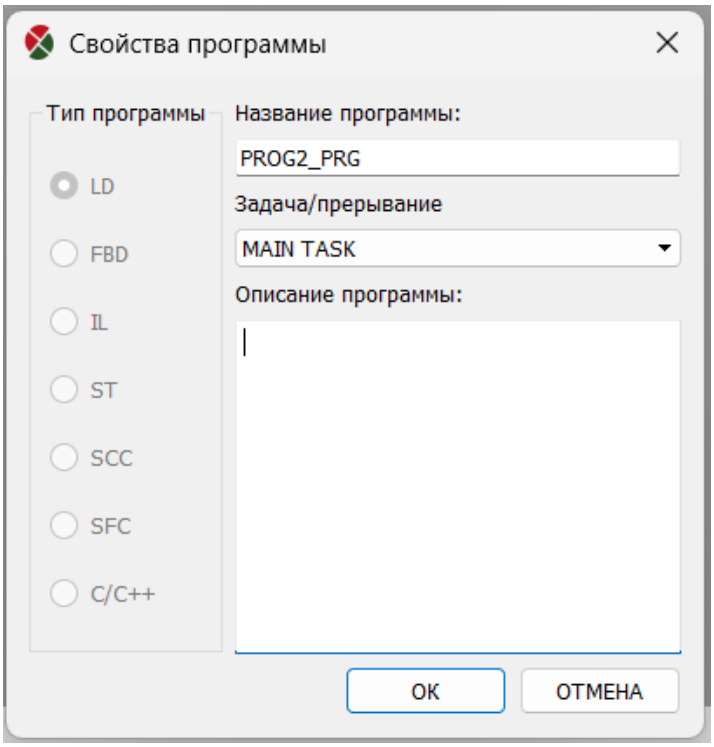


Рисунок 3.25 Описание программы

Установка пароля проекта

Чтобы поставить пароль на проект, используйте главное меню [Файл] - [Пароль] - [Пароль к файлу] и нажмите «Пароль к файлу», чтобы открыть диалоговое окно, показанное на рисунке 3.26.

Примечание
Пожалуйста, запомните свой пароль после создания, вы не сможете его восстановить, если забудете!

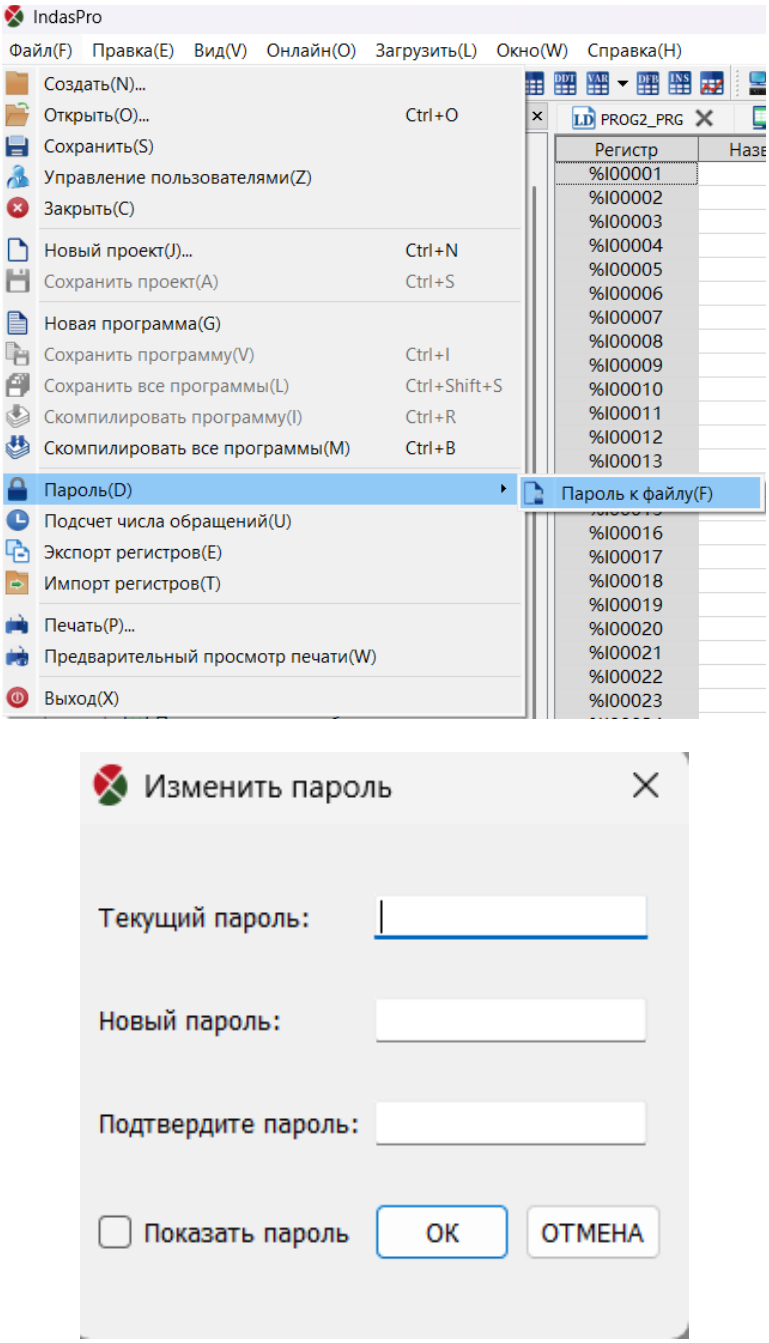


Рисунок 3.26 Создание пароля проекта

Экспорт программы

Чтобы экспортировать программу, щелкните правой кнопкой мыши по программе и выберите «Экспорт». Отобразится диалоговое окно, а затем выберите каталог для сохранения, как показано на рисунке 3.27.

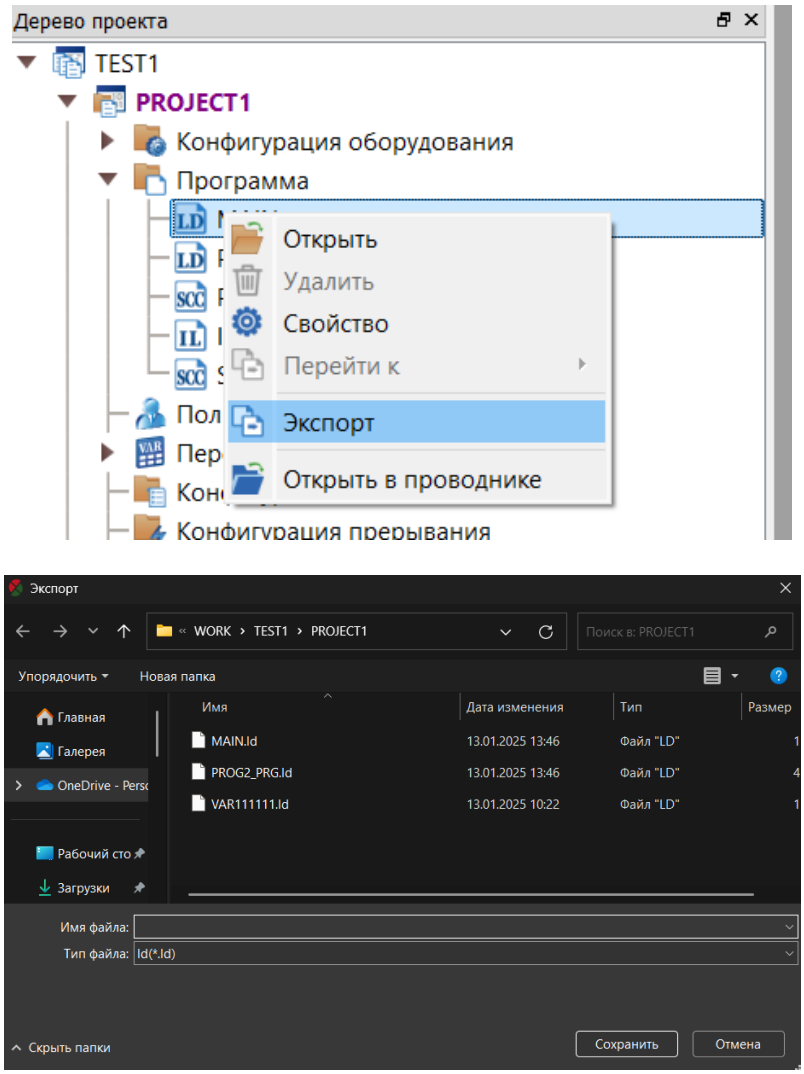


Рисунок 3.27 Экспорт программы

Импорт программы

Чтобы импортировать программу, щелкните [Программа] правой кнопкой мыши, а затем просто выберите файл программы для импорта и откройте его, как показано на рисунке 3.28.

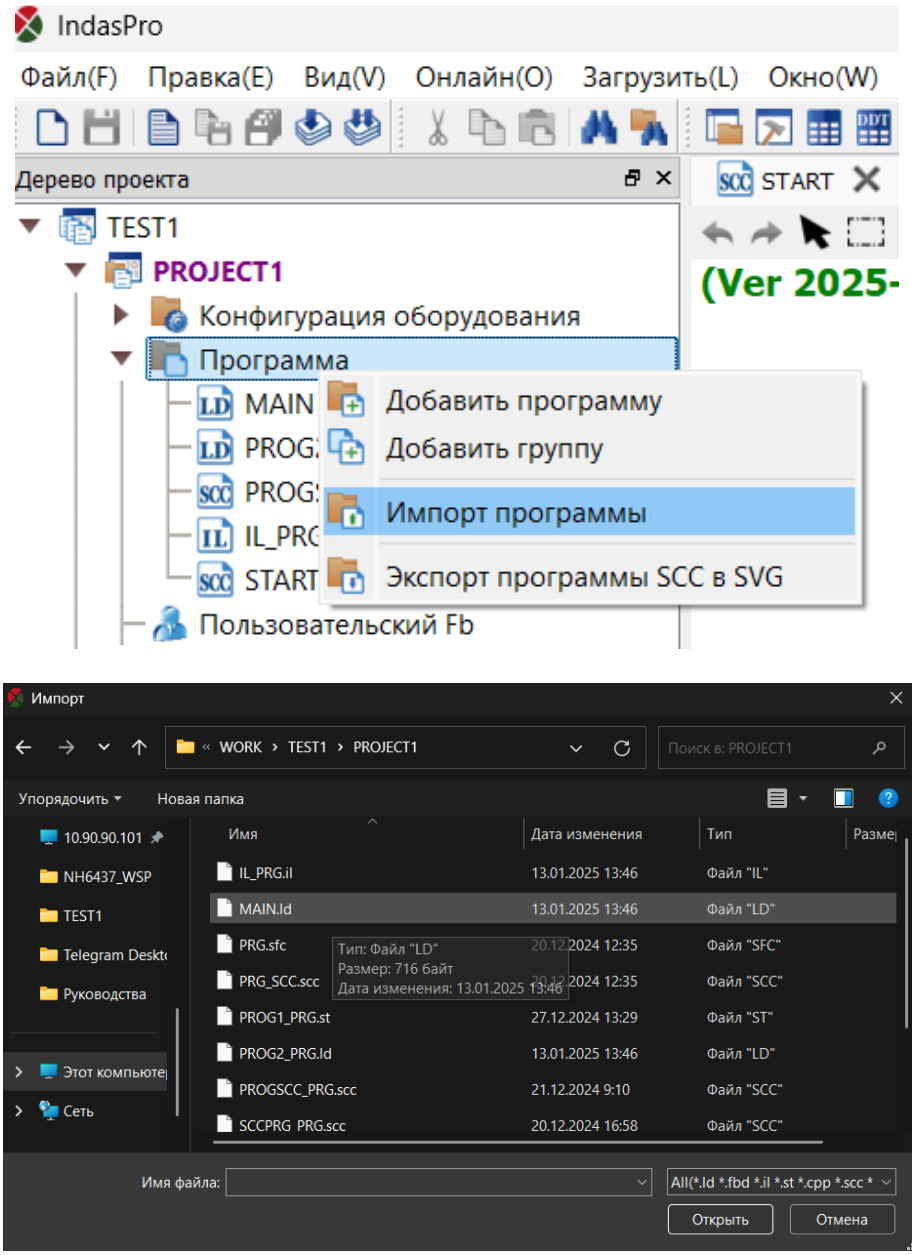


Рисунок 3.28 Импорт программы

Назначение задач

Чтобы назначить задачу программе, щелкните правой кнопкой мыши по программе (рисунок 3.29) и выберите задачу, к которой она принадлежит, в диалоговом окне «Свойства». По умолчанию вновь созданная программа принадлежит к основной задаче MAIN TASK.

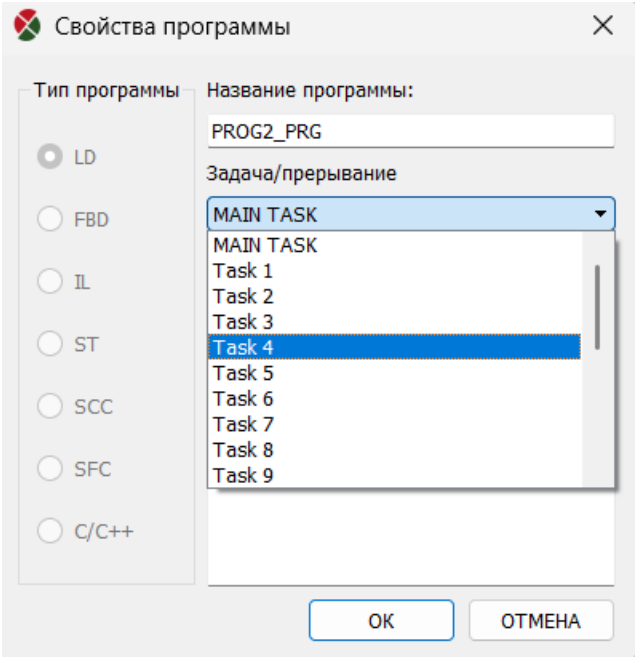


Рисунок 3.29 Назначение задачи

Управление задачами

Пользователи могут выделять несколько задач для одного проекта. Вызов программ зависит от выделения задач.

По умолчанию система работает как однозадачная. Программа MAIN вызывается автоматически и уникально. Через нее реализуются прямые или косвенные вызовы других программ без настройки задачи. Как правило, однозадачная среда может удовлетворить потребности управления ПЛК.

В многозадачной среде основная задача (автоматическое выполнение MAIN) выполняется в цикле с самым низким приоритетом. Программное обеспечение поддерживает максимум 16 задач, задача 16 с самым высоким приоритетом. Каждая программа должна принадлежать основной задаче или одной из задач 1–16. Программы в рамках одной задачи могут вызываться друг другом.

Все задачи и программы перечислены в [Конфигурация задач], как показано на рисунке 3.30.

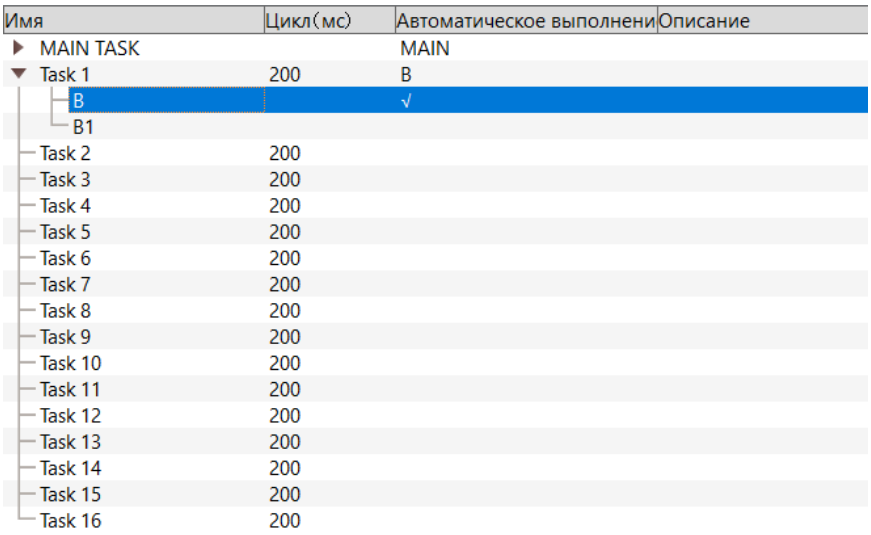


Рисунок 3.30 Конфигурация задачи

Программы B и B1 относятся к задаче 1. Выберите программу автоматического выполнения B и период времени 200 мс, как показано на рисунке 60. Минимальный период времени может быть установлен на 1 миллисекунду. В дополнение к автоматической программе, другие программы, относящиеся к задаче (например, B1), должны быть вызваны автоматической программой (например, B).

Управление прерываниями

Прерывание — это своего рода ответ и обработка внешних или внутренних событий ПЛК, включая события прерывания, программы прерывания и управление прерываниями. События прерывания являются причинами прерывания, такими как прерывание ввода-вывода, прерывание таймера и т. д. Когда происходят события прерывания, ПЛК прекращает сканирование для программы и передает права управления программам прерывания, принадлежащим этому событию прерывания. После выполнения программ прерывания право управления будет возвращено прерванным программам ПЛК для продолжения выполнения.

NAPLC поддерживает пять видов прерываний, включая прерывание таймера 1, прерывание таймера 2, прерывание таймера 3, прерывание таймера 4 и прерывание ввода-вывода. Прерывание 1 имеет самый низкий приоритет, а прерывание ввода-вывода — самый высокий приоритет. ПЛК обSlaveтывает события прерывания в соответствии с приоритетом.

Набор кодов функций, которые предъявляют высокие требования к точности реального времени и управления и могут быть связаны с событиями прерывания, можно скомпилировать в программу прерывания. Программа прерывания настраивается во время создания новой программы, и чем короче содержимое программы, тем лучше.

Управление прерываниями осуществляется ENI и DISI. Прерывания экранируются, когда ПЛК только что запущен. Только в ситуации, когда ENI включен и происходит соответствующее прерывание, может быть вызвана программа прерывания. В ситуации, когда прерывание запрещено, оборудование все еще отвечает на прерывание, но вызов программы прерывания не допускается.

Для прерывания таймера 1~4 пользователи должны установить цикл синхронизации, как показано на рисунке 3.31.

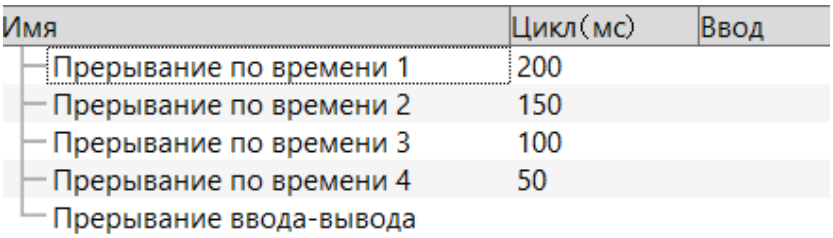


Рисунок 3.31 Диалоговое окно для настройки прерывания таймера

Защита проекта

Установите пароль файла для защиты конфигурации проекта и предотвращения изменения и доступа к программе без полномочий. Пароли файлов могут ограничить доступ к файлу, а пароли входа ограничивают подключение, загрузку и скачивание. Новый проект не имеет паролей файлов.

Подключение и отключение

Команда «Подключить» относится к коммуникационному соединению между INDAS PRO и ПЛК. Щелкните значок соединения на панели инструментов, и ПЛК перейдет в состояние соединения. Если соединение успешно, пользователи могут просматривать состояние выполнения программы, получать состояния регистров напрямую или вносить принудительные изменения в некоторые регистры для отладки программ. Если соединение не будет установлено в течение предписанного времени, будет сообщено об ошибке соединения.

Команда «Отключить» относится к отключению между программным обеспечением INDAS PRO и ПЛК. При отключении содержимое проекта может быть изменено, а проект и файл могут быть загружены.

Примечание

Подключение INDAS PRO осуществляется через порт программирования Ethernet. Пользователи должны открыть исходный завершенный проект разработки для подключения к ПЛК. Разные проекты не могут подключаться друг к другу. Программа выведет аварийную информацию «Версии проекта отличаются, пожалуйста, загрузите снова!» в окне выходной информации, показанном на рисунке 3.32. Подключение к новому ЦП может быть успешным только после загрузки проекта.

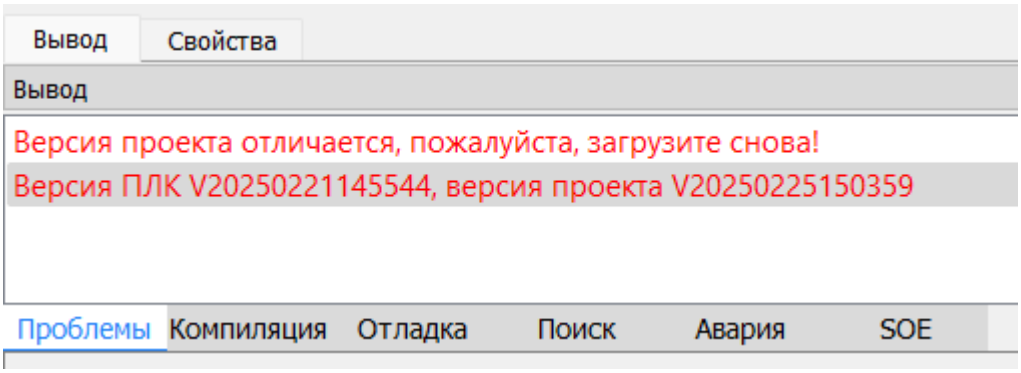


Рисунок 3.32 Несовпадение версий проекта ПЛК и ПК

Загрузка и выгрузка файла проекта

Ручная загрузка

Ручная загрузка используется для изменения сетевого IP-адреса ЦП, когда пользователи загружают файлы проекта на ЦП в первый раз или когда IP-адрес Ethernet ЦП забыт. Ручная загрузка автоматически загрузит весь файл проекта и файлы программы на ЦП.

Сначала переведите dip-переключатель CPU в состояние «DEBUG» и снова включите CPU. После этого индикатор R/F на панели CPU одновременно начнет мигать с частотой около 1 секунды, указывая на то, что CPU работает в состоянии DEBUG. В данный момент IP-адрес CPU - 192.168.1.66.

Измените IP-адрес в сетевом подключении ПК так, чтобы он находился в том же сегменте сети, что и IP-адрес ЦП по умолчанию (первые три сегмента одинаковы, последний сегмент отличается), как показано на рисунке 3.33.

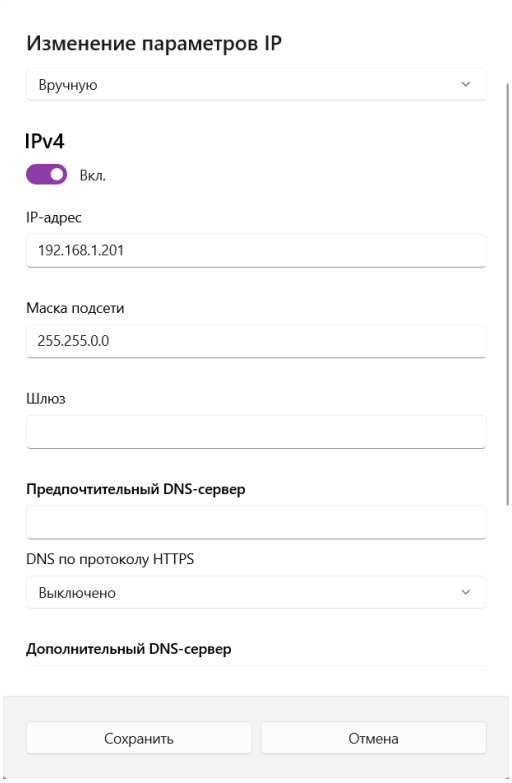


Рисунок 3.33 Модификация сетевого IP

Выбор меню INDAS PRO «Загрузить/Ручная загрузка» открывает следующее окно, показанное на рисунке 3.34.

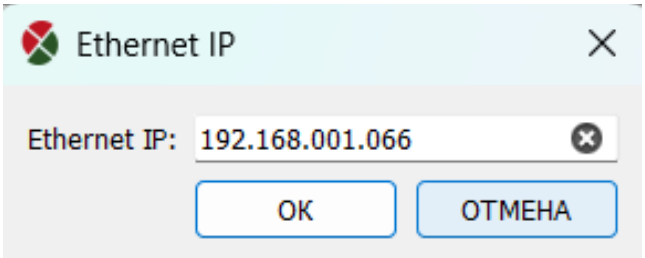


Рисунок 3.34 Ручная загрузка в ПЛК

Если сетевое оборудование не работает или адрес неверен, система сообщит об ошибке в окне выходной информации. В это время проверьте настройки сети.

С помощью команды ping 192.168.1.66 -t вы можете проверить, нормально ли работает сеть. Если при пинговании время отклика нормальное, но вы не можете загрузить программу, проверьте настройки брандмауэра вашего компьютера, чтобы увидеть, не заблокирована ли передача файлов или не занесен ли INDAS PRO в черный список и т. д. Для систем Win7 и Win10 обязательно измените настройки брандмауэра или отключите его, в противном случае вы не сможете загрузить.

После успешной загрузки переведите переключатель DIP на ЦП в положение RUN и снова включите ЦП. После успешного запуска индикатор R будет мигать раз в секунду, а индикатор A будет гореть, указывая на то, что ЦП работает нормально.

Загрузить проект

Скомпилированный файл проекта загружается в ПЛК. Как и в случае с онлайном, программное обеспечение для программирования автоматически ищет узел в сети и загружает его в соответствии с адресом Ethernet в конфигурации оборудования во время загрузки. Во время поиска и загрузки программное обеспечение для программирования отображает слово «загрузка», если узел ПЛК не найден, соединение не будет установлено. Для двойных систем программное обеспечение для программирования автоматически загружается для обоих ПЛК.

После загрузки программы в ПЛК необходимо сбросить модуль ЦП и перезапустить его снова, чтобы выполнить загруженную программу, в противном случае система выполнит программу до загрузки. Сброс можно выполнить с помощью команды сброса программного обеспечения для программирования.

Выгрузить проект

Для того, чтобы выгрузить файл проекта из ПЛК, необходимо:

1. Создать новое рабочее пространство
2. Выбрать пункт Загрузить/Выгрузить Всё. В отличие от загрузки, при выгрузке программное обеспечение выводит диалоговое окно с запросом адреса Ethernet IP ПЛК, где находится файл, который вы хотите загрузить. После нажатия кнопки подтверждения программное обеспечение для программирования выполнит поиск узла в сети на основе адреса, как показано на рисунке 3.35.

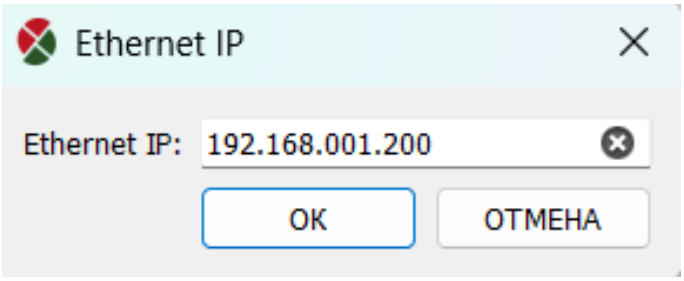


Рисунок 3.35 Выгрузка проекта

3. Задать название проекта

Примечание
Если в [Конфигурации ПЛК] не выбран параметр «Разрешить загрузку программы» (рисунок 3.36), файл проекта и файл программы не будут загружены, а при загрузке появится сообщение «Загрузка файла проекта не удалась!».

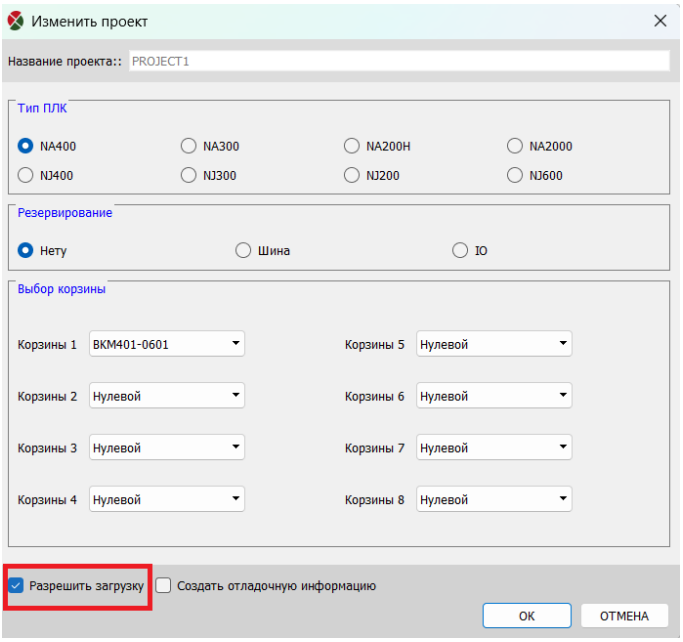


Рисунок 3.36 Флажок «Разрешить загрузку»

Загрузка и выгрузка программы

Загрузить программу

Если файл проекта не был изменен, а был изменен только файл программы, функцию загрузки программы можно использовать для обновления файла программы в ПЛК.

Если какой-либо программный файл не соответствует версии в ПЛК, информация о тревоге будет выведена в окне выходной информации, как показано на рисунке 3.37. Пользователи могут обновить программу, чтобы поддерживать согласованность версии.

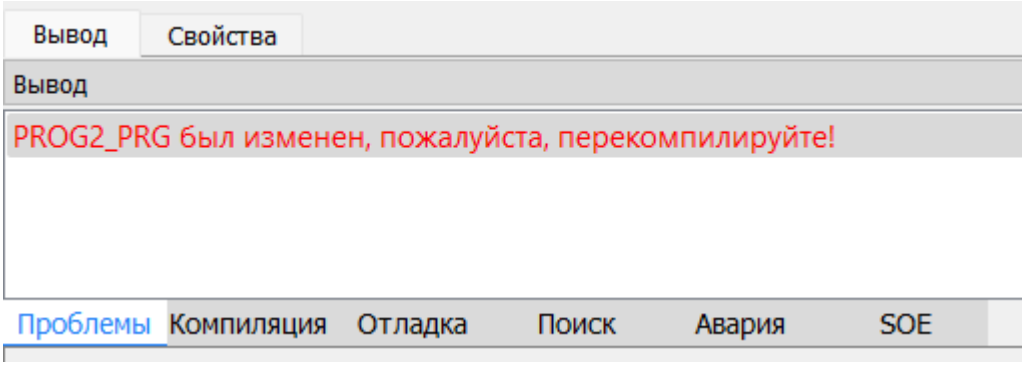


Рисунок 3.37 Информация о проблеме с проектом

Выгрузить программу

Загрузите все файлы программ из ПЛК, включая лестничные диаграммы, схемы функциональных блоков, Структурированный текст, списки инструкций, схемы управления последовательностью и т. д. При загрузке программы, как и при загрузке файлов проекта, необходимо ввести IP-адрес ПЛК, который будет загружен. Затем появится окно, в котором можно выбрать имя программы для загрузки или отметить их все.

Загрузить все

Загрузите файл проекта в ПЛК вместе со всеми программными файлами.

Загрузить все программы

Загрузите все программы в ПЛК.

Разница между загрузкой программ и загрузкой проектов

Загрузить проект

Содержимое загрузки: Файл конфигурации, за исключением программы, такой как конфигурация аппаратных модулей, списки программ, списки данных, информация о регистрах и т. д.

Применимый случай: списки программ не изменена (без добавления и удаления программ и без изменения имени программы). Однако конфигурация модулей, регистров, списки данных и другая информация изменены.

Загрузить программу

Содержимое загрузки: Загружайте только измененную программу.

Применимый случай: изменена только одна программа.

Загрузить все программы

Содержимое загрузки: Загрузить всю программу.

Применимый случай: изменено несколько программ, но конфигурация не изменена.

Скачать Все

Содержимое загрузки: Загрузить все программы и конфигурации.

Применимый случай: Все конфигурации оборудования и программы изменены.

Измененная конфигурация оборудования, таблица переменных, если она используется в программе, даже если сама программа не изменена, ее необходимо скомпилировать и загрузить полностью (например, переменная, добавленная в середину таблицы переменных, после изменения адреса переменной %V, программу необходимо перекомпилировать и загрузить).

Ручная загрузка

Загружаемый контент: контент такой же, как и у «загрузить все», включая всю конфигурацию проекта и программу.

Приложение:

Если вы не знаете IP-адрес модуля ЦП, войдите в режим отладки и используйте 192.168.1.66 для загрузки.

Не знаете нижнюю программу ЦП, когда мигает индикатор F, используйте адрес 192.168.1.66 для загрузки.

Вы уже знаете IP-адрес ЦП и хотите загрузить все конфигурации и программы проекта. (Ручная загрузка быстрее с помощью FTP-загрузки, другие методы загрузки относительно медленные с помощью сокета).

Управление данными

Стандарт МЭК 61131-3 определяет наиболее часто используемые типы данных для программирования ПЛК, и, таким образом, определение и использование этих типов данных единообразны в области ПЛК. Это имеет очевидные преимущества для производителей машин и установок, а также для инженеров, использующих несколько ПЛК и систем программирования от разных производителей: единообразные типы данных повышают переносимость программ ПЛК.

Тип данных

NAPLC обменивается следующими типами данных.

Тип	Имя	Количество цифр	Допустимый диапазон	Описание
BOOL	Булево	1	0 или 1	Хранится в битовых единицах и имеет только два состояния: 1 или 0.
BYTE	Байт	8	от 0 до 255	При использовании 8-битного регистра данных 8-битные данные могут быть независимы друг от друга, указывая только состояние текущего бита: 0 или 1; или они могут указывать на отсутствие символа. Целое число в диапазоне от 0 до 255.
WORD	Слово	16	0 до 65535	При использовании 16-битного регистра данных 16-битные данные могут быть независимы друг от друга, указывая только состояние текущего бита: 0 или 1; или они могут указывать на беззнаковое целое число в диапазоне от 0 до 65535.
DWORD	Двойное слово	32	0~4294967295	При использовании 32-битного регистра данных 32-битные данные могут быть независимы друг от друга, указывая только состояние текущего бита: 0 или 1; или они могут указывать на беззнаковое целое число в диапазоне от 0 до 4294967295.

SINT	Короткое целое число	8	-128 до +127	Использует 8-битный регистр данных для представления знакового целого числа в диапазоне от от -128 до +127.
INT	Целое число	16	-32768 до +32767	Использует 16-битный регистр данных для представления знакового целого числа в диапазоне от -32768 до +32767.
DINT	Длинное целое число	32	-2147483648~ +2147483647	Использует 32-битный регистр данных для представления знакового целого числа в диапазоне от -2147483648 до +2147483647.
REAL	Число с плавающей запятой	32		Представляет собой число с плавающей запятой. Точность — 6 знаков после запятой, округление значений свыше 6 знаков
USINT	Короткое целое число без знака	8	от 0 до 255	Использует 8-битный регистр данных для представления беззнакового целого числа в диапазоне от 0 до 255.
UINT	Целое число без знака	16	0 до 65535	Использует 16-битный регистр данных для представления беззнакового целого числа в диапазоне от 0 до 65535.
UDINT	Длинное целое число без знака	32	0~4294967295	Использует 32-битный регистр данных для представления беззнакового целого числа в диапазоне от 0 до 4294967295.

Столбец данных

Столбец данных включает регистры памяти, таблицу пользовательских типов, таблицу переменных, таблицу пользовательских функциональных блоков, таблицу примеров функциональных модулей и т. д., как показано на рисунке 4.1.

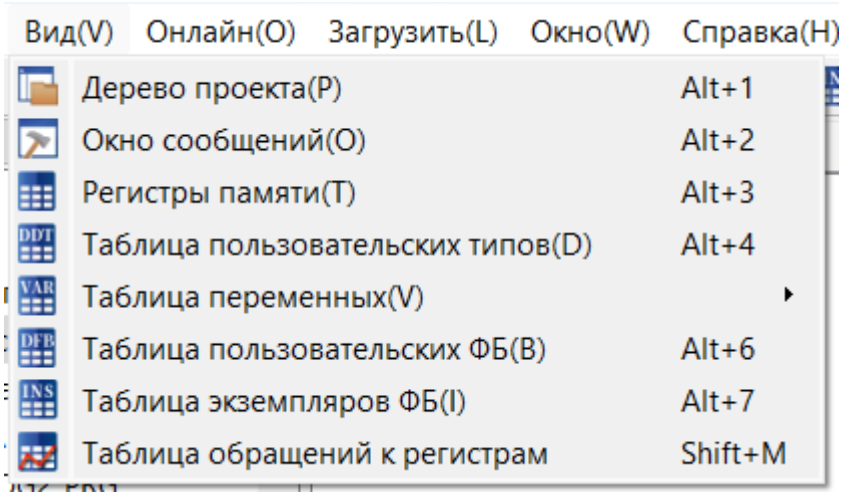


Рисунок 4.1 Структура столбца данных

Таблица пользовательских типов (DDT)

Основные функции

Определить типы данных структуры, которые могут использоваться в таблице переменных и таблице типов функциональных модулей. Каждый узел типа данных может иметь много дочерних узлов, а тип узла может быть либо базовым типом (базовые типы включают BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT), либо определяемым пользователем типом с максимальным количеством измерений узла 1024.

Примечания

[Имя переменной]: может начинаться только с буквы, состоящей из букв, цифр и подчеркивания. Если вы встретите размерность больше 1, то будет элемент массива, отмеченный [].

[Рекурсивные ссылки]: Узлы могут ссылаться друг на друга независимо от порядка, но если имеется рекурсивная ссылка, при анализе возникнет сообщение об ошибке.

[Модификация переменной]: отмечено как не проанализированное , изменение недействительно. Если проект компилируется с недопустимой переменной, проверьте этот пункт.

Описание функций контекстного меню (рисунок 4.2)

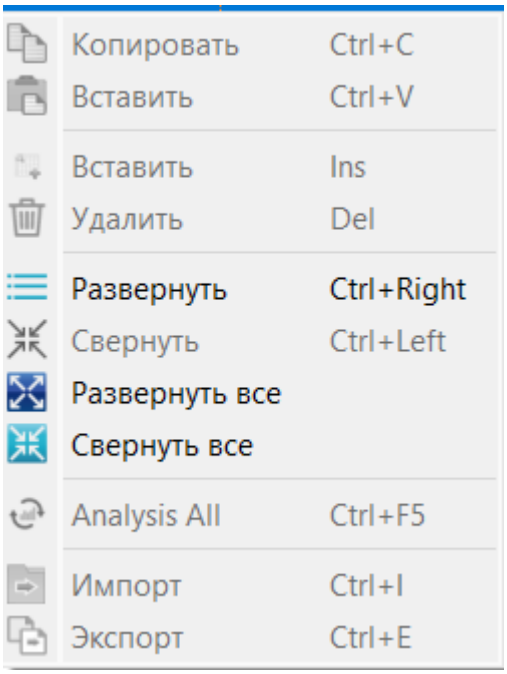


Рисунок 4.2 Контекстное меню для типа данных table

[**Копировать**]: Вы можете копировать узлы первого уровня.

[Вставить ]: Щелкните правой кнопкой мыши по пустой строке, чтобы вставить скопированный узел уровня 1, имя будет увеличиваться с 0 в соответствии с количеством вставок. Например, если имя скопированного узла уровня 1 — aa, имя первого вставленного узла — aa_0, имя второго вставленного узла — aa_1, …, а имя узла, вставленного в N-й раз, — aa_N-1. Дочерние элементы вставленного узла такие же, как у скопированного узла.

[**Вставить** ]: Щелкните существующий узел, которому должен предшествовать новый узел, и выберите [Вставить], чтобы вставить пустую строку перед выбранным узлом, при этом пользователь вводит содержимое двойным щелчком. Вставка может выполняться как в первичных, так и во вторичных узлах; эта функция недоступна для дочерних узлов пользовательского типа.

[**Удалить**]: Щелкните правой кнопкой мыши по опции удаления первичного и вторичного узлов, чтобы удалить узел. При удалении первичного узла его нельзя удалить, если он уже используется в таблице переменных или таблице типов функциональных модулей.

[**Развернуть**]: Щелкните по узлу, который необходимо развернуть, чтобы выбрать функцию развертывания и развернуть дочерние узлы этого узла.

[**Свернуть**]: Щелкните по узлу, который необходимо свернуть, чтобы выбрать функцию сворачивания, позволяющую свернуть дочерние узлы этого узла.

[**Развернуть все**]: Щелкните правой кнопкой мыши в любом месте таблицы типов данных и выберите функцию «Развернуть все», чтобы развернуть все дочерние узлы всей таблицы типов данных.

[**Свернуть все**]: Щелкните правой кнопкой мыши в любом месте таблицы типов данных и выберите функцию «Свернуть все», чтобы свернуть все дочерние узлы всей таблицы типов данных.

[**Тип анализа**]: После редактирования узла щелкните правой кнопкой мыши и выберите параметр анализа.

1. Если узел проанализирован успешно, значок изменится на ;
2. Если тип данных используется в таблице переменных или таблице типов функциональных модулей, он будет обновлен одновременно.
3. Если этот тип данных вызывает не анализируемый тип данных, то вызываемый тип также анализируется автоматически и одновременно.

Возможные причины неудачного анализа.

1. Имя анализируемого в данный момент узла совпадает с именем уже анализируемого узла.
2. В текущем узле для анализа имеется рекурсивный вызов, например, если анализируется узел, то дочерний элемент узла a имеет тип b, а дочерний элемент узла b имеет тип a. Это приводит к рекурсивному вызову.

	Значки для непроанализированных узлов
	Значки проанализированных узлов

[**Экспорт**]: Функция экспорта доступна для узлов первого уровня анализируемого DDT. При экспорте можно переименовывать экспортируемые узлы. Правила именования алфавитные, состоят из букв, цифр и подчеркиваний, без ограничений по длине на данный момент. Файл экспорта имеет суффикс. Ddt.

[**Импорт**]: Импортируйте узлы DDT из файла, вы можете изменить имя узла при импорте, имя узла не может совпадать с существующим именем узла в таблице DDT.

Если импортируемый узел ссылается на существующий узел в таблице DDT, он не будет повторяться, если содержимое то же самое, но если содержимое не то же самое, импорт не удастся.

Таблица переменных (VAR)

Основные функции

INDAS PRO имеет тип виртуальной переменной, называемой переменной, которая хранит данные в форме структуры данных (см. Рисунок 4.3). Текущее значение данных в переменной можно просмотреть онлайн через таблицу переменных.

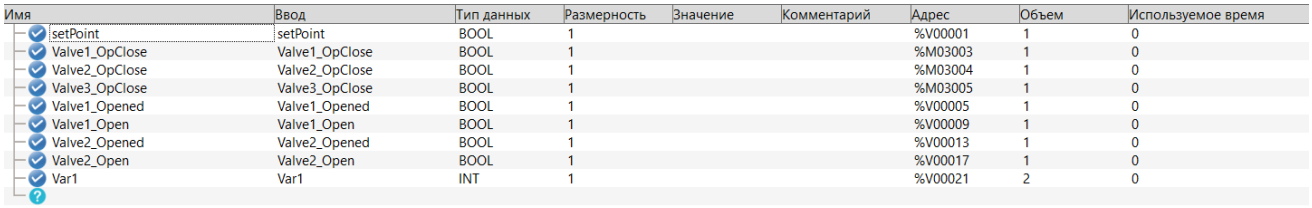
Размер пространства каждой переменной ЦП в серии NA следующий:

Тип процессора	Размер переменной	Диапазон адресов %V
CPU201-1101	32К Байт	%V00001-%V32768
CPU201-1102	32К байт	%V00001-%V32768
CPU401-1101	32К байт	%V00001-%V32768
CPU301-0101	32К байт	%V00001-%V32768
CPU301-0102	32К байт	%V00001-%V32768
CPU401-020X	20К байт	%V00001-%V20480
CPU401-030X	32К байт	%V00001-%V32768
CPU401-0401	32К байт	%V00001-%V32768
CPU401-0501	32К байт	%V00001-%V32768
CPU401-0221	32К байт	%V00001-%V32768
CPU401-0421	32К байт	%V00001-%V32768
CPU401-0521	32К байт	%V00001-%V32768
CPU401-0331	64К байт	%V00001-%V65536
CPU401-0431	64К байт	%V00001-%V65536
CPU401-0531	256К байт	%V00001-%V262144
CPU401-0211	32К байт	%V00001-%V32768
CPU401-0411	32К байт	%V00001-%V32768
CPU401-0511	32К байт	%V00001-%V32768
CPU402-0701	256К байт	%V00001-%V262144
CPU2001-2401	32К байт	%V00001-%V32768
CPU2001-2402	32К байт	%V00001-%V32768
CPU2001-2403	32К байт	%V00001-%V32768
CPU2001-2404	32К байт	%V00001-%V32768
CPU2001-2411	32К байт	%V00001-%V32768
CPU2001-2421	32К байт	%V00001-%V32768
CPU2002-2401	32К байт	%V00001-%V32768

CPU2002-2402	32K байт	%V00001-%V32768
CPU2002-2403	32K байт	%V00001-%V32768
CPU2002-2404	32K байт	%V00001-%V32768

Размер пространства каждой переменной ЦП в серии NJ следующий:

Тип процессора	Размер переменной	Диапазон адресов %V
CPU201-1102	32K Байт	%V00001-%V32768
CPU301-0101	32K Байт	%V00001-%V32768
CPU301-0102	32K Байт	%V00001-%V32768
CPU301-0301	256K Байт	%V00001-%V262144
CPU401-0211	32K Байт	%V00001-%V32768
CPU401-0411	32K Байт	%V00001-%V32768
CPU401-0511	32K Байт	%V00001-%V32768
CPU401-0611	32K Байт	%V00001-%V32768
CPU401-0441	64K Байт	%V00001-%V65536
CPU401-0541	256K Байт	%V00001-%V262144
CPU401-0451	64K Байт	%V00001-%V65536
CPU401-0551	256K Байт	%V00001-%V262144
CPU601-0501	256K Байт	%V00001-%V262144
CPU601-1501	256K Байт	%V00001-%V262144
CPU2001-2401	32K Байт	%V00001-%V32768
CPU2001-2402	32K Байт	%V00001-%V32768
CPU2001-2403	32K Байт	%V00001-%V32768
CPU2001-2404	32K Байт	%V00001-%V32768
CPU2002-2401	32K Байт	%V00001-%V32768
CPU2002-2402	32K Байт	%V00001-%V32768
CPU2002-2403	32K Байт	%V00001-%V32768
CPU2002-2404	32K Байт	%V00001-%V32768



Имя	Ввод	Тип данных	Размерность	Значение	Комментарий	Адрес	Объем	Используемое время
setPoint	setPoint	BOOL	1			%V00001	1	0
Valve1_OpClose	Valve1_OpClose	BOOL	1			%M03003	1	0
Valve2_OpClose	Valve2_OpClose	BOOL	1			%M03004	1	0
Valve3_OpClose	Valve3_OpClose	BOOL	1			%M03005	1	0
Valve1_Opened	Valve1_Opened	BOOL	1			%V00005	1	0
Valve1_Open	Valve1_Open	BOOL	1			%V00009	1	0
Valve2_Opened	Valve2_Opened	BOOL	1			%V00013	1	0
Valve2_Open	Valve2_Open	BOOL	1			%V00017	1	0
Var1	Var1	INT	1			%V00021	2	0

Рисунок 4.3 Таблица переменных

Заметки по редактированию свойств

[Имя]: Он может начинаться только с буквы и состоять из буквы, цифры и подчеркивания. Имя узла на первом уровне таблицы переменных не может совпадать с именем таблицы типов данных.

[Тип данных]: Помимо базового типа, типы, которые можно выбрать для переменных, включают типы всех узлов DDT, которые были проанализированы.

[Размерность]: Пока количество измерений не превышает диапазона соответствующего адреса, ограничений на количество измерений одной переменной нет.

[Адрес]: Адреса можно назначать вручную (%MW, %NW регистры в ЦП) или нет, но область данных %V назначается автоматически. Размер области %V, занимаемой различными базовыми типами, см. в таблице ниже.

Тип данных	Размер, занимаемый переменной %V (байт)	Тип данных	Размер, занимаемый переменной %V (байт)
BOOL	1	INT	2
BYTE	1	DINT	4
WORD	2	REAL	4
DWORD	4	USINT	1
SINT	1	UINT	2
		UDINT	4

[Значения переменных]: Диапазон значений для различных типов переменных следующий.

Тип	Минимальное значение	Максимальное значение
BOOL	0	1
BYTE	0	255
WORD	0	65535
DWORD	0	4294967295
SINT	-128	127
INT	- 32768	32767
DINT	-2147483648	2147483647

REAL	Точность — 6 знаков после запятой, округлено более 6 значений.	
USINT	0	255
UINT	0	65535
BOOL	0	4294967295

Список регистров измерения

Тип регистра измерения

Тип	Имя	Тип данных	Описание
I	Точка цифрового входа	WORD, 0 или 1	Состояние основных цифровых входных регистров
Q	Точка цифрового выхода	BOOL, 0 или 1	Состояние основных цифровых выходных регистров
IW	Точка аналогового входа	INT, сигнал напряжение-ток: от 0 до 20000 Температурный сигнал: -32768 до +32767	Текущее значение базовых аналоговых входных регистров
QW	Точка аналогового выхода	WORD, от 0 до 20000	Текущее значение базовых аналоговых выходных регистров
M	Битовый регистр	BOOL, 0 или 1	Доступная пользователю область хранения для переменных типа bool
MW	Регистр слов	WORD, от 0 до 65535	Доступная пользователю область хранения для переменных типа Word
N	Регистр битов удержания выключения питания	BOOL, 0 или 1	Отличие от регистра M заключается в самосохранении при отключении питания. Данные в регистре N не теряются при выключении ПЛК.
NW	Регистр слов удержания выключения питания	WORD, от 0 до 65535	Отличие от регистра M заключается в самосохранении при отключении питания. Данные в регистре NW не теряются при выключении ПЛК.
S	Системный регистр	BOOL, 0 или 1	В системе определен ряд битовых регистров, отражающих текущее состояние системы, причем каждая точка системного регистра имеет конкретные определения.
SW	Системный регистр слов	WORD, от 0 до 65535	Ряд регистров слов, определенных внутри для отражения текущего состояния системы, которые можно читать, но нельзя записывать.
T	Таймер	DWORD, от 0 до 60000	Таймер, предоставляемый системой пользователю

C	Счетчики	DWORD, от 0 до 4294967295	Счетчик, предоставляемый системой пользователю

Примечание

Регистры %M и %MW для CPU301-0301, CPU402-0701, CPU401-0441, CPU401-0541, CPU601-0501 являются регистрами хранения при отключении питания.

ПЛК предоставляет множество системных регистров для хранения рабочего состояния системы, которое может быть прочитано пользователем, что позволяет легко контролировать рабочие условия ПЛК в реальном времени. Системные регистры определены в следующей таблице.

Серийный номер	Имя	Описание
Системный битовый регистр		
%S0001	FIRST_SCAN	Первое сканирование
%S0002	ALWAYS_ON	Всегда включен
%S0003	ALWAYS_OFF	Всегда выключен
%S0004	T_SECOND	Таймер 1с
%S0006	PRG_OVERRUN	Переполнение при выполнении программы
%S0007	PRG_EXECERR	Ошибка выполнения программы
%S0010	BACKUP_OK	Резервное копирование Master-Slave является нормальным
%S0033	IO_COMERR	Ошибка связи модуля ввода-вывода
%S0034	IO_DIAGERR	Самодиагностика неисправностей в модулях ввода-вывода
%S0035	IO_CFGERR	Несоответствие типа модуля ввода-вывода
%S0036	COM_COMERR	Модуль связи с отказом связи
%S0037	COM_DIAGERR	Модуль связи с самодиагностикой неисправностей
%S0038	COM_CFGERR	Несоответствие типов модулей связи
%S0039	VER_DISMATCH	Версии ведущего и ведомого устройств несовместимы.
%S0041	IO_DOWNLOAD	Инициализация модуля ввода-вывода завершена
%S0042	COM_DOWNLOAD	Инициализация модуля связи завершена
Рабочее состояние ЦП		
%S0097	CPU1_MASTER	CPU1 — главный
%S0098	CPU1_FAULT	Ошибка CPU1
%S0099	CPU1_GPSLOST	Ошибка CPU1 GPS
%S0100	CPU1_SECOND	CPU1 второй свет
%S0101	CPU1_CAN1FLT	Ошибка CPU1 CAN1
%S0102	CPU1_CAN2FLT	Ошибка CPU1 CAN2
%S0103	CPU1_ETH1FLT	Сбой CPU1 ETH1

%S0104	CPU1_ETH2FLT	Сбой CPU1 ETH2
%S0105	CPU1_TASKFLT	Ошибка задачи CPU1
%S0106	CPU1_SELFON	CPU1 в работе
%S0107	CPU1_FATAL	Фатальный сбой CPU1
%S0114	CPU1_PEERON	CPU1 готов к обмену
%S0115	CPU1_PEERMST	CPU1 активный
%S0116	CPU1_STOP	Состояние остановки CPU1
%S0117	CPU1_DEBUG	Состояние отладки CPU1
%S0118	CPU1_NVRAMFLT	Сбой NVRAM CPU1
%S0120	CPU1_FIRST	CPU1 первый
%S0121	CPU2_MASTER	CPU2 как мастер
%S0122	CPU2_FAULT	Ошибка CPU2
%S0123	CPU2_GPSLOST	Ошибка CPU2 GPS
%S0124	CPU2_SECOND	CPU2 второй свет
%S0125	CPU2_CAN1FLT	Ошибка CPU2 CAN1
%S0126	CPU2_CAN2FLT	Ошибка CPU2 CAN2
%S0127	CPU2_ETH1FLT	Сбой CPU2 ETH1
%S0128	CPU2_ETH2FLT	Сбой CPU2 ETH2
%S0129	CPU2_TASKFLT	Ошибка задачи CPU2
%S0130	CPU2_SELFON	CPU2 Локальная Сохранитьона онлайн
%S0131	CPU2_FATAL	Фатальный сбой CPU2
%S0138	CPU2_PEERON	CPU2 готов к обмену
%S0139	CPU2_PEERMST	CPU2 активный
%S0140	CPU2_STOP	Состояние остановки CPU2
%S0141	CPU2_DEBUG	Состояние отладки CPU2
%S0142	CPU2_NVRAMFLT	Сбой NVRAM CPU2
%S0144	CPU2_FIRST	CPU2 первый
Рабочее состояние модуля		
%S0513	MDU001_COMERR	Ошибка связи модуля (адрес 001)
%S0514	MDU001_DIAGERR	Модуль (адрес 001) Самодиагностика неисправности
%S0515	MDU001_CFGERR	Несоответствие типа модуля (адрес 001)
%S0516	MDU001_RVS1	Модуль (адрес 001) зарезервирован 1
%S0517	MDU001_RVS2	Модуль (адрес 001) зарезервирован 2
%S0518	MDU001_RVS3	Модуль (адрес 001) зарезервирован 3
%S0519	MDU001_RVS4	Модуль (адрес 001) зарезервирован 4
%S0520	MDU001_RVS5	Модуль (адрес 001) зарезервирован 5
%S0521	MDU002_COMERR	Ошибка связи модуля (адрес 002)
%S0522	MDU002_DIAGERR	Модуль (адрес 002) Самодиагностика неисправности
%S0523	MDU002_CFGERR	Несоответствие типа модуля (адрес 002)
%S0524	MDU002_RVS1	Модуль (адрес 002) зарезервирован 1
%S0525	MDU002_RVS2	Модуль (адрес 002) зарезервирован 2
%S0526	MDU002_RVS3	Модуль (адрес 002) зарезервирован 3
%S0527	MDU002_RVS4	Модуль (адрес 002) зарезервирован 4
%S0528	MDU002_RVS5	Модуль (адрес 002) зарезервирован 5

…..	…..	…..
…..	…..	…..
Системный регистр слов		
%SW0001	TIME_YEAR	Время: Год
%SW0002	TIME_MONTH	Время: месяц
%SW0003	TIME_DAY	Время: День
%SW0004	TIME_HOUR	Время: Время
%SW0005	TIME_MINUTE	Время: минуты
%SW0006	TIME_SECOND	Время: секунды
%SW0007	TIME_MS	Время: миллисекунды
%SW0009	ALARM_PTR	Указатель тревоги
%SW0010	SOE_PTR	Указатель событий SOE
%SW0011	OVERRUN_INFO	Место переполнения при выполнении программы
%SW0014	EXERERR_INFO	Местоположение ошибки выполнения программы
%SW0018	CPU_TYPE	Тип ЦП
%SW0019	CPU_VER	Версия прошивки ЦП
%SW0021	COM1_SEND	COM1 Статус отправки
%SW0022	COM1_RECV	COM1 Статус получения
%SW0023	COM2_SEND	COM2 Статус отправки
%SW0024	COM2_RECV	COM2 Статус приема
%SW0025	STTM_YEAR	Дата запуска: Год
%SW0026	STTM _MONTH	Время запуска: месяц
%SW0027	STTM _DAY	Время начала: День
%SW0028	STTM _HOUR	Время запуска: часов
%SW0029	STTM _MINUTE	Время запуска: минут
%SW0030	STTM _SECOND	Время запуска: секунд
%SW0031	STTM _MS	Время запуска: миллисекунды
%SW0032	SCAN_TIME	Цикл сканирования

Бит флага связи последовательного порта ЦП NA200H:

Тип ЦП	**COM-порт**	**Системный регистр**	**Статус**	**Описание**
CPU200H	COM1	%SW0513	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0514	Статус Slave 17-32	
		%SW0515	Статус Slave 33-48	
		%SW0516	Статус Slave 49-64	
	COM2	%SW0517	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0518	Статус Slave 17-32	
		%SW0519	Статус Slave 33-48	
		%SW0520	Статус Slave 49-64	
	COM3	%SW0521	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0522	Статус Slave 17-32	
		%SW0523	Статус Slave 33-48	
		%SW0524	Статус Slave 49-64	
	COM4	%SW0525	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0526	Статус Slave 17-32	
		%SW0527	Статус Slave 33-48	

		%SW0528	Статус Slave 49-64	

NA2000 серийный номер:

Тип ЦП	COM-порт	Системный регистр	Статус	Описание
CPU2001	COM1	%SW0257	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0258	Статус Slave 17-32	
		%SW0259	Статус Slave 33-48	
		%SW0260	Статус Slave 49-64	
	COM2	%SW0261	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0262	Статус Slave 17-32	
		%SW0263	Статус Slave 33-48	
		%SW0264	Статус Slave 49-64	
	COM3	%SW0265	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0266	Статус Slave 17-32	
		%SW0267	Статус Slave 33-48	
		%SW0268	Статус Slave 49-64	
	COM4	%SW0269	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0270	Статус Slave 17-32	
		%SW0271	Статус Slave 33-48	
		%SW0272	Статус Slave 49-64	

Флаговый бит связи последовательного порта расширенного модуля NA2000:

Тип модуля расширения	COM-порт	Системные регистры	Статус	Описание
CMM2001-0201	COM1	%SW1029	Статус COM1 1-16	1: сбой коммуникации. 0: общая связь.
	COM2	%SW1030	Статус COM2 1-16	
	/	%SW1031	Статус COM1 1-16	
	/	%SW1032	Статус COM1 1-16	
CMM2001-0401	COM1	%SW1033	Статус COM1 1-16	1: сбой коммуникации. 0: общая связь.
	COM2	%SW1034	Статус COM2 1-16	
	COM3	%SW1035	COM3 1-16 Статус	
	COM4	%SW1036	Статус COM4 1-16	

Бит флага связи последовательного порта ЦП NA300:

Тип ЦП	COM-порт	Системный регистр	Статус	Описание
CPU301-0101	COM1	%SW0513	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0514	Статус Slave 17-32	
		%SW0515	Статус Slave 33-48	
		%SW0516	Статус Slave 49-64	
	COM2	%SW0517	Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0518	Статус Slave 17-32	
		%SW0519	Статус Slave 33-48	
		%SW0520	Статус Slave 49-64	

Если стойка ПЛК сконфигурирована с модулем связи последовательного порта CMM301-0401, каждый последовательный порт считывает бит флага связи ведомого устройства как главного устройства MODBUS.

Тип ЦП	Номер модуля CMM301-0401	Системный регистр	Статус	Описание
CPU301-0101 CPU301-0102	Модуль №1	%SW1953	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1954	COM2 Статус Slave 1-16	
		%SW1955	COM3 Статус Slave 1-16	
		%SW1956	COM4 Статус Slave 1-16	
	Модуль №2	%SW1957	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1958	COM2 Статус Slave 1-16	
		%SW1959	COM3 Статус Slave 1-16	
		%SW1960	COM4 Статус Slave 1-16	
	Модуль №3	%SW1961	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1962	COM2 Статус Slave 1-16	
		%SW1963	COM3 Статус Slave 1-16	
		%SW1964	COM4 Статус Slave 1-16	
	Модуль №4	%SW1965	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1966	COM2 Статус Slave 1-16	
		%SW1967	COM3 Статус Slave 1-16	
		%SW1968	COM4 Статус Slave 1-16	

Если модуль связи последовательного порта CMM401-0411 настроен на стойке ПЛК, каждый последовательный порт служит главной станцией MODBUS для считывания флага состояния связи подчиненной станции, как показано в следующей таблице. Номер модуля увеличивается слева направо, начиная с первой стойки в последовательности конфигурации.

Тип ЦП	Номер модуля CMM301-0401	Системный регистр	Статус	Описание
CPU401-020X CPU401-0221 CPU401-0211	Модуль №1	%SW0961	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0962	COM2 Статус Slave 1-16	
		%SW0963	COM3 Статус Slave 1-16	
		%SW0964	COM4 Статус Slave 1-16	
	Модуль №2	%SW0965	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0966	COM2 Статус Slave 1-16	
		%SW0967	COM3 Статус Slave 1-16	
		%SW0968	COM4 Статус Slave 1-16	
	Модуль №3	%SW0969	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0970	COM2 Статус Slave 1-16	
		%SW0971	COM3 Статус Slave 1-16	
		%SW0972	COM4 Статус Slave 1-16	
	Модуль №4	%SW0973	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW0974	COM2 Статус Slave 1-16	
		%SW0975	COM3 Статус Slave 1-16	
		%SW0976	COM4 Статус Slave 1-16	
CPU401-030X CPU401-0411 CPU401-0421 CPU401-0511 CPU401-0521	Модуль №1	%SW1953	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1954	COM2 Статус Slave 1-16	
		%SW1955	COM3 Статус Slave 1-16	
		%SW1956	COM4 Статус Slave 1-16	
	Модуль №2	%SW1957	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1958	COM2 Статус Slave 1-16	
		%SW1959	COM3 Статус Slave 1-16	

		%SW1960	COM4 Статус Slave 1-16	
	Модуль №3	%SW1961	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1962	COM2 Статус Slave 1-16	
		%SW1963	COM3 Статус Slave 1-16	
		%SW1964	COM4 Статус Slave 1-16	
	Модуль №4	%SW1965	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1966	COM2 Статус Slave 1-16	
		%SW1967	COM3 Статус Slave 1-16	
		%SW1968	COM4 Статус Slave 1-16	
	Модуль №5	%SW1969	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1970	COM2 Статус Slave 1-16	
		%SW1971	COM3 Статус Slave 1-16	
		%SW1972	COM4 Статус Slave 1-16	
	Модуль №6	%SW1973	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1974	COM2 Статус Slave 1-16	
		%SW1975	COM3 Статус Slave 1-16	
		%SW1976	COM4 Статус Slave 1-16	
	Модуль №7	%SW1977	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1978	COM2 Статус Slave 1-16	
		%SW1979	COM3 Статус Slave 1-16	
		%SW1980	COM4 Статус Slave 1-16	
	Модуль №8	%SW1981	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW1982	COM2 Статус Slave 1-16	
		%SW1983	COM3 Статус Slave 1-16	
		%SW1984	COM4 Статус Slave 1-16	
CPU401-0401 CPU401-0501 CPU401-0431 CPU401-0531 CPU401-0611	Модуль №1	%SW4001	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4002	COM2 Статус Slave 1-16	
		%SW4003	COM3 Статус Slave 1-16	
		%SW4004	COM4 Статус Slave 1-16	
	Модуль №2	%SW4005	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4006	COM2 Статус Slave 1-16	
		%SW4007	COM3 Статус Slave 1-16	
		%SW4008	COM4 Статус Slave 1-16	
	Модуль №3	%SW4009	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4010	COM2 Статус Slave 1-16	
		%SW4011	COM3 Статус Slave 1-16	
		%SW4012	COM4 Статус Slave 1-16	
	Модуль №4	%SW4013	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4014	COM2 Статус Slave 1-16	
		%SW4015	COM3 Статус Slave 1-16	
		%SW4016	COM4 Статус Slave 1-16	
	Модуль №5	%SW4017	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4018	COM2 Статус Slave 1-16	
		%SW4019	COM3 Статус Slave 1-16	
		%SW4020	COM4 Статус Slave 1-16	
	Модуль №6	%SW4021	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4022	COM2 Статус Slave 1-16	
		%SW4023	COM3 Статус Slave 1-16	
		%SW4024	COM4 Статус Slave 1-16	
	Модуль №7	%SW4025	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4026	COM2 Статус Slave 1-16	
		%SW4027	COM3 Статус Slave 1-16	
		%SW4028	COM4 Статус Slave 1-16	
	Модуль №8	%SW4029	COM1 Статус Slave 1-16	1: сбой коммуникации. 0: общая связь.
		%SW4030	COM2 Статус Slave 1-16	
		%SW4031	COM3 Статус Slave 1-16	
		%SW4032	COM4 Статус Slave 1-16	

Свойства регистров измерения

Существует два типа атрибутов регистров измерения: общие для всех типов регистров измерения и специфичные для отдельных регистров измерения.

Общие Атрибуты

[**Серийный номер**]: Серийный номер используется для различения регистров измерения. Серийный номер генерируется автоматически и не может быть изменен. Серийный номер может быть напрямую использован в качестве объекта операции функционального модуля или инструкции. Серийный номер состоит из двух частей: типа переменной и серийного номера переменной, например, %I00001, %MW0100 и т. д. Тип регистра измерения указывает тип текущей переменной, например, %I для регистров измерения цифрового входа, %MW для регистров слов; номер переменной указывает номер текущей переменной, и номер переменной не может превышать своего максимального значения. Максимальное значение отличается для разных типов регистров измерения и разных типов ЦП.

Тип	**Имя**	**Максимум серии CPU401-02**	**Максимум серии CPU401-03**	**Максимум серии CPU401-04**	**Максимум серии CPU401-05**	**CPU401-1101 CPU201-1101**
I	Точка цифрового входа	512	1024	2048	2048	512
Q	Точка цифрового выхода	512	1024	2048	2048	512
IW	Аналоговая входная точка	128	256	512	512	128
QW	Аналоговая выходная точка	128	256	512	512	128
M	Битовый регистр	4096	8192	16384	16384	2048
MW	Регистр слов	4096	8192	16384	16384	8192
N	Регистр битов удержания выключения питания	1024	2048	4096	4096	512
NW	Регистр слов удержания выключения питания	1024	2048	4096	4096	512
S	Системный регистр	1024	2048	4096	4096	1024
SW	Системный регистр слов	1024	2048	4096	4096	1024
T	Таймер	256	512	1024	1024	256
C	Счетчики	256	512	1024	1024	256

[**Имя**]: Каждая переменная может быть определена с именем, которое может использоваться непосредственно как объект операции в программах, таких как лестничные

диаграммы и диаграммы управления последовательностью, и может отображаться непосредственно в программе. Например, %I00001 определен как DL_ON (выключатель замкнут), поместите контакт в область редактирования лестничной диаграммы и введите DL_ON в качестве параметра, который будет автоматически распознан программным обеспечением после компиляции. Если определение переменной изменилось и переменная «выключатель замкнут» была изменена на 2-ю точку, просто определить имя %I0002 как DL_ON без каких-либо изменений в лестничной диаграмме. Имя может быть определено на английском или китайском языке.

[**Описание**]: Каждая переменная также может иметь определенное описание, которое является более подробным описанием переменной. Например, имя %I00001 по-прежнему DL_ON, а описание определено как «Breaker Closure», поэтому, если вы не понимаете значение DL_ON при чтении программы релейной логики, вы можете перейти к описанию. Однако содержимое описания не отображается в релейной логике.

[**Число обращений**]: Счетчик использования показывает, сколько раз текущая переменная использовалась в программе. Его можно получить через [Файл] / [Число обращений], сохраните диск и не нужно будет снова считать в следующий раз. Это особенно полезно для таймеров и счетчиков, чтобы проверить, использовались они или нет, чтобы избежать путаницы, вызванной повторным использованием.

Особые атрибуты

[**Адрес модуля**]: Используется только для фактических регистров измерения, таких как точка цифрового входа I, точка цифрового выхода Q, точка аналогового входа IW, точка аналогового выхода QW. Адрес модуля — это адрес модуля, в котором находится текущая переменная, который автоматически генерируется после определения модуля и не может быть изменен. Адрес модуля для шасси 1 — от 1 до 15, для шасси 2 — от 16 до 30 и т. д. Например, если модуль, в котором находится текущая TP, находится в слоте 3 шасси 2, то адрес модуля — 18.

[**Фильтр**]: Используется только для цифровых входов. Каждый цифровой входной сигнал может быть отфильтрован в течение определенного периода времени для предотвращения помех. Время фильтрации измеряется в единицах 10 мс и может быть изменено от 0 до 250 мс. Значение времени фильтрации по умолчанию составляет 10 мс. Если входное значение превышает диапазон, программное обеспечение выведет окно с предупреждением.

[**Форсировать**]: Используется только для фактических регистров измерения, таких как точка цифрового входа I, точка цифрового выхода Q, точка входа режима IW и точка выхода режима QW. После форсирования переменной вы можете установить значение переменной в соответствии с вашими требованиями, независимо от текущего значения сигнала. Форсирование можно выполнить только в режиме онлайн, дважды щелкнув по столбцу «Форсировать» переменной, которую нужно форсировать.

Символ √ указывает на то, что точка принудительно установлена, дважды щелкните еще раз, когда столбец принудительной установки пуст, указывая на то, что точка не принудительно установлена. В принудительном состоянии состояние сигнала, сканируемое системой, больше не отправляется в память регистра.

[**Значение**]: Отражает текущее значение фактических регистров измерения переменной цифрового входа I, переменной цифрового выхода Q, аналогового входа IW и аналогового выхода QW в состоянии онлайн; отражает текущее значение виртуальных регистров измерения битового регистра M, регистра слов MW, регистра битов удержания при отключении питания N, регистра слов удержания при отключении питания NW, системного битового регистра S и системного регистра слов SW в состоянии онлайн. Для системных регистров S и SW они могут быть только прочитаны, но не записаны.

[**Тип сигнала**]: Используется только для аналоговых входных модулей. Аналоговые входные модули могут иметь различные типы сигналов, такие как напряжение, ток и RTD, и существуют различные диапазоны для одного и того же типа входного сигнала. Для регистров измерения модуля тока типы сигналов могут быть 4–20 мА, 0–20 мА и 0–10 мА; для регистров измерения модуля напряжения типы сигналов могут быть 0–5 В, 1–5 В, -5–5 В, 0–10 В, -10–10 В; для регистров измерения модуля ввода RTD типы сигналов могут быть Pt100, Cu50, Cu53, Cu100, Pt1000 и Ni Cu100, Pt1000 и Ni1000.

[**Предустановленное значение**]: Используется только для таймера T и счетчика C. Проверьте предустановленное время таймера или предустановленное значение счетчика в режиме онлайн.

[**Текущее значение**]: Используется только для таймера T и счетчика C. Просмотр текущего времени таймера или текущего значения счетчика в режиме онлайн.

[**Измерение**]: Используется только для переменных. Максимальное количество измерений переменной — 4096. Индексы для переменных начинаются с 0. Например, если V0001 определен как status, а количество измерений — 100, то можно использовать status[0] — status[99]. Если переменная является одномерной, вам не нужно вводить индекс, только имя переменной, например, V0001 (с именем status) и измерением 1, вы

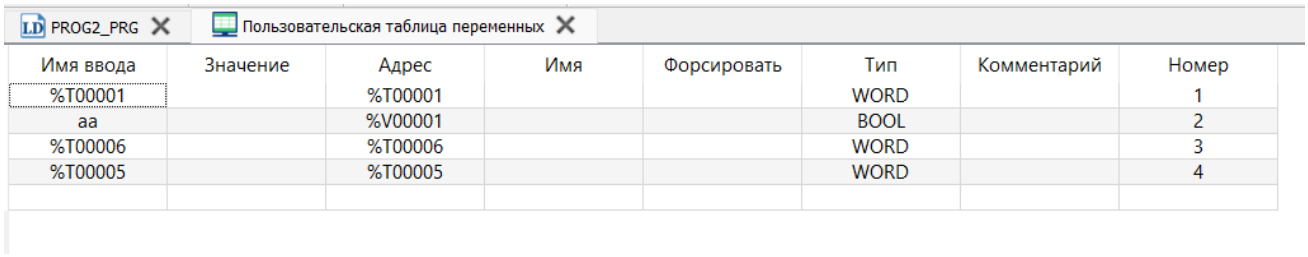
можете использовать status вместо status[0]. Имя переменной должно быть определено во время использования, например, V0001 с именем status можно использовать только с status[0], а не с V0001[0].

[**Тип**]: Используется только для переменных. Выберите тип данных переменной через раскрывающееся меню, которое может быть любым типом данных в ПЛК, включая BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT и другие типы (подробнее о типах данных см. в разделе «Типы данных» этой главы), но каждая группа переменных может быть определена только как один и тот же тип данных.

[Смещение нуля]: Используется только для переменной аналогового входа IW, которая компенсирует погрешность собранного сигнала.

Пользовательская таблица переменных

Изменить/просмотреть значение переменной онлайн. Пользовательская таблица переменных может определять точки, которые пользователи хотят наблюдать, так что пользователи могут наблюдать только нужные им переменные при отладке программ, что удобно для отладки. В дереве проекта выберите Данные/ Пользовательская таблица переменных, и пользовательская таблица переменных появится. Если она используется впервые, таблица будет пустой, как показано на рисунке 4.4.



Имя ввода	Значение	Адрес	Имя	Форсировать	Тип	Комментарий	Номер
%T00001		%T00001			WORD		1
aa		%V00001			BOOL		2
%T00006		%T00006			WORD		3
%T00005		%T00005			WORD		4

Рисунок 4.4 Пользовательская таблица переменных

Добавить точку

Один раз щелкните правой кнопкой мыши и выберите [Добавить из программы]. Появится окно [Выбор программы]. Выберите программу с переменными, после чего все регистры и переменные, используемые в программе, будут введены в эту таблицу, как показано на рисунке 4.5.

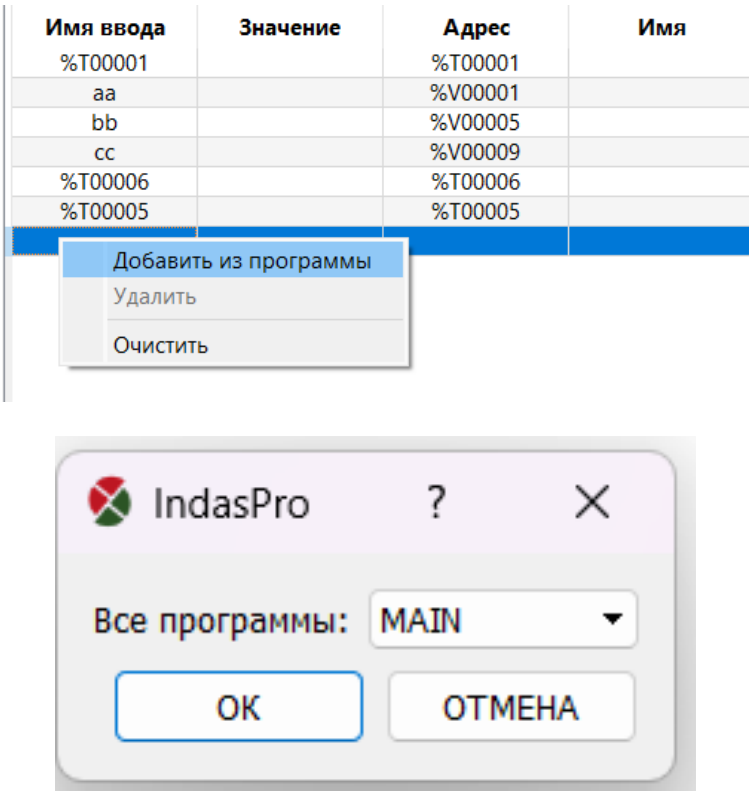


Рисунок 4.5 Добавление переменных из программы

После добавления переменных пользователи могут наблюдать значения, имена, форсирование переменных, как показано на рисунке 4.6.

Имя ввода	Значение	Адрес	Имя	Форсировать	Тип	Комментарий	No.
setPoint		%V00001			BOOL		1
Valve2_Opened		%V00013			BOOL		2
Valve2_Open		%V00017			BOOL		3
%T00001		%T00001			WORD		4

Рисунок 4.6 Пользовательская таблица переменных

[**Форсировать**]:Для %M、%MW、%N、%NW、%M、%MW、%N、%NM%Q、%QW и переменных измените переменные, дважды щелкнув значение переменных. Для %I и %IW дважды щелкните мышью сначала в опции [Force] и отметьте выбор силы, затем вы можете изменить значение.

[**Удаление и редактирование переменной**]: Если некоторые переменные не нужны, их также можно удалить или изменить на другие переменные. Просто щелкните правой кнопкой мыши по панели, чтобы сделать выбор, как показано на рисунке 4.7.

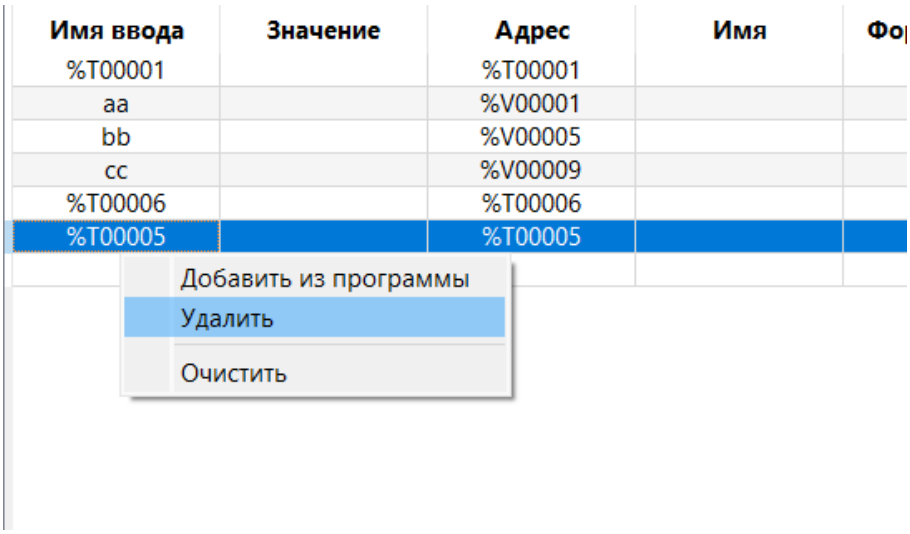


Рисунок 4.7 Удаление переменной из таблицы

[**Добавить пользовательскую таблицу переменных**]: В соответствии со своими потребностями пользователи могут создать ряд пользовательских таблиц переменных, до 16 таблиц. Щелкните правой кнопкой мыши на «Пользовательская таблица переменных» и выберите «Добавить пользовательскую таблицу переменных», как показано на рисунке 4.8.

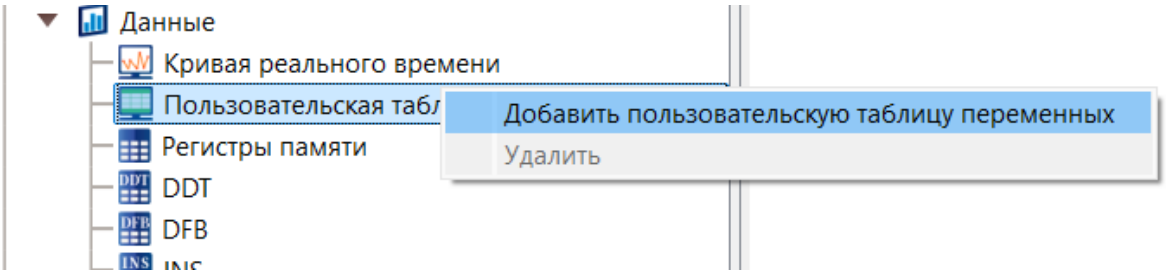


Рисунок 4.8 Добавление пользовательской таблицы переменных

[**Удалить пользовательскую таблицу переменных**]: Если вы не хотите сохранять и удалять вновь созданную регистры памяти, очистите таблицу переменных, а затем закройте диалоговое окно, система автоматически удалит таблицу переменных.

Пользовательская таблица переменных сохраняется в текущем компьютере, используемом для отладки. При следующем выходе в онлайн пользовательская таблица переменных останется в состоянии последнего выхода. Пользовательская таблица переменных не загружается в ПЛК.

Метод адресации

Адресация, т.е. метод доступа к точкам измерения для модулей лестничных функций и других методов программирования. Доступны следующие типы адресации.

1. Немедленное обращение

CONSTANT используется непосредственно как объект доступа. Модуль функции назначения, показанный на рисунке 4.9 ниже, выполняет функцию назначения 100 %MW0001.

В случае режима «немедленной адресации» вход 100 представляет собой режим немедленной адресации.

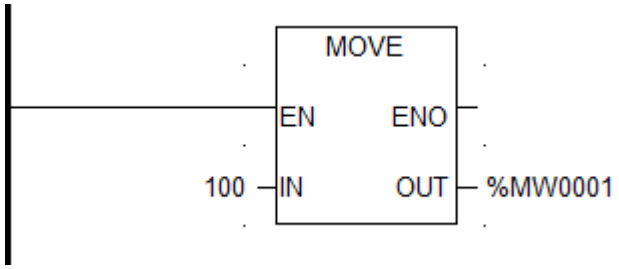


Рисунок 4.9 Немедленная адресация

Примечание
Если входные данные шестнадцатеричные, в конце данных следует добавить H, а если первые данные — буква, данные должны начинаться с 0. Например, если вы хотите ввести B021H, вы должны написать 0B021H, в противном случае системный компилятор сообщит об ошибке.

2. Прямая адресация

Прямая адресация основана на типе переменной плюс регистр переменной как объект доступа.

Как особый случай, использование имени переменной также является прямой адресацией, поскольку имя переменной соответствует указанной точке измерения. Например, если имя переменной %I00001 определено как GD_ON, вы можете использовать GD_ON непосредственно в программе, ссылаясь на %I00001.

3. Косвенная адресация

Косвенная адресация означает, что номер переменной объекта доступа не является константой, а другой переменной, т. е. в качестве номера переменной используется

текущее значение переменной. В отличие от прямой адресации, которая начинается с 0, косвенная адресация начинается с 0, т. е. для косвенно адресуемой переменной %I[%MW0001] текущее значение %MW0001 равно 1, а переменная, к которой она обращается, равна %I0002, а не %I00001.

Лестница, показанная на рисунке 4.10, %Q[%MW0001] — это метод косвенной адресации. Операция, выполняемая лестницей, следующая: %MW0001 присваивается значение 0, а катушка выводит 1 для открытой точки %Q0001.

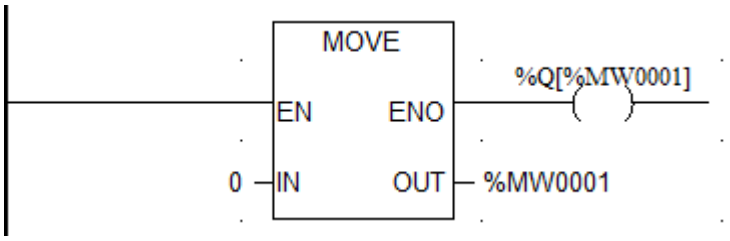


Рисунок 4.10 Косвенная адресация

Как показано на схеме на рисунке 4.11, %Q[%MW0001+1] также адресуется косвенно. Тип регистра — Q, а номер точки — не CONSTANT, а квадратичное выражение. Схема выполняет следующую операцию: %MW0001 присваивается значение 0, затем к %MW0001 добавляется 1, и катушка выдает 1 для открытой переменной %Q[1] (%Q0002).

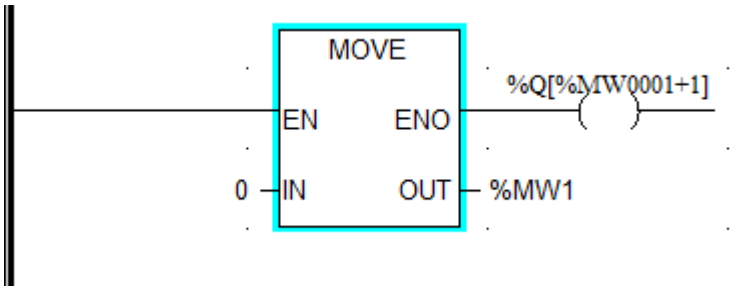


Рисунок 4.11 Косвенная адресация

Базовый функциональный блок

Обзор

В графическом языке LD и FBD функциональный блок представляет собой рамку с входами и выходами. Входы всегда находятся слева, а выходы справа. Имя функционального блока, то есть тип функционального блока, показанного в центре рамки. Все основные функциональные блоки в этой главе могут быть вызваны в LD, FBD, ST, IL.

Изменение свойства

Свойство каждого функционального блока может быть изменено. EN/ENO может быть отображено/скрыто в FBD. Некоторые функциональные блоки могут добавлять количество входов.

Как показано на рисунке 5.1, в редакторе LD выберите функциональный блок, свойство которого вы хотите наблюдать (функциональный блок ADD на рисунке), а затем выберите свойство, щелкнув правой кнопкой мыши. Появится диалоговое окно, показанное на рисунке 5.2. Количество входов можно изменить, используя раскрывающийся список номеров входов.

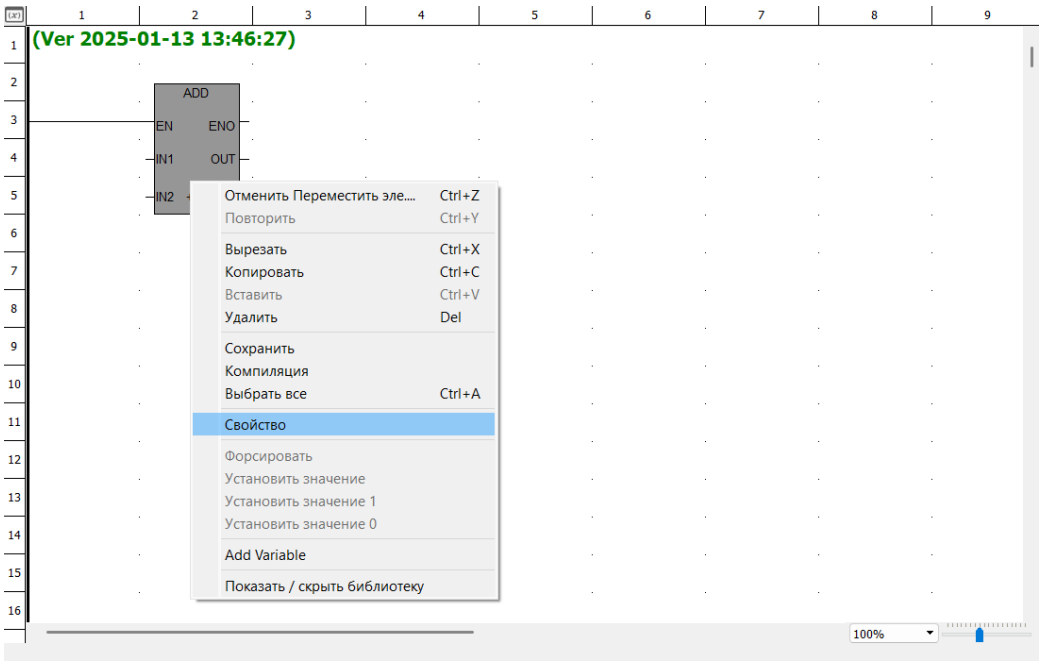


Рисунок 5.1 Открытие «свойства» функционального блока

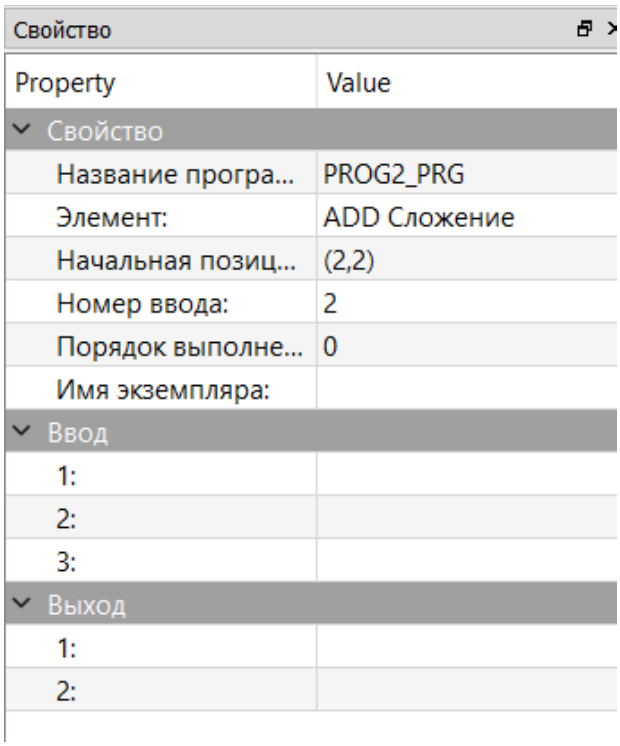


Рисунок 5.2 Диалоговое окно изменения свойства

AH/ENO

Все функциональные блоки могут быть сконфигурированы с EN Input и ENO Output. Если EN равен 0 при вызове функционального блока, определенный алгоритм не будет выполнен, а ENO будет установлен на 0. Если EN равен 1, определенный алгоритм будет выполнен, а ENO будет установлен на 0. Если во время выполнения алгоритма произойдет ошибка, ENO также будет установлен на 0.

Арифметическая операция

Функциональные блоки арифметической операции можно рассматривать как два процесса. Выполнить арифметическую операцию, а затем присвоить значение назначенным точкам. Входные данные арифметической операции не изменятся.

Функция арифметической операции будет выполнена, когда EN удовлетворяет условию.

Тип	Описание
ADD	OUT = IN1 + IN2 + … + INn
SUB	OUT = IN1 - IN2
MUL	OUT = IN1 * IN2 * … * INn
DIV	OUT = IN1 / IN2
MOD	OUT = IN1 % IN2
DIVMOD	DV = IN1 / IN2 MD = IN1 % IN2
INC	INOUT = INOUT + 1
DEC	INOUT = INOUT - 1
NEG	OUT = 0 - IN
SIGN	IF IN < 0, OUT = 1 IF IN ≥ 0, OUT = 0
SQRT	OUT = $\sqrt{IN}$
ABS	OUT = | IN |
LOG	OUT = $\log_{10}^{IN}$
LN	OUT = ln IN, i.e., OUT = $\log_{e}^{IN}$ (e = 2.718282)
EXP	OUT = eIN (e = 2.718282)
EXPT	OUT = IN1IN2
SIN	OUT = sin IN
COS	OUT = cos IN
TAN	OUT = tan IN
ASIN	OUT = asin IN
ACOS	OUT = acos IN
ADD	OUT = atan IN

ADD

Описание функции

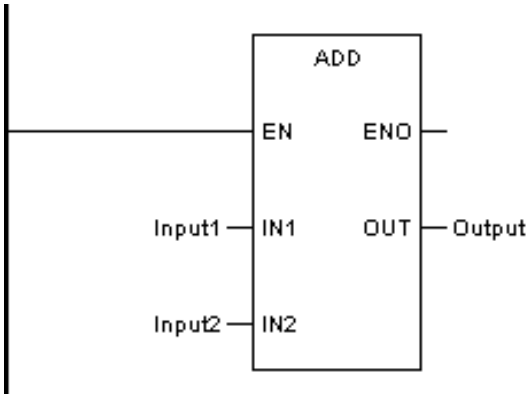
- Этот функциональный блок складывает все входные значения и присваивает результат выходу.
- Максимальное количество входов: 8.

Уравнение

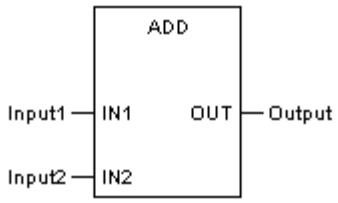
OUT = IN1 + IN2 + … + INn

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
ADD	Input2
ST	Output

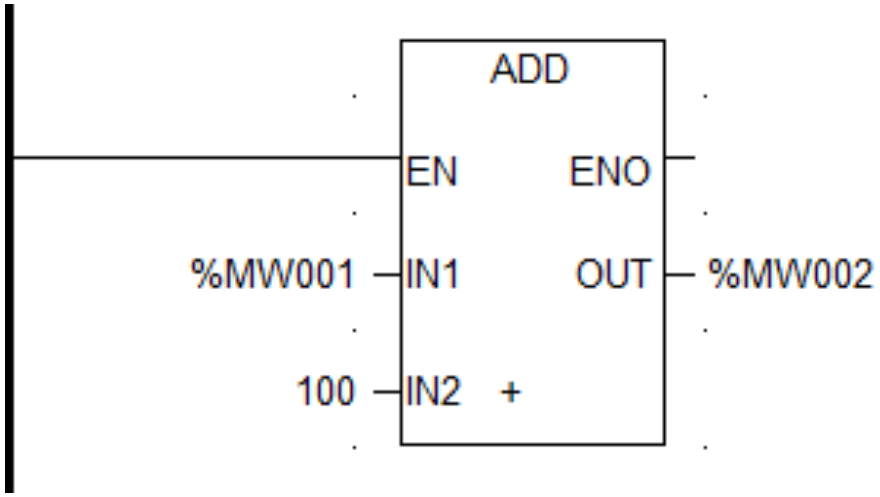
Форма на языке ST:

Output := ADD (Input1, Input2);

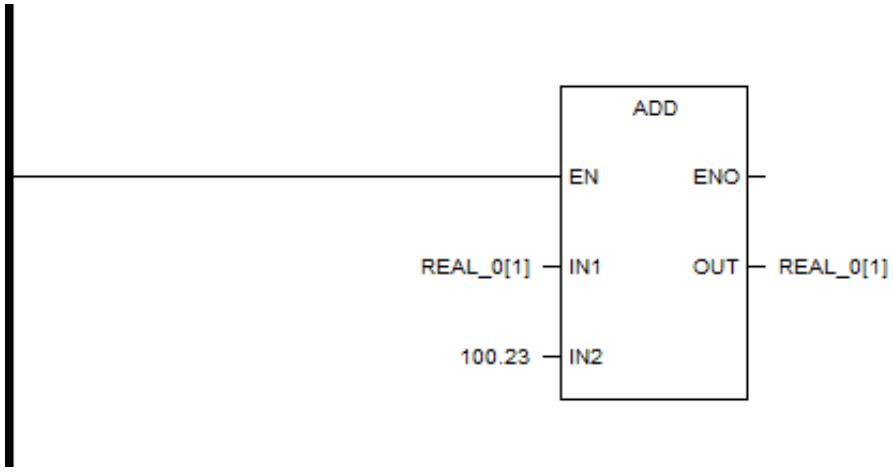
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Слагаемое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Слагаемое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Сумма	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

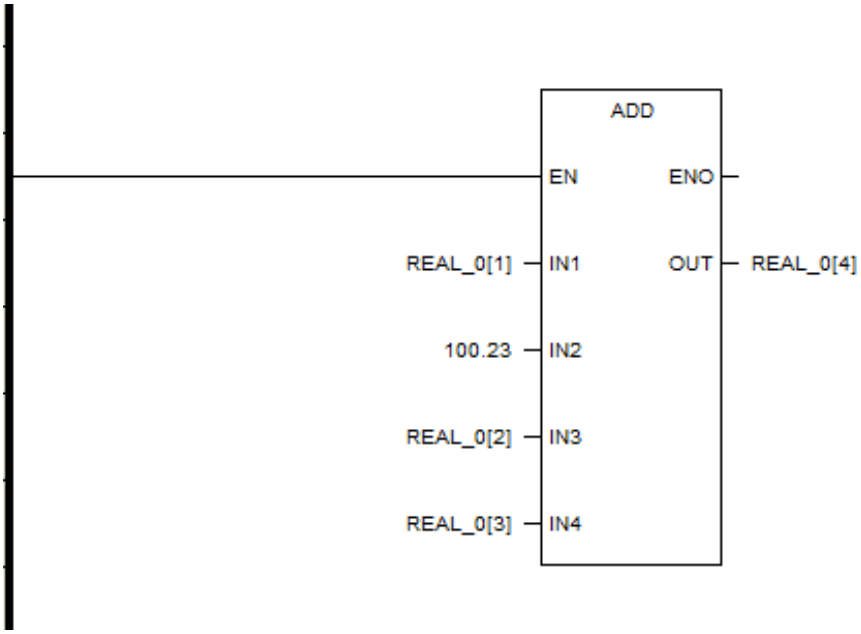
ПРИМЕР 1: Сложение целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Добавление переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



ПРИМЕР 3: Сложение нескольких чисел



SUB

Описание функции

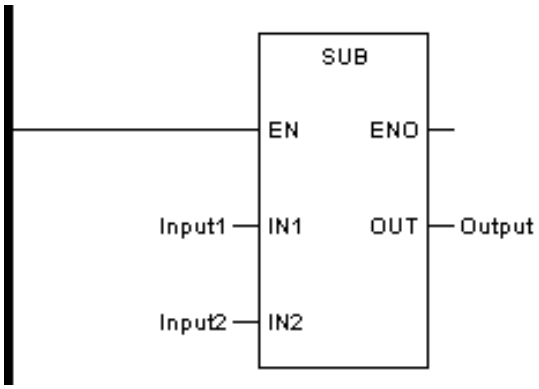
- Этот функциональный блок вычитает значение IN2 из значения IN1 и присваивает результат выходу.

Уравнение

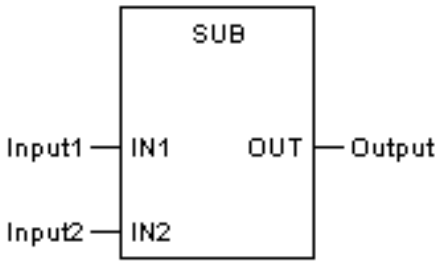
OUT = IN1 - IN2

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
SUB	Input2
ST	Output

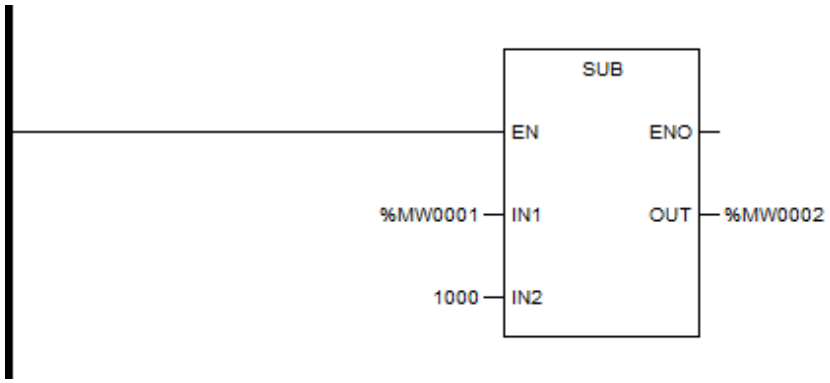
Форма на языке ST:

Output: = SUB (Input1, Input2);

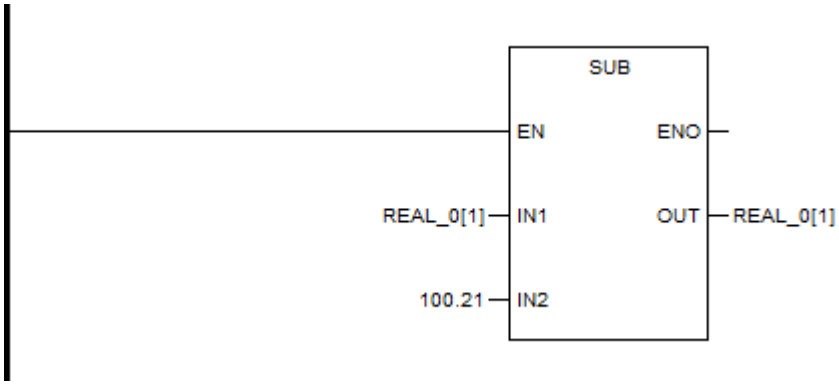
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Уменьшаемое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вычитаемое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Разность	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Вычитание целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Вычитание переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



MUL

Описание функции

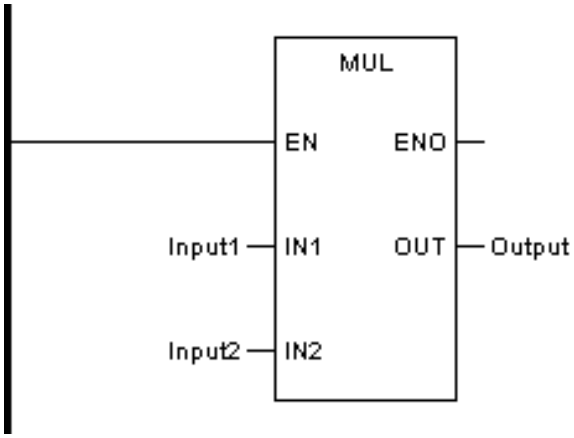
- Этот функциональный блок перемножает все входные значения и присваивает результат выходу.
- Максимальное количество входов: 8.

Уравнение

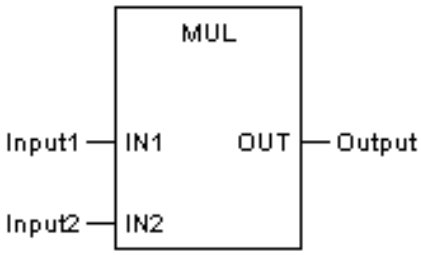
OUT = IN1 * IN2 * … * INn

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

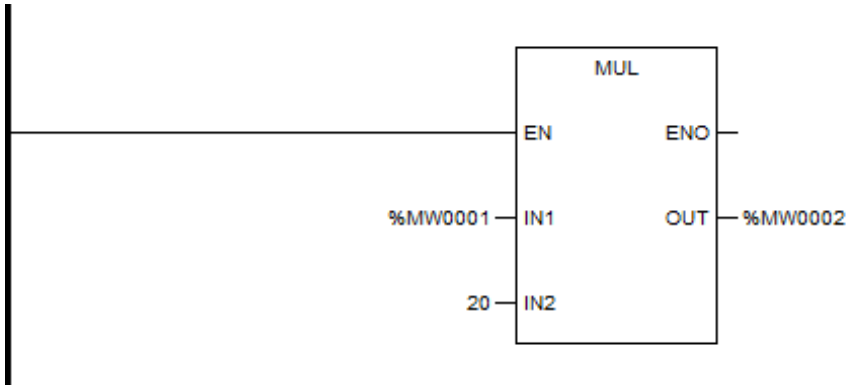
LD	Input1
MUL	Input2
ST	Output

Форма на языке ST:

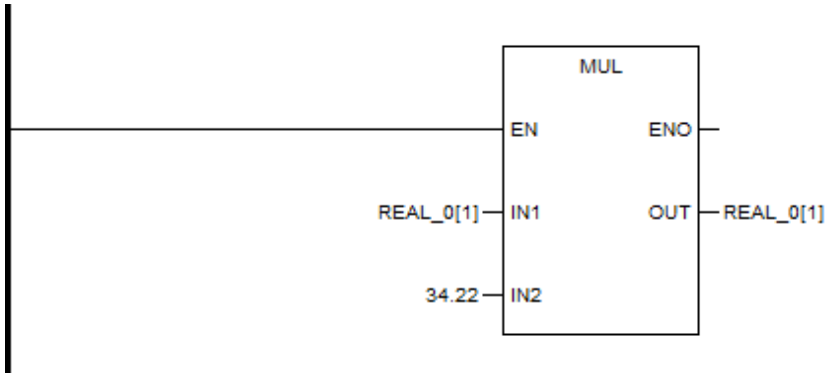
Output := MUL (Input1, Input2);

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Множимое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Множитель	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Произведение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

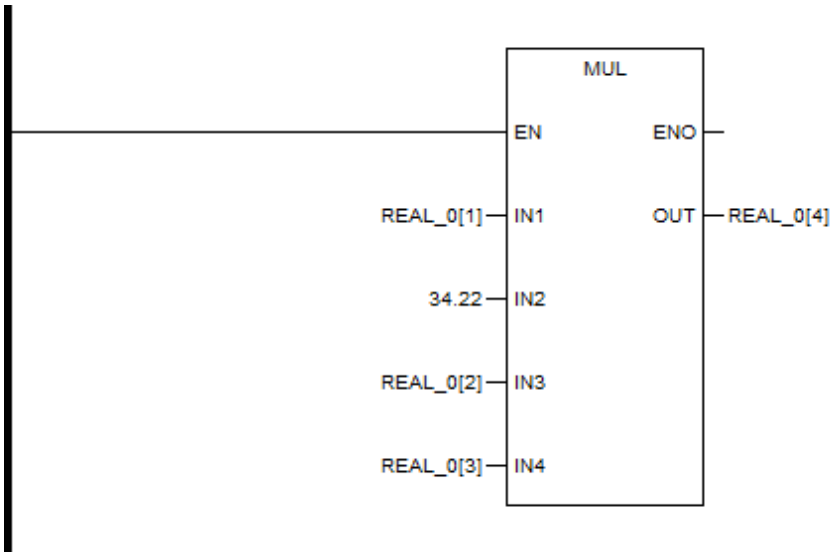
ПРИМЕР 1: Умножение целых чисел (тип регистра MW — WORD)



ПРИМЕР 2: Умножение переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



ПРИМЕР 3: Умножение нескольких чисел



DIV

Описание функции

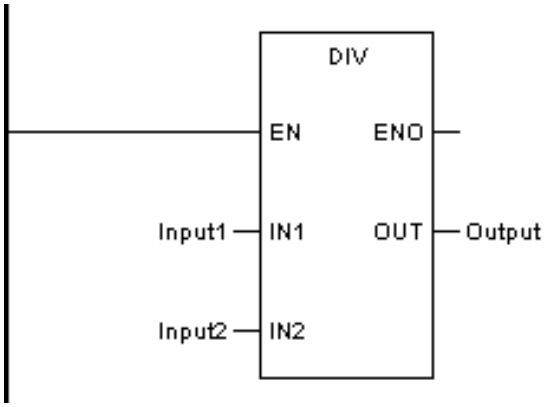
- Этот функциональный блок делит значение IN1 на значение IN2 и присваивает результат выходу.
- При делении целочисленного типа данных все десятичные знаки в результате игнорируются, например, 5÷2＝2 ; -5÷2＝-2.

Уравнение

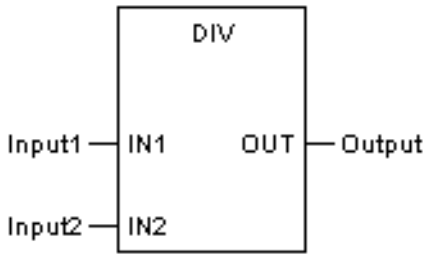
OUT = IN1 / IN2

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
DIV	Input2
ST	Output

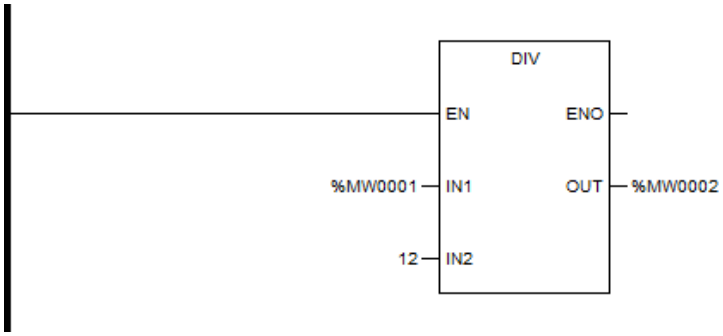
Форма на языке ST:

Output := DIV (Input1, Input2);

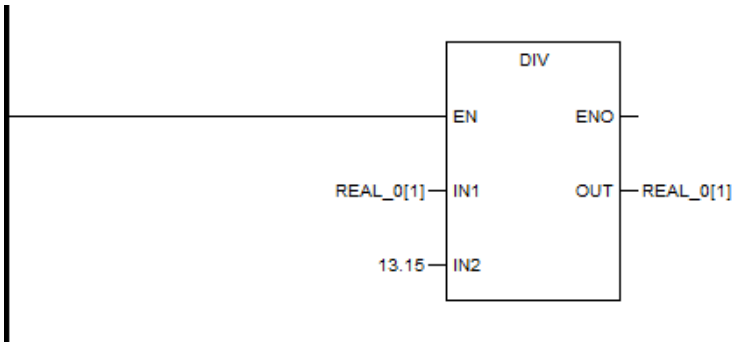
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Делимое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Делитель	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Частное	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Деление целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Деление переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)


DIV
EN
ENO
REAL_0[1]
IN1
OUT
REAL_0[1]
13.15
IN2

MOD

Описание функции

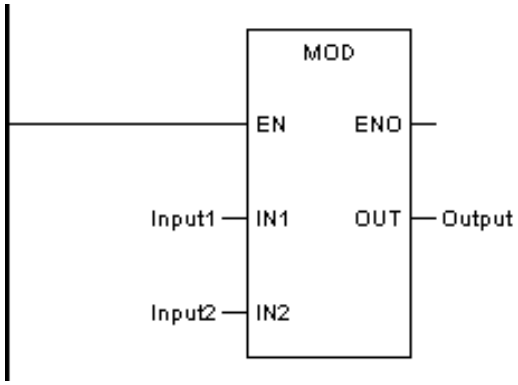
- Этот функциональный блок делит значение IN1 на значение IN2 и присваивает остаток выходу.

Уравнение

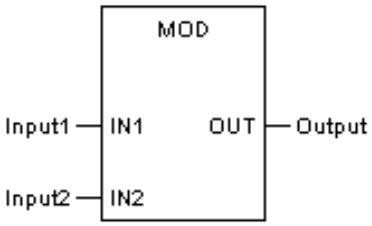
OUT = IN1% IN2

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
MOD	Input2
ST	Output

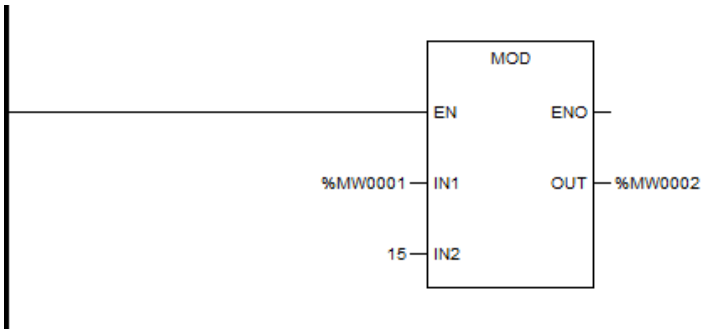
Форма на языке ST:

OUT := MOD (Input1, Input2);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Делимое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Делитель	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Остаток	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Остаток целого числа (тип регистра MW — WORD)



DIVMOD

Описание функции

- Этот функциональный блок делит значение IN1 на значение IN2. Затем блок присваивает остаток MD и присваивает частное DV.

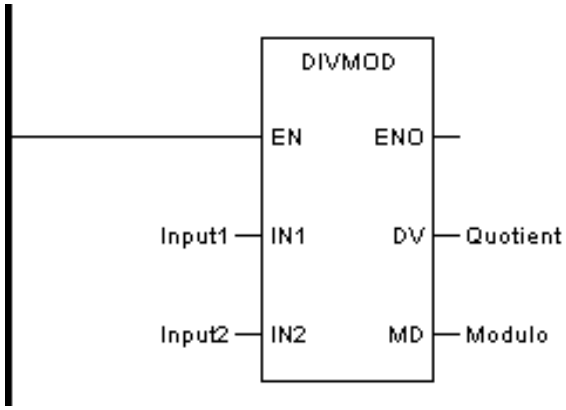
Уравнение

DV=IN1/IN2

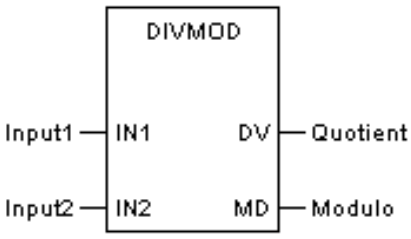
MD = IN1% IN2

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL DIVMOD (IN1:=Вход1, IN2:=Вход2, DV=>Частное, MD=>Модуль)

Форма на языке ST:

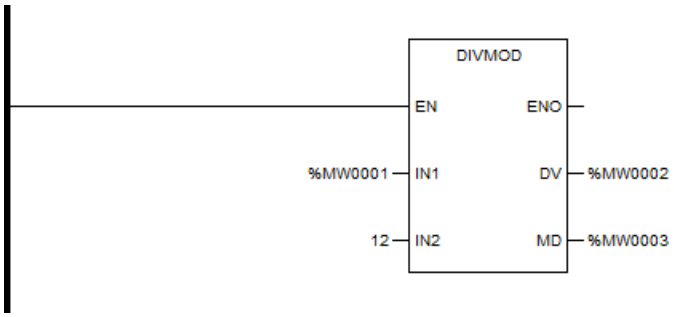
DIVMOD (IN1:=Вход1, IN2:=Вход2, DV=>Частное, MD=>Модуль)

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра

IN1	Input1	Делимое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Делитель	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DV	Quotient	Частное	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR
MD	Modulo	Остаток	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: DIVMOD целого числа (тип регистра MW — WORD)



ПРИБАВИТЬ 1 - INC

Описание функции

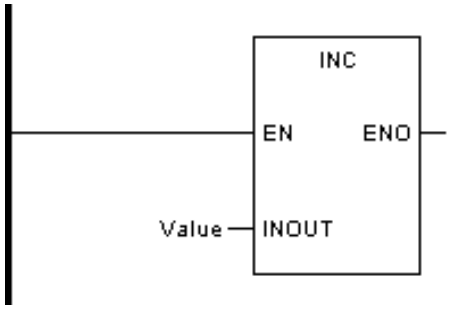
- Каждый раз при выполнении функционального блока значение INOUT будет увеличиваться на 1.

Уравнение

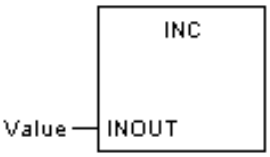
INOUT=INOUT+1

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL INC (значение)

Форма на языке ST:

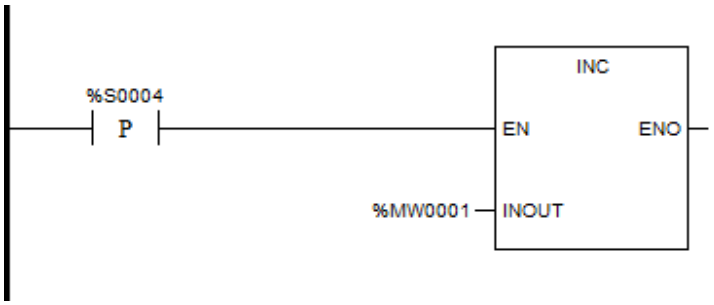
INC (значение);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
INOUT	Input1	К переменной необходимо прибавить 1, и она будет являться выходом результатов операции.	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР: Плюс 1 за каждую секунду.

(Тип регистра MW — WORD, а %S0004 — таймер на 1 с)



ОТНЯТЬ 1 - DEC

Описание функции

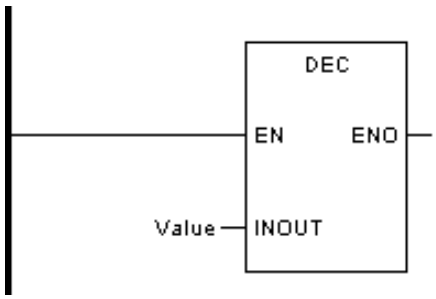
- Каждый раз при выполнении функционального блока значение INOUT будет вычитаться на 1.

Уравнение

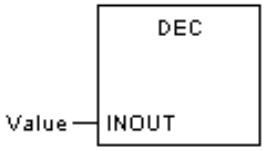
INOUT=INOUT-1

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL DEC (Значение)

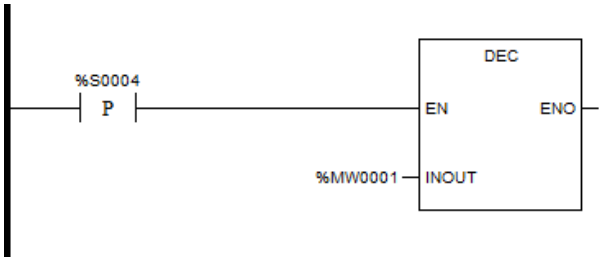
Форма на языке ST:

DEC (Значение);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
INOUT	Input1	Переменную необходимо вычесть из 1, и она является результатом операции.	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР: Минус 1 каждую секунду. (Тип регистра MW — WORD, а %S0004 — таймер на 1 с)



NEG

Описание функции

- Функциональный блок принимает отрицательное число на входном значении и присваивает результат выходу.

  Поменяйте знак, например:

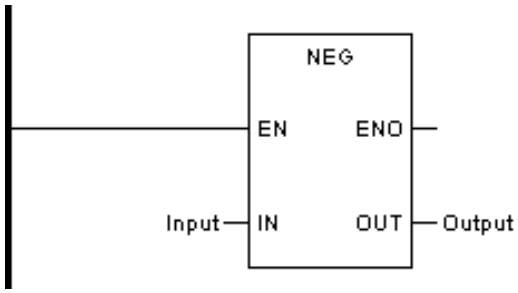
  300 -> -300

Уравнение

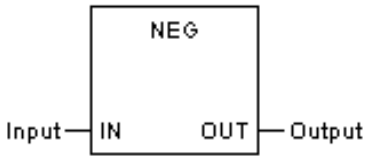
OUT=0 – IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
NEG	
ST	Output

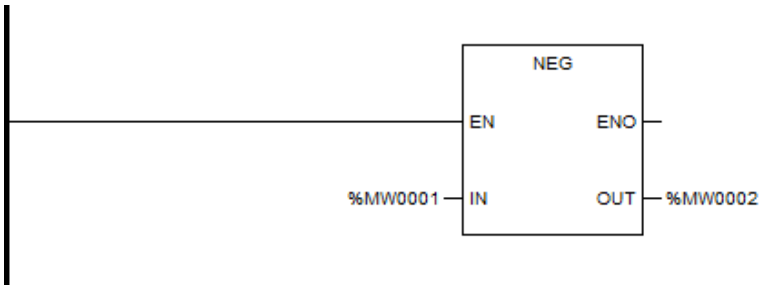
Форма на языке ST:

Output := NEG (Input)

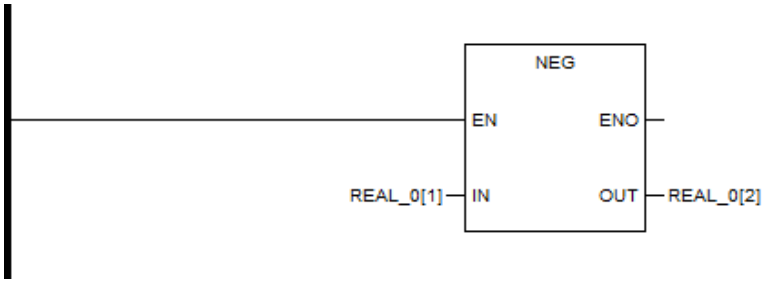
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input1	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Отрицатель-ный выход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: NEG для целого числа (тип регистра MW — WORD)



ПРИМЕР 2: NEG для переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



SIGN

Описание функции

- Функциональный блок используется для обнаружения отрицательного знака.

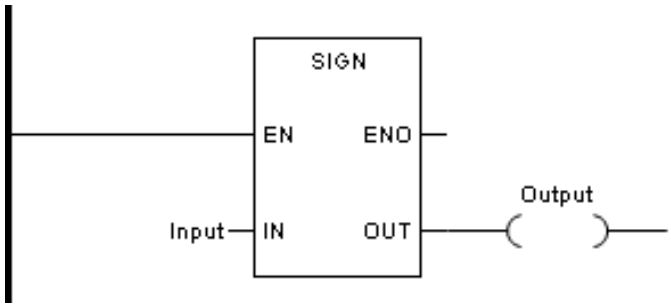
Уравнение

IF IN < 0, OUT = 1

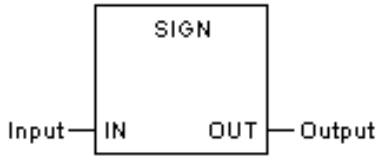
IF IN ≥ 0, OUT = 0

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
SIGN	
ST	Output

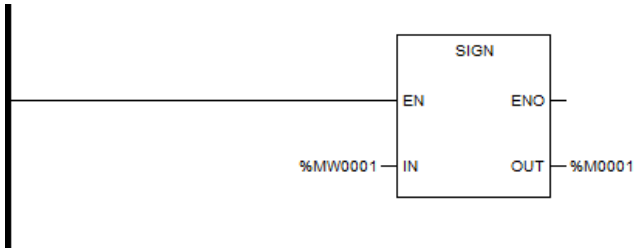
Форма на языке ST:

Output := SIGN (Input)

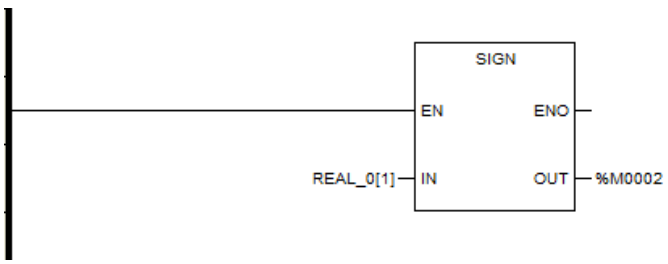
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: ЗНАК для целого числа (тип регистра MW — WORD)



ПРИМЕР 2: ЗНАК для переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



SQRT

Описание функции

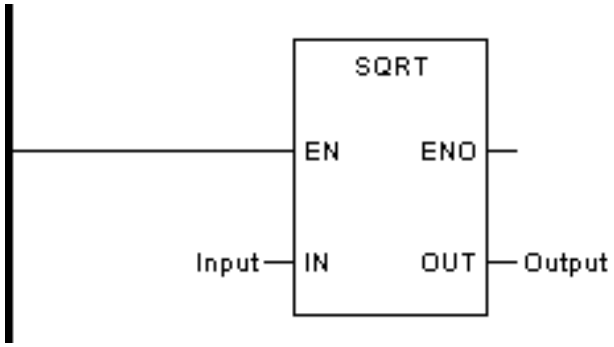
- Этот функциональный блок используется для вычисления квадратного корня переменной.
- Рассчитываемая переменная должна быть больше или равна нулю.

Уравнение

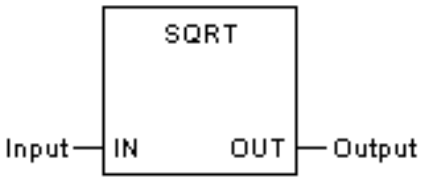
OUT = $\sqrt{IN}$

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
SQRT	
ST	Output

Форма на языке ST:

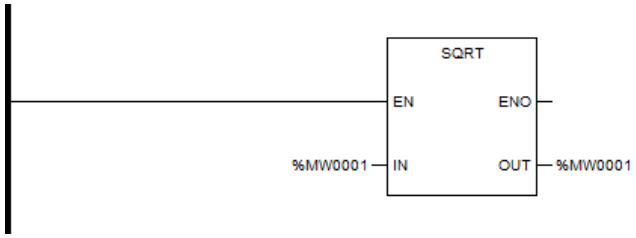
Output := SQRT (Input)

Описание параметра

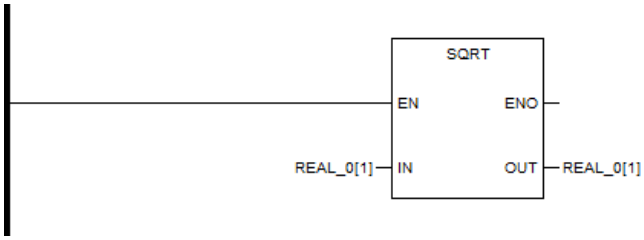
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Квадратный корень	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: SQRT для целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного квадратного корня может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: SQRT для переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



ABS

Описание функции

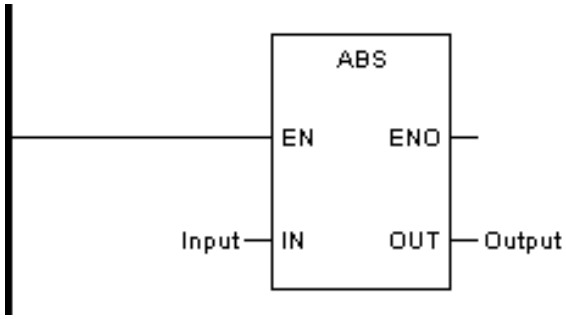
- Этот функциональный блок используется для вычисления абсолютного значения входного сигнала, а затем присваивает результат выходному сигналу.

Уравнение

OUT = |IN|

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма на языке IL:

LD	Input
ABS	
ST	Output

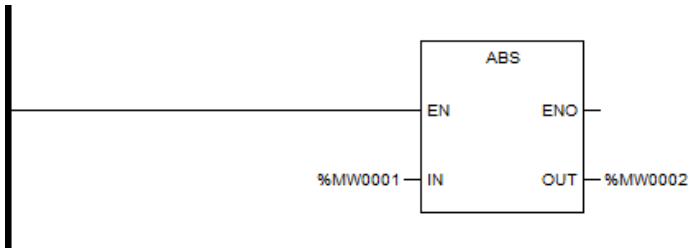
Форма на языке ST:

Output := ABS (Input)

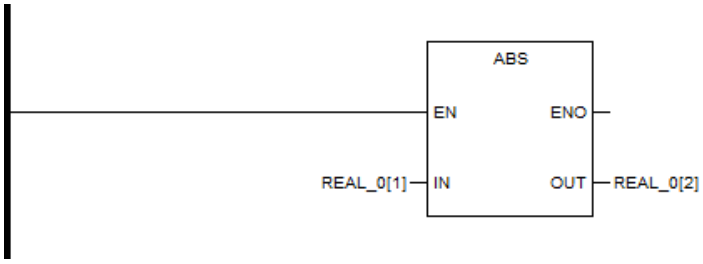
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Абсолютное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Абсолютное значение для целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Абсолютное значение для переменной с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



LOG

Описание функции

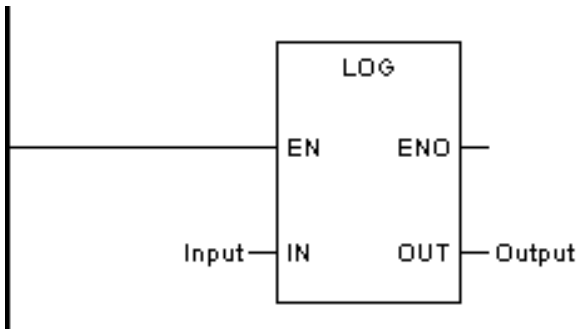
- Этот функциональный блок используется для вычисления логарифма по основанию 10.

Уравнение

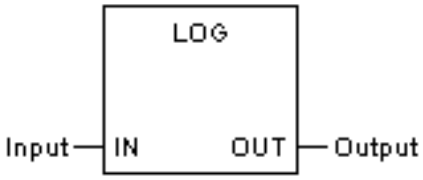
OUT = $\log_{10}^{IN}$

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
LOG	
ST	Output

Форма на языке ST:

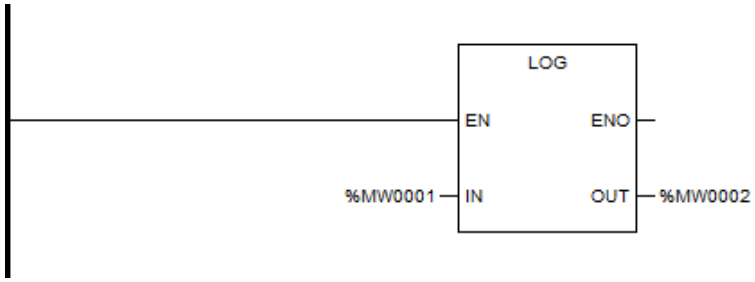
Output := LOG (Input)

Описание параметра

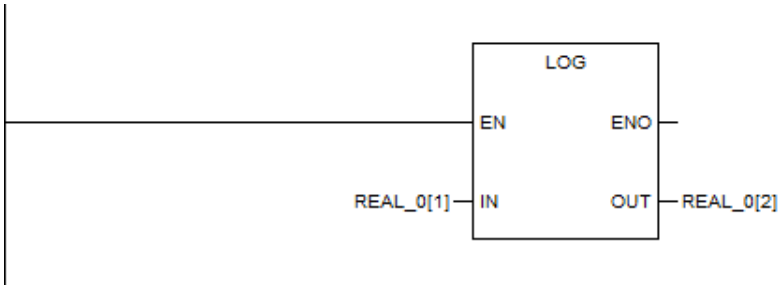
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Логарифм	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Логарифм целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного логарифма может отображать только целые части, если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Логарифм числа с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



LN

Описание функции

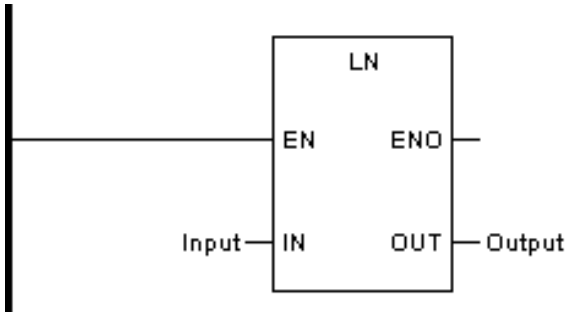
Этот функциональный блок используется для вычисления Неперова логарифма.

Уравнение

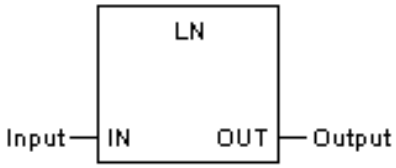
OUT = $\log_e^{IN}$, e=2.718282

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
LN	

ST	Output

Форма на языке ST:

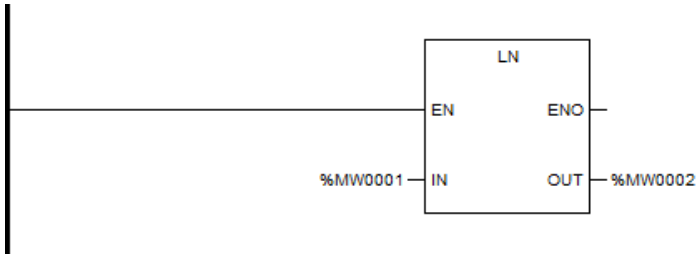
Output := LN (Input)

Описание параметра

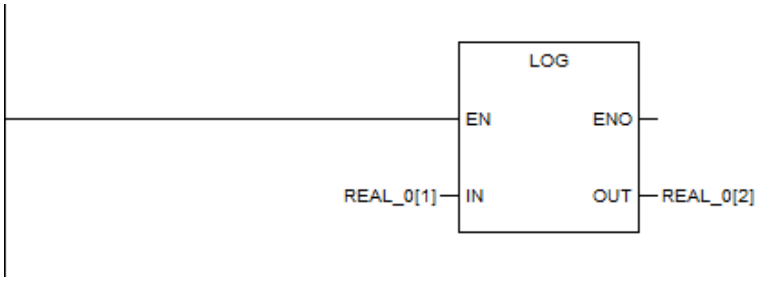
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Неперов логарифм	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Неперовский логарифм целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного логарифма Непера может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Неперовский логарифм числа с плавающей запятой (REAL_0[1] и REAL_0[2] — определяемая пользователем переменная с плавающей запятой)



EXP

Описание функции

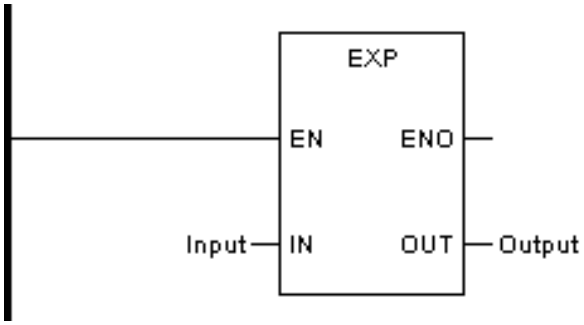
- Этот функциональный блок используется для вычисления натурального экспоненциального значения.

Уравнение

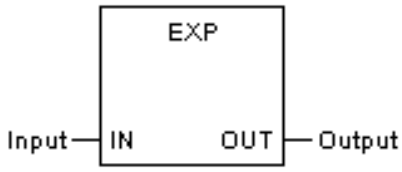
OUT = e^{IN}

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
EXP	
ST	Output

Форма на языке ST:

Output := EXP (Input)

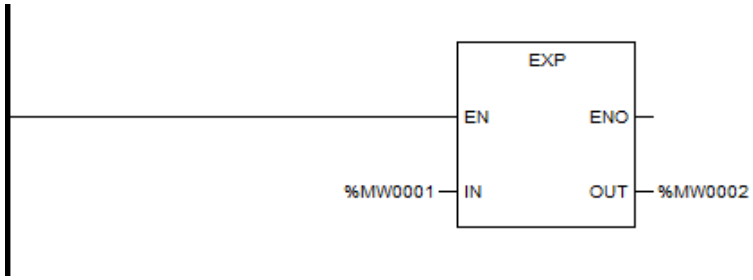
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

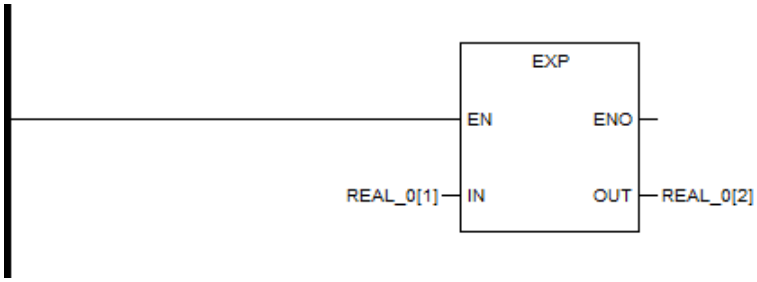
OUT	Output	Натуральная экспонента	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Натуральная экспонента целого числа (тип регистра MW — WORD)

Примечание: целочисленный экспоненциальный вывод может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Натуральная экспонента числа с плавающей запятой (REAL_0[1] — определяемая пользователем переменная с плавающей запятой)



EXPT

Описание функции

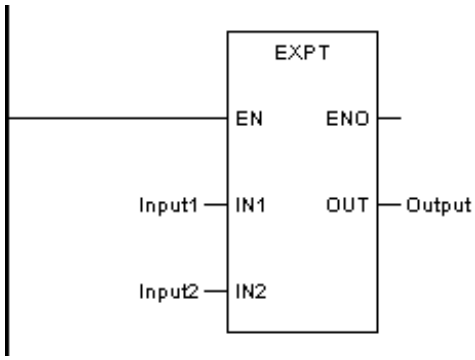
- Этот функциональный блок используется для вычисления экспоненты.

Уравнение

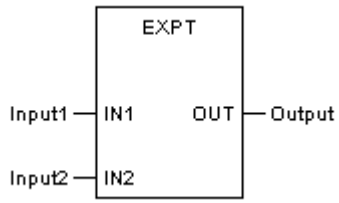
$OUT = IN1^{IN2}$

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
EXPT	Input2
ST	Output

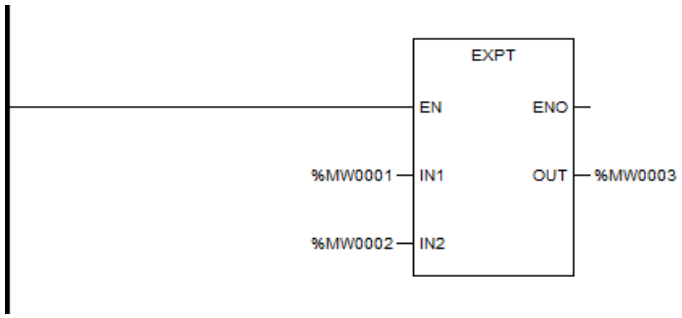
Форма на языке ST:

Output := EXPT (Input1, Input2);

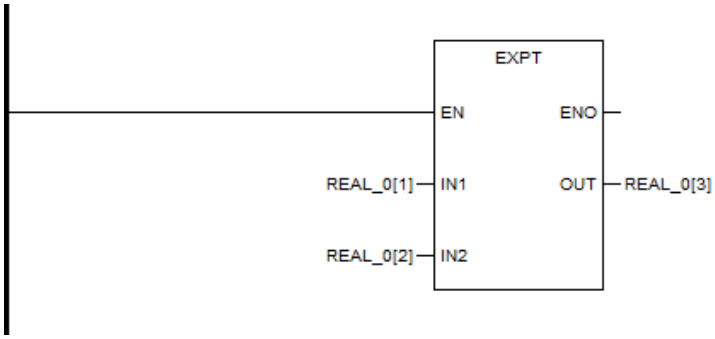
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	База	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Индекс	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Экспоненциальный выход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Экспонента целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Экспонента числа с плавающей запятой (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)



SIN

Описание функции

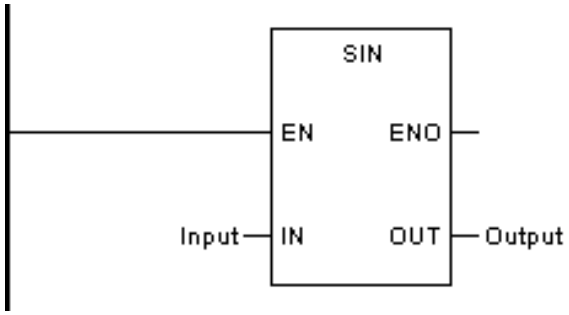
- Этот функциональный блок используется для вычисления значения синуса угла.

Уравнение

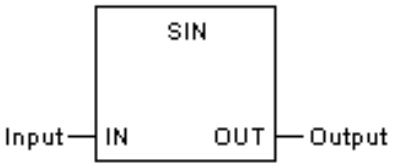
OUT = sin IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
SIN	
ST	Output

Форма на языке ST:

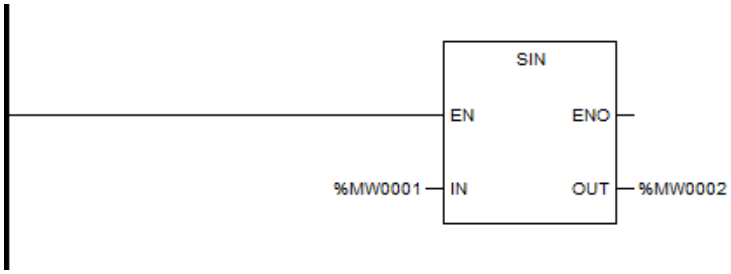
Output := SIN (Input);

Описание параметра

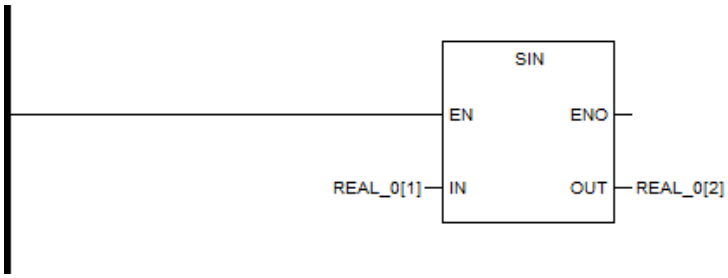
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Значение синуса	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1:Значение синуса целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного синуса может отображать только целые части, если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Значение синуса числа с плавающей запятой (REAL_0[1], REAL_0[2] — определяемая пользователем переменная с плавающей запятой)



COS

Описание функции

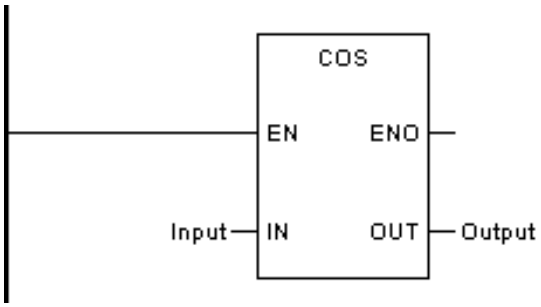
- Этот функциональный блок используется для вычисления значения косинуса угла.

Уравнение

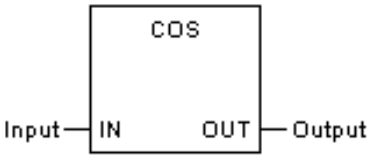
OUT = cos IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input

COS	
ST	Output

Форма на языке ST:

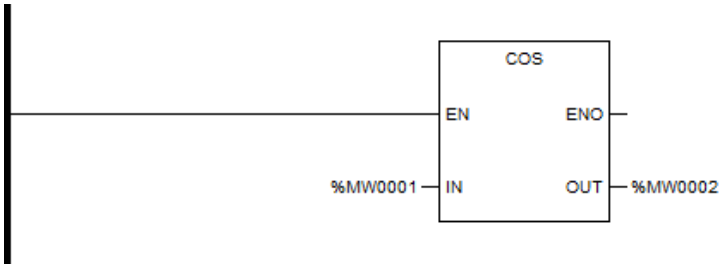
Output := COS (Input);

Описание параметра

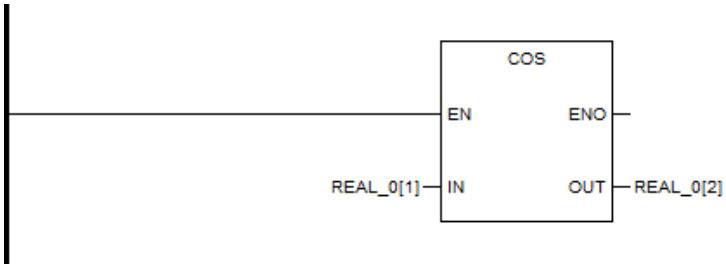
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Значение косинуса	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Значение косинусадлцелое число (тип регистра MW — WORD)

Примечание: вывод целочисленного косинуса может отображать только целые части, если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Значение косинуса для Плавающая (REAL_0[1], REAL_0[2] — определяемая пользователем переменная с плавающей запятой)



TAN

Описание функции

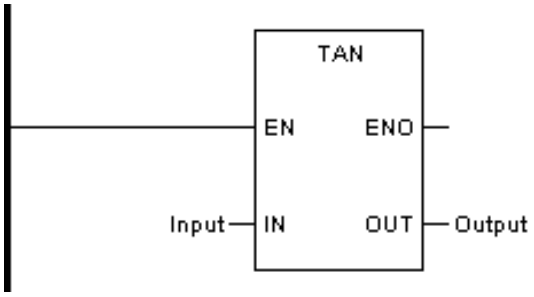
- Этот функциональный блок используется для расчета значения тангенса угла.

Уравнение

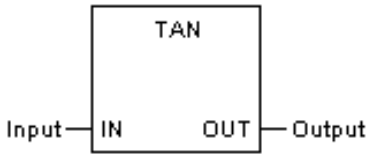
OUT = tan IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
TAN	
ST	Output

Форма на языке ST:

Output := TAN (Input);

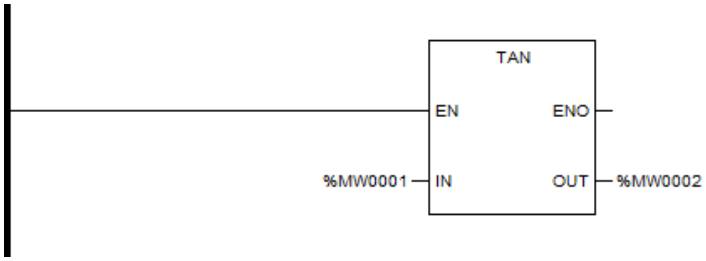
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра

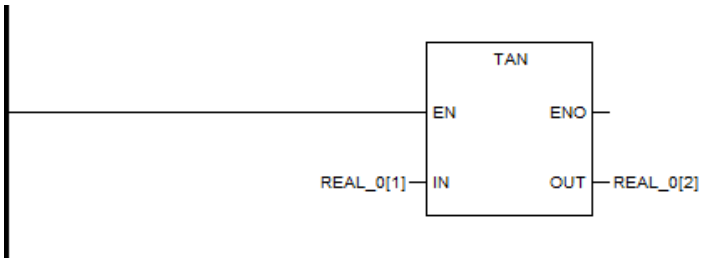
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Значение тангенса	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Значение тангенса целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного тангенса может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Значение тангенса числа с плавающей запятой (REAL_0[1], REAL_0[2] — определяемая пользователем переменная с плавающей запятой)



ASIN

Описание функции

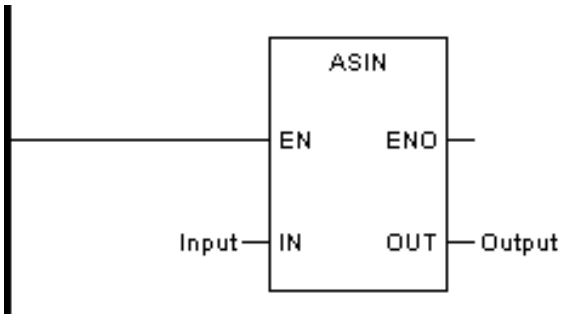
- Этот функциональный блок используется для вычисления значения арксинуса угла.

Уравнение

OUT = asin IN

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма на языке IL:

LD	Input
ASIN	
ST	Output

Форма на языке ST:

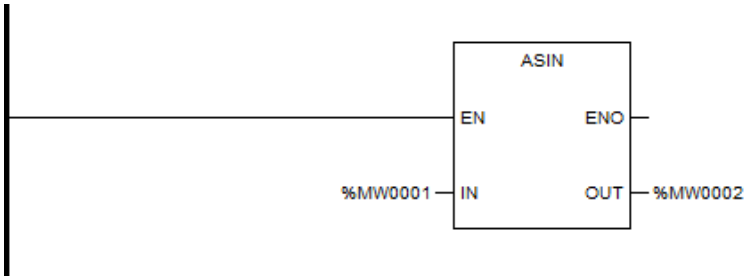
Output := ASIN (Input);

Описание параметра

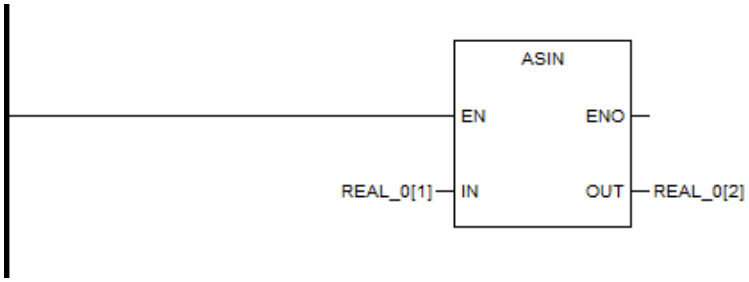
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Значение арксинуса	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Значение арксинуса целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного арксинуса может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Значение арксинуса числа с плавающей запятой (REAL_0[1], REAL_0[2] — определяемая пользователем переменная с плавающей запятой)



ACOS

Описание функции

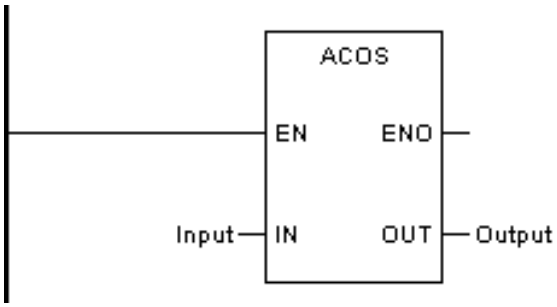
- Этот функциональный блок используется для вычисления значения арккосинуса угла.

Уравнение

OUT = acos IN

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма на языке IL:

LD	Input
ACOS	
ST	Output

Форма на языке ST:

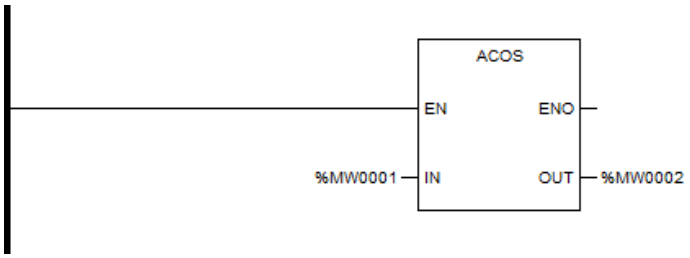
Output := ACOS (Input);

Описание параметра

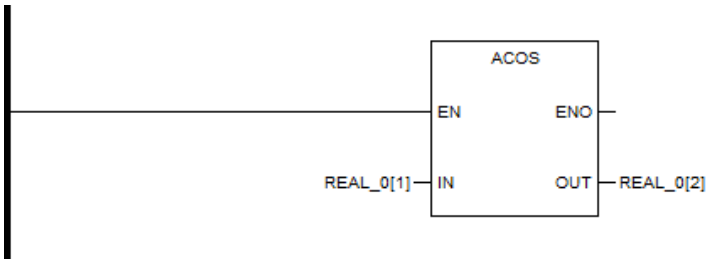
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Значение арккосинуса	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Значение арккосинуса целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного арккосинуса может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Значение арккосинуса числа с плавающей запятой (REAL_0[1], REAL_0[2] — определяемая пользователем переменная с плавающей запятой)



ATAN

Описание функции

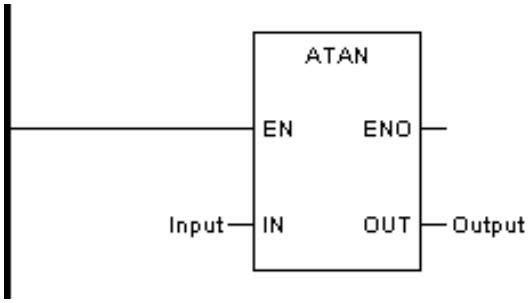
- Этот функциональный блок используется для вычисления значения арктангенса угла.

Уравнение

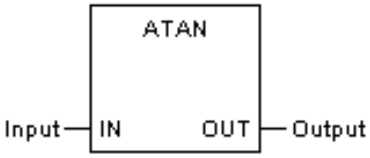
OUT = atan IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
ATAN	
ST	Output

Форма на языке ST:

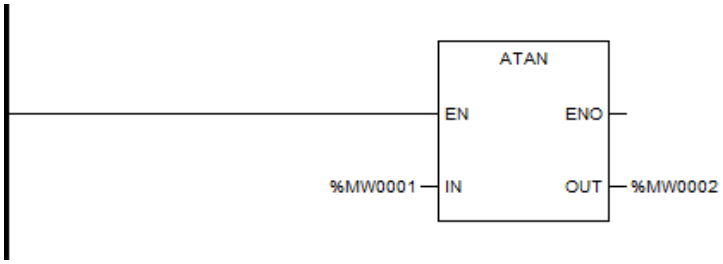
Output := ATAN (Input);

Описание параметра

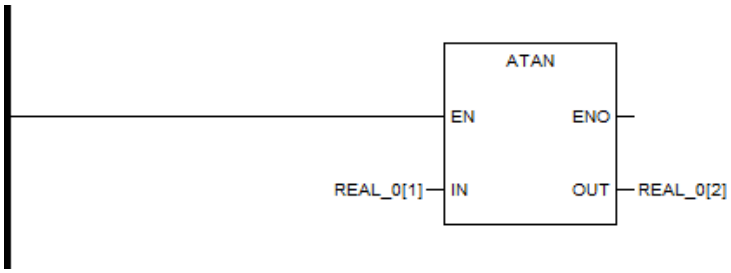
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Значение арктангенса	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Значение арктангенса целого числа (тип регистра MW — WORD)

Примечание: вывод целочисленного арктангенса может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Значение арктангенса числа с плавающей запятой (REAL_0[1], REAL_0[2] — определяемая пользователем переменная с плавающей запятой)


ATAN
EN
ENO
REAL_0[1]
IN
OUT
REAL_0[2]

Статистическая операция

Тда	Описание
MIN	Минимум: OUT = MIN {IN1, IN2,…, INn} (n≤8)
MAX	Максимум: OUT = MAX {IN1，IN2，…，INn} (n≤8）
AVE	Среднее: OUT = $\frac{IN1 + IN2 + \ldots INn}{n}$ (n≤8)
LIMIT	Ограниченное значение: OUT = IN, IF (IN ≥ MN) & (IN ≤ MX) OUT = MN, IF (IN < MN) OUT = MX, IF (IN > MX)
SEL	Выбор между 0/1: G = 0 -> OUT = IN0 G = 1 -> OUT = IN1
MUX	Множественный выбор: K = 0 -> OUT = IN0 K = 1 -> OUT = IN1 K = 2 -> OUT = IN2 …… K = n -> OUT = INn （n≤6）

MIN

Описание функции

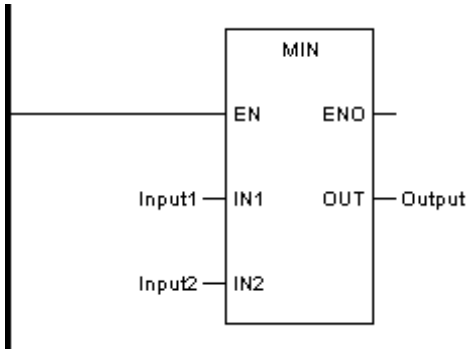
- Этот функциональный блок присваивает выходному значению минимальное входное значение.
- До 8 входов.

Уравнение

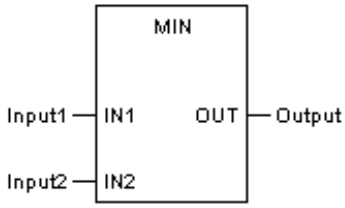
OUT = MIN {IN1，IN2，…，INn} （n≤8）

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
MIN	Input2
ST	Output

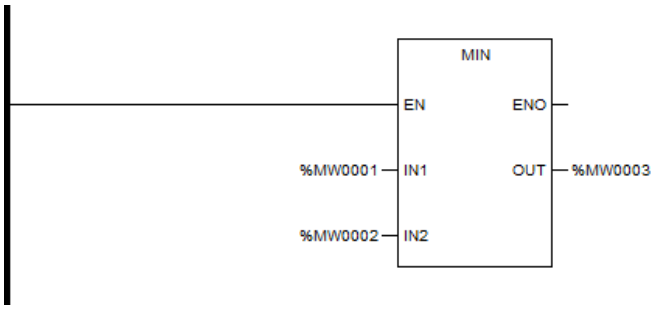
Форма на языке ST:

Output := MIN (Input1, Input2);

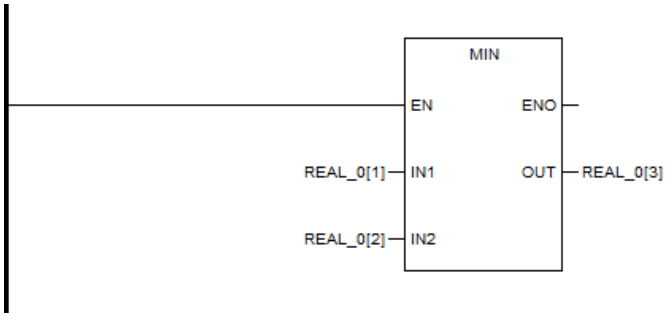
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Минимальный выход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

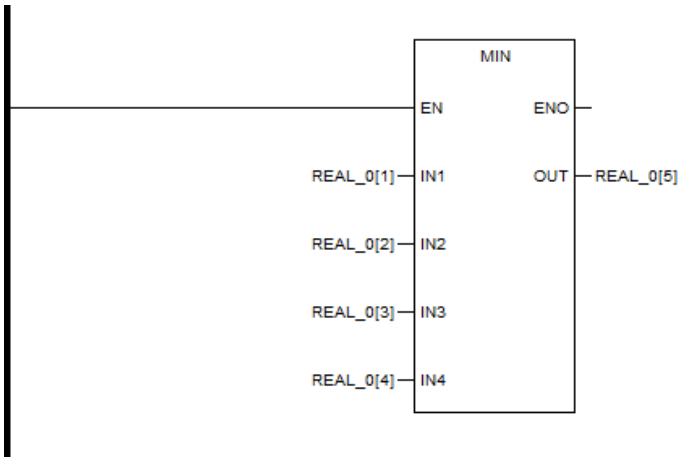
ПРИМЕР 1: Минимальное значение целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Минимальное значение числа с плавающей запятой (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: Минимальное значение среди нескольких чисел (щелкните правой кнопкой мыши по функциональному блоку, выберите свойство и измените количество входов)



MAX

Описание функции

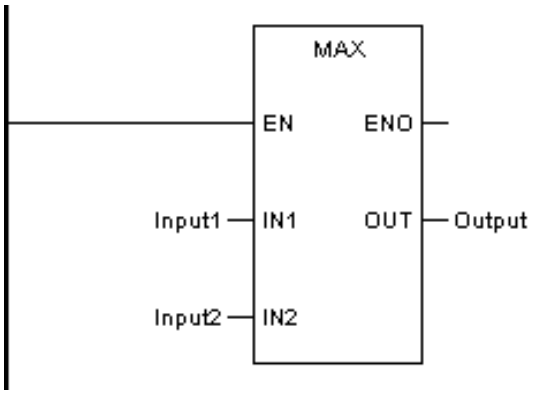
- Этот функциональный блок присваивает максимальное входное значение выходу.
- До 8 входов.

Уравнение

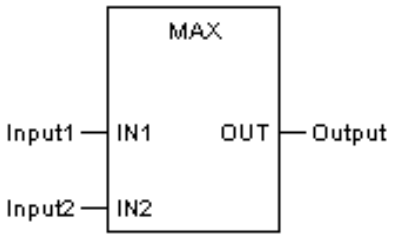
OUT = MAX {IN1，IN2，…，INn} （n≤8）

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
MAX	Input2
ST	Output

Форма на языке ST:

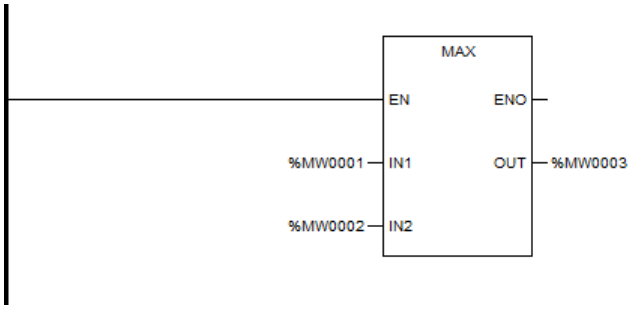
Output := MAX (Input1, Input2);

Описание параметра

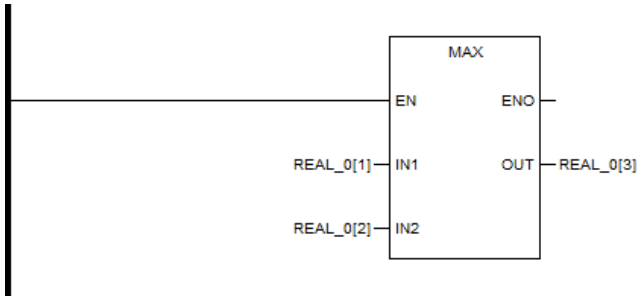
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Максимальная производительность	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

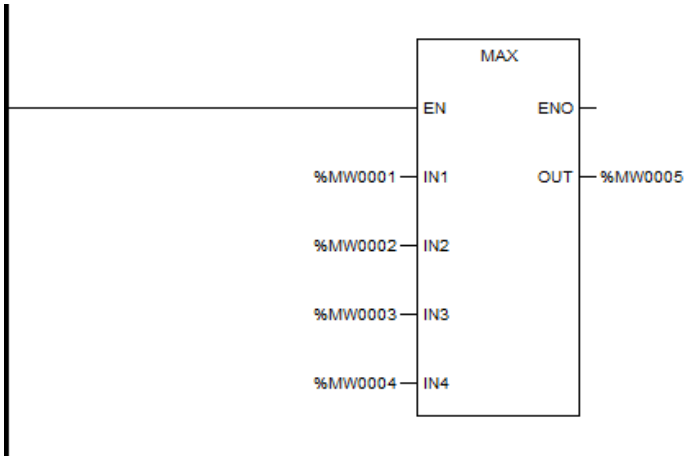
ПРИМЕР 1: Максимальное значение целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Максимальное значение числа с плавающей запятой (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: Максимальное значение среди нескольких чисел (щелкните правой кнопкой мыши по функциональному блоку, выберите свойство и измените количество входов)



AVE

Описание функции

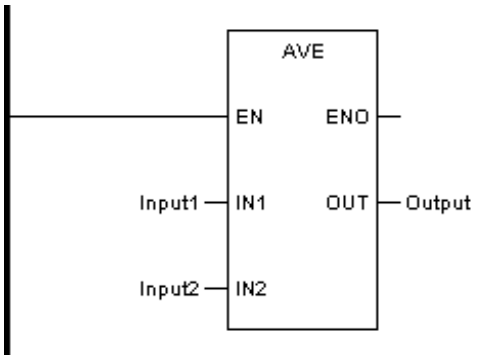
- Этот функциональный блок присваивает выходное среднее значение входов
- До 8 входов.

Уравнение

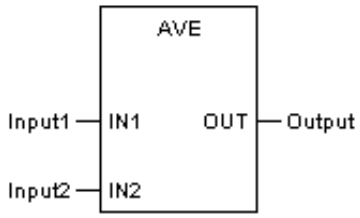
$$OUT = \frac{IN1 + IN2 + \ldots INn}{n} \quad (n \leq 8)$$

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
AVE	Input2
ST	Output

Форма на языке ST:

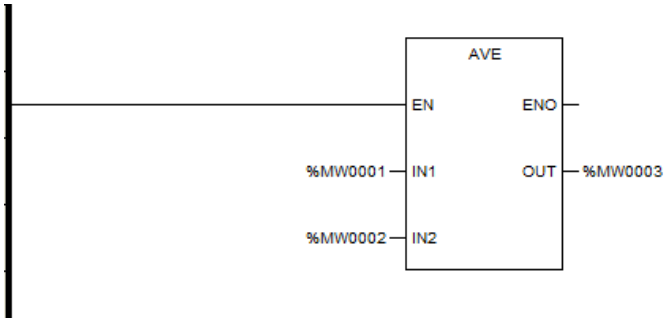
Output := AVE (Input1, Input2);

Описание параметра

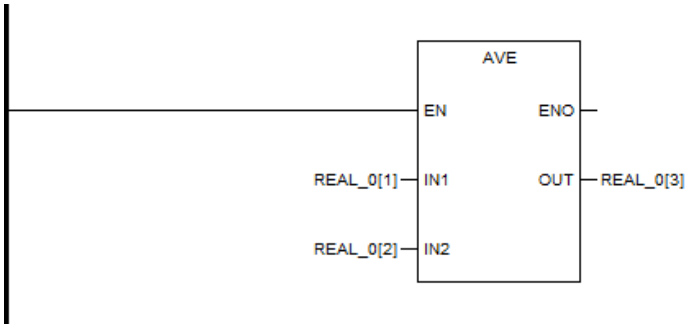
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Средний выход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Среднее значение целого числа (тип регистра MW — WORD)

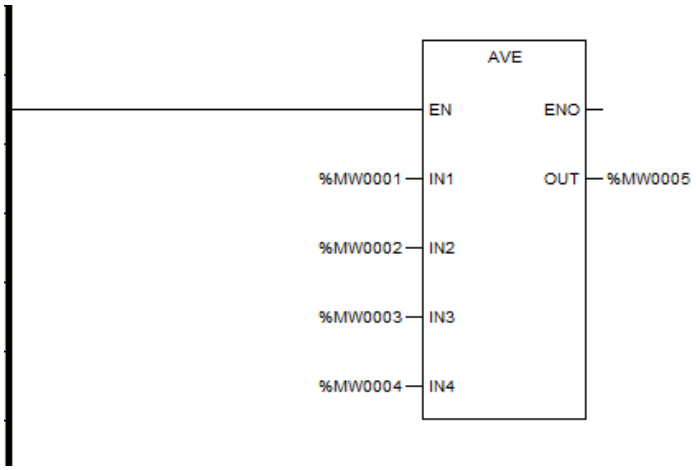
Примечание: вывод целочисленного квадратного корня может отображать только целые части; если вы хотите отобразить десятичные дроби, вы можете вывести их как числа с плавающей запятой.



ПРИМЕР 2: Среднее значение числа с плавающей запятой (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: Среднее значение среди нескольких чисел (щелкните правой кнопкой мыши по функциональному блоку, выберите свойство и измените количество входов)



LIMIT

Описание функции

- Если входное значение не меньше нижнего предела и не больше верхнего предела, функциональный блок назначит входное значение выходу.
- Если входное значение меньше нижнего предела, функциональный блок назначит нижний предел выходу.
- Если входное значение больше верхнего предела, функциональный блок назначит верхний предел выходу.

Уравнение

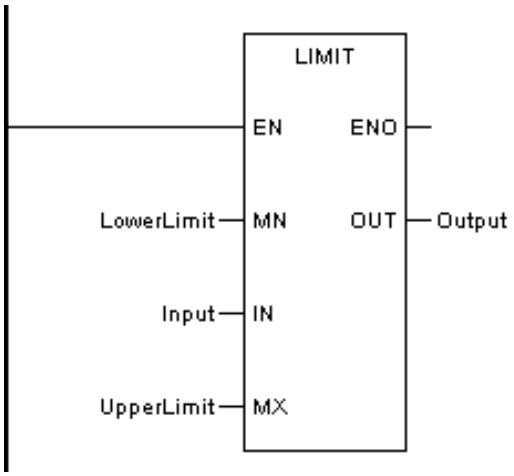
OUT = IN, IF (IN ≥ MN) & (IN ≤ MX)

OUT = MN, IF (IN < MN)

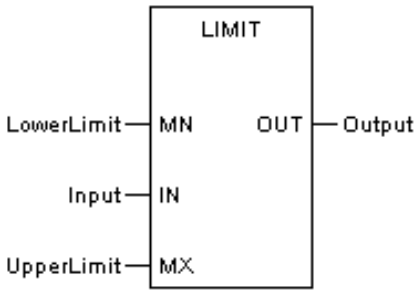
OUT = MX, IF (IN > MX)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	LowerLimit
LIMIT	Input, UpperLimit
ST	Output

Форма на языке ST:

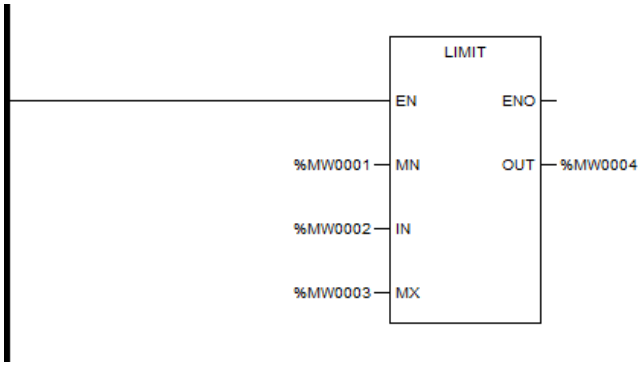
Output := LIMIT (LowerLimit, Input, UpperLimit);

Описание параметра

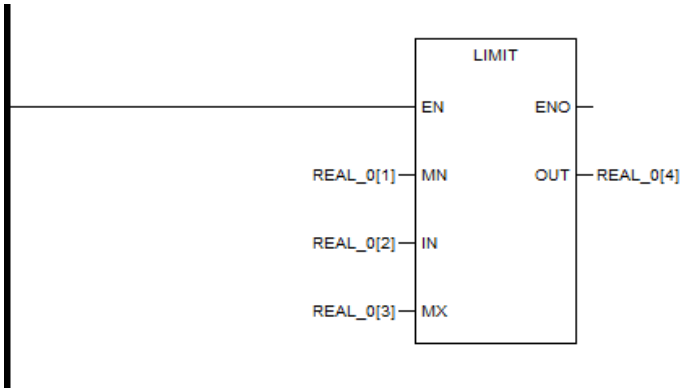
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
MN	LowerLimit	Нижний пре-дел	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

MX	UpperLimit	Верхний предел	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Средний выход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Предельное значение целого числа (тип регистра MW — WORD)



ПРИМЕР 2: Предельное значение числа с плавающей запятой (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)



Выбор между 0/1 - SEL

Описание функции

- Функциональный блок используется для выбора между двумя входными значениями. В зависимости от статуса входа G функциональный блок назначает выходу IN0 или IN1.

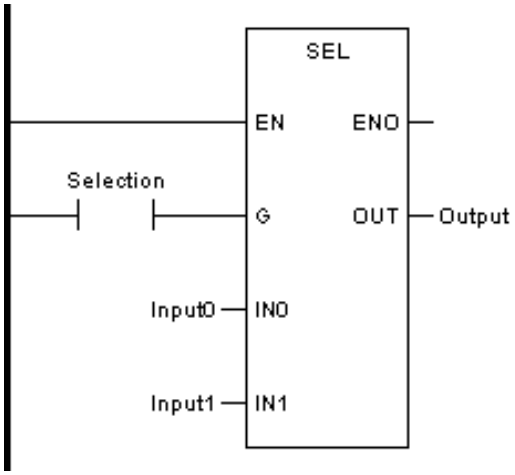
Уравнение

G = 0 -> OUT = IN0

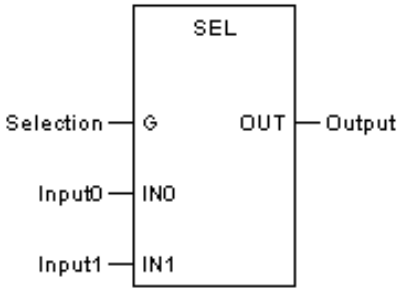
G = 1 -> OUT = IN1

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Selection
SEL	Input0, Input1
ST	Output

Форма на языке ST:

Output := SEL (Selection, Input0, Input1);

Описание параметра

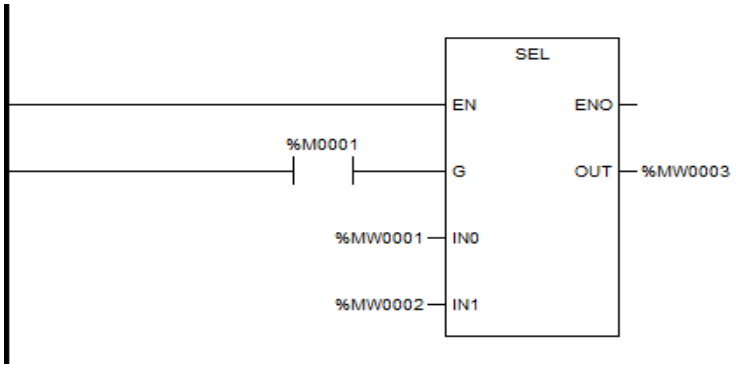
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**

G	Selection	Выбор вход-ных данных	BOOL	CONSTANT, I, Q, M, N, S, VAR
IN0	Input0	Вход 0	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Выбор между 0/1 для целого числа (тип регистра MW — WORD)

ЕСЛИ %M0001=0, %MW0003=%MW0001;

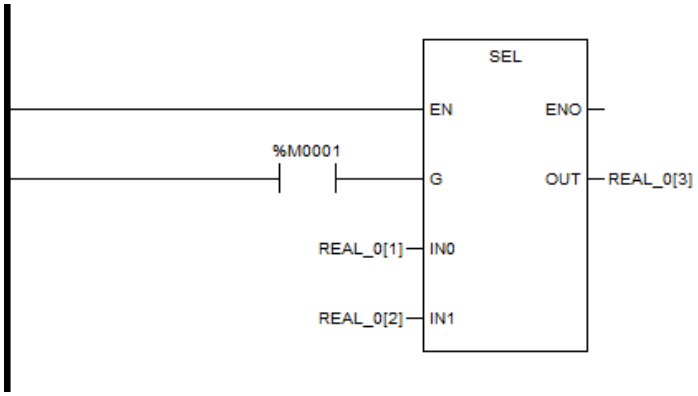
ЕСЛИ %M0001=1, %MW0003=%MW0001.



ПРИМЕР 2: Выбор между 0/1дляПлавающие (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)

ЕСЛИ %M0001=0, REAL_0[3] =REAL_0[1];

ЕСЛИ %M0001=1, REAL_0[3] =REAL_0[2];



MUX

Описание функции

- Функциональный блок назначает вход выходу в соответствии с входным значением K.
- Входы до 7.

Уравнение

K = 0 -> OUT = IN0

K = 1 -> OUT = IN1

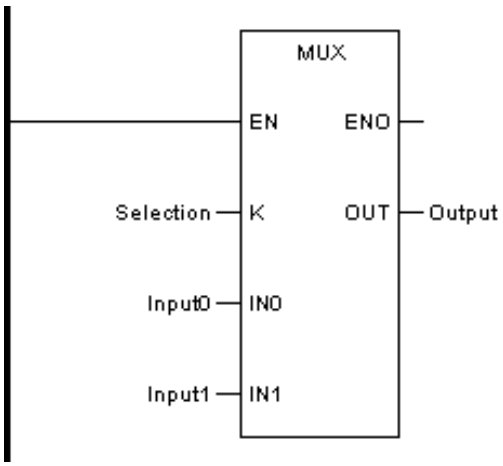
K = 2 -> OUT = IN2

……

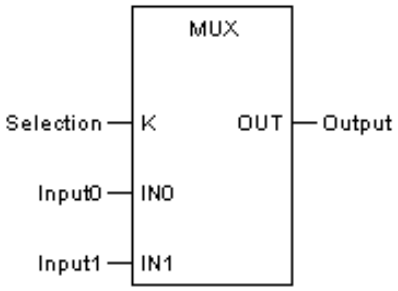
K = n -> OUT = INn （n≤6）

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Selection
MUX	Input0, Input1
ST	Output

Форма на языке ST:

Output := MUX (Selection, Input0, Input1);

Описание параметра

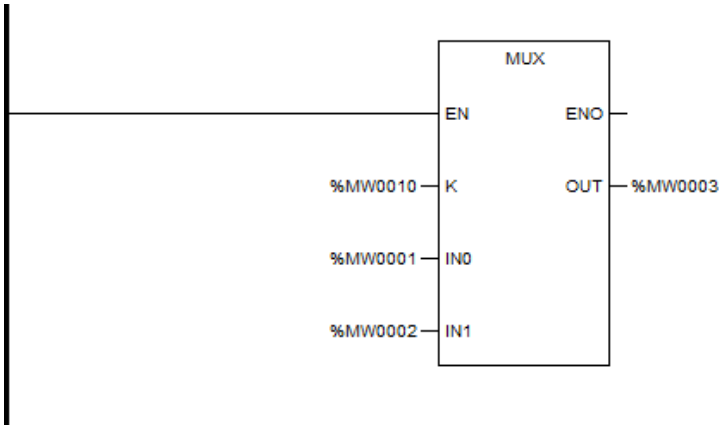
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
K	Selection	Выбор входных данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, I, Q, M, N, S, VAR
IN0	Input0	Вход 0	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: Множественный выбор целого числа (тип регистра MW — WORD)

Примечание: 0≤%MW0010≤6, в этом примере 0≤%MW0010≤1

%M0001=0, %MW0003=%MW0001;

%M0001=1, %MW0003=%MW0001.



ПРИМЕР 2: Множественный выбор числа с плавающей запятой (REAL_0[1], REAL_0[2] и REAL_0[3] — определяемые пользователем переменные с плавающей запятой)

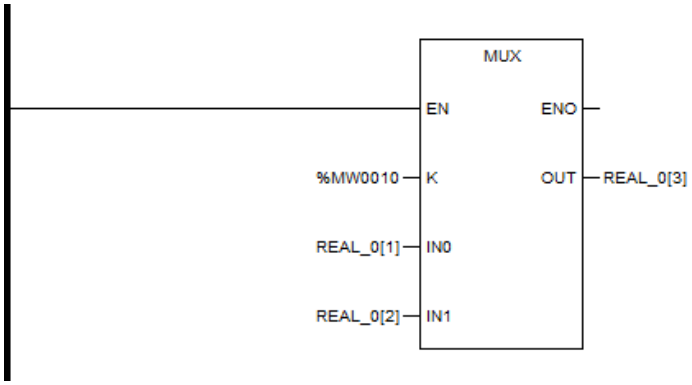
Примечание: 0≤%MW0010≤6, в этом примере 0≤%MW0010≤1

%M0001=0, %MW0003=%MW0001;

%M0001=1, %MW0003=%MW0001.

IF %M0001=0, REAL_0[3] =REAL_0[1];

IF %M0001=1, REAL_0[3] =REAL_0[2];



ПРИМЕР 3: Выбор среди нескольких чисел (щелкните правой кнопкой мыши по функциональному блоку, выберите свойство и измените количество входов)

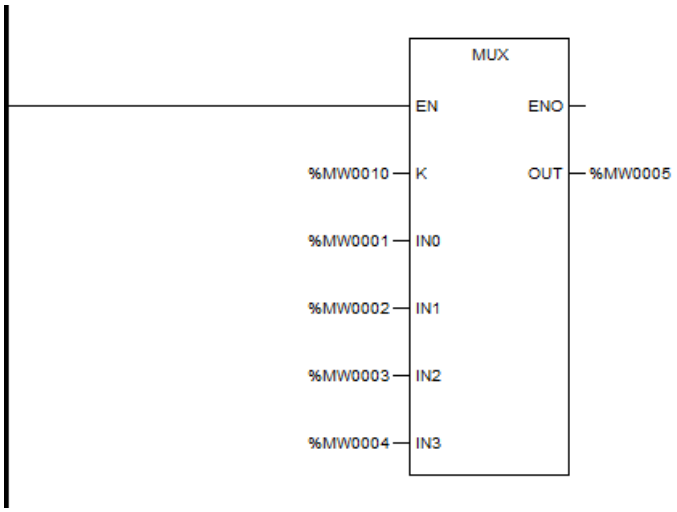
Примечание: 0≤%MW0010≤6, в этом примере 0≤%MW0010≤3

%MW0010=0, %MW005=%MW0001;

%MW0010=1, %MW005=%MW0002;

%MW0010=2, %MW005=%MW0003;

%MW0010=3, %MW005=%MW0004;


MUX
EN
ENO
%MW0010
K
OUT
%MW0005
%MW0001
IN0
%MW0002
IN1
%MW0003
IN2
%MW0004
IN3

Логическая операция

Функциональный блок логических операций выполняет битовую операцию над значением входа, т. е. независимо от типа данных или регистров, функциональные блоки логических операций всегда рассматривают их как битовую строку, состоящую из каждого бита данных. Каждый бит данных независим друг от друга. RESET и BCLR напрямую работают с входными данными. Для этих двух функциональных блоков данные конца входа будут изменены. Для других функциональных блоков логических операций результаты будут отправлены в назначенные переменной, а входное значение не будет изменено. Входные данные могут быть 8-битными, 16-битными или 32-битными, начиная с 0 справа налево.

Тда	Описание
AND	Логическое И: Два входных данных сравниваются побитно, начиная с самого младшего бита. Если два бита данных равны 1, соответствующий бит выходных данных равен 1. Если один или два бита равны 0, соответствующий бит выходных данных равен 0.
OR	Логическое ИЛИ: Два входных данных сравниваются бит за битом, начиная с самого младшего бита. Если один из битов данных равен 1, соответствующий бит выходных данных равен 1. Если два бита оба равны 0, соответствующий бит выходных данных равен 0.
NOT	Логическое НЕ: Входные данные меняются местами по битам.
XOR	Исключающее ИЛИ: Два входных данных сравниваются побитно, начиная с самого младшего бита. Если два бита данных одинаковы, соответствующий бит выходных данных равен 1. Если два бита данных различны, соответствующий бит выходных данных равен 0.
SHL	Сдвиг влево: все биты целочисленных данных сдвигаются влево на указанное количество бит, а биты данных, оставшиеся пустыми, заполняются нулями.
SHR	Сдвиг вправо: все биты целочисленных данных сдвигаются вправо на указанное количество бит, а оставшиеся пустыми биты данных заполняются нулями.
ROL	Циклический сдвиг влево: Все биты входных данных сдвигаются влево на указанное количество бит. Самый старший бит, который сдвигается, будет помещен на самый младший бит.
ROR	Циклический сдвиг вправо: Все биты входных данных сдвигаются вправо на указанное количество бит. Самый низкий бит, который выдвигается, будет помещен на самый высокий бит.
BSET	Установить бит: Назначенный бит входных данных равен 1.

BCLR	Очистить бит: Назначенный бит входных данных равен 0.
BTST	Проверить бит: Проверяет, равен ли определенный бит входных данных 0 или 1. Когда проверяемый бит равен 1, выходной конец данных пропустит ток или установит 1 в назначенных точках; когда проверяемый бит равен 0, выходной конец данных не пропустит ток или установит 0 в назначенных точках.
R_TRIG	Обнаружение восходящего фронта: Когда назначенная точка изменится с 0 на 1, выход Q превратится в 1 и сохранится в течение одного цикла. В следующем периоде сканирования Q вернется к 0 до следующего изменения с 0 на 1.
F_TRIG	Обнаружение нисходящего фронта: Когда назначенная точка изменится с 1 на 0, выход Q превратится в 1 и сохранится в течение одного цикла. В следующем периоде сканирования Q вернется к 0 до следующего изменения с 1 на 0.
SET	Установить: Установить 1 в назначенный бит. Эта функция заставит переменную поддерживать 1 до тех пор, пока не будет блока функции сброса. В противном случае назначенная точка будет поддерживать 1.
RESET	Сброс: Установите 0 в назначенный бит. Эта функция заставит переменную поддерживать 0 до тех пор, пока не будет установлен функциональный блок. В противном случае назначенная точка будет поддерживать 0.
SR	Нестабильное состояние (Установить и держать)
RS	Нестабильное состояние (Сбросить)

AND

Описание функции

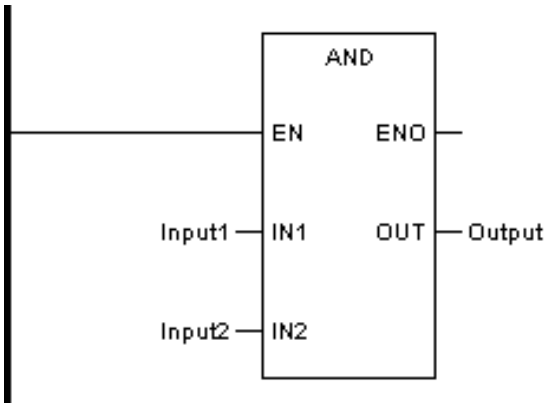
- Этот функциональный блок выполняет операцию «И»над битовой последовательностью входа и присваивает результат выходу.
- Входы до 8.

Уравнение

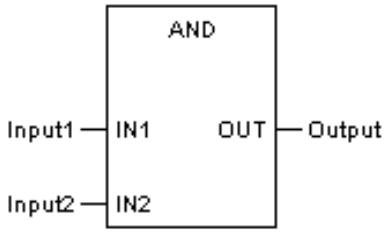
OUT = IN1 AND IN2 AND … AND INn

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
AND	Input2
ST	Output

Форма на языке ST:

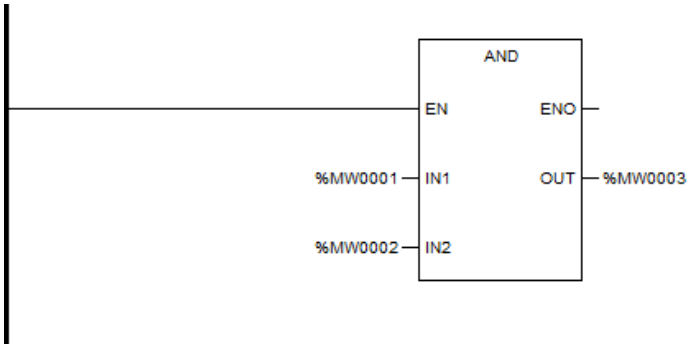
Output := AND (Input1, Input2);

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BOOL[*1], BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, VAR
IN2	Input2	Вход 2	BOOL[*1], BYTE, WORD, DWORD, SINT, INT, DINT, REAL USINT, UINT, UDINT	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, VAR
OUT	Output	Выход	BOOL[*1], BYTE, WORD, DWORD, SINT, INT, DINT, REAL USINT, UINT, UDINT	Q, M, N, MW, NW, VAR
[*1] AND для BOOL не поддерживается LD				

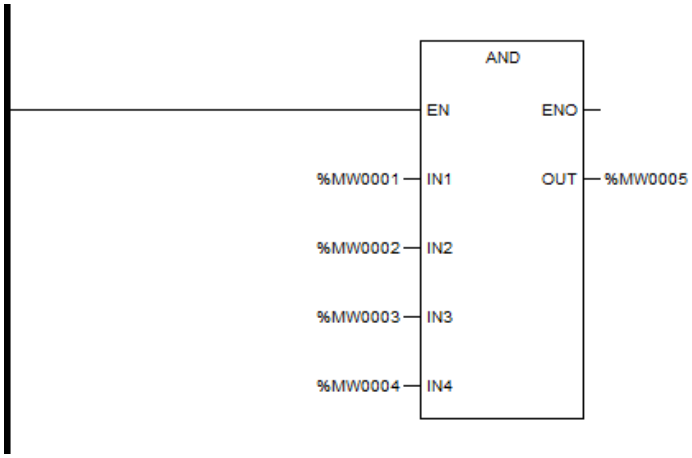
ПРИМЕР 1: AND целых чисел (тип регистра — WORD)

Примечание: в двоичном виде.



ПРИМЕР2: AND нескольких целых чисел

Примечание: в двоичном виде.



OR

Описание функции

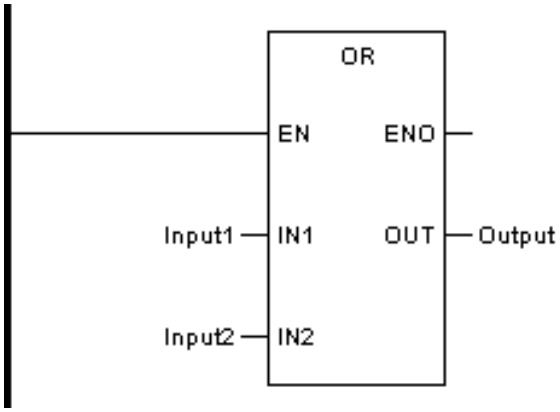
- Этот функциональный блок выполняет операцию ИЛИ над битовой последовательностью входа и присваивает результат выходу.
- Входы до 8.

Уравнение

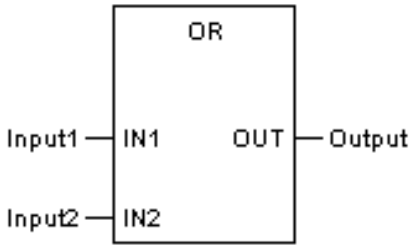
OUT = IN1 OR IN2 OR … OR INn

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
OR	Input2
ST	Output

Форма на языке ST:

Output := OR (Input1, Input2);

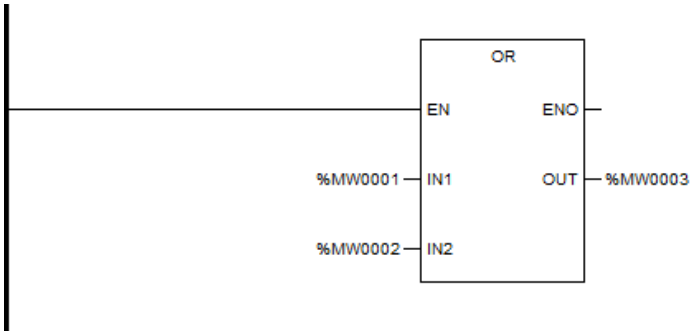
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BOOL[*1], BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, VAR
IN2	Input2	Вход 2	BOOL[*1], BYTE, WORD, DWORD, SINT, INT,	CONSTANT, I, Q, IW, QW, M, MW,

			DINT, REAL USINT, UINT, UDINT	N, NW, S, SW, VAR
OUT	Output	Выход	BOOL*1, BYTE, WORD, DWORD, SINT, INT, DINT, REAL USINT, UINT, UDINT	Q, M, N, MW, NW, VAR
*1 OR для BOOL не поддерживается LD				

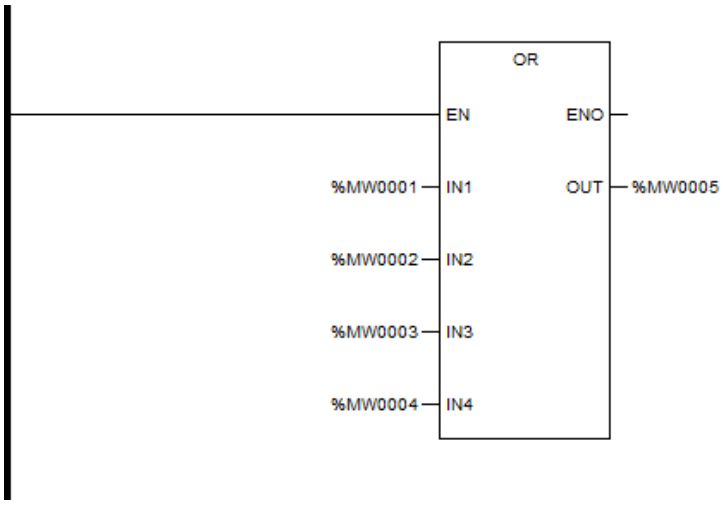
ПРИМЕР 1: OR целых чисел (тип регистра — WORD)

Примечание: в двоичном виде.



ПРИМЕР2: OR нескольких целых чисел

Примечание: в двоичном виде.



NOT

Описание функции

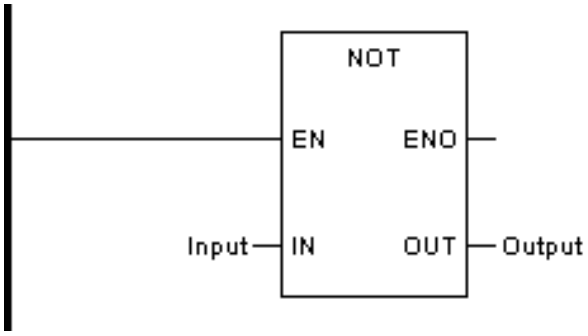
- Этот функциональный блок выполняет операцию НЕ над битовой последовательностью входа и присваивает результат выходу.

Уравнение

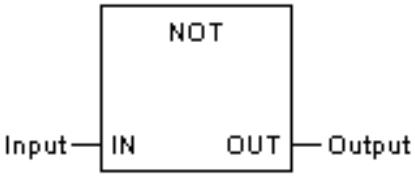
OUT = NOT IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
NOT	
ST	Output

Форма на языке ST:

Output := NOT (Input);

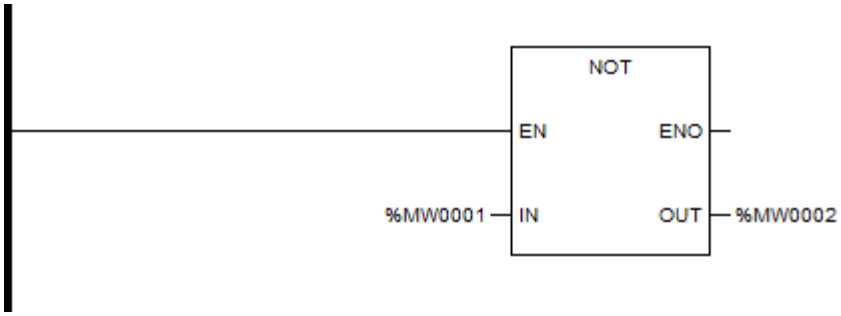
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: NOT целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.



XOR

Описание функции

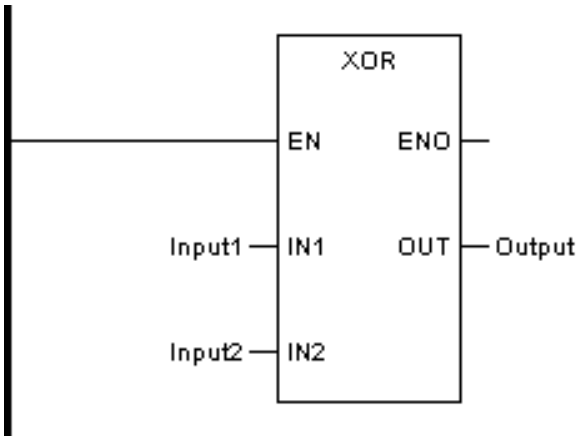
- Этот функциональный блок выполняет операцию XOR над битовой последовательностью входа и присваивает результат выходу.
- Входы до 8.

Уравнение

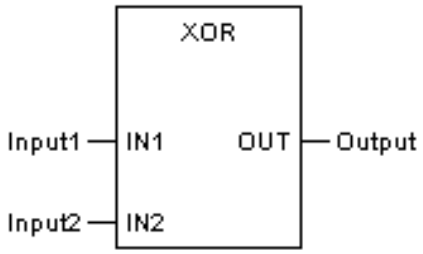
OUT = IN1 XOR IN2 XOR … XOR INn

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input1
XOR	
ST	Output

Форма на языке ST:

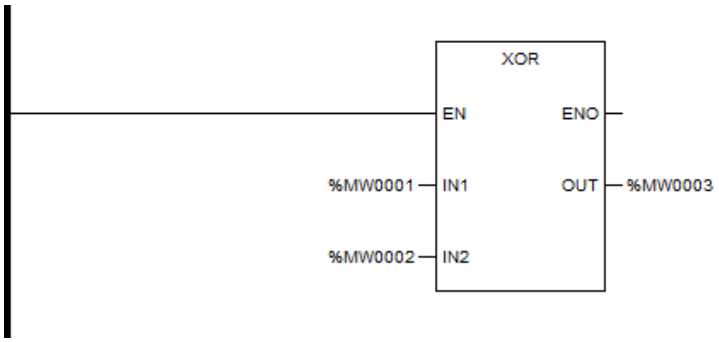
Output := XOR (Input1, Input2);

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

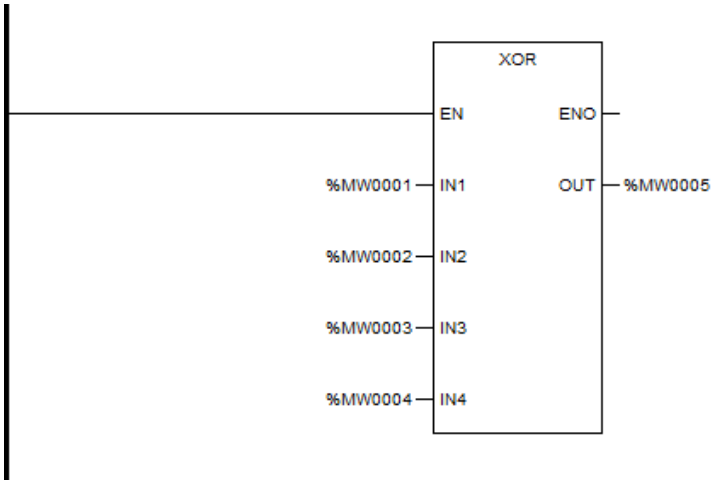
ПРИМЕР 1: XOR целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.



ПРИМЕР2: XOR нескольких целых чисел

Примечание: в двоичном виде.



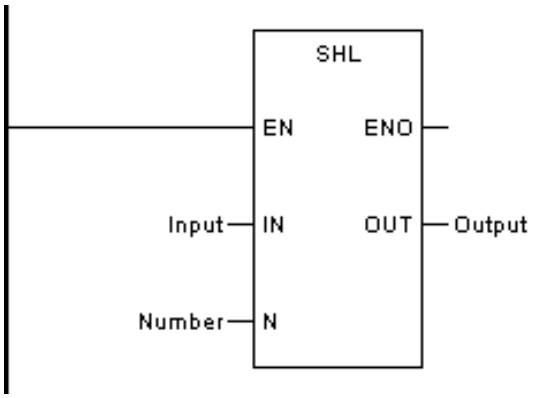
SHL

Описание функции

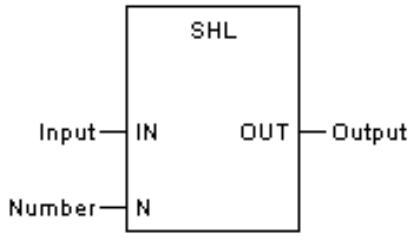
- Этот функциональный блок перемещает все биты целочисленных данных влево на указанное количество бит, а оставшиеся пустыми биты данных заполняются нулями. Результат будет присвоен выходу.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
SHL	Number
ST	Output

Форма на языке ST:

Output := SHL (Input, Number);

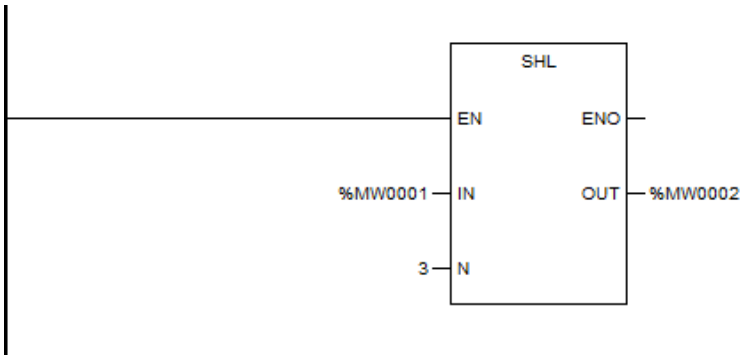
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР: SHL целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=5（00000101）После перемещения влево на N=3 бита, %MW0002=40（00101000)



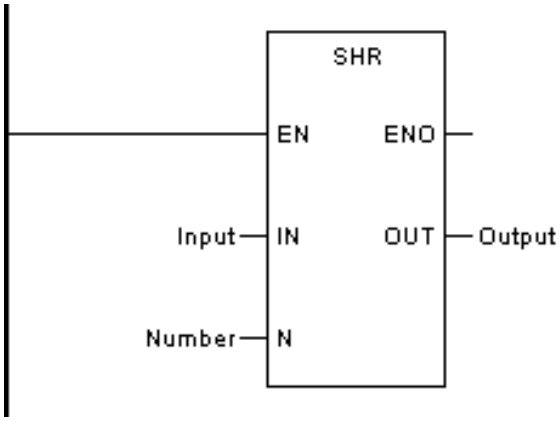
SHR

Описание функции

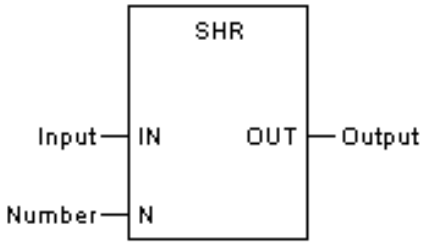
- Этот функциональный блок перемещает все биты целочисленных данных вправо на указанное количество бит, а оставшиеся пустыми биты данных заполняются нулями. Результат будет присвоен выходу.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
SHR	Number
ST	Output

Форма на языке ST:

Output := SHR (Input, Number);

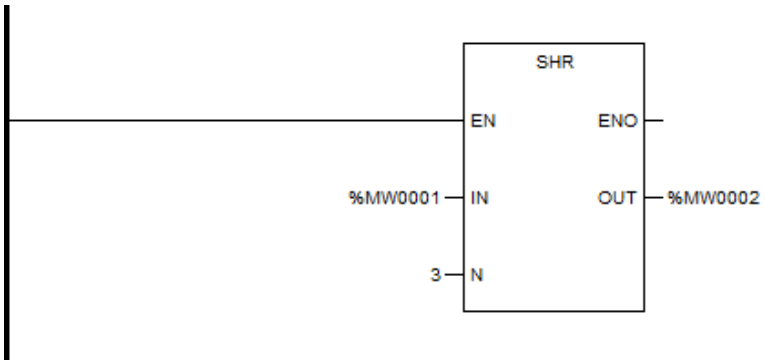
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР: SHR целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=40（00101000）После сдвига вправо на N=3 бита, %MW0002=5（00000101）



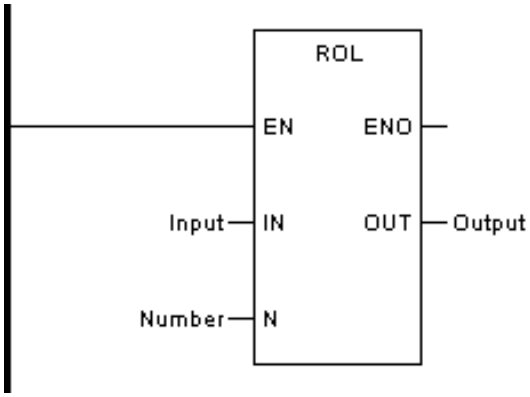
ROL

Описание функции

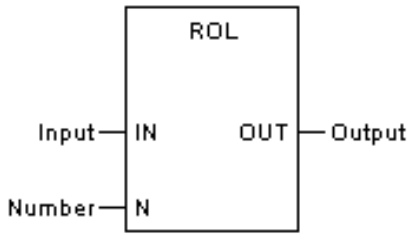
- Этот функциональный блок перемещает все биты целочисленных данных влево на n бит в циклическом режиме. Результат будет назначен на выход.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
ROL	Number
ST	Output

Форма на языке ST:

Output := ROL (Input, Number);

Описание параметра

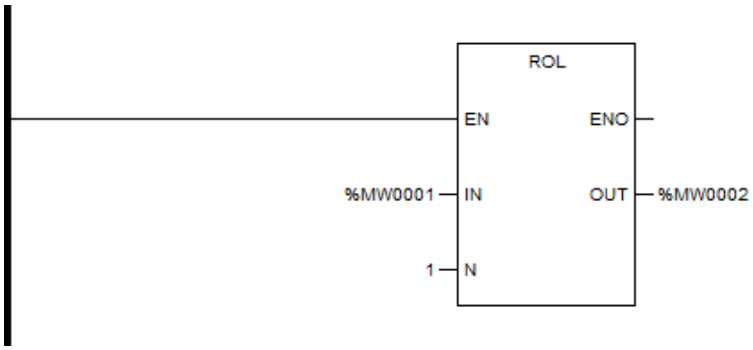
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР: ROL целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=22181（0101011010100101）После перемещения влево на N=1 бит,

%MW0002=44362（1010110101001010）, т.е. бит 15 превращается в бит 0, остальные перемещаются влево на 1 бит.



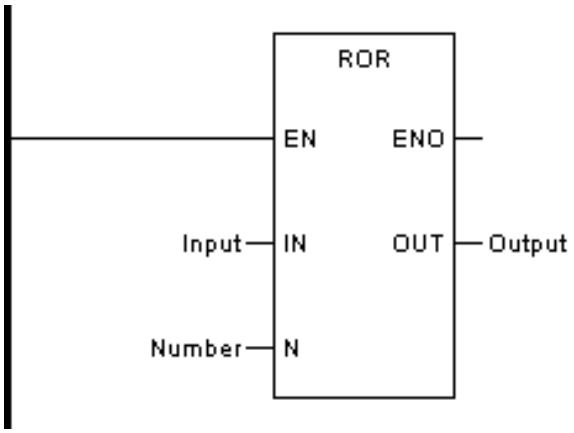
ROR

Описание функции

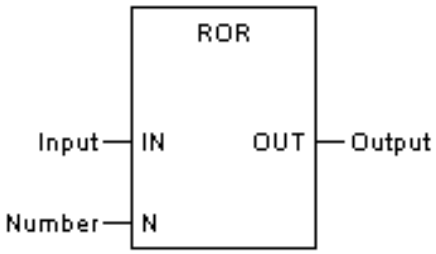
- Этот функциональный блок перемещает все биты целочисленных данных вправо на n бит в циклическом режиме. Результат будет назначен на OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

LD	Input
ROR	Number
ST	Output

Форма на языке ST:

Output := ROR (Input, Number);

Описание параметра

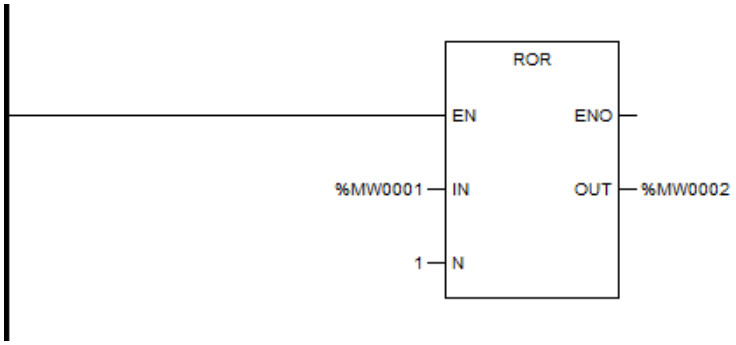
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР 1: ROR целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=22181（0101011010100101）После перемещения вправо на N=1 бит,

%MW0002=43858（1010101101010010）, т.е. бит 0 превращается в бит 15, остальные перемещаются вправо на 1 бит.



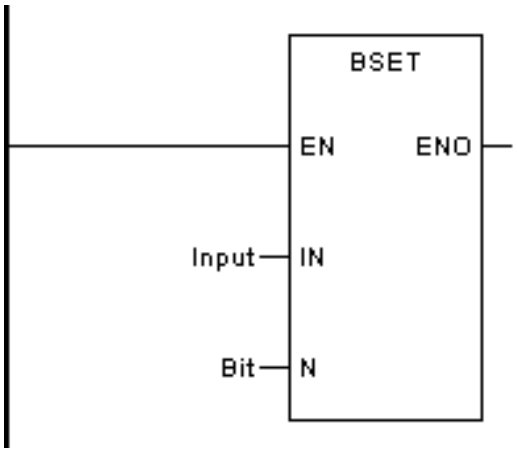
BSET

Описание функции

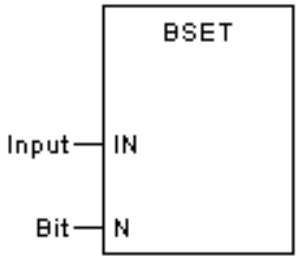
- Этот функциональный блок устанавливает бит n последовательности битов входа равным 1 и присваивает результат выходу.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL BSET (IN:=Вход, N:=Bit)

Форма на языке ST:

BSET (IN:=Вход, N:=Bit);

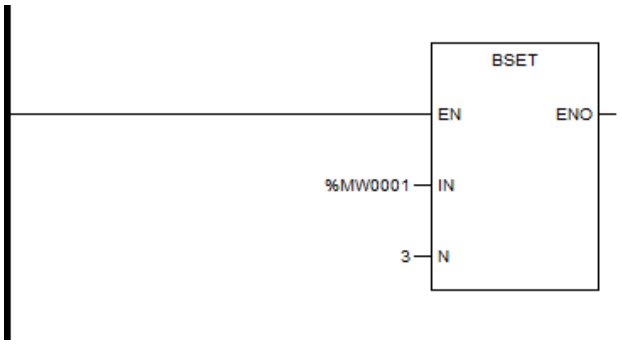
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: BSET целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=8 (00001000), после выполнения функционального блока бит 3 последовательности бит будет установлен как 1, а затем %MW0001=12 (00001100)



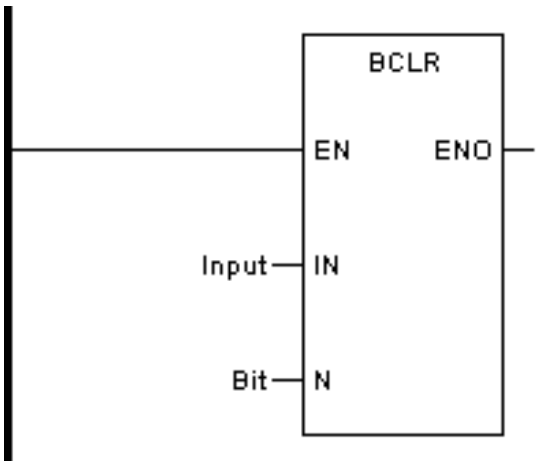
BCLR

Описание функции

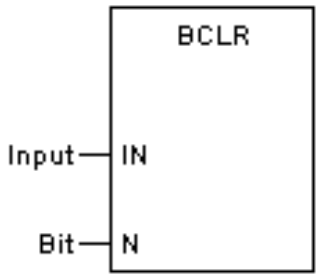
- Этот функциональный блок устанавливает бит n битовой последовательности входа как 0 и присваивает результат выходу.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL BCLR (IN:=Вход, N:=Bit)

Форма на языке ST:

BCLR (IN :=Вход, N :=Bit);

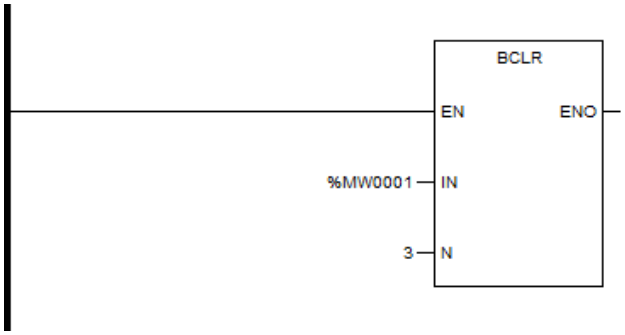
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Bit	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: BCLR целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=12 (00001100), после выполнения функционального блока бит 3 последовательности бит будет установлен как 0, а затем %MW0001=8 (00001000)



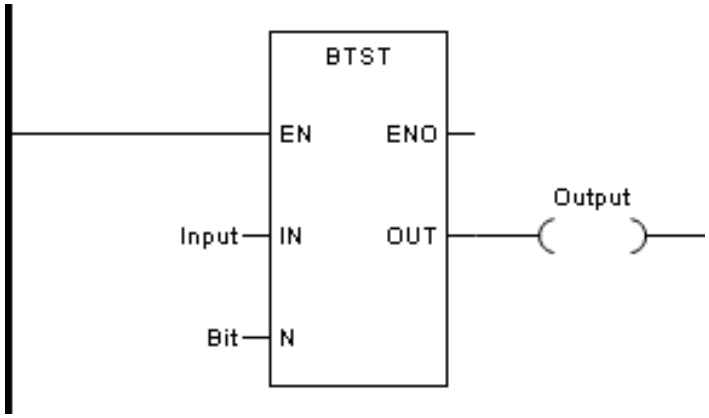
BTST

Описание функции

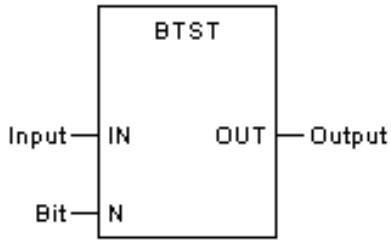
- Этот функциональный блок обнаруживает бит n битовой последовательности входа. Когда тестируемый бит равен 1, выход пропустит ток или установит назначенную точку на 1; Когда тестируемый бит равен 0, выход пропустит ток или установит назначенную точку на 0.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL BTST (IN:=Вход, N:=Бит, OUT=>Output)

Форма на языке ST:

BTST (IN:=Вход, N:=Бит, OUT=>Output);

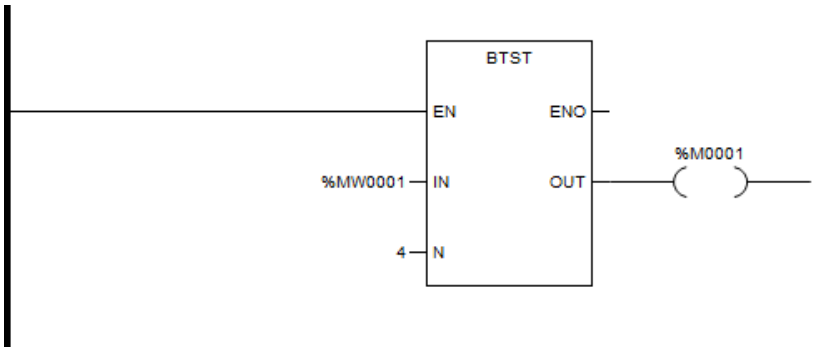
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Bit	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	OUT	BOOL	Q, M, N, VAR

ПРИМЕР: BTST целых чисел (тип регистра MW — WORD)

Примечание: в двоичном виде.

%MW0001=12 (00001100), после обнаружения бита 4 OUT %M0001 равен 1.



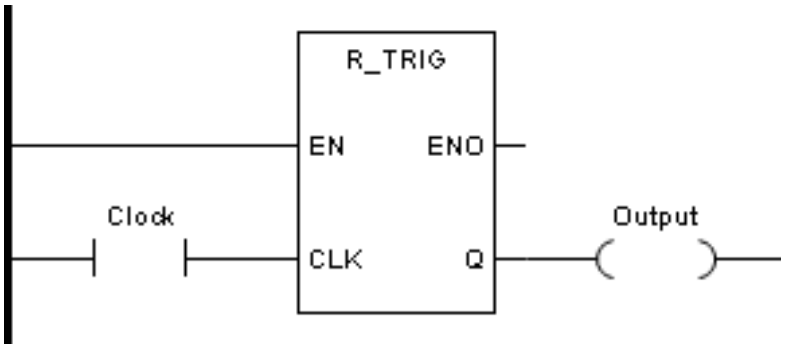
R_TRIG

Описание функции

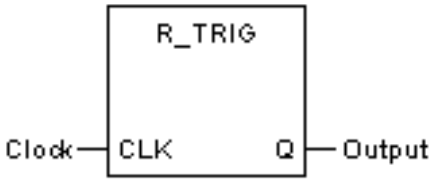
- Этот функциональный блок обнаруживает восходящий фронт.
- Если функциональный блок обнаруживает изменение CLK с 0 на 1, выход Q изменится на 1, сохранится в течение одного цикла, а затем вернется к 0.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL R_TRIG (CLK:=Clock, Q=>Output)

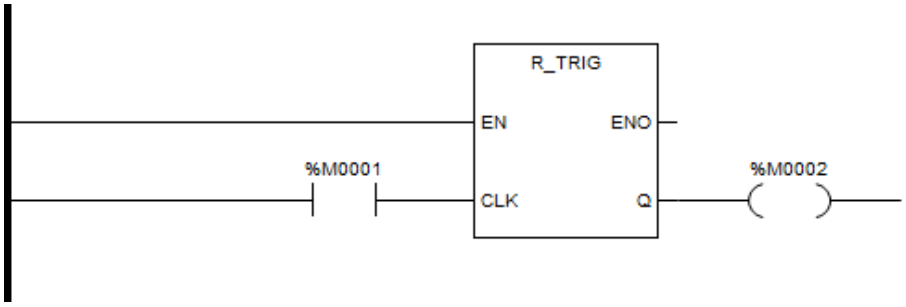
Форма на языке ST:

R_TRIG (CLK:=Clock, Q=>Output);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
CLK	Clock	Вход	BOOL	CONSTANT, I, Q, M, N, S, VAR
Q	Output	OUT	BOOL	Q, M, N, VAR

ПРИМЕР: Обнаружение восходящего фронта



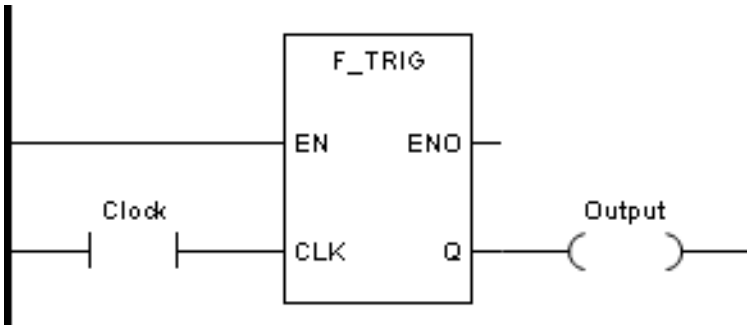
F_TRIG

Описание функции

- Этот функциональный блок обнаруживает задний фронт.
- Если функциональный блок обнаруживает изменение CLK с 1 на 0, выход Q изменится на 1, сохранится в течение одного цикла, а затем вернется к 0.

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма на языке IL:

CAL F_TRIG (CLK:=Clock, Q=>Output)

Форма на языке ST:

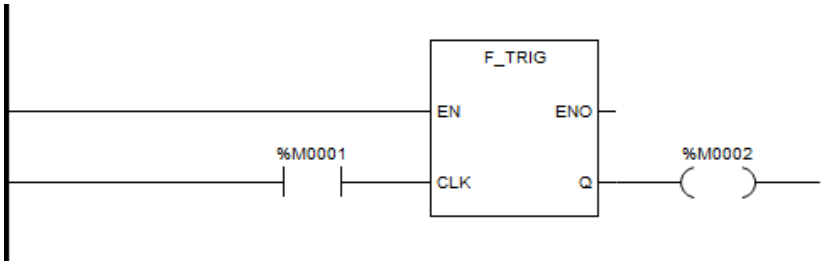
F_TRIG (CLK:=Clock, Q=>Output);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра

CLK	Clock	Вход	BOOL	CONSTANT, I, Q, M, N, S, VAR
Q	Output	OUT	BOOL	Q, M, N, VAR

ПРИМЕР: Обнаружение нисходящего фронта



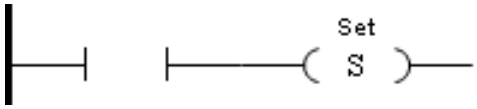
SET

Описание функции

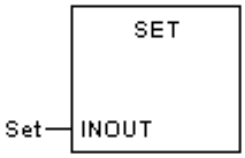
- Этот функциональный блок устанавливает соответствующий бит на 1. Функция та же, что и у set coil -(S)-.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

```
CAL   SET (Set)
```

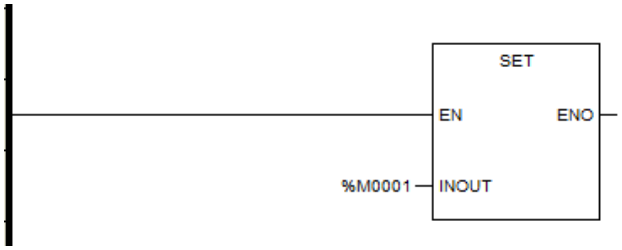
Форма на языке ST:

SET (Set);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
INOUT	Set	Заданное значение	BOOL	Q, M, N, VAR

ПРИМЕР: Установить 1 на %M0001



RESET

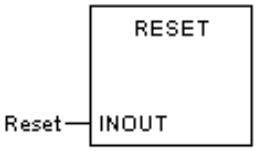
Описание функции

- Этот функциональный блок устанавливает соответствующий бит на 0. Функция та же, что и у set coil -(R)-.

ВЫЗОВ

Форма в LD:

Форма в FBD:



Форма на языке IL:

CAL RESET (Reset)

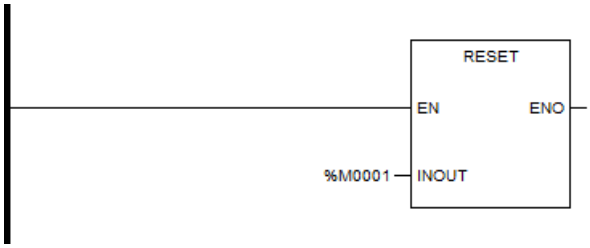
Форма на языке ST:

RESET (Reset);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
INOUT	Set	Заданное значение	BOOL	Q, M, N, VAR

ПРИМЕР: Сброс %M0001



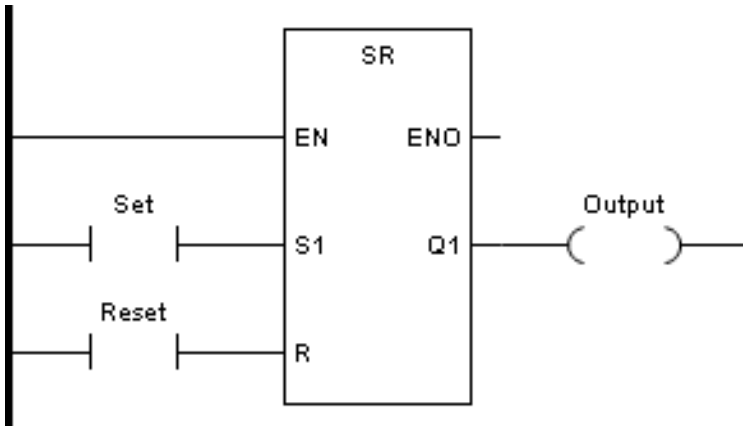
Bistable state (Set first) - SR

Описание функции

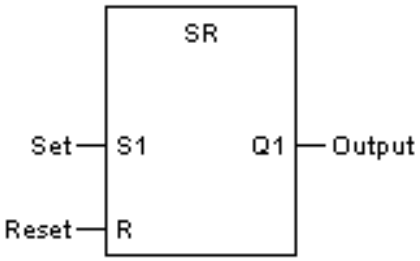
- Этот функциональный блок представляет собой запоминающее устройство SR со свойством SET first.
- Когда вход S1 становится 1, выход Q1 становится 1. Даже если вход S1 снова станет 0, Q1 все равно сохранится. Когда вход R становится 1, выход Q1 снова станет 0. Если входы S1 и R одновременно равны 1, Q1 будет установлен как 1.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL SR (S1:=Set, R:=Reset, Q1=>Output)

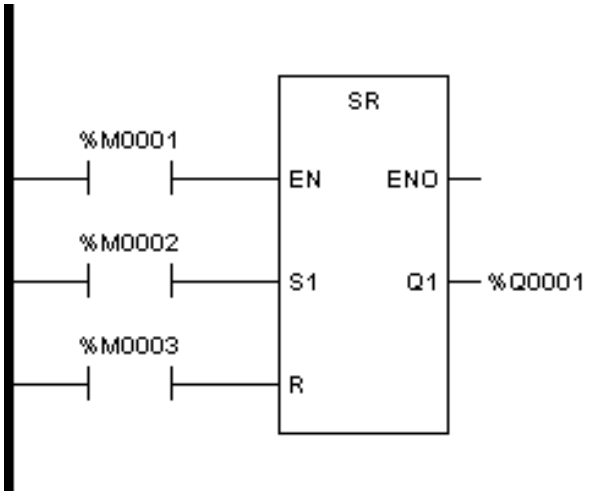
Форма на языке ST:

SR (S1:=Set, R:=Reset, Q1=>Output);

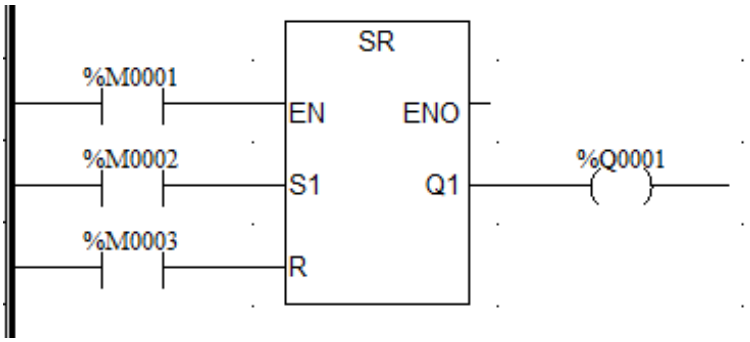
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
S1	Set	Установить (1)	BOOL	CONSTANT, I, Q, M, N, S, VAR
R	Reset	Сбросить	BOOL	CONSTANT, I, Q, M, N, S, VAR
Q	Output	OUT	BOOL	Q, M, N, VAR

ПРИМЕР:



Описание: На рисунке выше, когда выход осуществляется с использованием прямого адреса и %M0001 не равен 1, %Q0001 останется их текущим значением. Когда ПЛК перезапустится и вызовет этот функциональный блок первым, начальное состояние Q1 не изменится.



Описание: На рисунке выше, когда выход с использованием катушки и %M0001 не равен 1, %Q0001 будет принудительно установлен в 0. Когда ПЛК перезапускается и вызывает этот функциональный блок в первый раз, начальное состояние Q1 равно 0.

Bistable state (Reset first) - RS

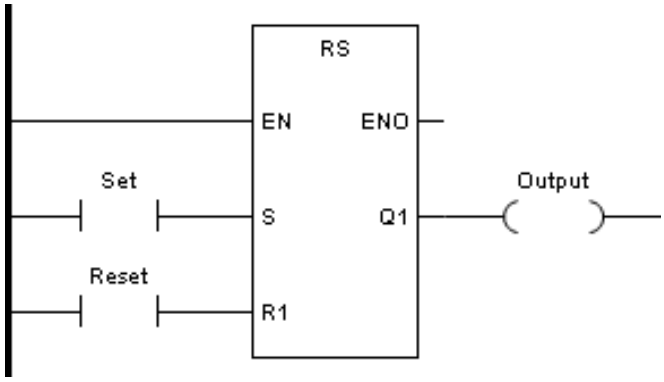
Описание функции

- Этот функциональный блок представляет собой RS-запоминающее устройство со свойством RESET first.

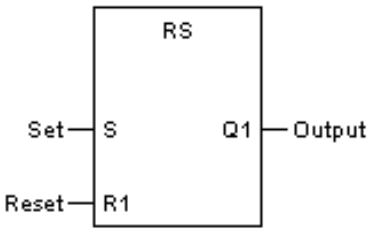
- Когда вход S1 становится 1, выход Q1 становится 1. Даже если вход S1 снова станет 0, Q1 все равно сохранится. Когда вход R становится 1, выход Q1 снова станет 0. Если вход S1 и R одновременно равны 1, Q1 будет установлен в 0.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL RS (S1:=Set, R:=Reset, Q1=>Output)

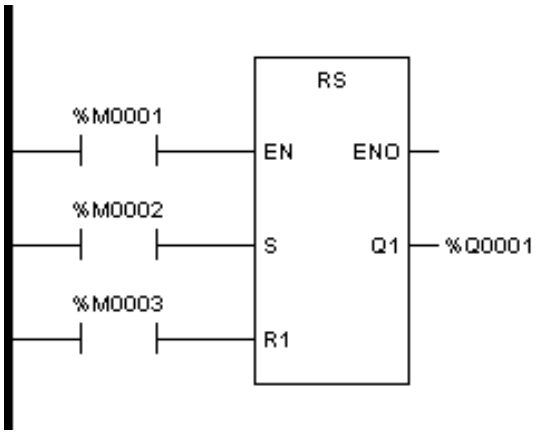
Форма на языке ST:

RS (S1:=Set, R:=Reset, Q1=>Output);

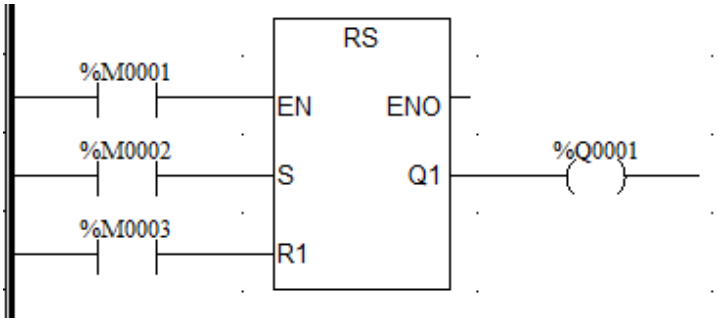
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
S1	Set	Набор	BOOL	CONSTANT, I, Q, M, N, S, VAR
R	Reset	Сброс (первый)	BOOL	CONSTANT, I, Q, M, N, S, VAR
Q	Output	OUT	BOOL	Q, M, N, VAR

ПРИМЕР:



Описание: На рисунке выше, когда выход осуществляется с использованием прямого адреса и %M0001 не равен 1, %Q0001 останется их текущим значением. Когда ПЛК перезапустится и вызовет этот функциональный блок первым, начальное состояние Q1 не изменится.



Описание: На рисунке выше, когда выход с использованием катушки и %M0001 не равен 1, %Q0001 будет принудительно установлен в 0. Когда ПЛК перезапускается и вызывает этот функциональный блок в первый раз, начальное состояние Q1 равно 0.

Операции отношения

Функциональные блоки реляционных операций используются для сравнения значений входных концов. Когда требования функционального блока выполнены, выходной конец разрешит текущий проход или назначит значение назначенной точке. Реляционная операция не изменит данные входного конца.

Тип	Описание
EQ	Равен: определяет IN1=IN2
NE	Не равен：определяет IN1≠IN2
GT	Больше, чем：определяет B1>B2
GE	Больше или равен：определяет IN1≥IN2
LT	Меньше, чем：определяет IN1< IN2
LE	Меньше, чем：определяет IN1≤IN2

Равно - EQ

Описание функции

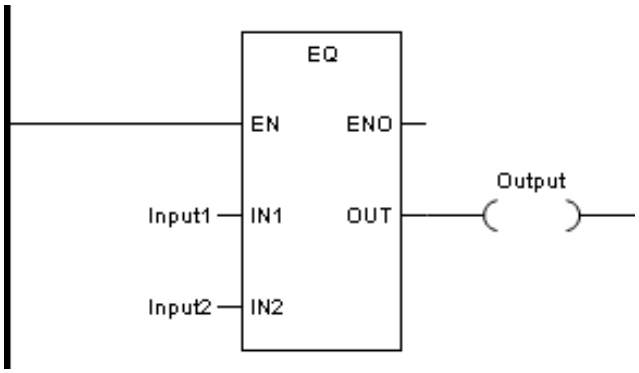
- Этот функциональный блок используется для определения того, равны ли значения входов. Если входы равны, OUT равен 1, в противном случае — 0.
- Входов — до 8.

Выражение

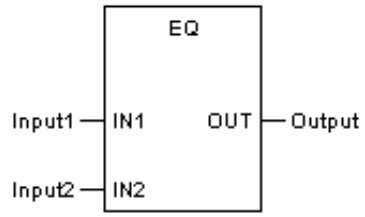
OUT = 1, IF (IN1 = IN2) AND (IN2 = IN3) AND … AND (INn-1 = INn)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Input1
EQ	Input2
ST	OUT

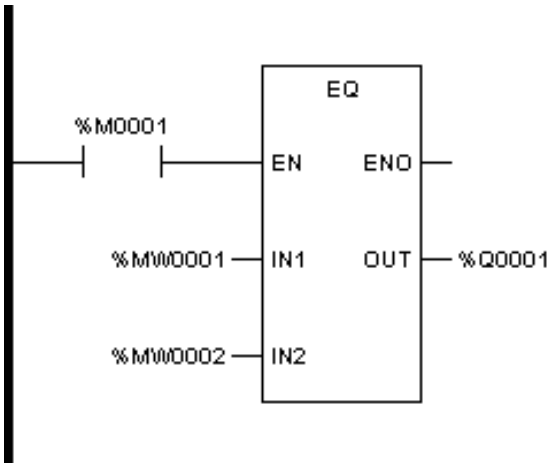
Форма в ST:

OUT := EQ (Input1, Input2);

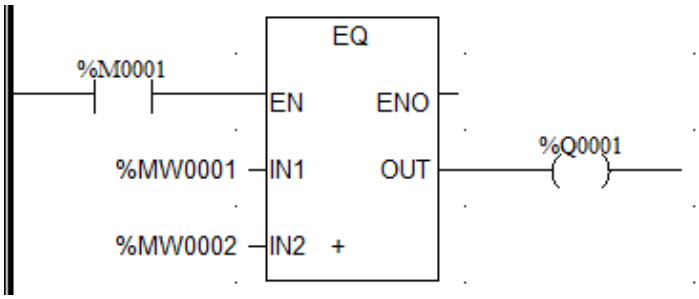
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Bit	Количество бит	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: EQ для целого числа (тип регистра MW — WORD)

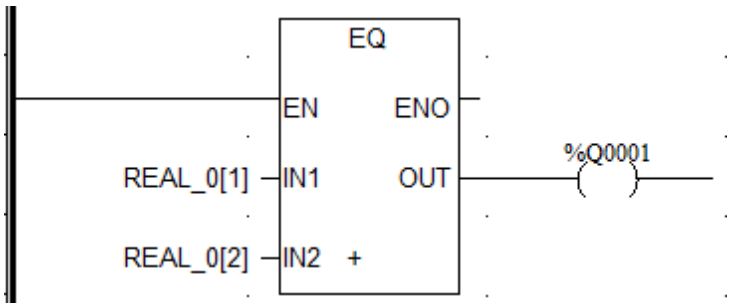


Описание: В приведенном выше LD, применяющем прямой адрес для вывода, когда %M0001=0, %Q0001 сохраняет текущее значение.

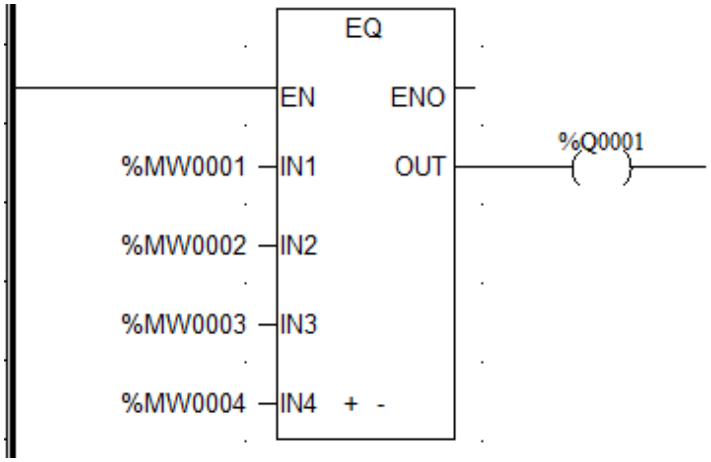


Описание: В приведенном выше LD с использованием катушки для выхода, когда %M0001=0, %Q0001 принудительно устанавливается в 0.

ПРИМЕР 2: EQ для чисел с плавающей запятой. (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: EQ для множества входов



Описание: На рисунке выше %Q0001 равен 1 только тогда, когда все входные данные равны.

Не равно - NE

Описание функции

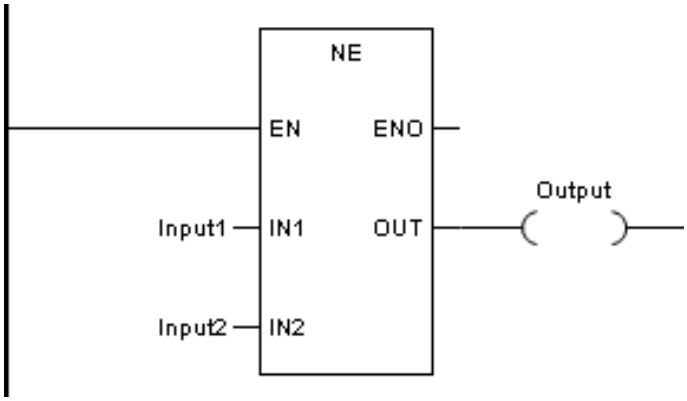
- Этот функциональный блок используется для определения того, являются ли значения входов НЕ равными. Если входы НЕ равны, OUT равен 1, в противном случае — 0.

Выражение

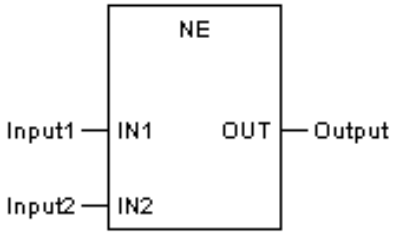
OUT = 1, IF B1≠B2

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Input1
NE	Input2
ST	OUT

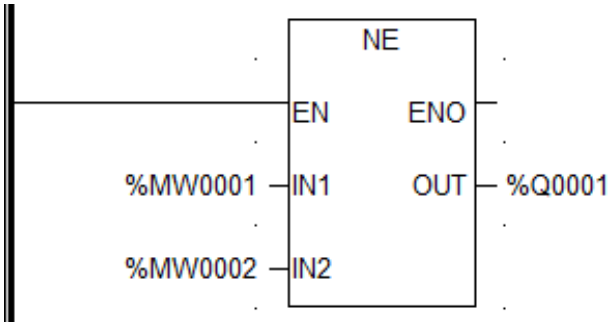
Форма в ST:

OUT := NE (Input1, Input2);

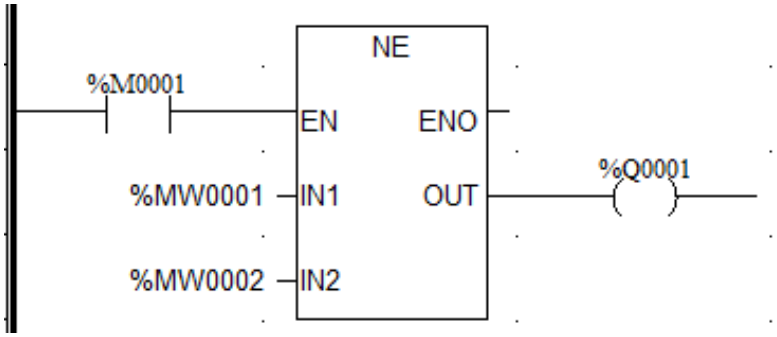
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: NE для целого числа (тип регистра MW — WORD)

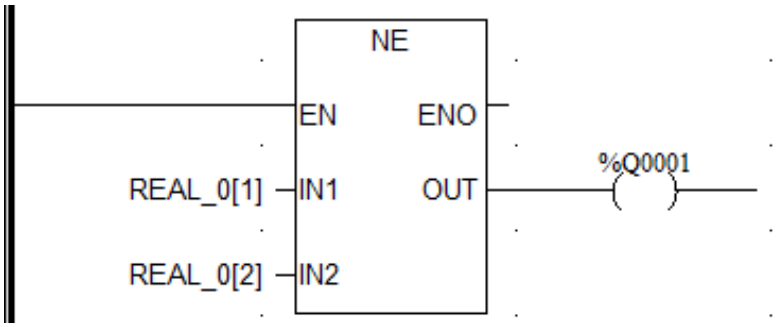


Описание: В приведенном выше LD, применяющем прямой адрес для вывода, когда %M0001=0, %Q0001 сохраняет текущее значение.



Описание: В приведенном выше LD с использованием катушки для выхода, когда %M0001=0, %Q0001 принудительно устанавливается в 0.

ПРИМЕР 2: NE для чисел с плавающей запятой. (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



Больше, чем - GT

Описание функции

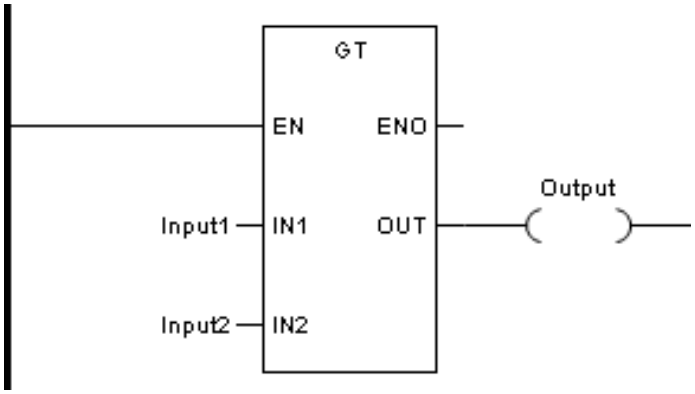
- Этот функциональный модуль проверяет, является ли значение последовательного входа убывающей последовательностью.
- Входы — до 8.

Выражение

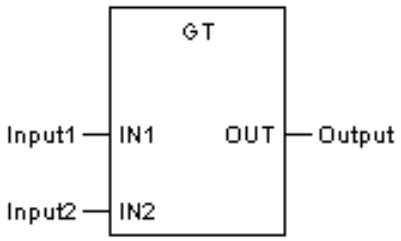
OUT = 1, IF (IN1>IN2) AND (IN2>IN3) AND … AND (INn-1>INn)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Input1
GT	Input2
ST	OUT

Форма в ST:

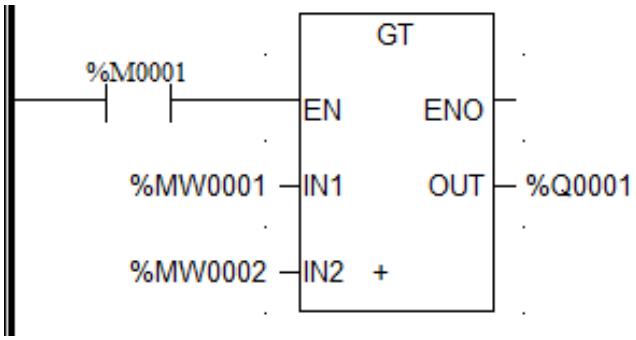
OUT := GT (Input1, Input2);

Описание параметра

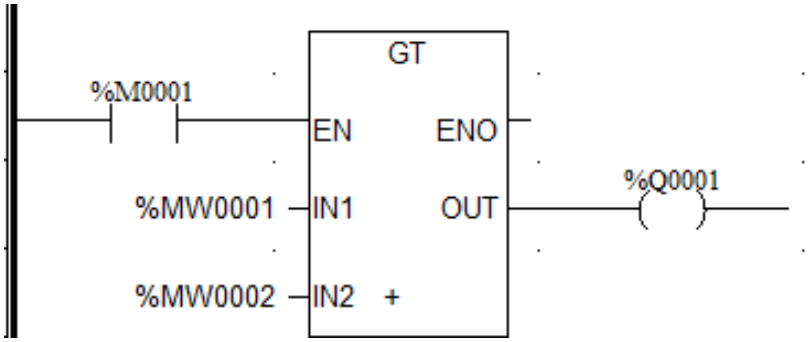
Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: GT для целого числа (тип регистра MW — WORD)

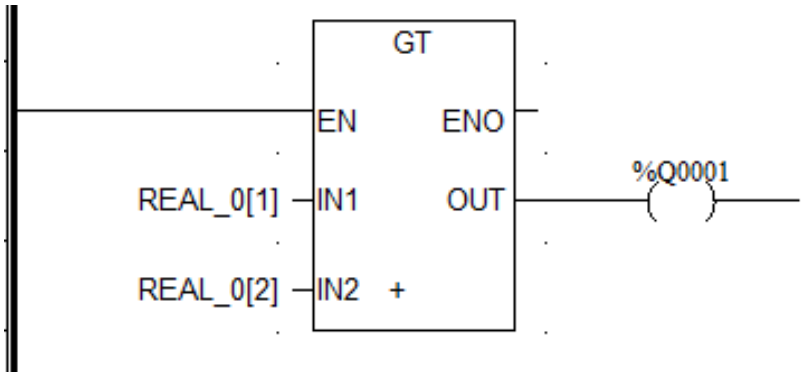
Описание: В приведенном выше LD, применяющем прямой адрес для вывода, когда %M0001=0, %Q0001 сохраняет текущее значение.



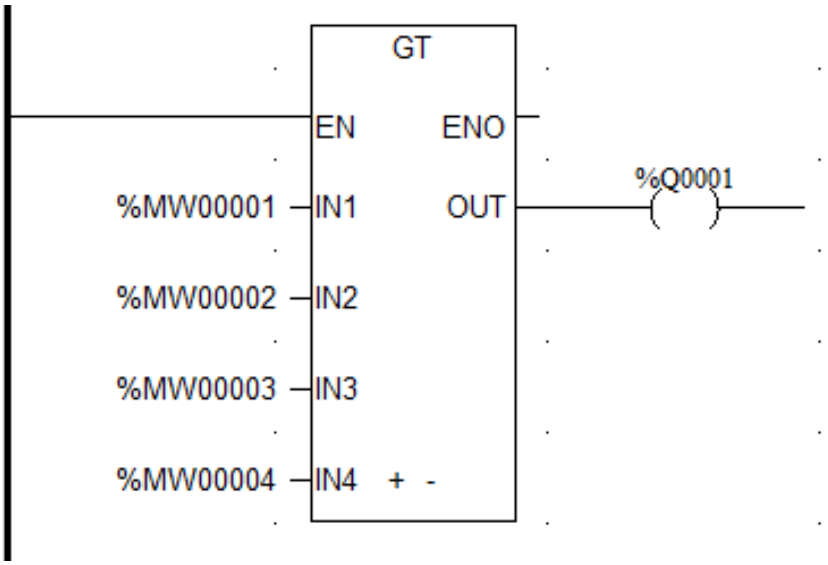
Описание: В приведенном выше LD с использованием катушки для выхода, когда %M0001=0, %Q0001 принудительно устанавливается в 0.



ПРИМЕР 2: GT для чисел с плавающей запятой. (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: GT для нескольких входов



Описание: На рисунке выше, только когда %MW0001>%MW0002>%MW0003>%MW0004, %Q0001 равен 1.

Больше или равно - GE

Описание функции

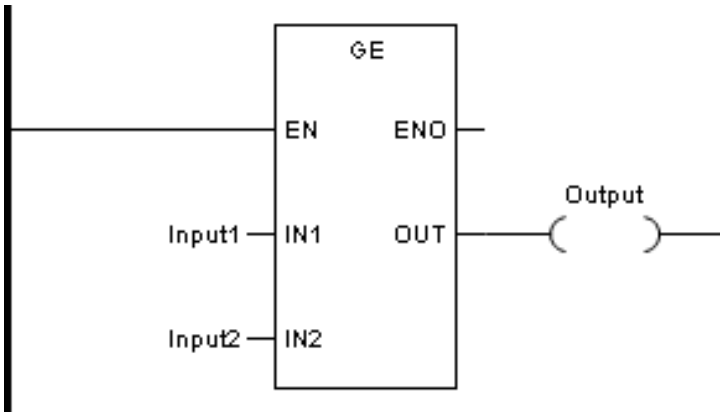
- Этот функциональный модуль проверяет, является ли значение последовательного входа убывающей последовательностью ИЛИ равным.
- Входы — до 8.

Выражение

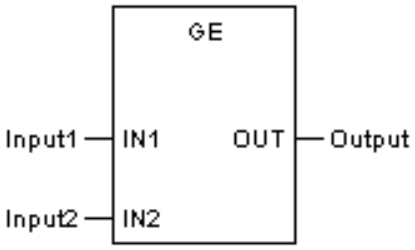
OUT = 1, IF (IN1≥IN2) AND (IN2≥IN3) AND … AND (INn-1≥INn)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Input1
GE	Input2
ST	OUT

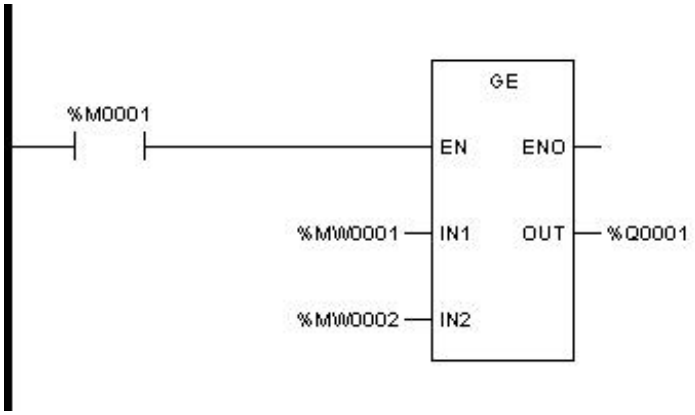
Форма в ST:

OUT := GE (Input1, Input2);

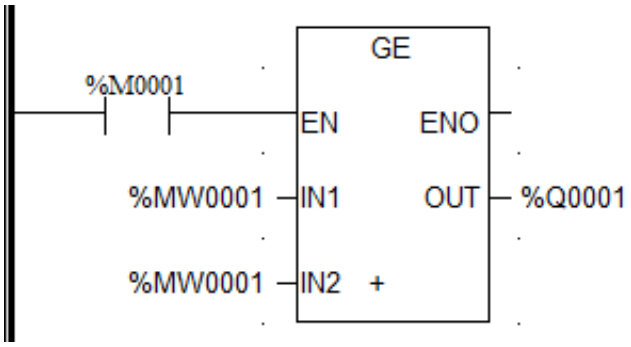
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: GE для целого числа (тип регистра MW — WORD)

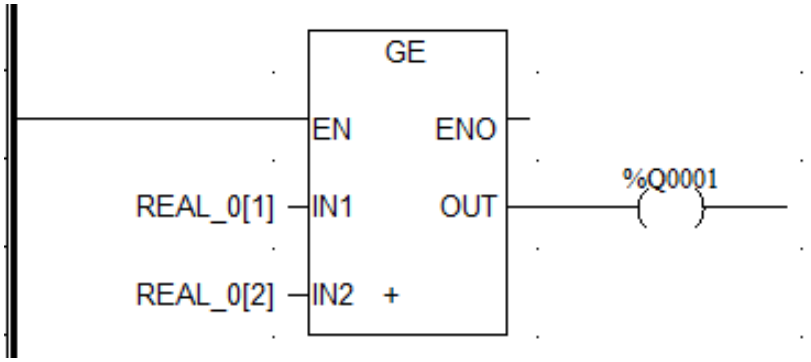


Описание: В приведенном выше LD, применяющем прямой адрес для вывода, когда %M0001=0, %Q0001 сохраняет текущее значение.

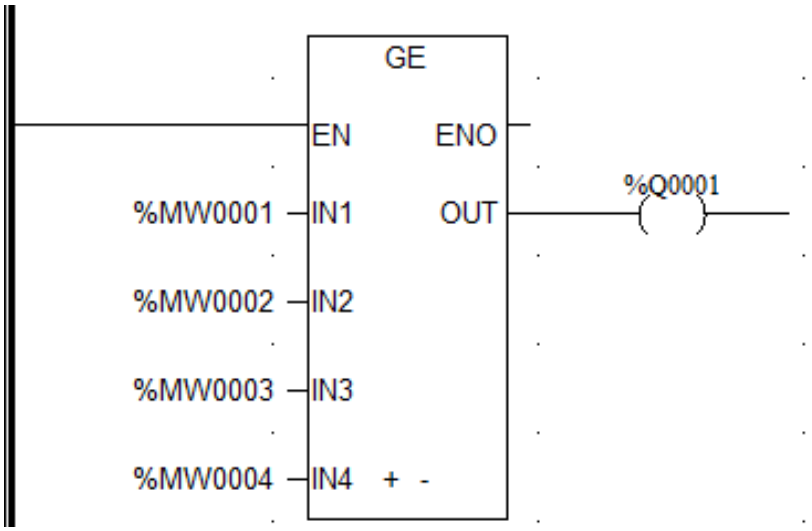


Описание: В приведенном выше LD с использованием катушки для выхода, когда %M0001=0, %Q0001 принудительно устанавливается в 0.

ПРИМЕР 2: GT для чисел с плавающей запятой. (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: GT для нескольких входов



Описание: На рисунке выше, только когда %MW0001≥%MW0002≥%MW0003≥%MW0004, %Q0001 равен 1.

Меньше, чем - LT

Описание функции

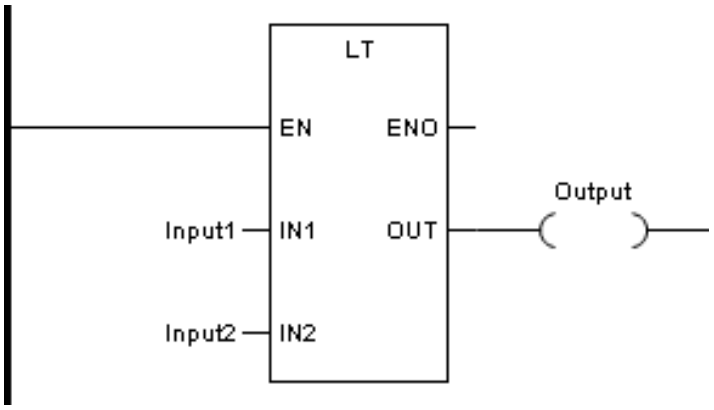
- Этот функциональный модуль проверяет, является ли значение последовательного входа возрастающей последовательностью.
- Входы — до 8.

Выражение

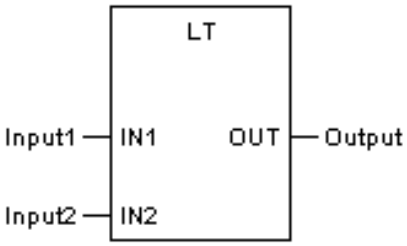
OUT = 1, IF (IN1<IN2) AND (IN2<IN3) AND … AND (INn-1<INn)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Input1
LT	Input2
ST	OUT

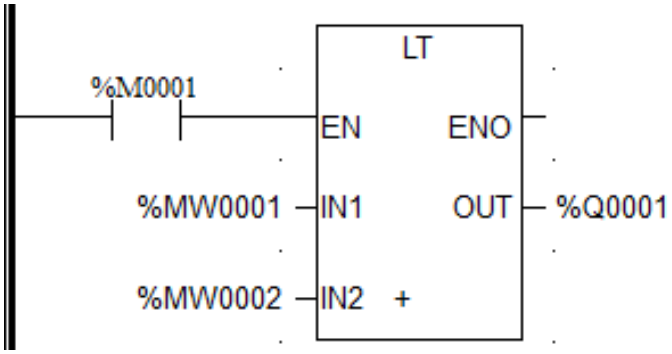
Форма в ST:

OUT := LT (Input1, Input2);

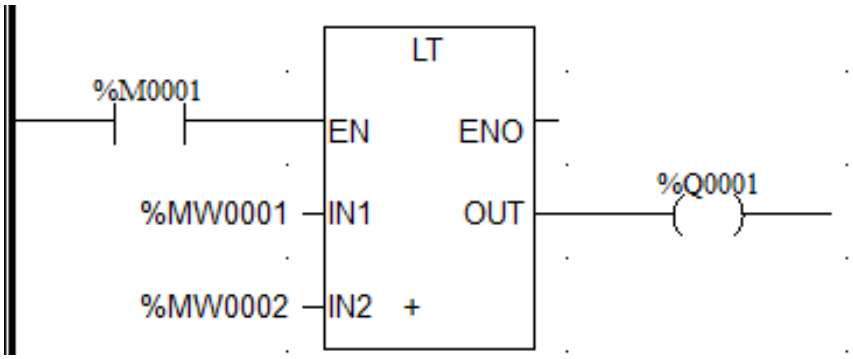
Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: LT для целого числа (тип регистра MW — WORD)

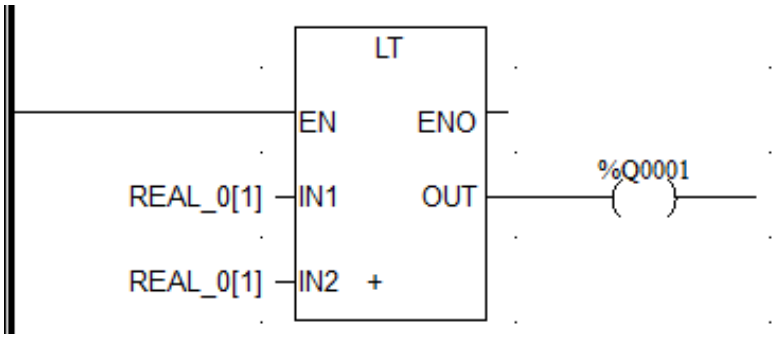


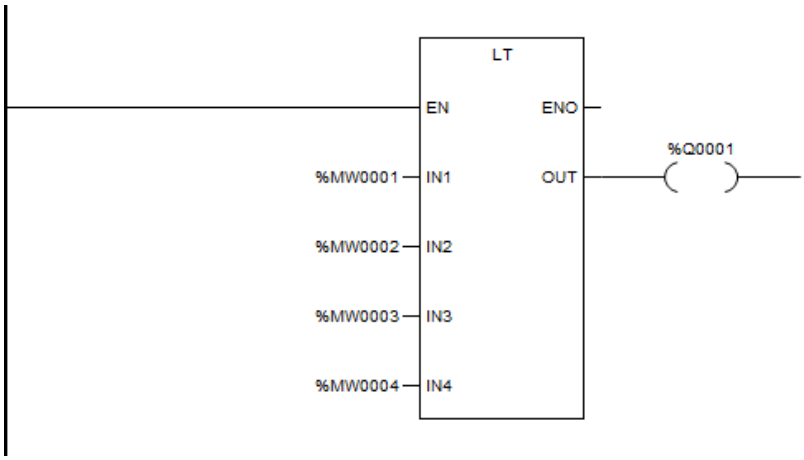
Описание: В приведенном выше LD, применяющем прямой адрес для вывода, когда %M0001=0, %Q0001 сохраняет текущее значение.



Описание: В приведенном выше LD с использованием катушки для выхода, когда %M0001=0, %Q0001 принудительно устанавливается в 0.

ПРИМЕР 2: LT для чисел с плавающей запятой. (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)





Описание: На рисунке выше, только когда %MW0001＜%MW0002＜%MW0003＜%MW0004, %Q0001 равен 1.

Меньше или равно - LE

Описание функции

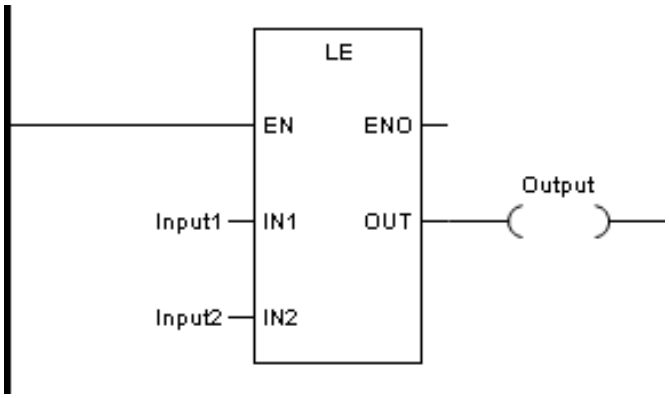
- Этот функциональный модуль проверяет, является ли значение последовательного входа возрастающей последовательностью ИЛИ равным.
- Входы — до 8.

Выражение

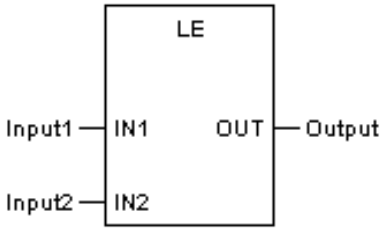
OUT = 1, IF (IN1≤IN2) AND (IN2≤IN3) AND … AND (INn-1≤INn)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Input1
LE	Input2
ST	OUT

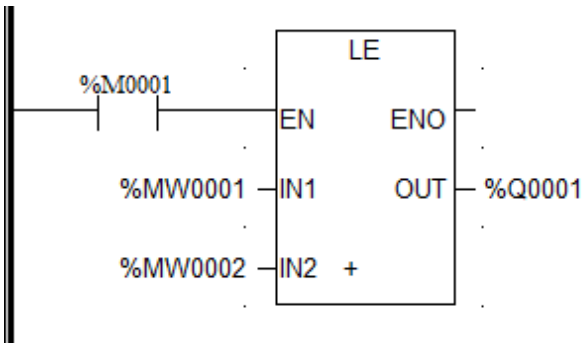
Форма в ST:

OUT := LE (Input1, Input2);

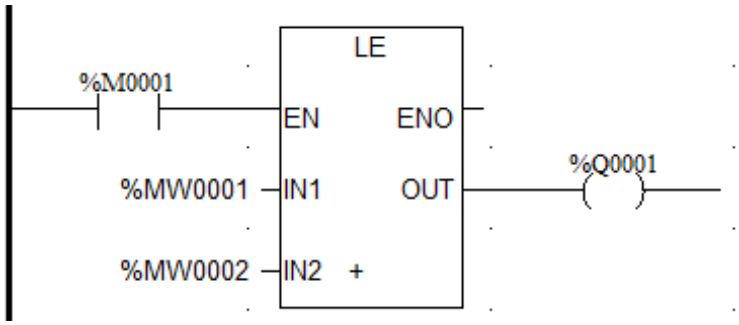
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход 1	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
IN2	Input2	Вход 2	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL	Q, M, N, VAR

ПРИМЕР 1: LE для целого числа (тип регистра MW — WORD)

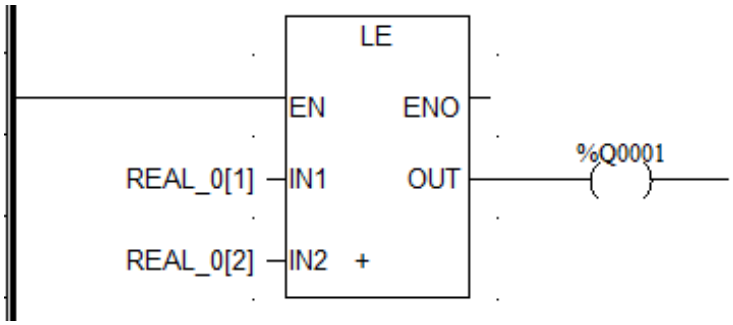


Описание: В приведенном выше LD, применяющем прямой адрес для вывода, когда %M0001=0, %Q0001 сохраняет текущее значение.

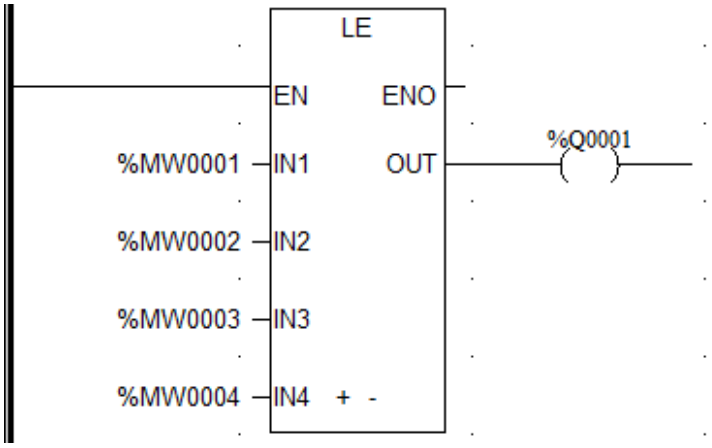


Описание: В приведенном выше LD с использованием катушки для выхода, когда %M0001=0, %Q0001 принудительно устанавливается в 0.

ПРИМЕР 2: LE для чисел с плавающей запятой. (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



ПРИМЕР 3: LE для нескольких входов



Описание: На рисунке выше, только когда %MW0001≤%MW0002≤%MW0003≤%MW0004, %Q0001 равен 1.

Преобразование данных

Функциональные блоки преобразования данных в основном используются для преобразования между целыми числами и BCD, целыми числами и кодом Грея, градусами и радианами.

Тип	Описание
INT_TO_BCD	целое число в код BCD
BCD_TO_INT	BCD-код в целое число
INT_TO_GRY	Целое число в код Грея
GRY_TO_INT	Код Грея в целое число
DEG_TO_RAD	Градусы в радианы
RAD_TO_DEG	Радианы в градусы

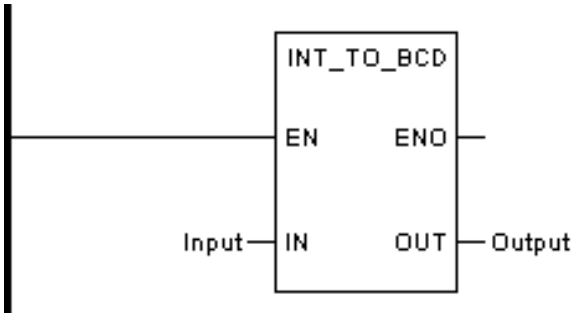
Целое число в код BCD - INT_TO_BCD

Описание функции

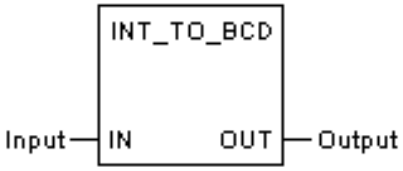
- Этот функциональный блок используется для преобразования входного целого числа в код BCD и присваивает результат OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Вход
INT_TO_BCD	
ST	OUT

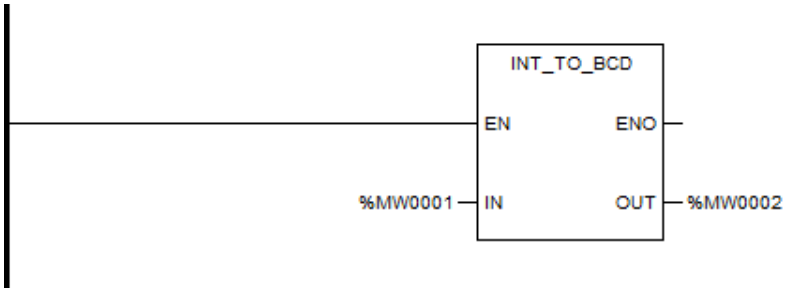
Форма в ST:

OUT := INT_TO_BCD (Вход);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Целое число в код BCD (тип регистра MW — WORD)



Описание: Преобразованное значение можно просмотреть в режиме онлайн-формате в двоичном виде.

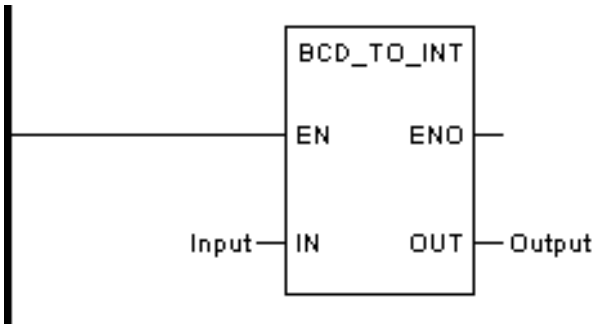
Код BCD в целое число - BCD_TO_INT

Описание функции

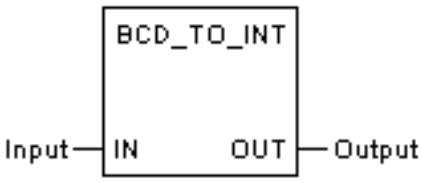
- Этот функциональный блок используется для преобразования входного кода BCD в целое число и присваивает результат OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Вход
BCD_TO_INT	
ST	OUT

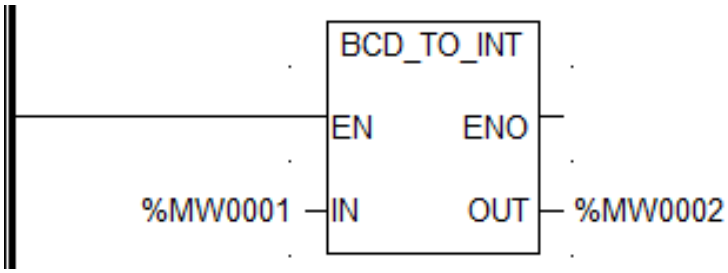
Форма в ST:

OUT := BCD_TO_INT (Вход);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Преобразование кода BCD в целое число (тип регистра MW — WORD)



Описание: Преобразованное значение можно просмотреть в режиме онлайн-формате в двоичном виде.

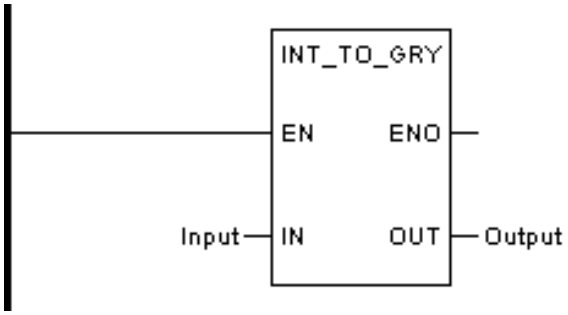
Целое число в код Грея - INT_TO_GRY

Описание функции

- Этот функциональный блок используется для преобразования входного целого числа в код Грея и присваивает результат OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма в IL:

LD	Вход

INT_TO_GRY	
ST	OUT

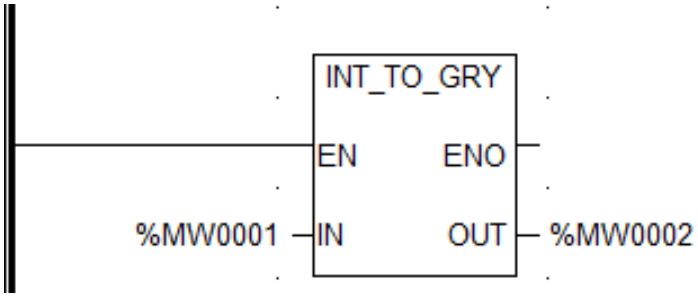
Форма в ST:

OUT := INT_TO_GRY (Вход);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Целое число в код Грея (тип регистра MW — WORD)



Описание: Преобразованное значение можно просмотреть в режиме онлайн-формате в двоичном виде.

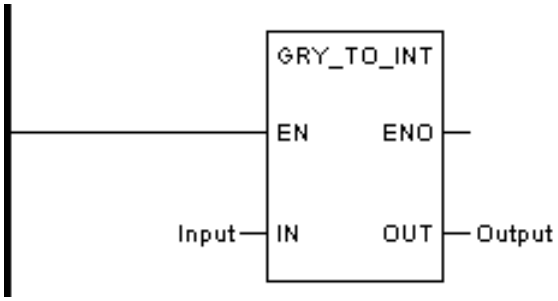
Код Грея в целое число - GRY_TO_INT

Описание функции

- Этот функциональный блок используется для преобразования входного кода Грея в целое число и присваивает результат OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма в IL:

LD	Вход
GRY_TO_INT	
ST	OUT

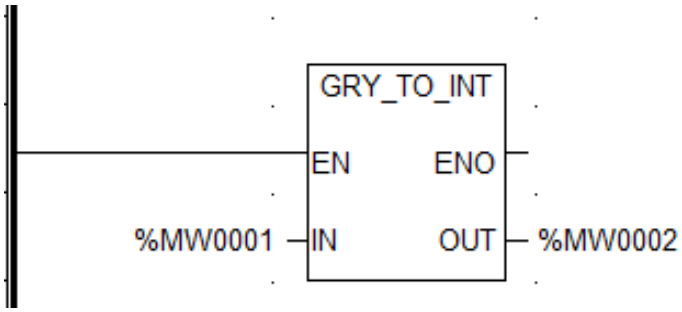
Форма в ST:

OUT := GRY_TO_INT (Вход);

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Код Грея в целое число (тип регистра MW — WORD)



Описание: Преобразованное значение можно просмотреть в режиме онлайн-формате в двоичном виде.

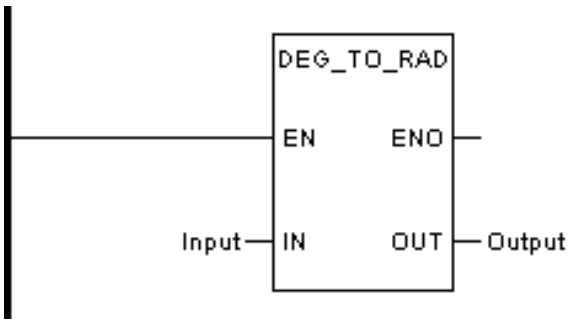
Градусы в радианы - DEG_TO_RAD

Описание функции

- Этот функциональный блок используется для преобразования угла, отображаемого в градусах, в радианы и присваивает результат OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:

Форма в IL:

LD	Вход

DEG_TO_RAD	
ST	OUT

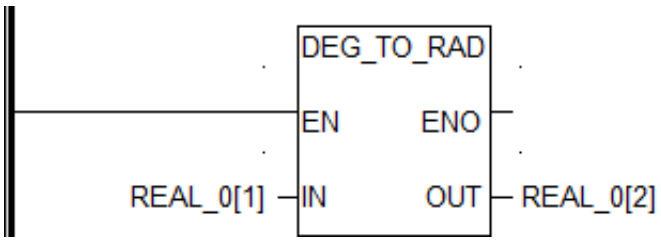
Форма в ST:

OUT := DEG_TO_RAD (Вход);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Градусы в радианы (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



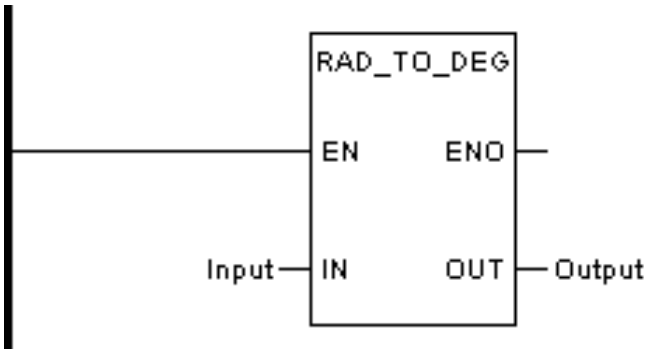
Радианы в градусы - RAD_TO_DEG

Описание функции

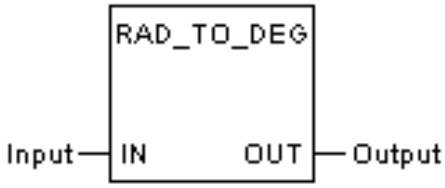
- Этот функциональный блок используется для преобразования угла, отображаемого в радианах, в градусы и присваивает результат OUT.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Вход
RAD_TO_DEG	
ST	OUT

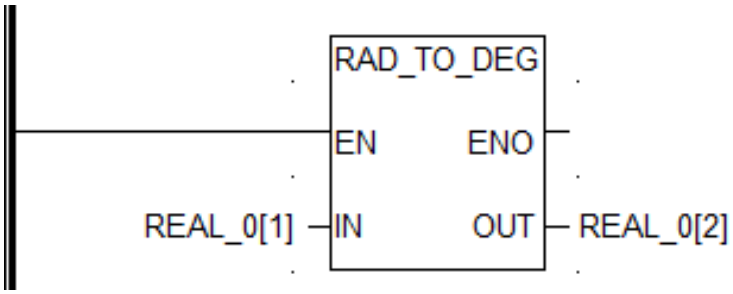
Форма в ST:

OUT := RAD_TO_DEG (Вход);

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Вход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Радианы в градусы (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)


RAD_TO_DEG
EN
ENO
REAL_0[1]
IN
OUT
REAL_0[2]

Перемещение данных

Тип	Описание
MOVE	Перемещение данных
BLKMOV	Перемещение блока
BLKCLR	Очистка блока
ETHMOV	Передача данных Ethernet
COMMOV	Передача данных связи
READ	Считать данные специального модуля
WRITE	Записать данные в специальный модуль
XMT	Передача данных по свободному порту
RCV	Считать данные через свободный порт
LRC	Продольный контроль избыточности
CRC	Рассчитать контрольный код CRC
MODRW	Чтение и запись данных MODBUS

Перемещение данных - MOVE

Описание функции

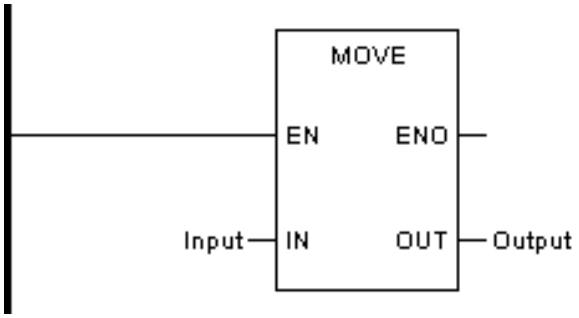
- Этот функциональный блок используется для копирования данных из одной точки в другую. Эта операция не изменит исходные данные.

Выражение

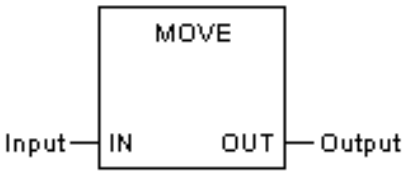
OUT = IN

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

LD	Вход
ST	OUT

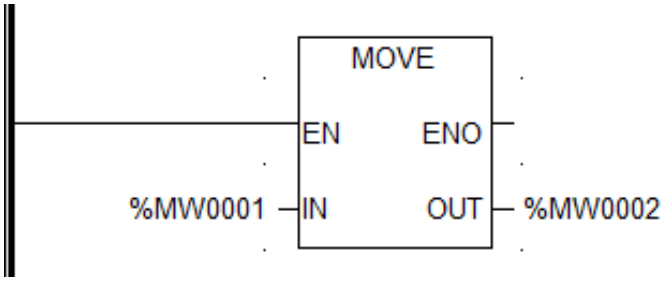
Форма в ST:

OUT := Вход;

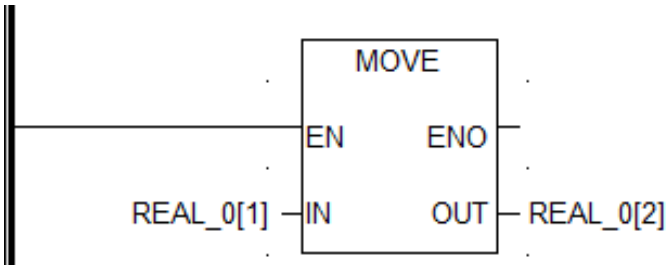
Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN1	Input1	Вход	BOOL[*1], BYTE, WORD, DWORD, SINT, INT, DINT, REAL USINT, UINT, UDINT	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, VAR
IN2	Input2	Выход	BOOL[*1], BYTE, WORD, DWORD, SINT, INT, DINT, REAL USINT, UINT, UDINT	Q, M, N, MW, NW, VAR
OUT	OUT			
[*1] BOOL не поддерживается LD				

ПРИМЕР 1: Перемещение данных Integer



ПРИМЕР 2: Перемещение данных с плавающей запятой (REAL_0[1] и REAL_0[2] — определяемые пользователем переменные с плавающей запятой)



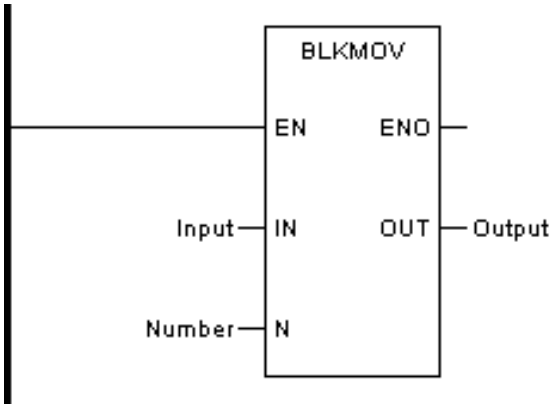
Перемещение блока - BLKMOV

Описание функции

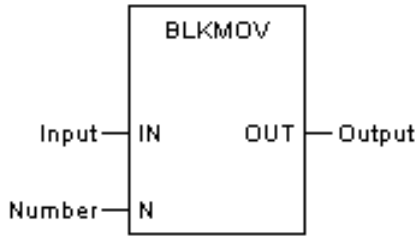
- Этот функциональный блок копирует заданный объем входных данных из одной области памяти в другую область той же самой памяти.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL BLKMOV (IN:=Вход, N:=Число, OUT=>Output)

Форма в ST:

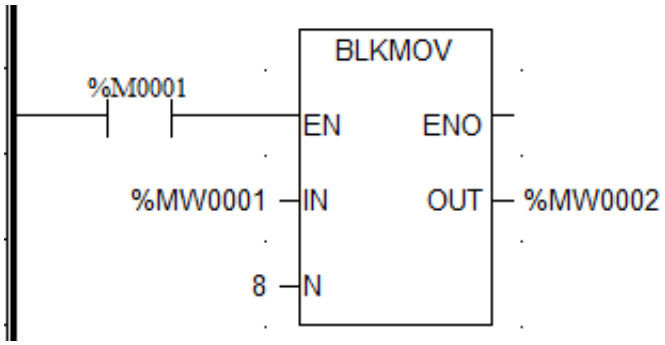
BLKMOV (IN:=Вход, N:=Число, OUT=>Output);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Исходные данные для копирования	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT ,REAL, USINT, UINT, UDINT	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, VAR
N	Number	Количество скопированных данных, за единицу измерения принимается тип входных данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Выход	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT ,REAL, USINT, UINT, UDINT	M, N, MW, NW, VAR

ПРИМЕР:

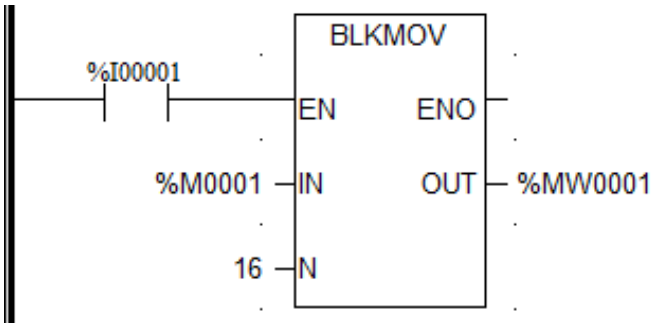
1. Перемещение данных Integer пакетами



Описание: В приведенном выше LD, когда %M0001=1, 8 последовательных регистров слов, начиная с начального адреса %MW0001, копируются партиями в другие 8 последовательных регистров слов, начиная с начального адреса %NW0001.

Регистр	Значение		Регистр	Значение
%MW0001	10		%NW0001	10
%MW0002	20	→	%NW0002	20
%MW0003	30		%NW0003	30
%MW0004	40	→	%NW0004	40
%MW0005	50	→	%NW0005	50
%MW0006	60		%NW0006	60
%MW0007	70		%NW0007	70
%MW0008	80		%NW0008	80

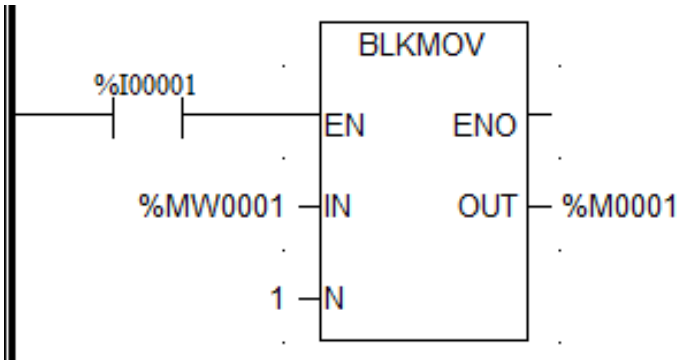
2. Перемещение состояний битов в регистр слов партиями



Описание: В приведенном выше LD, когда %I00001=1, 16 последовательных битовых регистров, начиная с начального адреса %M0001, копируются в другие 16 последовательных слов-регистров, начиная с начального адреса %MW0001. Если число перемещаемых битов равно 32, будут заняты два слова-регистра, т. е. первые 16 бит хранятся в %MW0001, последние хранятся в %MW0002.

Регистр	**Значение**		**Регистр**	**Значение**
%M0001	1		%MW0001	7
%M0002	1	→	%MW0002	0
%M0003	1		%MW0003	0
%M0004	0	→	%MW0004	0
%M0005	0		%MW0005	0
%M0006	0	→	%MW0006	0
%M0007	0		%MW0007	0
%M0008	0		%MW0008	0

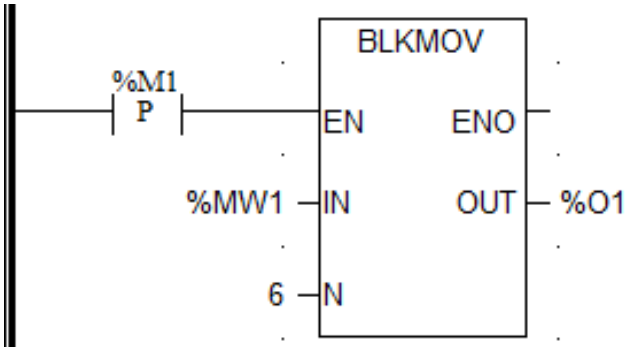
3. Перемещение словесных данных в битовый регистр



Описание: В приведенном выше LD, когда %I00001=1, значения регистра %MW0001 копируются в 16 последовательных битовых регистров. Если число перемещаемых слов равно 2, будут заняты 32 битовых регистра, т. е. %M0001~%M0016 сохраняют первые 16 (значение %MW0001), а %M0017~%M0032 сохраняют последние 16 (значение %MW0002).

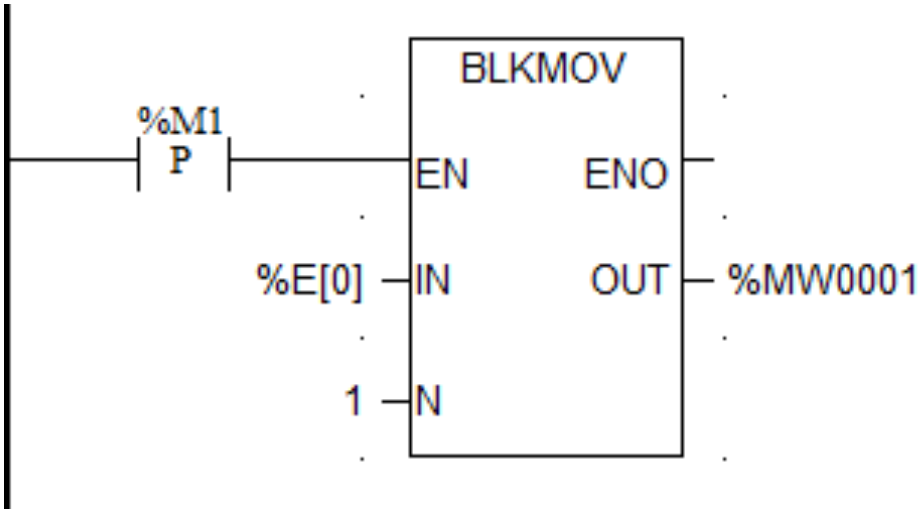
Регистр	Значение		Регистр	Значение
%MW0001	7		%M0001	1
		→	%M0002	1
		→	%M0003	1
		→	%M0004	0
		→	%M0005	0
			%M0006	0
			%M0007	0
			%M0008	0

4. Изменение системного времени SW1-SW6



Описание: В LD выше запишите значение в регистр %O1 (то есть %SW1~%SW6, всего 6 слов, регистр системного времени) с помощью функционального блока BLKMOV. В качестве входа можно использовать %MW, %NW, %V, чтобы хранить год, месяц, день, час, минуту и секунду отдельно. Обратите внимание на диапазон данных. Если диапазон данных неверен, запись будет неудачной, например, значение месяцев больше 12 или часов больше 24.

5. Прочитать событие SOE



Описание: В LD выше, для чтения событий SOE из модулей SOE используется функциональный блок BLKMOV. Каждый ЦП может хранить до 256 событий SOE, что соответствует %E[0] ~ %E[255]. N относится к количеству событий, которые должны быть прочитаны. Каждое событие SOE занимает 16 бит. Конкретная форма показана в следующей таблице:

Формат событий SOE:

Адрес (бит)	**Имя**	**Описание**
1	Метка события	1
2	Свойство	1 : ВЫКЛ→ВКЛ ; 0 : ВКЛ→ВЫКЛ
3	Год	Более 2000
4	Месяц	
5	День	
6	Час	
7	Минута	
8	Второй	
9	Миллисекунда	
10		
11	Количество регистров измерения	
12		
13	Измерение ценности	0/1
14		
15	Сохранение данных	0
16		

%SW0010, указатель SOE, хранит адрес текущего события, начальный адрес которого 10031 (адрес в виде слова). Указатель добавит 8 с созданием одного события SOE. Когда значение достигает 256, указатель снова начнется с 10031. Таким же образом, хранение событий SOE по %E[**] также применяет режим прокрутки. Когда значение достигает %E[255], следующее событие будет сохранено, начиная с %E[0].

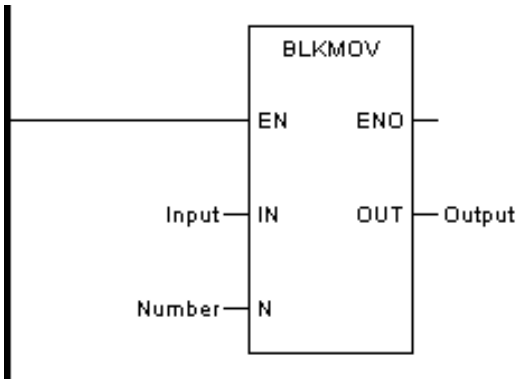
Очистить блок - BLKCLR

Описание функции

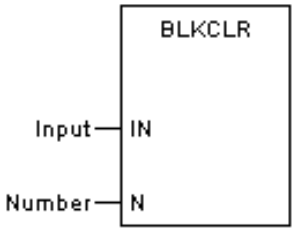
- Этот функциональный блок может установить все назначенные входные блоки данных в 0. N относится к числу данных, которые необходимо очистить. Единица измерения та же, что и у типа данных IN. Если IN — регистр MW, N очищает N регистры MW, возвращая им 0.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL BLKCLR (IN:=Вход, N:=Number)

Форма в ST:

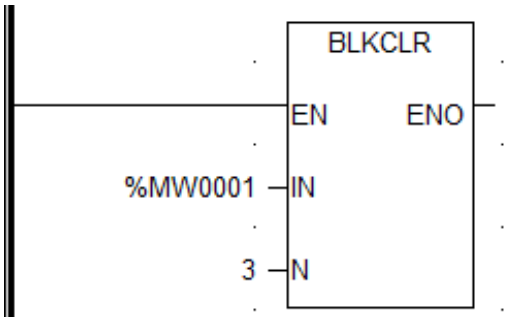
BLKCLR (IN:=Вход, N:=Number);

Описание параметра

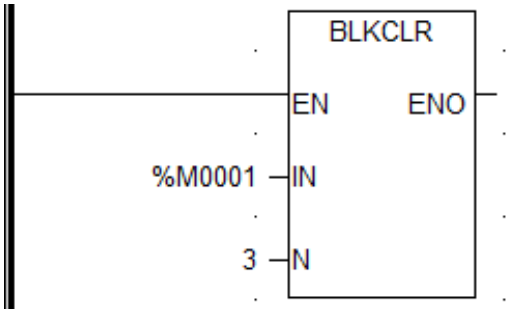
Диаграмма	Параметр	Описание	Тип данных	Тип регистра

IN	Input	Точки, которые будут очищены до 0	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT ,REAL, USINT, UINT, UDINT	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, VAR
N	Number	Количество регистров, которые необходимо очистить, (единицей измерения является тип входных данных.)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Очистка блока регистра MW, т. е. очистка данных в %MW0001, %MW0002 и %MW0003.



ПРИМЕР 2: Очистка блока регистра M, т. е. очистка данных в %M0001, %M0002 и %M0003.



Перемещение данных через Ethernet - ETHMOV

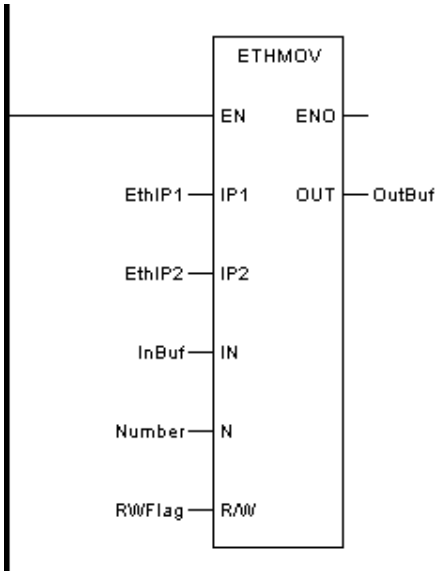
Описание функции

- Этот функциональный блок используется для реализации обмена данными с другими ПЛК через Ethernet. Цикл выполнения не более 1 с. И функциональный

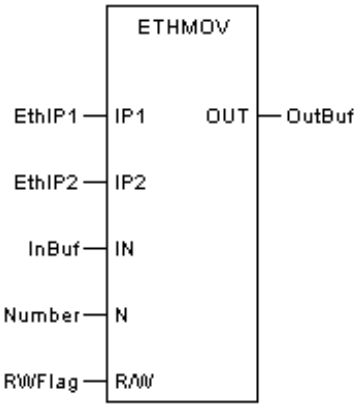
блок запускается по восходящему или нисходящему фронту EN. С помощью программного обеспечения INDAS PRO версии выше V5.1 пользователи могут использовать интерфейс Ethernet для чтения и записи данных других ПЛК через основную конфигурацию MODBUS/TCP.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL ETHMOV (IP1:=EthIP1, IP2:=EthIP2, IN:=Вход, N:=Number,

R/W:=RWFlag, OUT=>Output)

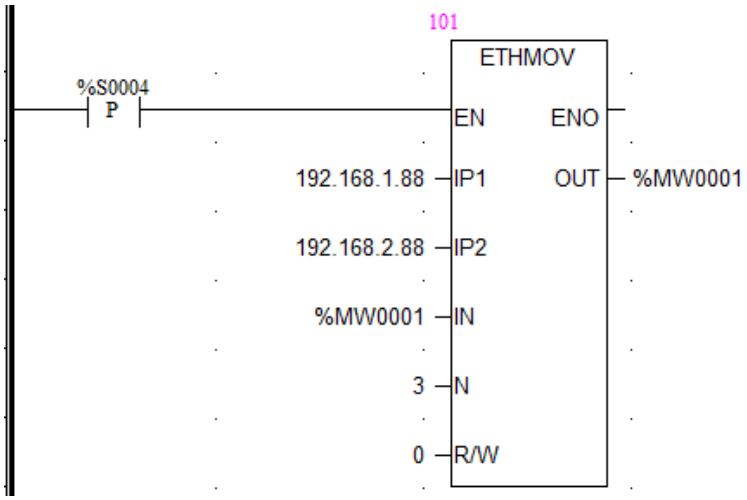
Форма в ST:

ETHMOV (IP1:=EthIP1, IP2:=EthIP2, IN:=Вход, N:=Число, R/W:=RWFlag, OUT=>Output);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IP1	EthIP1	Другой ПЛК Ethernet IP1	DWORD	например, 10.40.0.22
IP2	EthIP2	Другие ПЛК Ethernet IP2	DWORD	например, 10.40.1.22
IN	Input	Область хранения данных главного ПЛК	WORD	MW
N	Number	Объем данных с единицей измерения WORD.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
R/W	RWFlag	Чтение - 0. Запись - 1.	BOOL	CONSTANT, I, Q, M, N, S, VAR
OUT	Output	Область хранения данных других ПЛК.	WORD	MW

ПРИМЕР: Обмен данными в %MW0001~%MW0003 IP-адресов 192.168.1.88 и 192.168.2.88, с одним обменом в секунду в режиме только для чтения.



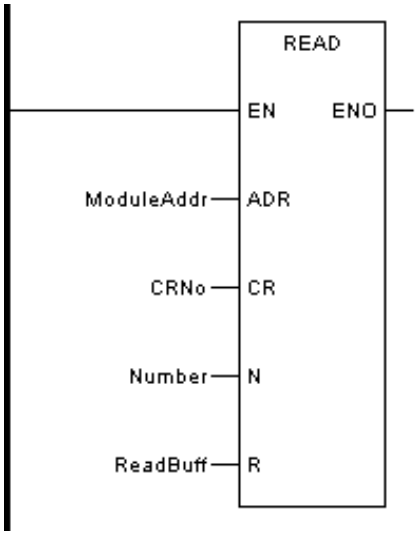
Чтение данных специальной модели - READ

Описание функции

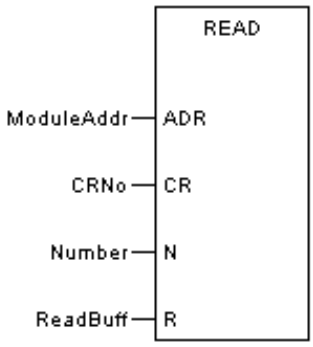
- Этот функциональный блок используется для чтения данных CR специальной модели. При выполнении блока данные CR специальной модели в адресе ADR считываются в область R ПЛК. Чтение N данных CR происходит один раз.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL READ (ADR:=ModuleAddr, CR:=CRNo, N:=Number, R:=ReadBuff)

Форма в ST:

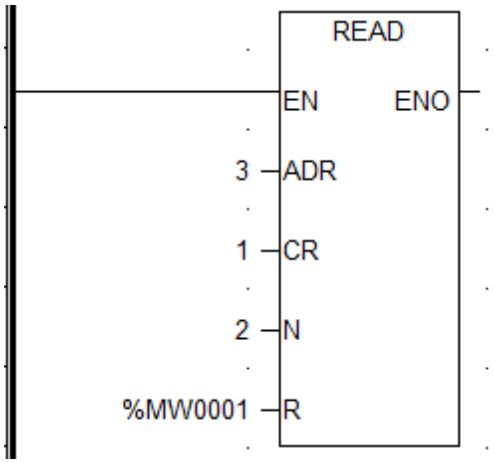
READ (ADR:=ModuleAddr, CR:=CRNo, N:=Number, R:=ReadBuff);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра

ADR	ModuleAddr	Адрес специальной модели	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
CR	CRNo	Начальный порядковый номер считываемого CR равен 1, каждый CR занимает 2 байта.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Номер CR, который нужно прочитать	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
R	ReadBuff	Буфер считанных данных используется для хранения считанных данных.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Пример: прочитать данные CR в специальном модуле с адресом 3, номер CR равен 2, прочитать из CR1, то есть прочитать данные CR1 и CR2, и поместить их в %MW0001, %MW0002 и %MW0003, %MW0004 (каждый CR составляет 2 байта).



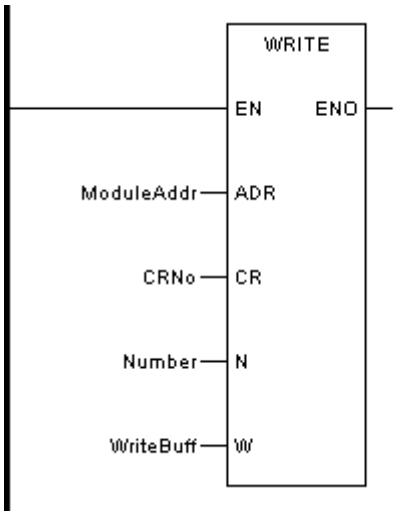
Запись данных специальной модели - WRITE

Описание функции

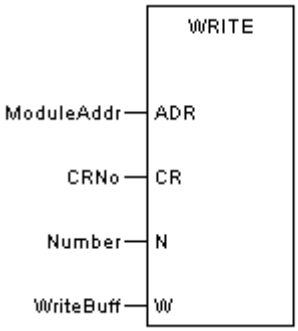
- Этот функциональный блок используется для записи данных CR в специальную модель. Когда блок выполняется, данные в области W записываются в адрес CR в ADR специальной модели, и одновременно записывается N CR.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL WRITE (ADR:=ModuleAddr, CR:=CRNo, N:=Number, R:=WriteBuff)

Форма в ST:

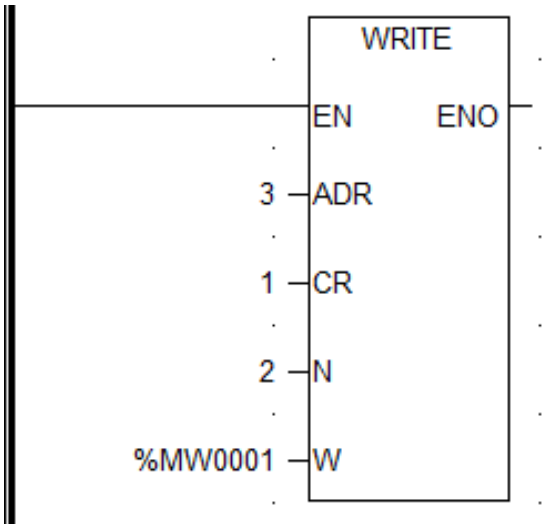
WRITE (ADR:=ModuleAddr, CR:=CRNo, N:=Number, R:=WriteBuff);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
ADR	ModuleAddr	Адрес специальной модели	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
CR	CRNo	Начальный порядковый номер считываемого CR равен 1, каждый CR занимает 2 байта.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

N	Number	Номер CR, который нужно прочитать	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
R	ReadBuff	Буфер считанных данных используется для хранения считанных данных.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР: Номер CR, который нужно записать, равен 2, начиная с CR1. Сохраните данные, которые нужно записать, в %MW0001, %MW0002, %MW0003 и %MW0004 (каждый CR занимает 2 байта), а затем запишите данные в CR1 и CR2.



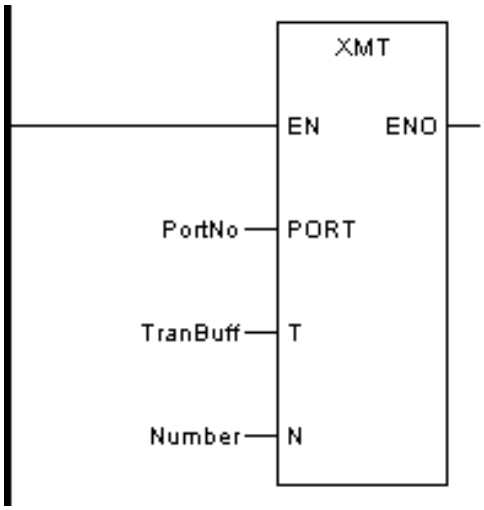
Передача данных через порт - XMT

Описание функции

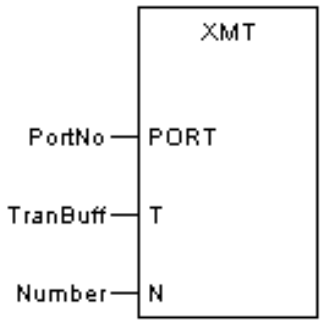
- Данный функциональный блок используется для передачи данных в режиме свободного порта.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL XMT (PORT:=PortNo, T:=TranBuff, N:=Number)

Форма в ST:

XMT (PORT:=PortNo, T:=TranBuff, N:=Number)

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
PORT	PortNo	Номер последовательного порта (1 или 2).	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
T	TranBuff	Буфер передачи, используемый для хранения данных, подлежащих передаче.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
N	Number	Количество байтов, которые необходимо передать.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

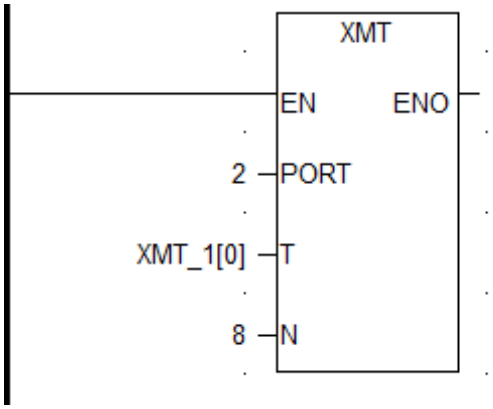
Состояние передачи

%SW0021(Состояние передачи COM1) %SW0023(Состояние передачи COM2)
0 : Передача в процессе
1 : Успешная передача
2 : Неудачная передача

Регистры системы передачи и приема CPU201-1101 и CPU401-1101 с 4 последовательными портами приведены в следующей таблице:

Номер последовательного порта	**COM1**	**COM2**	**COM3**	**COM4**
Регистр статуса передачи	%SW0021	%SW0023	%SW0041	%SW0043
Регистр статуса получения	%SW0022	%SW0024	%SW0042	%SW0044

ПРИМЕР:



Имя	Ввод	Тип данных	Размерность	Значение	Комментарий	Адрес	Объем	Число обращений
XMT_1	XMT_1	BYTE[]	100			%V00017	100	0

Описание: На схеме выше XMT запускается по переднему или заднему фронту. 8 байтов, начиная с XMT 1[0] в массиве XMT_1, отправляются из последовательного порта 2.

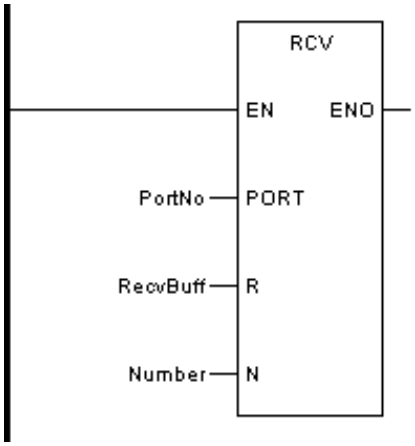
Получение данных через порт RCV

Описание функции

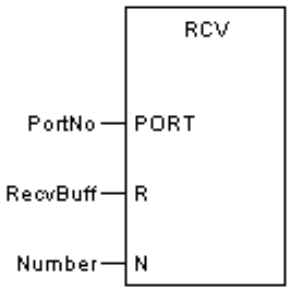
- Данный функциональный блок используется для приема данных в режиме свободного порта.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL RCV (PORT:=PortNo, T:=TranBuff, N:=Number)

Форма в ST:

RCV (PORT:=PortNo, T:=TranBuff, N:=Number)

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
PORT	PortNo	Номер последовательного порта (1 или 2).	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

T	RecvBuff	Приемный буфер, используемый для хранения полученных данных.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
N	Number	Количество байтов, которые необходимо передать.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

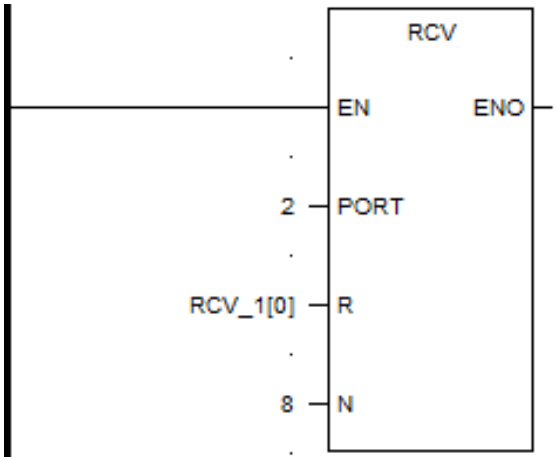
Состояние передачи

%SW0022(Состояние приема COM1) %SW0024(Состояние приема COM2)
0：при получении
1：получить желаемую длину
2：получить конечную отметку
3：получение истекло
4: интервалы между символами слишком длинные
5：больше максимального количества символов сообщения
8：Ошибка последовательного порта

Регистры системы передачи и приема CPU201-1101 и CPU401-1101 с 4 последовательными портами приведены в следующей таблице.

Номер последовательного порта	**COM1**	**COM2**	**COM3**	**COM4**
Регистр статуса передачи	%SW0021	%SW0023	%SW0041	%SW0043
Регистр статуса получения	%SW0022	%SW0024	%SW0042	%SW0044

ПРИМЕР:



Имя	Ввод	Тип данных	Размерность	Значение	Комментарий	Адрес	Объем
RCV_1	RCV_1	BYTE[]	100			%V00117	100

Описание: На схеме выше блок функции приема получает данные из последовательного порта 2 и сохраняет данные в массиве 8-байтовых регистров, начиная с RCV_1[0].

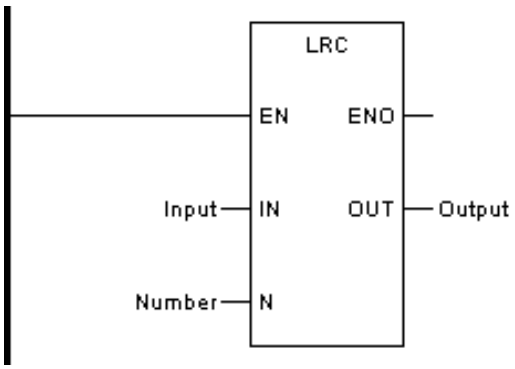
Расчет контрольного кода LRC - LRC

Описание функции

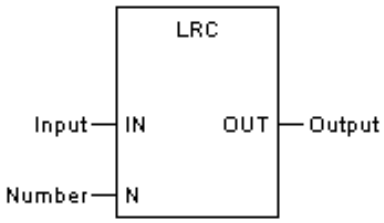
- Данный функциональный блок используется для расчета контрольного кода LRC.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL LRC (IN:=Вход, N:=Number, OUT=>Output)

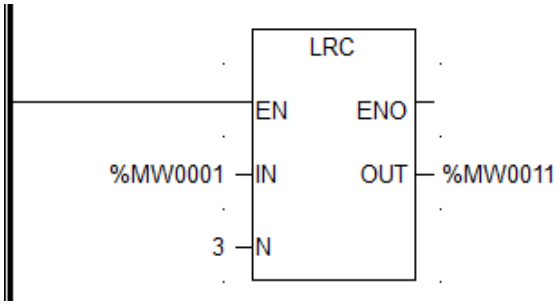
Форма в ST:

LRC (IN:=Вход, N:=Число, OUT=>Output)

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Рассчитываемая область хранения данных LRC	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
N	Number	Количество байт LRC, которое необходимо рассчитать	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Результат хранения LRC	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР. Вычислить данные из 3 байтов в области хранения данных LRC, начиная с %MW0001, и сохранить результат в %MW0011 и %MW0012.



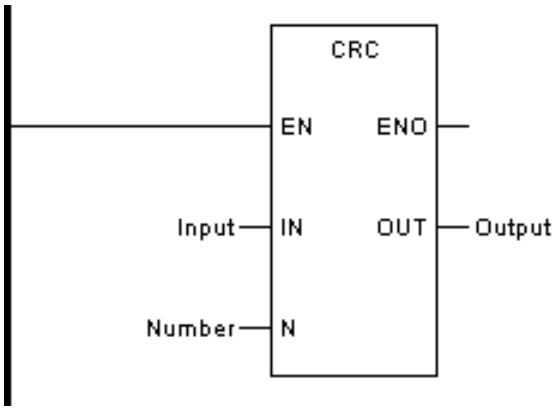
Расчет контрольного кода CRC - CRC

Описание функции

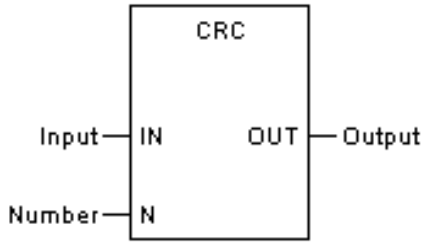
- Данный функциональный блок используется для вычисления контрольного кода CRC.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL CRC (IN:=Вход, N:=Number, OUT=>Output)

Форма в ST:

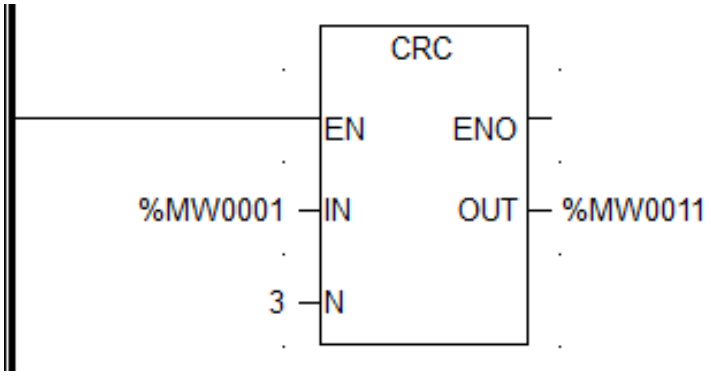
CRC (IN:=Вход, N:=Number, OUT=>Output)

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра

IN	Input	Область хранения данных CRC, подлежащая расчету	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
N	Number	Количество байтов CRC, которые необходимо рассчитать	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
OUT	Output	Результат хранения CRC	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР: Вычислить данные из 3 байтов в области хранения данных CRC, начиная с %MW0001, и сохранить результат в %MW0011 и %MW0012.



Чтение и запись данных MODBUS - MODRW

Описание функции

Этот функциональный блок используется для реализации функции чтения и записи данных стандартного главного протокола MODBUS, для автоматического анализа сообщений протокола MODBUS и для проверки CRC и длины данных. Пользователям нужно только заполнить адрес чтения-записи данных и данные функции, чтобы реализовать функцию связи. Вызов функционального блока требует триггера синхронизации, а периодический интервал — это период для чтения и записи данных с минимальным значением 50 мс. Интервал времени связи должен быть скорректирован в соответствии со временем ответа ведомого устройства во время связи, в противном случае данные могут быть прочитаны неправильно или не могут быть прочитаны.

Показатели коммуникационных состояний:

CW21 (состояние отправки COM1), SW23 (состояние отправки COM2)

0: в процессе передачи

1: успешная передача

2: сбой передачи

SW22 (состояние приема COM1), SW24 (состояние приема COM2)

0: в приеме

1: успешно получено

2: неисправность последовательного порта

3: тайм-аут приема

4: характерный интервал слишком длинный

5: больше максимального количества символов в сообщениях

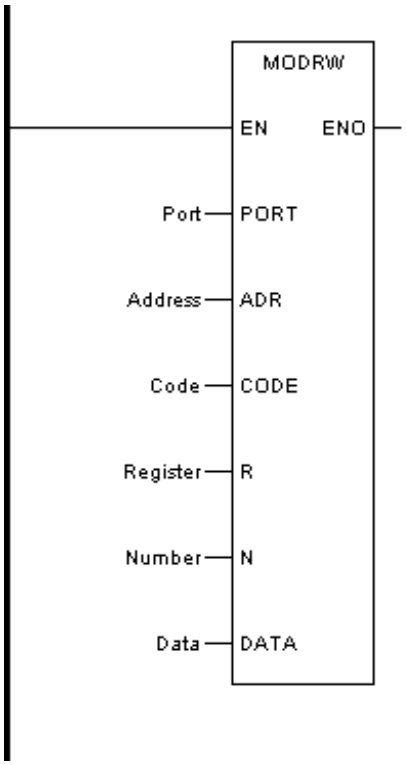
7: неправильное ответное сообщение

8: неправильное сообщение о запросе

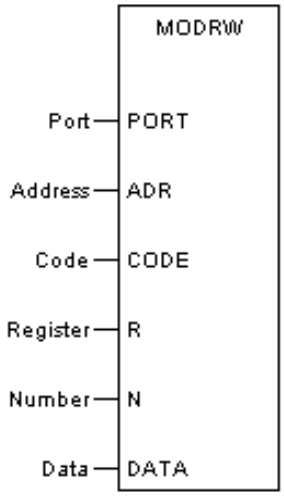
9: неправильная проверка

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL MODRW (PORT:=Порт, ADR:=Адрес, CODE:=Код, R:=Регистр, N:=Номер, DATA:=Данные)

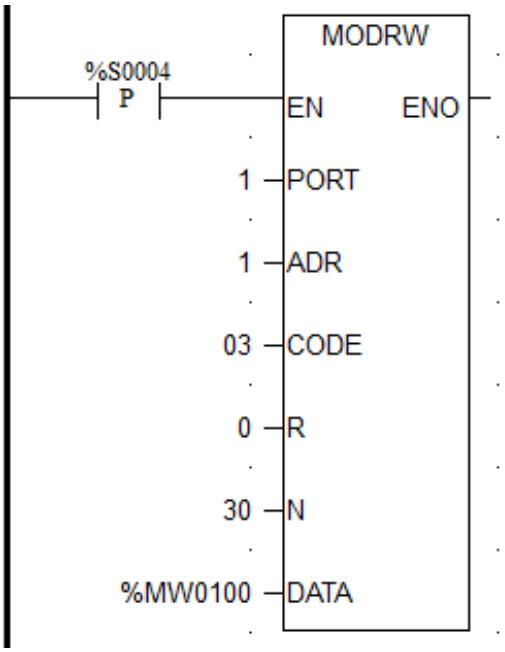
Форма в ST:

MODRW (PORT:=Порт, ADR:=Адрес, CODE:=Код, R:=Регистр, N:=Номер, DATA:=Данные);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
PORT	Port	Номер последовательного порта (1 ИЛИ 2), конкретное определение параметра, относящегося к конфигурации ЦП.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
ADR	Adress	Адрес ведомого устройства MODBUS в диапазоне от 1 до 255.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
CODE	Code	Стандартный функциональный код протокола MODBUS, в настоящее время поддерживающий следующие функциональные коды: 01, 02, 03, 04, 05, 06, 15, 16.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
R	Register	Адрес регистров данных.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Количество регистров чтения-записи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	Data	Буфер данных чтения-записи, хранящий данные для передачи и получения.	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	IW, QW, MW, NW, SW, I, Q, M, N, S, VAR

ПРИМЕР:



Описание: В приведенном выше примере показано, что каждую секунду последовательный порт 1 ЦП используется для считывания данных 30 последовательных регистров, начиная с 0, который поступает от ведомого устройства с адресом 1, с использованием кода функции 03. И считанные данные сохраняются в 30 последовательных регистрах, начиная с %MW100.

Примечания: этот функциональный блок не может использоваться в CPU401-1101 и CPU201-1101 и использовать интерфейс конфигурации MODBUS RTU master для завершения чтения и записи MODBUS. Между тем, все модули, которые могут конфигурировать MODBUS MASTER в модуле, не могут использовать этот функциональный блок.

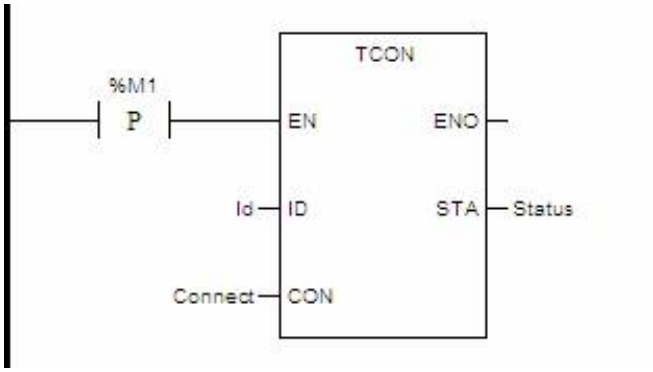
Активировать соединение Ethernet - TCON

Описание функции

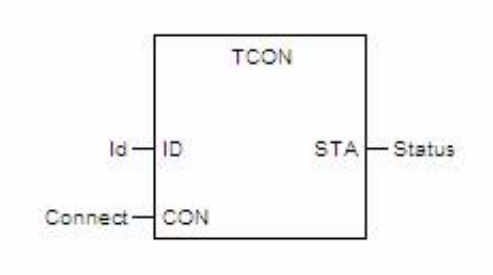
- Данный функциональный блок используется для активации Ethernet-соединения.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL TCON (ID:=Id, CON:=Подключение, STA=>Статус)

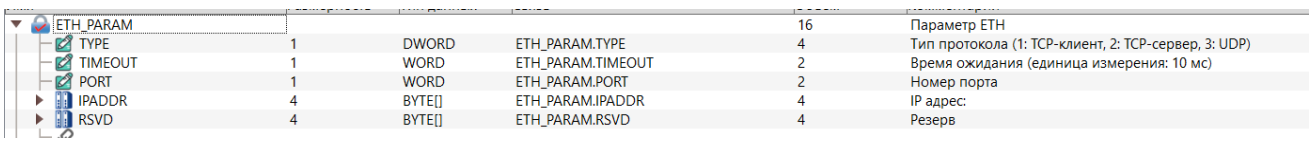
Форма на языке ST:

TCON (ID:=Id, CON:=Подключение, STA=>Статус);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
ID	Id	Id подключения, который является уникальным во всем проекте.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
CON	Connect	Параметр Ethernet-подключения.	ETH_PARAM	VAR
STA	Status	Состояние вывода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Описание CON



Для TCP-клиента: TYPE=1; TIMEOUT — это тайм-аут соединения/передачи/приема, с единицей измерения 10 мс. IPADDR указывает IP-адрес однорангового узла (самый младший байт идет первым). PORT указывает номер порта однорангового узла.

Для TCP-сервера: TYPE=2; TIMEOUT — тайм-аут передачи/приема, единица измерения 10 мс. PORT — номер локального порта; другие можно игнорировать.

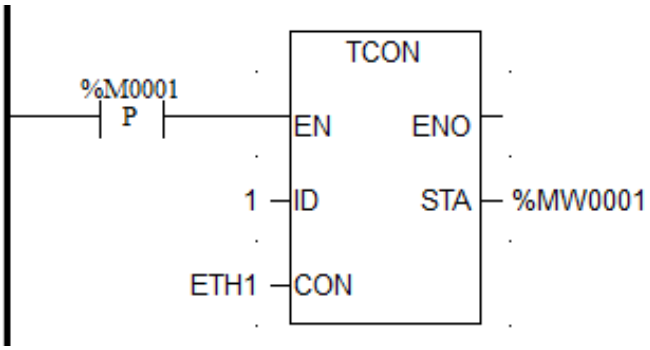
Для UDP: TYPE=2; TIMEOUT — тайм-аут передачи/приема, единица измерения 10 мс. PORT — номер локального порта; другие можно игнорировать.

Описание STA:

- 0x01：В работе
- 0x02：успешная операция
- 0x22：ID находится в соединении.
- 0x25: Тайм-аут передачи/приема
- 0x28：Слишком частая отправка/получение
- 0x81：Внутренняя ошибка
- 0x83：ID не подключен.
- 0x84：Неправильное подключение
- 0x86：Неверный тип
- 0x87：Достигнуто максимальное количество подключений.
- 0x88：Подключение происходит слишком часто.
- 0x89：Неправильная передача/прием

Для Id STA больше 0x80 необходимо повторно активировать Ethernet-соединение, вызвав TCON для реализации отправки и получения данных.

ПРИМЕР: (ETH - это определяемый пользователем тип переменной ETH_PARAM в таблице переменных)



Описание: Непосредственно после включения %M0001, происходит подключение Ethernet с ID 1.

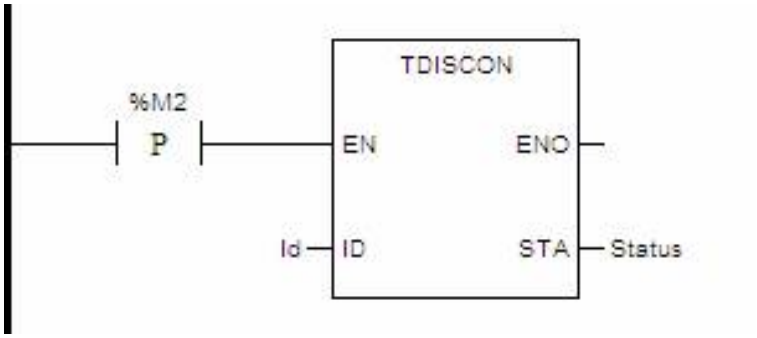
Отключение Ethernet-соединения - TDISCON

Описание функции

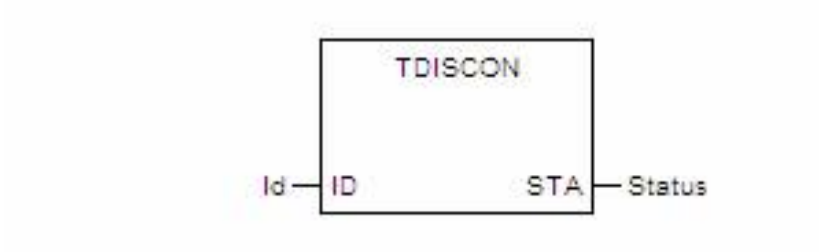
- Данный функциональный блок используется для отключения Ethernet-соединения.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL TDISCON (ID:=Id, STA=>Статус)

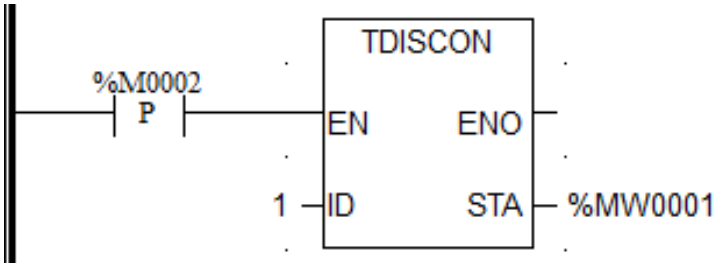
Форма на языке ST:

TDISCON (ID:=Id, STA=>Статус);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
ID	Id	Id соединения, который соответствует Id в TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
STA	Status	Состояние вывода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: Сразу после включения %M0002 Ethernet с Id 1 отключается.

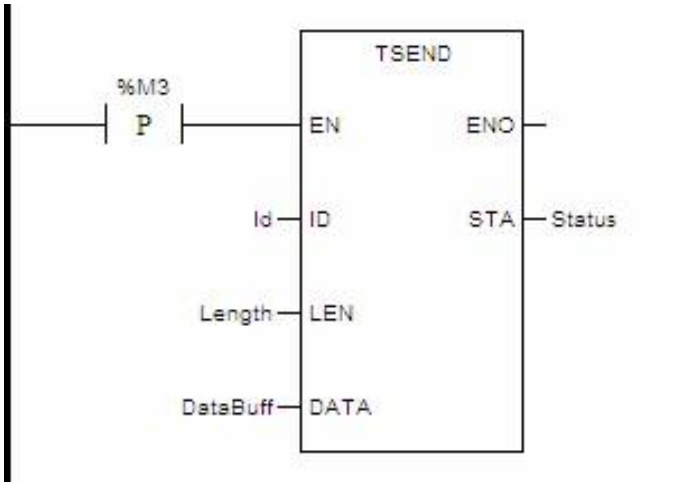
Отправка данных TCP - TSEND

Описание функции

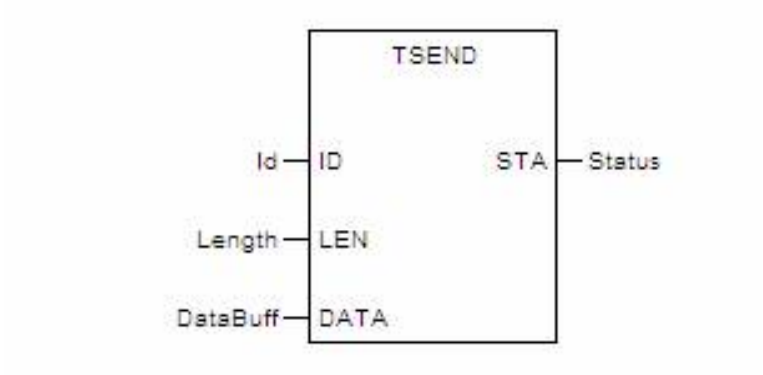
- Данный функциональный блок используется для отправки данных TCP.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL TSEND (ID:=Id, LEN:=Length, Data:=DataBuff, STA=>Status)

Форма на языке ST:

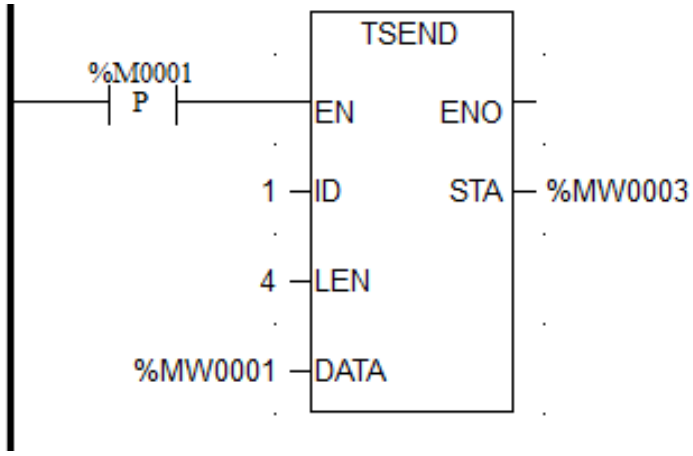
TSEND (ID:=Id, LEN:=Длина, Data:=DataBuff, STA=>Статус);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
ID	Id	ID соединения, который совпадает с ID в TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
LEN	Length	Длина отправленных данных в байтах. Максимальное значение — 8192 байта.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	DataBuff	Буфер отправки данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

STA	Status	Состояние вывода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: Сразу после подключения %M0001 функциональный блок TSEND отправляет данные, Id отправки равен 1, а длина байтов равна 4, т. е. данные отправляются в %MW0001 и %MW0002, а состояние вывода отображается в %MW0003.

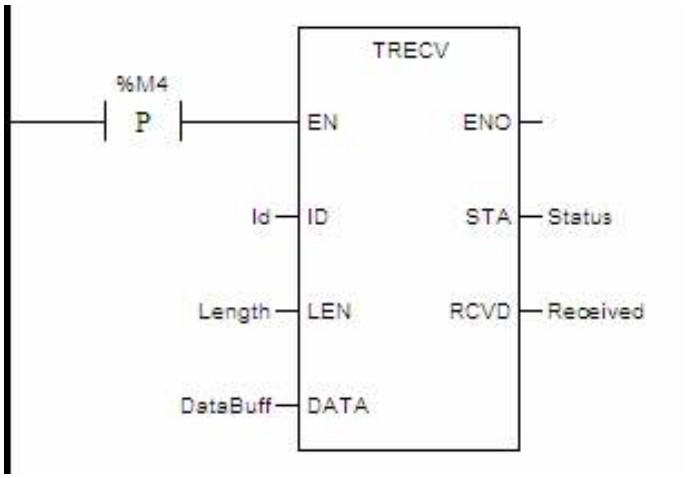
Прием данных TCP - TRECV

Описание функции

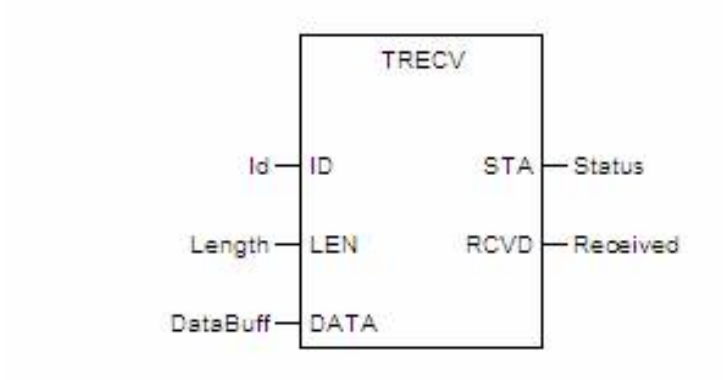
- Данный функциональный блок используется для приема данных TCP.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL TRECV (ID:=Id, LEN:=Length, Data:=DataBuff, STA=>Status, RCVD=>Received)

Форма на языке ST:

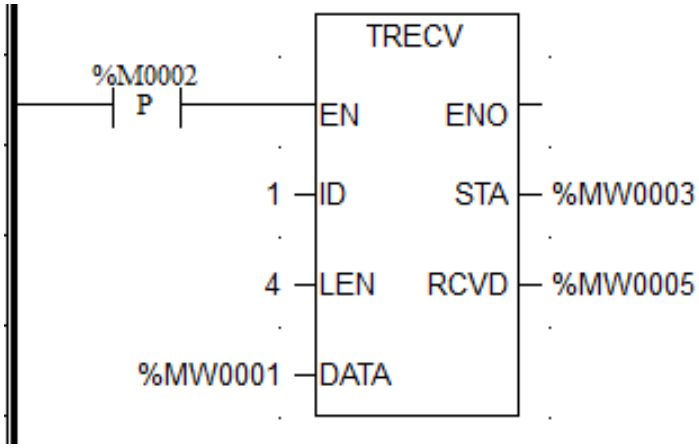
TRECV (ID:=Id, LEN:=Length, Data:=DataBuff, STA=>Status, RCVD=>Received);

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
ID	Id	ID соединения, который совпадает с ID в TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
LEN	Length	Длина отправленных данных в байтах. Максимальное значение — 8192 байта.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	DataBuff	Буфер отправки данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

STA	Status	Состояние вывода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
RCVD	Received	Длина полученных данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: Сразу после подключения %M0002 функциональный блок TRECV получает данные, Id приема равен 1, а длина байтов равна 4, т.е. сохраняет полученные данные в %MW0001 и %MW0002 и отображает состояние вывода в %MW0003. Количество полученных байтов сохраняется в %MW0005.

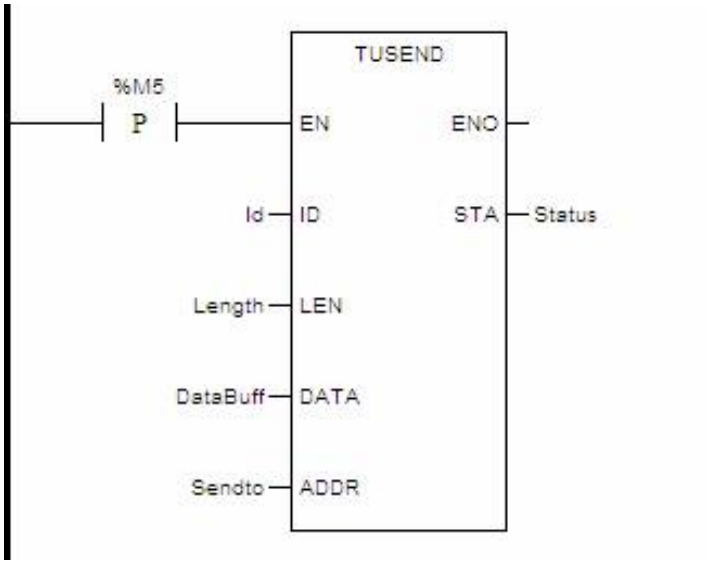
Отправка данных UDP - TUSEND

Описание функции

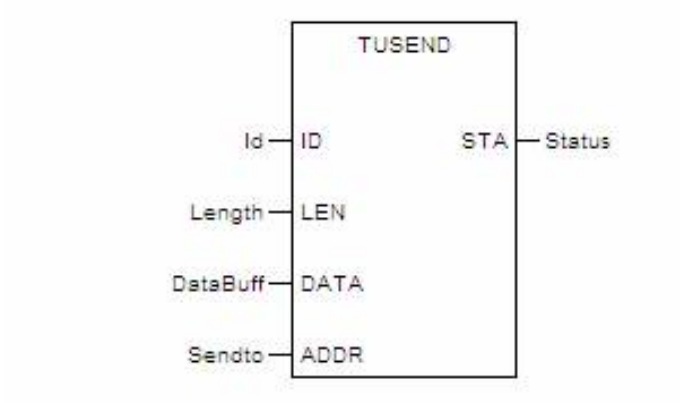
- Данный функциональный блок используется для отправки данных UDP.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL TUSEND (ID:=Id, LEN:=Length, Data:=DataBuff, ADDR:=Sendto, STA=>Status)

Форма на языке ST:

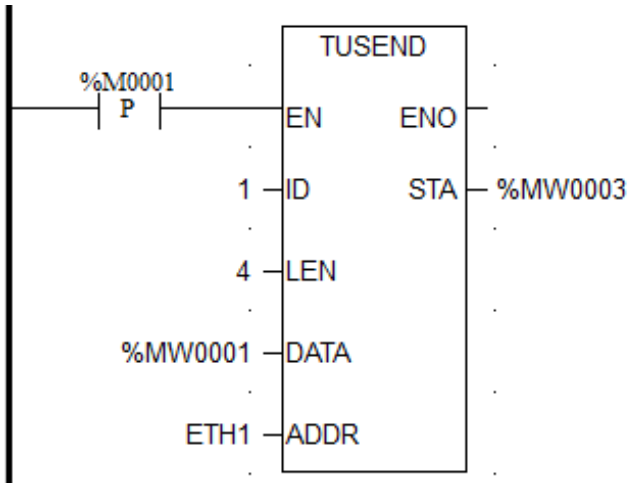
TUSEND (ID:=Id, LEN:=Длина, Данные:=DataBuff, ADDR:=Отправить, STA=>Статус);

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
ID	Id	ID соединения, который совпадает с ID в TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
LEN	Length	Длина отправленных данных в байтах. Максимальное значение — 1472 байта.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

DATA	DataBuff	Буфер отправки данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
ADDR	Status	Параметр Ethernet однорангового узла. IPADDR — это IP-адрес однорангового узла, а PORT — это номер порта одноранговой узла.	ETH_PARAM	VAR
STA	Received	Статус вывода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: Сразу после подключения %M0001 функциональный блок TUSEND отправляет данные, Id отправки равен 1, а длина байтов равна 4, т. е. данные отправляются в %MW0001 и %MW0002. ETH1 — это определяемый пользователем тип данных переменной ETH_PARAM. IP-адрес и PORT можно задать в таблице переменных. Состояние вывода отображается в %MW0003.

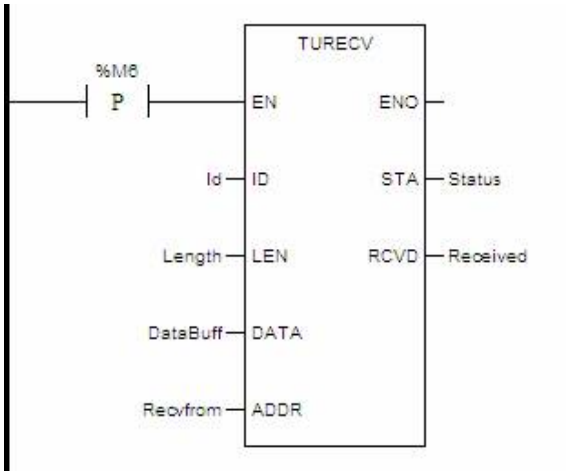
Получение данных UDP - TURECV

Описание функции

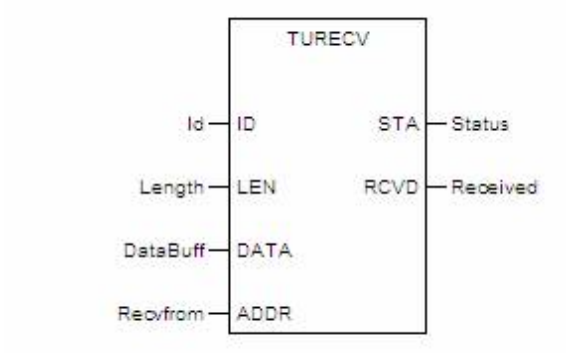
- Данный функциональный блок используется для приема данных UDP.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в IL:

CAL TURECV (ID:=Id, LEN:=Length, Data:=DataBuff, ADDR:=Recvfrom, STA=>Status, RCVD=>Received)

Форма на языке ST:

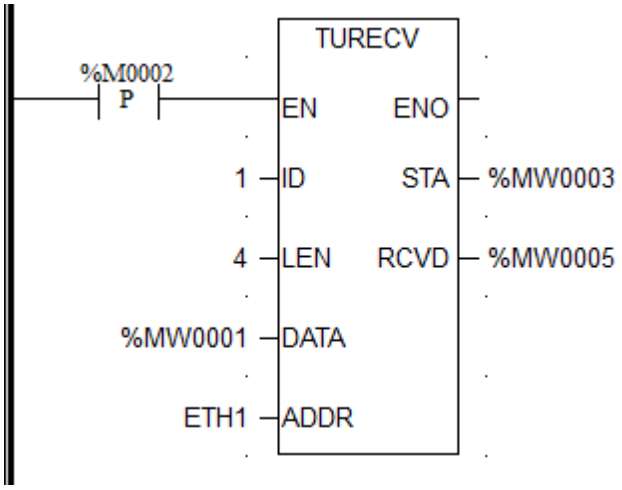
TURECV (ID:=Id, LEN:=Length, Data:=DataBuff, ADDR:=Recvfrom, STA=>Status, RCVD=>Received);

Описание параметра

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
ID	Id	ID соединения, который совпадает с ID в TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

LEN	Length	Длина отправленных данных в байтах. Максимальное значение — 1472 байта.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	DataBuff	Буфер отправки данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
ADDR	Recvfrom	После получения заполните параметр Ethernet одноранго-вого узла. IPADDR — IP-адрес одноранго-вого узла, а PORT — номер порта однорангового узла.	ETH_PARAM	VAR
STA	Received	Статус вывода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: Сразу после подключения %M0002 функциональный блок TURECV получает данные, Id приема равен 1, а длина байтов равна 4, т.е. полученные данные сохраняются в %MW0001 и %MW0002. ETH1 — это определяемый пользователем тип данных переменной ETH_PARAM. IP-адрес и PORT можно задать в таблице переменных. Состояние вывода отображается в %MW0003, а количество полученных байтов сохраняется в %MW0005.

Таймеры

Таймер используется для установки времени в соответствии с требованиями пользователя. Запустите отсчет времени при определенных условиях и выведите 1 после отсчета времени. Хотя один таймер может использоваться в программе много раз, текущее значение таймера с тем же порядковым номером остается тем же. Поэтому, если нет необходимости, каждый таймер лучше использовать только один раз, а таблица регистров в программном обеспечении указывает время использования каждого таймера.

Тда	Описание
TON	Таймер задержки включения (секунда)
TOF	Таймер задержки выключения (секунда)
TP	Импульсный таймер (секунда)
TMR	Таймер задержки включения со сбросом (секунда)
TON_MS	Таймер задержки включения (миллисекунда)
TOF_MS	Таймер задержки выключения (миллисекунда)
TP_MS	Импульсный таймер (миллисекунда)
TMR_MS	Таймер задержки включения со сбросом (миллисекунда)
TON_M	Таймер задержки включения (минута)
TOF_M	Таймер задержки выключения (минута)
TP_M	Импульсный таймер (минута)
TMR_M	Таймер задержки включения со сбросом (минута)
TON_H	Таймер задержки включения (час)
TOF_H	Таймер задержки выключения (час)
TP_H	Импульсный таймер (час)
TMR_H	Таймер задержки включения со сбросом (час)

Таймер задержки включения - TON

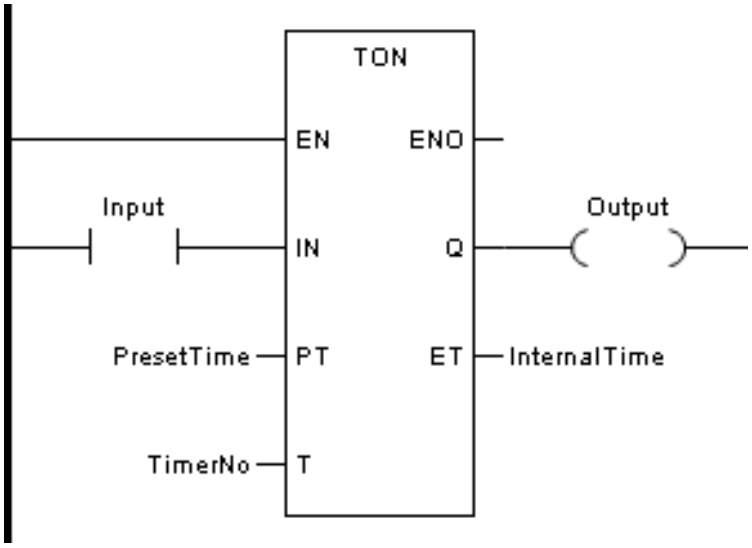
Описание функции

- Этот функциональный блок используется для задержки включения. Когда этот функциональный блок вызывается в первый раз, начальное состояние ET равно 0
- Функциональные блоки одной и той же функции

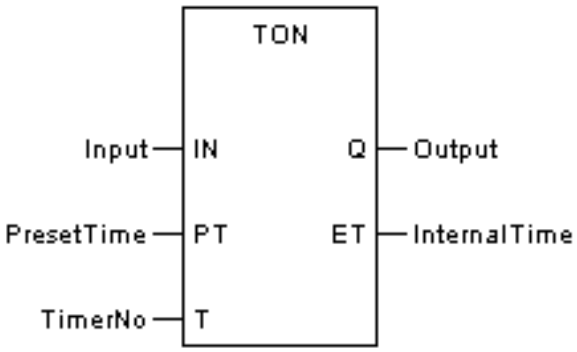
- TON_MS Таймер задержки включения (миллисекунда)
- TON_M Таймер задержки включения (минута)
- TON_H Таймер задержки включения (час)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL TON (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output, ET=>Внтуреннее время) (параметр ET можно удалить)

Форма на языке ST:

TON (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output, ET=>Внутреннее время);

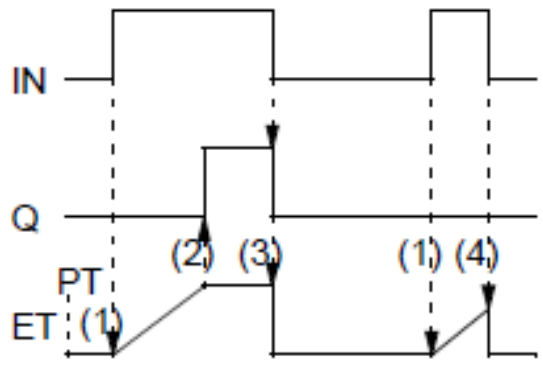
(Параметр ET можно удалить)

Описание параметров

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
IN	Input	Задержка включения	BOOL	CONSTANT, I, Q, M, N, S,VAR
PT	PresetTime	Сброс значения таймера	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
T	TimerNo	Номер таймера		T
Q	Output	Выход	BOOL	Q, M, N, VAR
ET	InternalTime	Текущее значение таймера.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Диаграмма последовательности

Форма задержки включения TON:



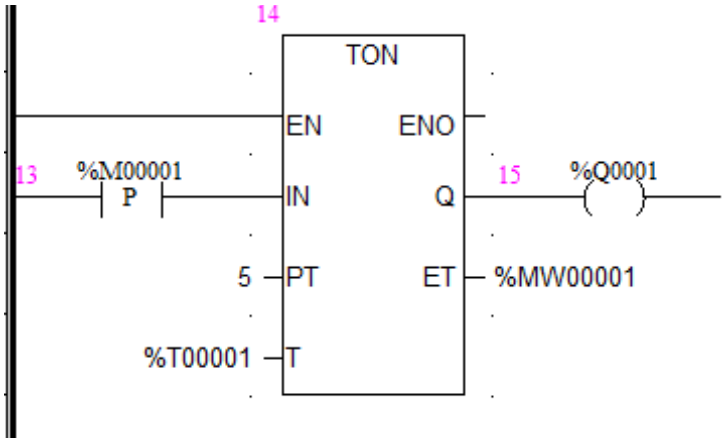
(1) ЕСЛИ IN=1, начинается внутреннее время ET.

(2) ЕСЛИ внутреннее время достигает значения PT, то Q=1.

(3) ЕСЛИ IN=0, то Q=0 и внутреннее время останавливается или сбрасывается.

(4) ЕСЛИ IN=0 до того, как внутреннее время достигнет значения PT, то внутреннее время останавливается или сбрасывается, и Q не изменится на 1.

ПРИМЕР:



Описание: Когда %M0001 изменяется с 0 на 1 и начинается отсчет времени задержки включения, через 5 с %Q0001 разрешит текущий проход, а %MW0001 отобразит текущее значение таймера.

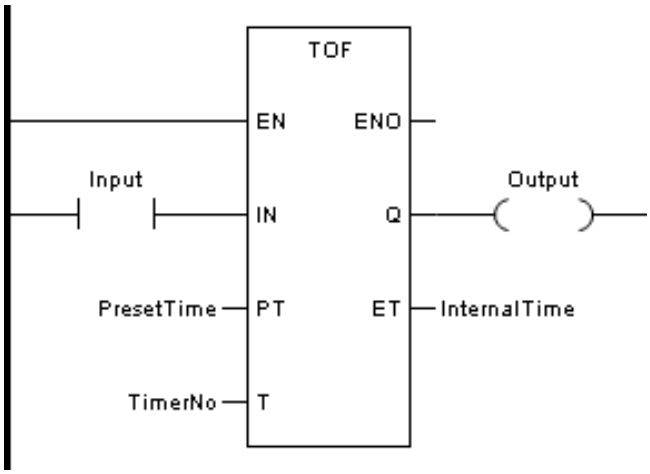
Таймер задержки выключения - TOF

Описание функции

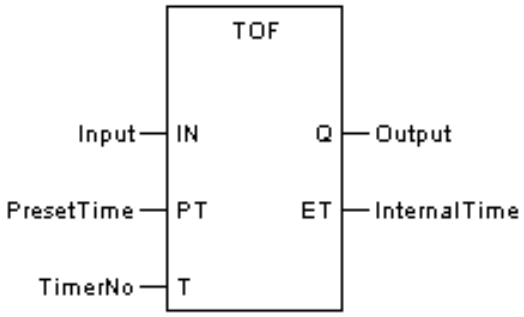
- Этот функциональный блок используется для отсчета времени задержки выключения. Когда этот функциональный блок вызывается в первый раз, начальное состояние ET равно 0
- Функциональные блоки одной и той же функции
  - TOF_MS Таймер задержки выключения (миллисекунда)
  - TOF_M Таймер задержки выключения (минута)
  - TOF_H Таймер задержки выключения (час)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL TOF (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output, ET=>Внтуреннее время) (параметр ET можно удалить)

Форма на языке ST:

TOF (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output, ET=>Внутреннее время);

(Параметр ET можно удалить)

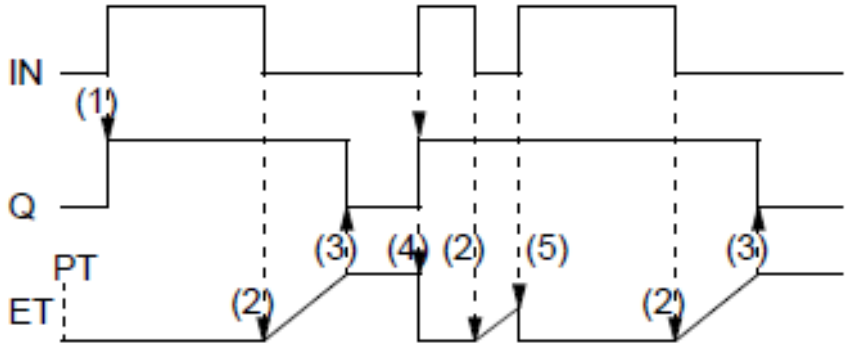
Описание параметров

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Задержка включе-ния	BOOL	CONSTANT, I, Q, M, N, S, VAR
PT	PresetTime	Сброс значения таймера	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
T	TimerNo	Номер таймера		T

Q	Output	Выход	BOOL	Q, M, N, VAR
ET	InternalTime	Текущее значение таймера.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Диаграмма последовательности

Форма задержки выключения TOF:



(1) ЕСЛИ IN=1, Q=1.

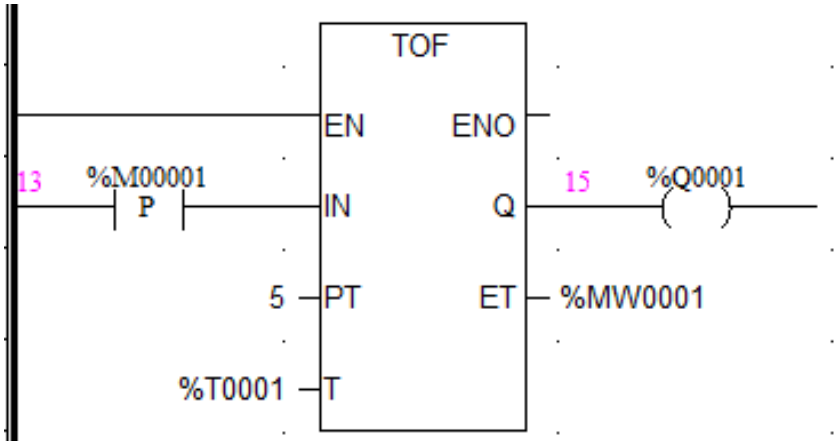
(2) ЕСЛИ внутреннее время достигает значения PT, то Q=0.

(3) ЕСЛИ IN=0, то начинается внутреннее время ET.

(4) ЕСЛИ IN=1, то Q=1 и внутреннее время останавливается или сбрасывается.

(5) ЕСЛИ IN=1 до того, как внутреннее время достигнет значения PT, то внутреннее время останавливается и сбрасывается, а Q не будет сброшен до 0.

ПРИМЕР:



Описание: Когда %M0001 изменяется с 1 на 0 и начинается отсчет времени задержки выключения, через 5 с %Q0001 отключится, а %MW0001 отобразит текущее значение таймера.

Примечание: Если %M0001=1, %Q0001 немедленно включит ток, и таймер не запустится.

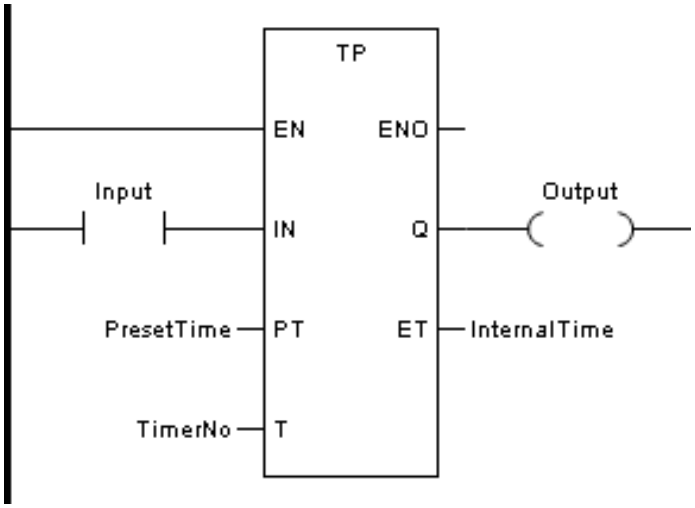
Импульсный таймер - TP

Описание функции

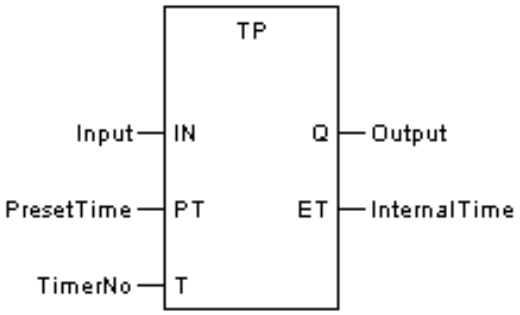
- Этот функциональный блок используется для генерации импульса с определенным временем. Когда этот функциональный блок вызывается в первый раз, начальное состояние ET равно 0.
- Функциональные блоки одной и той же функции
  - TP_MS Импульсный таймер (миллисекунда)
  - TP_M Импульсный таймер (минута)
  - TP_H Импульсный таймер (секунда)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL TP (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output, ET=>Внтуреннее время) (параметр ET можно удалить)

Форма на языке ST:

TP (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output, ET=>Внутреннее время);

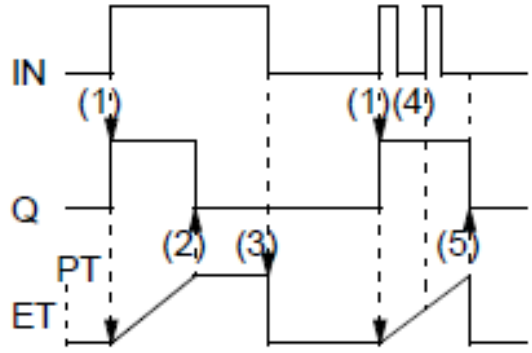
(Параметр ET можно удалить)

Описание параметров

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Задержка включения	BOOL	CONSTANT, I, Q, M, N, S, VAR
PT	PresetTime	Сброс значения таймера	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
T	TimerNo	Номер таймера		T
Q	Output	Выход	BOOL	Q, M, N, VAR
ET	InternalTime	Текущее значение таймера.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Диаграмма последовательности

Форма импульса синхронизации TP:



(1) ЕСЛИ IN=1, Q=1 и начинается внутреннее время ET.

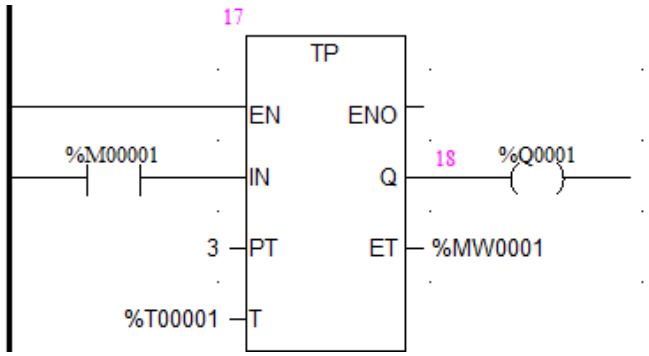
(2) ЕСЛИ внутреннее время достигает значения PT, то Q=0.

(3) ЕСЛИ IN=0, то внутреннее время останавливается или сбрасывается.

(4) Если внутреннее время не достигло значения PT, то на внутреннее время не влияют время на IN.

(5) ЕСЛИ IN=1 до того, как внутреннее время достигнет значения PT, то внутреннее время останавливается и сбрасывается, а Q=0.

ПРИМЕР:



Описание: Когда %M0001 изменяется с 0 на 1 и начинается отсчет времени импульса, через 3 с %Q0001 отключается и сбрасывается, а %MW0001 отображает текущее значение таймера.

Примечание: Если %M0001=1, %Q0001 немедленно активируется.

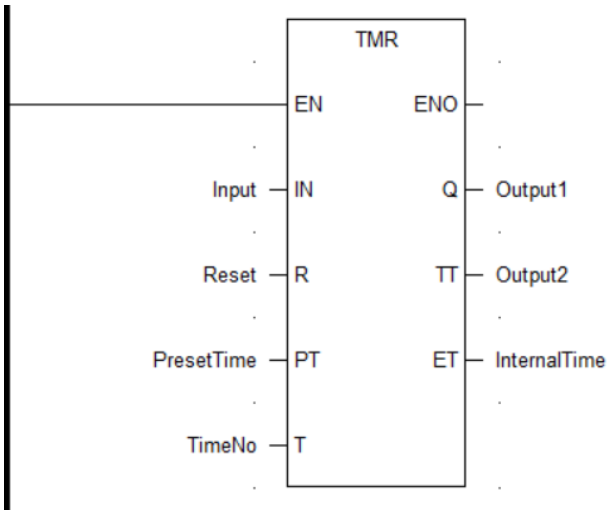
Таймер задержки выключения с функцией сброса - TMR

Описание функции

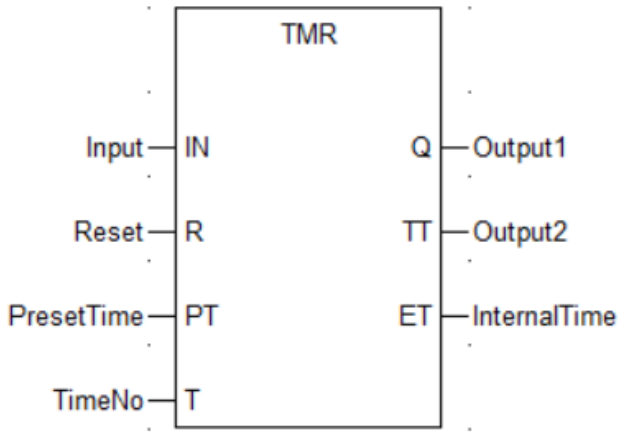
- Этот функциональный блок используется для отсчета времени задержки включения с функцией сброса. Когда этот функциональный блок вызывается в первый раз, начальное состояние ET равно 0. Когда вход установлен на 1, таймер начинает отсчет времени и OUTPUT2 = 1. При достижении времени сброса PT, Output1 = 1 и Output2 = 0. После сброса перезапустите отсчет времени.
- Функциональные блоки одной и той же функции
    - TMR_MS Время задержки включения с функцией сброса (миллисекунды)
    - TMR_M Время задержки включения с функцией сброса (минуты)
    - TMR_H Время задержки включения с функцией сброса (время)

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL TMR (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output1, TT=>Output2，ET=>Внутреннее время)；（Параметр ET можно удалить）

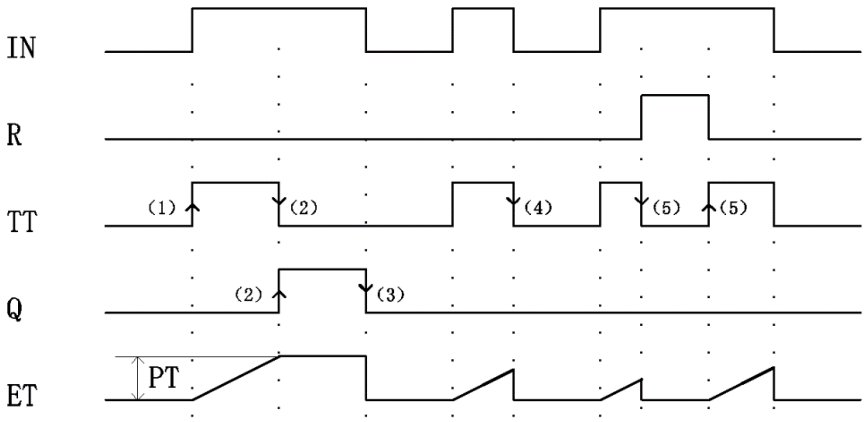
Форма на языке ST:

TMR (IN:=Вход, PT:=Предустановленное время, T:=Номер таймера, Q=>Output1, TT=>Output2，ET=>Внтуреннее время); (параметр ET можно удалить)

Описание параметра

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
IN	Input	Задержка включе-ния	BOOL	CONSTANT, I, Q, M, N, S, VAR
R	Reset	Сброс	BOOL	CONSTANT, I, Q, M, N, S, VAR
PT	PresetTime	Сброс значения таймера	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
T	TimerNo	Номер таймера		T
Q	Output1	Выход 1	BOOL	Q, M, N, VAR
TT	Output2	Выход 2	BOOL	Q, M, N, VAR
ET	InternalTime	Текущее значе-ние таймера.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Диаграмма последовательности



(1) ЕСЛИ IN=1, то одновременно начинается внутреннее время ET и TT=1.

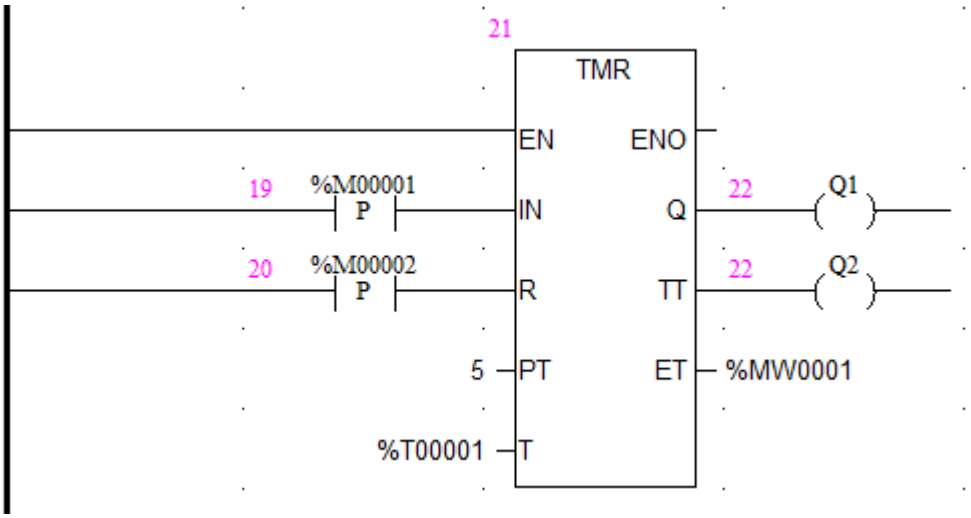
(2) ЕСЛИ внутреннее время достигает значения PT, то Q=1 и TT=0.

(3) ЕСЛИ IN=0, то Q=0 и внутреннее время останавливается и сбрасывается.

(4) ЕСЛИ внутреннее время достигает значения PT до того, как IN=0, то внутреннее время останавливается или сбрасывается, пока Q не изменится на 1.

(5) ЕСЛИ R=1, то каким бы ни было значение IN, TT, Q, ET все обращаются в 0. Когда R=0, следуйте шагам выше и начните снова. Но TT следует за изменением IN, пока ET не достигнет значения PT, Q станет 1, а T станет 0.

ПРИМЕР:



Описание: Когда %M1=1, T1 запускается, а Q2 немедленно устанавливает 1; через 5 с Q1 устанавливает 1, а Q2 устанавливает 0. %MW0001 отображает текущее значение. %M2 — это функция сброса.

Счётчики

Счетчик — это функциональный блок, который подсчитывает назначенные импульсы и выдает 1 при достижении заданного значения.

Тип	Описание
CTU	Подсчёт: начальное значение равно 0. Каждый раз, когда на вход поступает импульс с восходящим фронтом, к текущему значению счетчика добавляется 1. Когда текущее значение больше или равно предустановленному значению, счетчик выдает 1. Когда сброс равен 1, текущее значение счетчика сбрасывается до 0.
CTD	Обратный отсчет: начальное значение — это предустановленное значение. Каждый раз, когда на вход поступает импульс с нарастающим фронтом, текущее значение счетчика уменьшается на 1. Когда текущее значение уменьшается до 0, счетчик выводит 1. Когда конец нагрузки равен 1, текущее значение счетчика сбрасывается до предустановленного значения.
CTUD	Счёт вверх/вниз: выполняет функцию одновременного сложения и вычитания счётчиков.

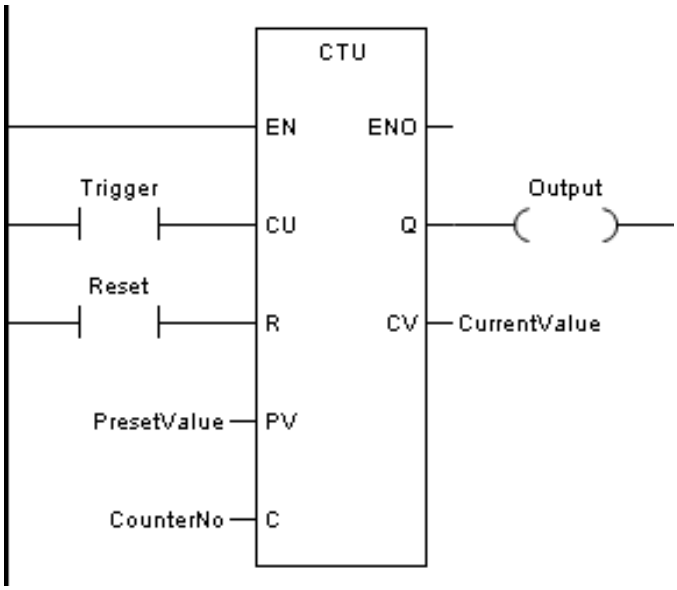
Счётчик прямого отсчёта - CTU

Описание функции

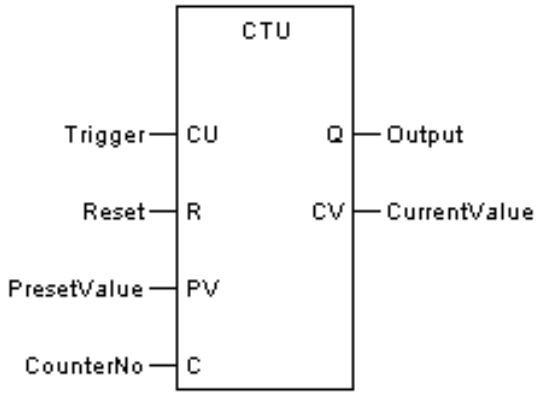
- Этот функциональный блок используется для подсчета.
- Сигнал «1» на входе R назначит 0 для выхода CV. Каждый раз, когда вход CU изменяется с 0 на 1, CV увеличивается на 1. Когда CV ≥ PV, Q выдает 1.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL CTU (CU:=Trigger, R:=Reset, PV:=PresetValue, C:=CounterNo, Q=>Output, CV=>CurrentValue) (параметр CV можно удалить)

Форма на языке ST:

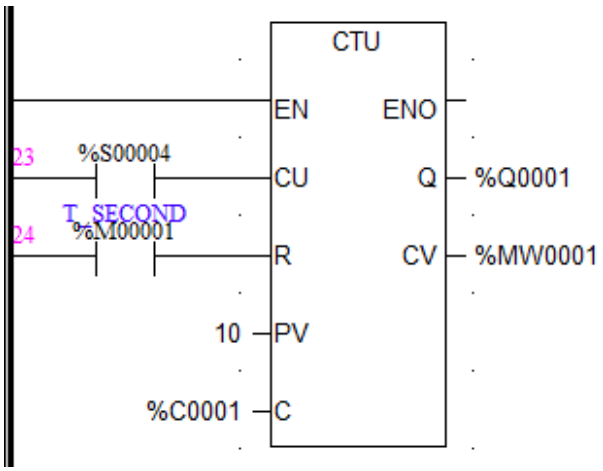
CTU (CU:=Trigger, R:=Reset, PV:=PresetValue, C:=CounterNo, Q=>Output, CV=>CurrentValue); (параметр CV можно удалить)

Описание параметров

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
CU	Trigger	Вход триггера	BOOL	CONSTANT, I, Q, M, N, S, VAR
R	Reset	Сброс	BOOL	CONSTANT, I, Q, M, N, S, VAR

PV	PresetValue	Предустанов-ленное значение счетчика	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
C	CounterNo	Номер счётчика.		C
Q	Output	Выход	BOOL	Q, M, N, VAR
CV	CurrentValue	Текущее значе-ние счетчика	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: %S0004 — это системный секундный таймер. Функция счетчика подсчета выше на схеме — добавлять 1 каждую секунду. Когда %MW0001≥10, т.е. через 10 с, %Q0001=1, и в это время, если %M0001=1, счетчик можно сбросить, т.е. %MW0001=0, %Q0001=0.

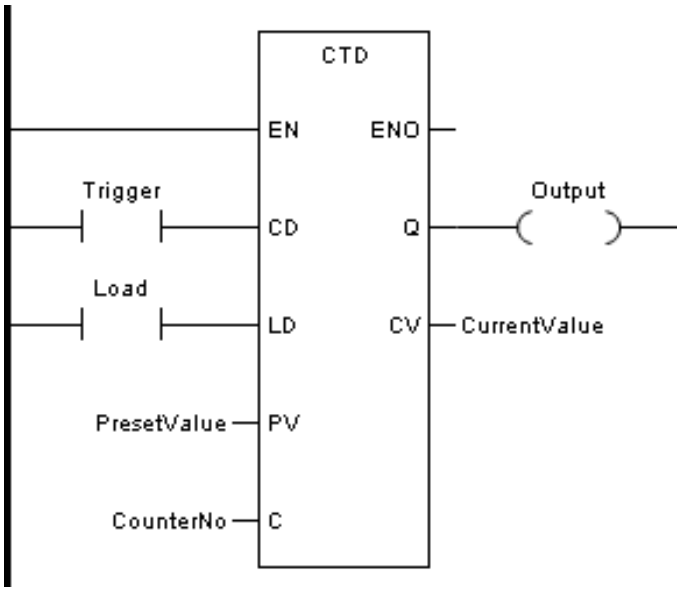
Счетчик обратного отсчета - CTD

Описание функции

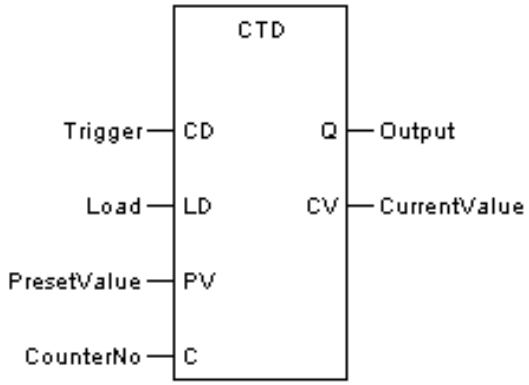
- Этот функциональный блок может вести обратный отсчет.
- Сигнал «1» на входе LD назначит значение PV для выхода CV. Каждый раз, когда вход CD изменяется с 0 на 1, CV уменьшается на 1. Когда CV≤0, Q выдает 1.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL CTD (CD:=Trigger, LD:=Load, PV:=PresetValue, C:=CounterNo, Q=>Output, CV=>CurrentValue) (параметр CV можно удалить)

Форма на языке ST:

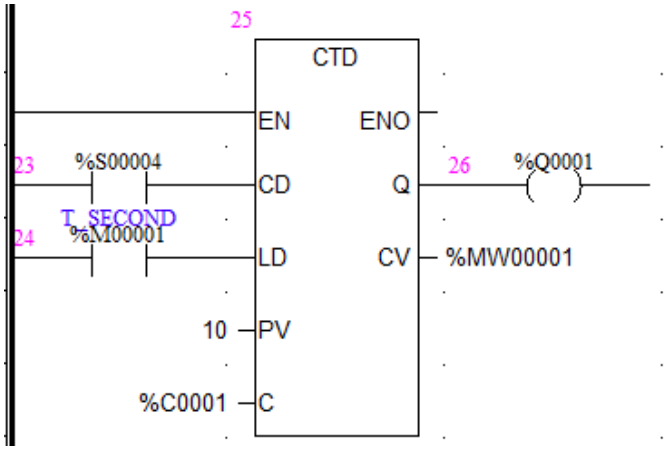
CTD (CD:=Trigger, LD:=Load, PV:=PresetValue, C:=CounterNo, Q=>Output, CV=>CurrentValue); (параметр CV можно удалить)

Описание параметров

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
CD	Trigger	Вход триггера	BOOL	CONSTANT, I, Q, M, N, S, VAR

LD	Load	Загрузка начального значения	BOOL	CONSTANT, I, Q, M, N, S, VAR
PV	PresetValue	Предустановленное значение счетчика	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
C	CounterNo	Номер счётчика.		C
Q	Output	Выход	BOOL	Q, M, N, VAR
CV	CurrentValue	Текущее значение счетчика	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: %S0004 — это системный секундный таймер. Функция счетчика обратного отсчета выше на схеме — уменьшать на 1 каждую секунду. Когда %MW0001≤0, т.е. через 10 с, %Q0001=1, а если %M0001=1, счетчик можно сбросить, т.е. %MW0001=10, %Q0001=0.

Счетчик прямого/обратного счета - CTUD

Описание функции

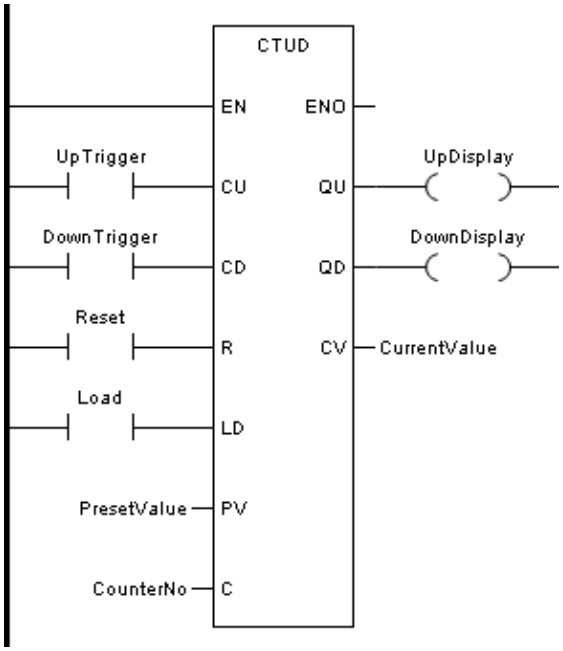
- Этот функциональный блок используется для счета вверх/вниз.
- Сигнал «1» на входе R назначит 0 для выхода CV. Сигнал «1» на входе LD назначит предустановленное значение для выхода CV. Каждый раз, когда вход CD

изменяется с 0 на 1, CV уменьшается на 1. Когда CV≤0, Q выдает 1. Каждый раз, когда вход CU изменяется с 0 на 1, CV увеличивается на 1.

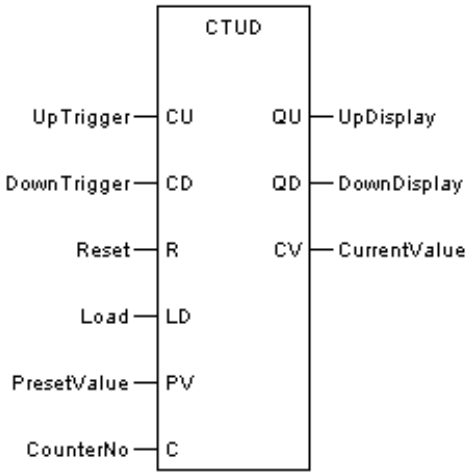
- Если на входах R и LD присутствует сигнал 1, приоритет имеет вход R.
- Когда CV ≥ PV, QU=1.
- Когда CV ≤ 0, QD=1.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL CTUD (CU:=UpTrigger, CD:=DownTrigger, R:=Reset, LD:=Load, PV:=PresetValue, C:=CounterNo, QU=>UpDisplay, QD=>DownDisplay, CV=>CurrentValue)

(Параметр CV можно удалить)

Форма на языке ST:

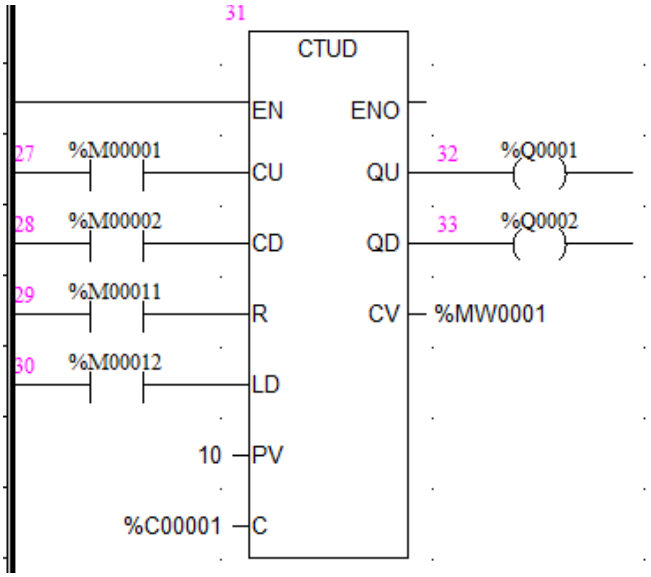
CTUD (CU:=UpTrigger, CD:=DownTrigger, R:=Reset, LD:=Load, PV:=PresetValue, C:=CounterNo, QU=>UpDisplay, QD=>DownDisplay, CV=>CurrentValue);

(Параметр CV можно удалить)

Описание параметров

Диаграмма	**Параметр**	**Описание**	**Тип данных**	**Тип регистра**
CU	UpTrigger	Входной триггер для подсчета	BOOL	CONSTANT, I, Q, M, N, S, VAR
CD	DownTrigger	Входной триггер для обратного отсчета	BOOL	CONSTANT, I, Q, M, N, S, VAR
R	Reset	Сброс	BOOL	CONSTANT, I, Q, M, N, S, VAR
LD	Load	Загрузка начального значения	BOOL	CONSTANT, I, Q, M, N, S, VAR
PV	PresetValue	Предустановленное значение счетчика	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
C	CounterNo	Номер счётчика.		C
QU	UpDisplay	Дисплей для подсчета	BOOL	Q, M, N, VAR
QD	DownDisplay	Дисплей для обратного отсчета	BOOL	Q, M, N, VAR
CV	CurrentValue	Текущее значение счетчика	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

ПРИМЕР:



Описание: Каждый раз, когда %M0001 переходит от 0 к 1, запускается счетчик прямого счета, и CV присваивается значение 0. И каждый раз, когда происходит изменение %MW0001 от 0 к 1, значение CV будет прибавляться к 1. Когда CV≥PV, %Q0001 выводит 1, и счетчик прямого счета останавливается. Когда проводится %M0011, счетчик прямого счета сбрасывается, CV=0 и %Q0001=0.

Каждый раз, когда %M0002 переходит от 0 к 1, запускается счетчик обратного отсчета, и CV присваивается значению PV. И каждый раз, когда происходит изменение %MW0002 от 0 к 1, значение CV будет уменьшаться на 1. Когда CV≤0, %Q0001 выводит 1, и счетчик обратного отсчета останавливается. Когда выполняется %M0012, счетчик обратного отсчета перезагружает значение PV, в это время CV=PV, и %Q0001 сбрасывается.

%M0001 и %M0002 не могут быть выполнены одновременно, т. е. прямой и обратный отсчет не могут быть выполнены одновременно. Если %M0011 и %M0012 выполняются одновременно, сброс счетчика прямого отсчета будет иметь приоритет.

Управление

Блоки функций управления в основном используются для завершения работы функционального блока, включая следующие инструкции:

Тип	Описание
CALL	Вызвать: вызов подпрограммы
EXEC	Выполнить: выполнить назначенную программу SCC
KILL	Прекратить: прекратить выполнение программы SCC
LOCK	Блокировать: Блокировка назначенной программы SCC
UNLOCK	UNLOCK: Разблокировать назначенную программу SCC

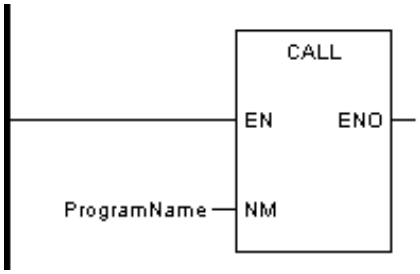
CALL

Описание функции:

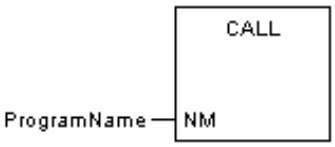
- При выполнении функционального блока CALL программа немедленно просканирует назначенную подпрограмму, а после сканирования вернется к месту функционального блока CALL и продолжит сканирование следующих программ. Вызываемая подпрограмма должна существовать.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

Название программы CAL

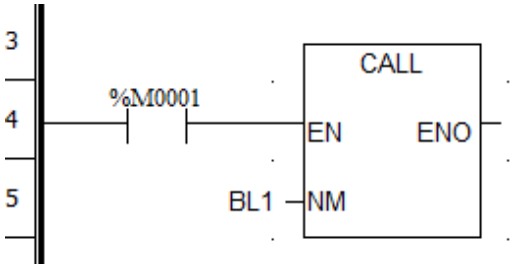
Форма на языке ST:

ИмяПрограммы ();

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
NM	ProgramName	Название подпро-граммы		

ПРИМЕР:



Описание: Щелкните правой кнопкой мыши LD, и появится новый интерфейс программы. Введите имя подпрограммы и выполнить подпрограмму BL1.

Выполнение программы SCC- EXEC

Описание функции:

- При выполнении функционального блока EXEC программа немедленно выполнит назначенный SCC, а исходная программа продолжит выполнение, не дожидаясь завершения сканирования SCC.

ВЫЗОВ

Форма в LD:

Форма в FBD:

Форма на языке IL:

CAL EXEC (SCCName)

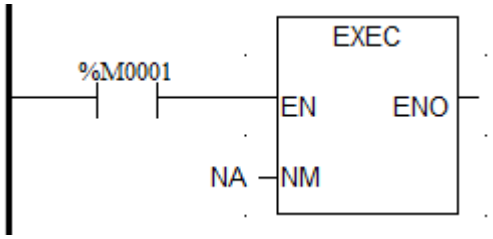
Форма на языке ST:

EXEC (имя_SCC);

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
NM	SCCName	Название подпро-граммы SCC		

ПРИМЕР:



Описание: Если %M0001=1, выполняется подпрограмма NA.

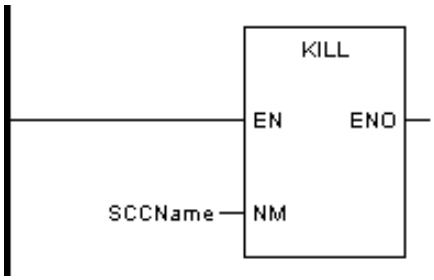
Прекращение выполнения программы SCC- KILL

Описание функции:

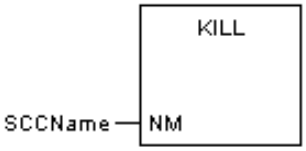
- При выполнении функционального блока KILL программа немедленно завершит назначенный SCC, а исходная программа продолжит выполнение.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL KILL (SCCName)

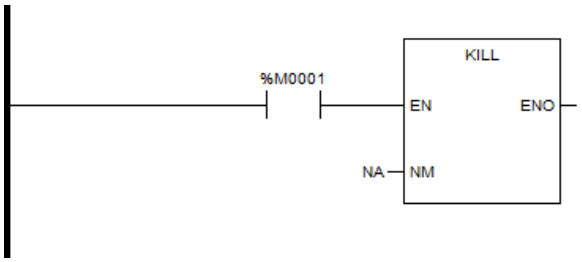
Форма на языке ST:

УБИТЬ (SCCName);

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
NM	SCCName	Название подпро-граммы SCC		

ПРИМЕР:



Описание: Если %M0001=1, завершить подпрограмму NA.

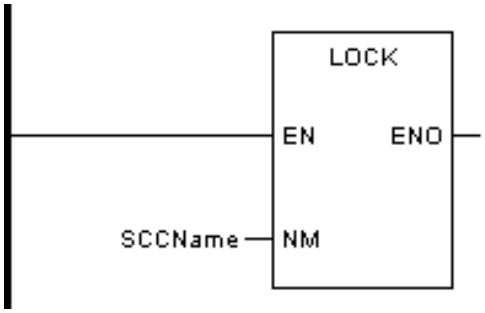
Заблокировать SCC - LOCK

Описание функции:

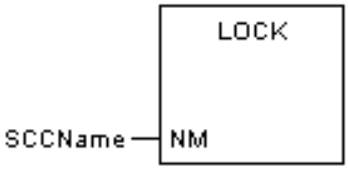
- При выполнении функционального блока LOCK программа немедленно заблокирует назначенный SCC, а исходная программа продолжит выполнение.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL LOCK (SCCName)

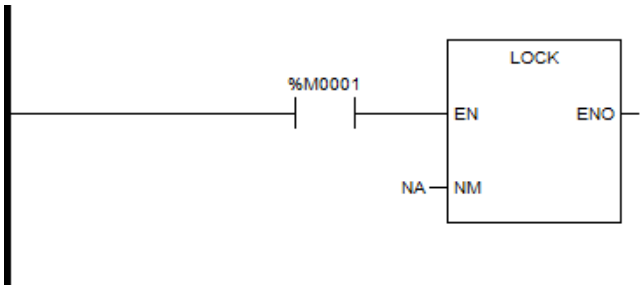
Форма на языке ST:

LOCK (SCCName);

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
NM	SCCName	Название подпрограммы SCC		

ПРИМЕР:



Описание: Если %M0001=1, заблокировать подпрограмму NA.

Разблокировать SCC - UNLOCK

Описание функции:

- При выполнении функционального блока UNLOCK программа немедленно разблокирует назначенный SCC, а исходная программа продолжит выполнение.

ВЫЗОВ

Форма в LD:

Форма в FBD:

Форма на языке IL:

CAL UNLOCK (SCCName)

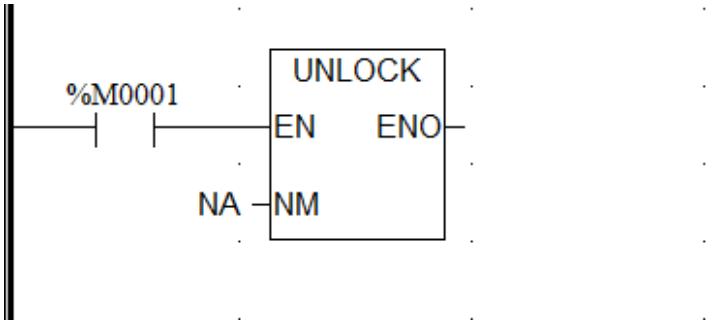
Форма на языке ST:

UNLOCK (SCCName);

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
NM	SCCName	Название подпрограммы SCC		

ПРИМЕР:



Описание: Если %M0001=1, разблокировать заблокированную подпрограмму NA.

Тип	Описание
PULSE	Импульсный выход
AOUT	Аналоговый вывод
FORCE	Принудительное присваивание значений
UNFORCE	Отмена принудительного присваивания
SWITCH	Переключение между Master и Slave
ENI	EN для прерывания
DISI	Макса прерываний
PWM	Широтно-импульсная модуляция

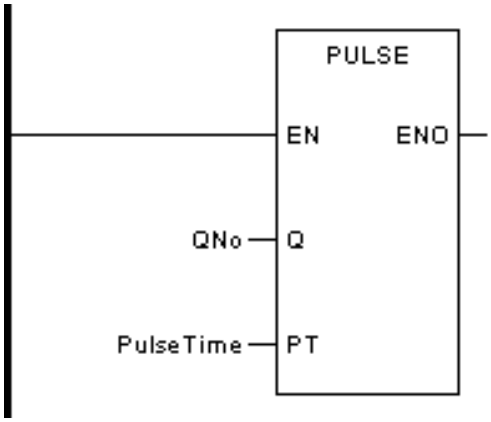
Импульсный выход - PULSE

Описание функции

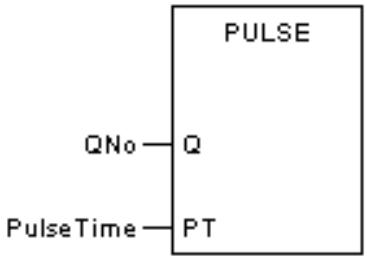
- Этот функциональный блок используется для вывода назначенной ширины импульса в цифровую выходную точку. Выполнение функционального блока установит назначенную цифровую выходную точку как 1. И начнет отсчет времени в соответствии с назначенной шириной импульса (т.е. временем, в течение которого цифровой выходной сигнал остается равным 1). По завершении отсчета времени цифровая выходная точка автоматически сбросится на 0.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL PULSE (Q:=QNo, PT:=PulseTime)

Форма на языке ST:

PULSE (Q:=QNo, PT:=PulseTime);

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
Q	QNo	Номер выход-ного параметра		B
PT	PulseTime	Длительность импульса, (мс); Эффективное значение 10～600000.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР:

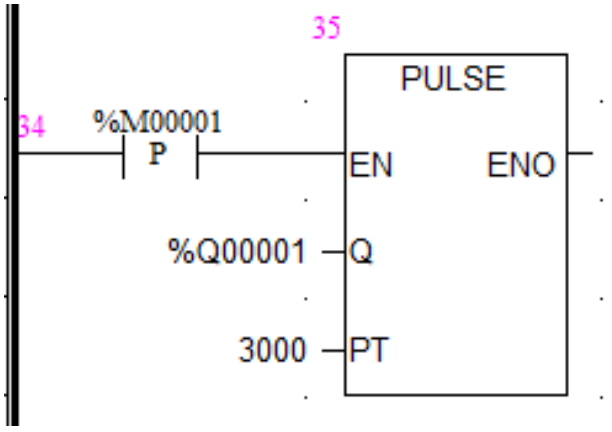
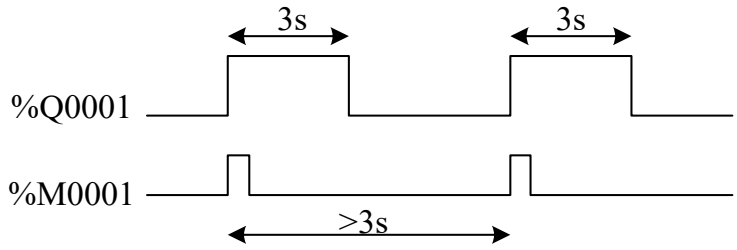


Диаграмма последовательности:



Описание: Каждый раз, когда %M0001 изменяется с 0 на 1, %Q0001 выдает импульсный выход с периодом 3 с. Период изменения %M0001 с 0 на 1 должен быть больше периода импульсного выхода %Q00001.

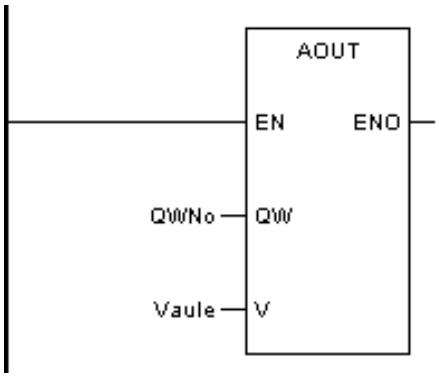
Аналоговый вывод - AOUT

Описание функции

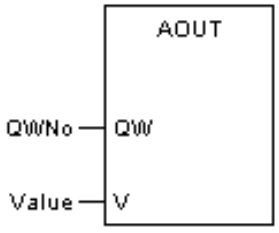
- Этот функциональный блок выводит значение на аналоговую выходную точку.

ВЫЗОВ

Форма в LD :



Форма в FBD:



Форма на языке IL:

CAL AOUT (QW:=QWNo, V:=Value)

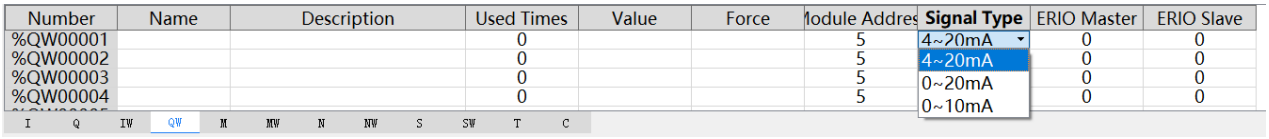
Форма на языке ST:

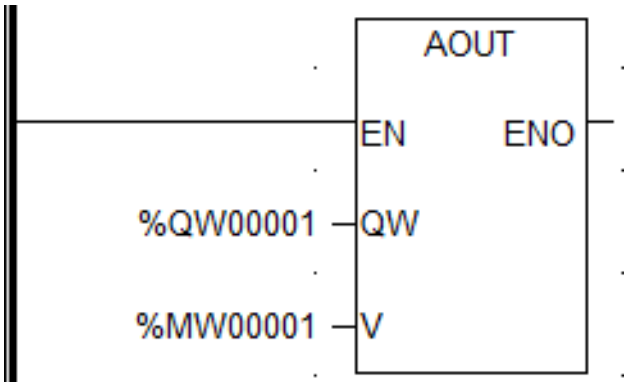
AOUT (QW:=QWNo, V:=Value);

Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
QW	QWNo	Регистр аналогового выхода		QW
V	Value	Значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР:





Описание: %MW0001 — буфер аналоговых выходов. Используя блок AOUT, можно передавать данные в регистр аналогового выхода %QW0001, а тип %QW0001 можно выбрать в таблице регистров памяти QW.

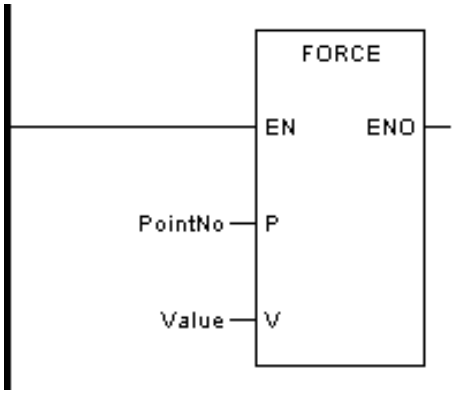
Принудительное присваивание значений регистров - FORCE

Описание функции

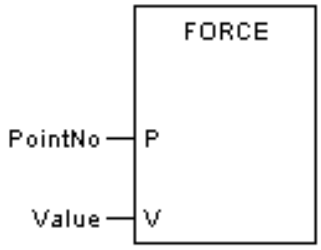
- Этот функциональный блок принудительно устанавливает базовую точку (I, Q, IW, QW) в качестве назначенного значения.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL FORCE (P:=Номер точки, V:=Значение)

Форма на языке ST:

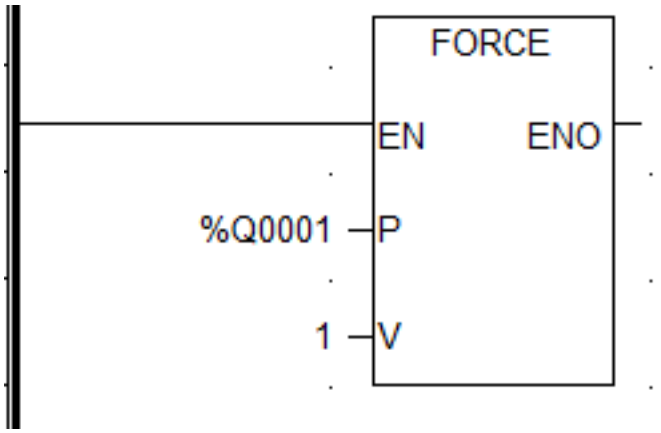
FORCE (P:=Номер точки, V:=Значение);

Описание параметров:

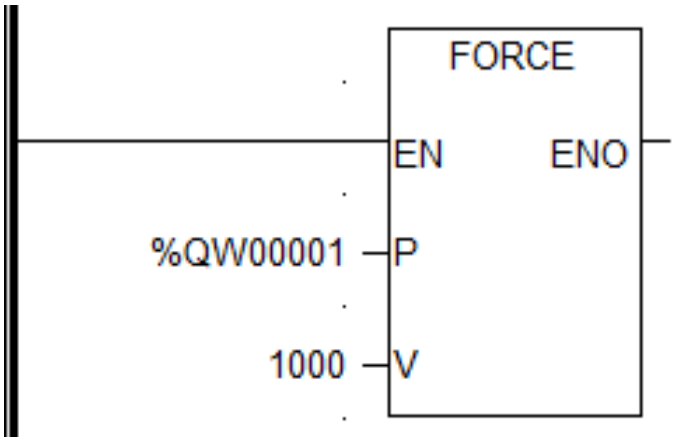
Диаграмма	Параметр	Описание	Тип данных	Тип регистра
P	PointNo	Номер регистра		I, Q, IW, QW

V	Value	Значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

ПРИМЕР 1: Принудительная точка Q. Принудительно присвойте %Q0001 значение 1.



ПРИМЕР 2: Принудительно установите %QW0001 равным 10000, а тип выходного типа %QW0001 — 4~20 мА. Соответствующая форма выходных данных — 4000~20000. В это время принудительно установите аналоговый выходной сигнал равным 10 мА.



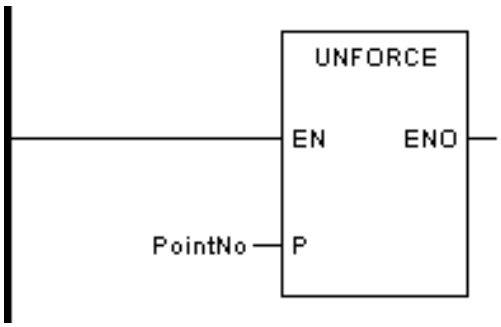
Отмена присваивания - UNFORCE

Описание функции

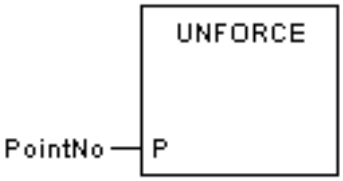
- Этот функциональный блок отменяет форсирование (принудительное присваивание) базовой точки (I, Q, IW, QW).

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL UNFORCE (пункт №)

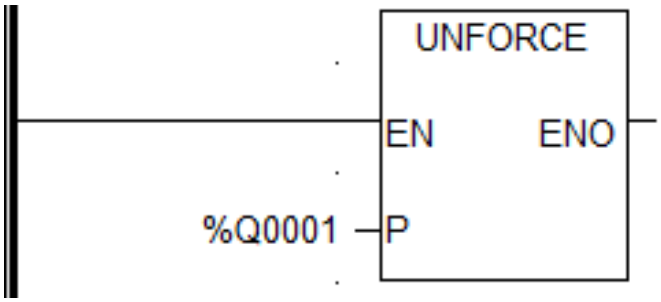
Форма на языке ST:

UNFORCE (пункт №);

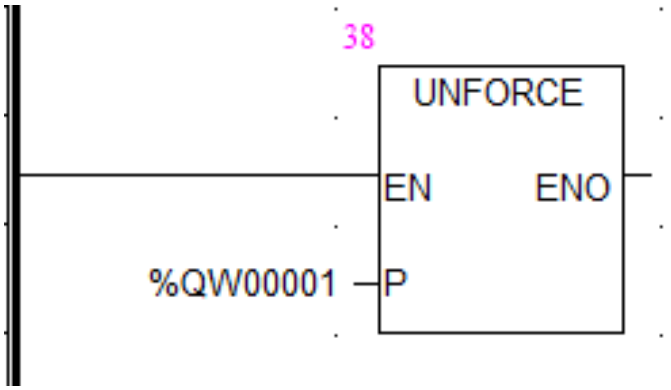
Описание параметров

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
P	PointNo	Номер регистра		I, Q, IW, QW

ПРИМЕР 1: Отменить принудительное применение точки %Q0001.



ПРИМЕР 2: Отменить присваивание точки %QW0001.



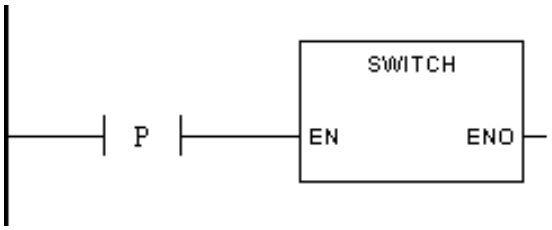
Переключение Master-Slave - SWITCH

Описание функции

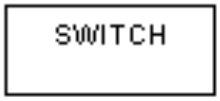
- Этот функциональный блок используется для переключения главного-ведомого устройства в двухпроцессорной системе. Когда у главного устройства есть неисправность, этот функциональный блок одновременно превратит его в подчиненное устройство, а подчиненное устройство в главное. Не зацикливайте переключение главного/ведомого устройства повторно, используйте команду обнаружения переднего или заднего фронта для запуска.

ВЫЗОВ

Форма в LD:



Форма в FBD:



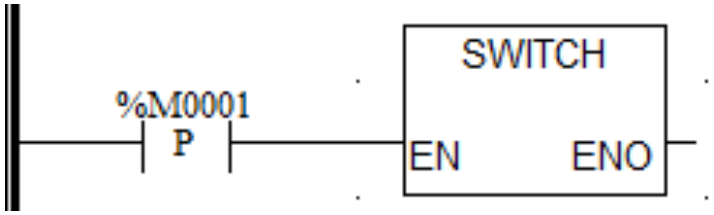
Форма на языке IL:

CAL SWITCH ()

Форма на языке ST:

CAL SWITCH ()

ПРИМЕР: При выполнении %M0001 выполнить переключение ведущего и ведомого устройств.



Прерывание EN - ENI

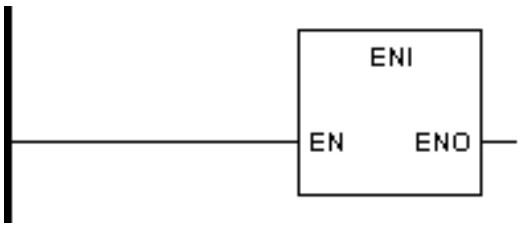
Описание функции

- После выполнения функционального блока, в случае прерывания, разрешается выполнение предопределенной программы прерывания. Перед выполнением прерывания таймера и прерывания ввода-вывода функциональный блок должен

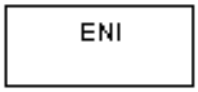
быть вызван один раз. Нет необходимости запускать его в каждом периоде сканирования.

ВЫЗОВ

Форма в LD:



Форма в FBD:



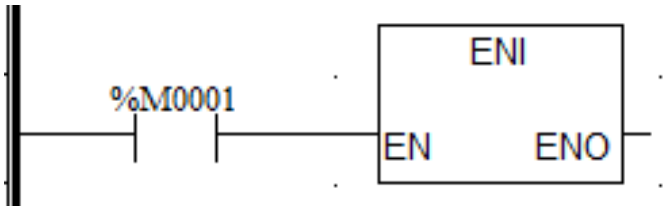
Форма на языке IL:

```
CAL    ENI ()
```

Форма на языке ST:

```
ENI ();
```

ПРИМЕР: При выполнении %M0001 происходит прерывание и выполняется предопределенная программа прерывания.

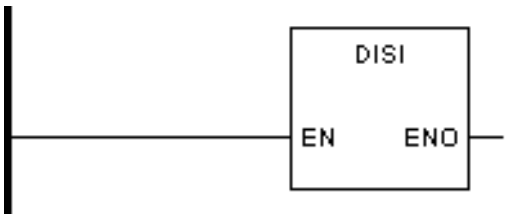


Описание функции

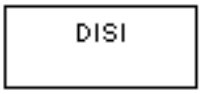
- После выполнения функционального блока, в случае прерывания, предопределенная программа прерывания не может быть выполнена.

ВЫЗОВ

Форма в LD:



Форма в FBD:



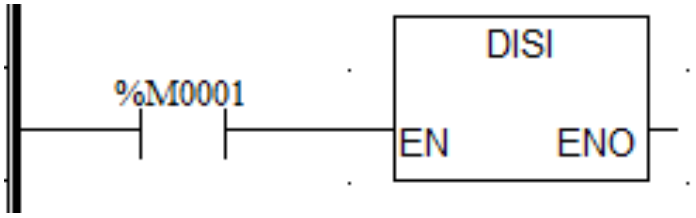
Форма на языке IL:

CAL DISI ()

Форма на языке ST:

DISI ();

ПРИМЕР: При выполнении %M0001 запрещается выполнение предопределенной программы прерывания.



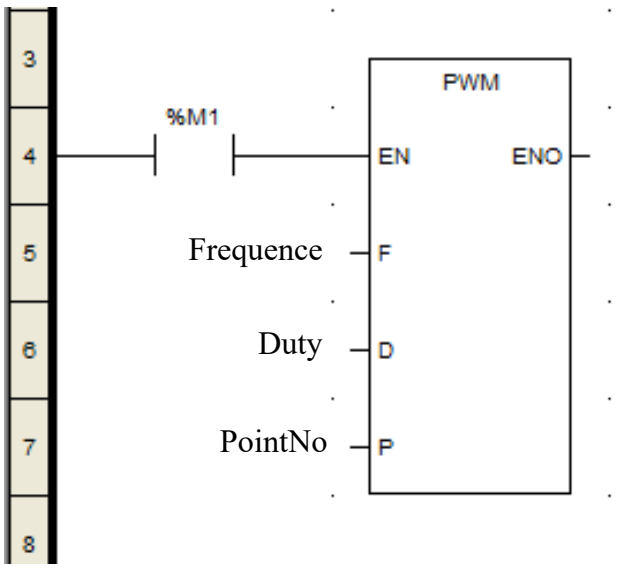
Широтно-импульсная модуляция - PWM

Описание функции

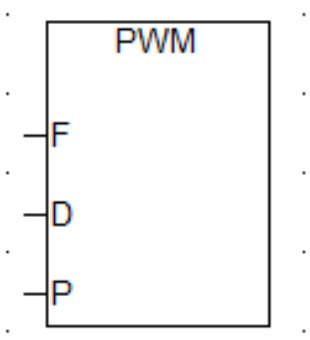
- Этот функциональный блок используется для вывода импульса с назначенной частотой и рабочим циклом на высокоскоростной выходной канал импульсов для управления устройствами (такими как шаговый двигатель, сервоконтроллер и т. д.). Когда блок включен, выход направления Q не контролируется функциональным блоком.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Описание параметров:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
EN	EN	Сигнал включения модуля, если 1, выход включен; если значение равно 0, выход отключен	1 бит	M
F	Frequence	Частота выходного импульса (0,1 Гц) от 100 до 1000000	32 бита	CONSTANT, IW, QW, MW, NW, SW, DW
D	Duty	Рабочий период выходного импульса, в диапазоне от 0 до 100	16 бит	CONSTANT, IW, QW, MW, NW, SW
P	PointNo	Импульсный выход Q		B
ENO	ENO	Состояние модуля	1 бит	M

ПРИМЕР:

На рисунке выше M350 — это включение инструкции. Частота 1000; Привод третьего вала. M350 включает открытый канал 3 для вывода неограниченных импульсов с частотой 1000 и рабочим циклом 50.

Примечание

ШИМ-инструкция не ограничивается выключателем предельной защиты, поэтому следует соблюдать осторожность, чтобы не допустить удара вала в оба конца направляющей при его движении.

Специальные блоки

Тип	Описание
PID	ПИД модификация
FOPEN	Открытие файла
FCLOSE	Закрытие файла
FREAD	Чтение файла
FWRITE	Запись в файл

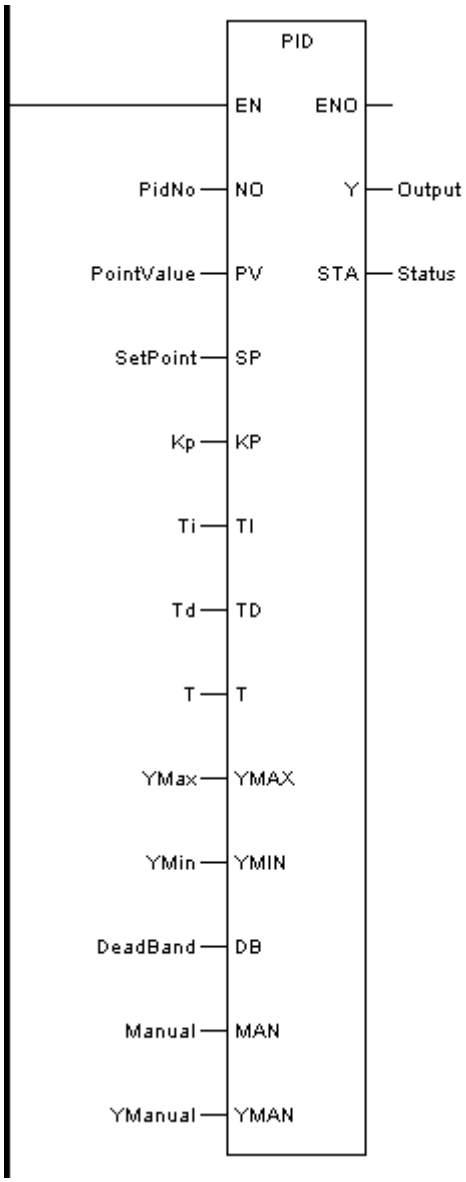
PID

Описание функции

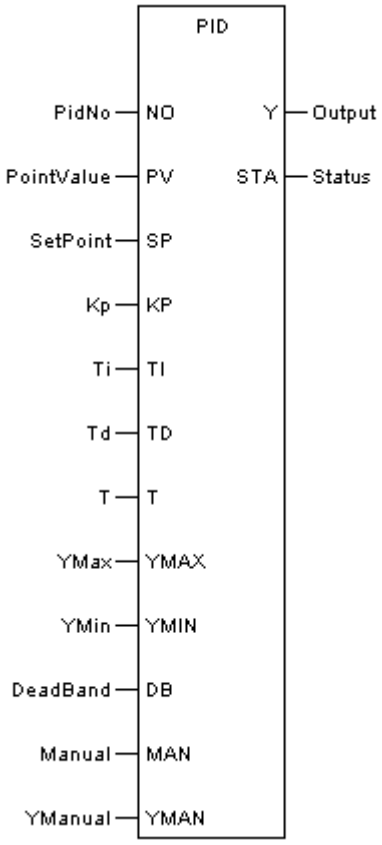
- Этот функциональный блок используется для реализации управления PID-регулятором, обычно для управления аналоговым каналом. NAPLC поддерживает 32 блока ПИД для одновременного использования.

ВЫЗОВ

Форма в LD:


PID
EN
ENO
PidNo
NO
Y
Output
PointValue
PV
STA
Status
SetPoint
SP
Kp
KP
Ti
TI
Td
TD
T
T
YMax
YMAX
YMin
YMIN
DeadBand
DB
Manual
MAN
YManual
YMAN

Форма в FBD:



Описание параметров:

Диа-грамма	Параметр	Описание	Тип данных	Тип регистра
NO	PidNo	Номер PID-регулятора	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
PV	PointValue	Значение регистра	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	IW, MW, NW, VAR
SP	SetPoint	Заданное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
KP	Kp	Пропорциональный коэффициент	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
TI	Ti	Время интегрирования (с), минимум 0,01	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
TD	Td	Время дифференцирования (с)	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

T	T	Время выборки (с): ▪ минимум 0,01, ▪ максимум 20	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
YMAX	YMax	максимальное выходное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
YMIN	YMin	минимальное выходное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DB	DeadBand	Мертвая зона	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
MAN	Manual	Выбор ручного/автоматического режим ▪ 1 ручной, ▪ 0 автоматический	BOOL	CONSTANT, I, Q, M, N, S, VAR
YMAN	YManual	Значение ручного вывода	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
Y	Output	Выходное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR
STA	Status	Состояние выполнения: 1. сбой инициализации 2. ПИД в использовании 3. выход регулятора. 4. ежегодный 5. Выход больше максимального. 6. выход меньше максимального 7. максимум/минимум недействительный	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, VAR

Уравнение расчета ПИД

$$Y = K_p * (E(t) + \frac{T}{T_I} \int E(t)dt + \frac{T_D}{T} * dE(t)/dt)$$

Дополнительные инструкции

[PV]: Значение точки. Поскольку формула расчета PID не преобразует диапазон входных данных, единицы данных PV, SP и Y должны быть согласованы. Чтобы выход Y существенно изменился, значение PV не должно быть слишком малым, и его лучше преобразовать в стандартное кодовое значение 0-20000 путем линейного преобразования.

[SP]: Заданное значение. Поскольку формула расчета ПИД не преобразует диапазон входных данных, единицы данных PV, SP и Y должны быть согласованы. Чтобы выход Y значительно изменился, значение PV не должно быть слишком малым, и его лучше преобразовать в стандартное кодовое значение 0-20000 путем линейного преобразования.

[КП]: коэффициент пропорциональности.

- KP>0: противоречащий регулированию
- KP<0: повышение регуляции

Предлагаемый параметр для KP составляет от 0,5 до 5. Значение по умолчанию — 2.

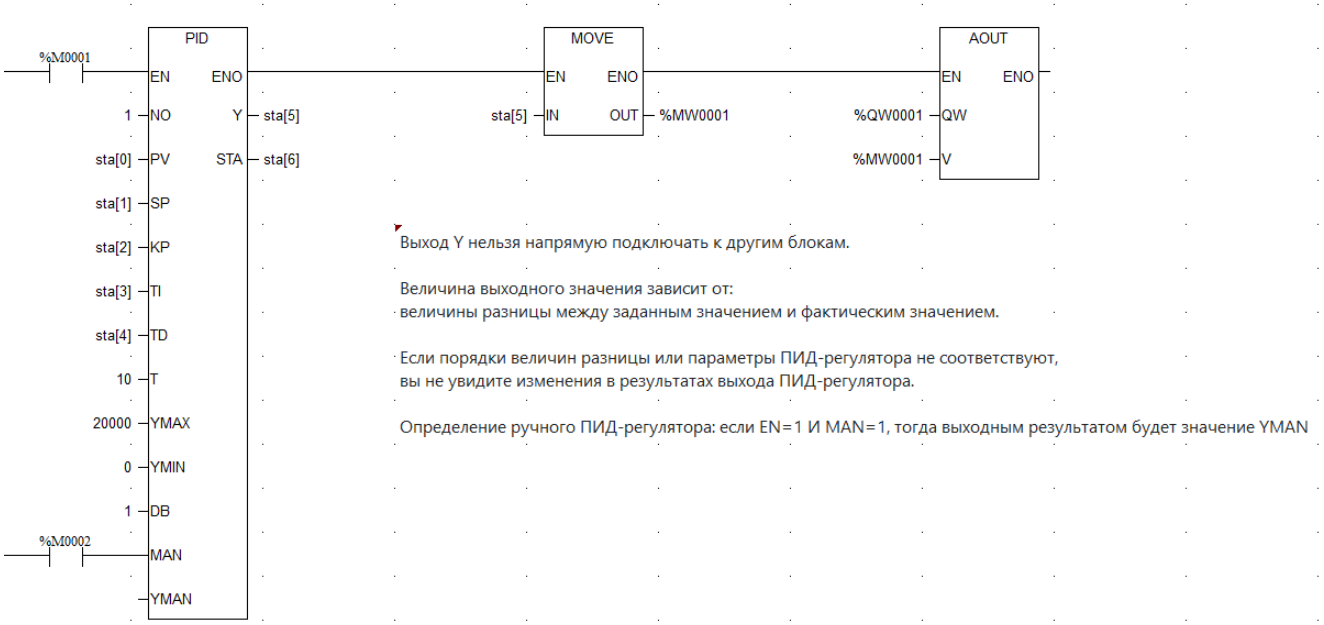
[TI]: Время интегрировании, рекомендуется до 100. Чем меньше значение, тем сильнее интегральная функция. Если вы не используете интегральную функцию, вы можете ввести большое значение, например, 20000. Значение по умолчанию — 20.

[TD]: Время дифференцирования. Общее регулирование PID температуры не может использовать дифференциальную функцию. Заполнение 0 делает дифференциальную функцию недействительной. Если используется дифференциальная функция, рекомендуется 10.

[T]: Время выборки, а также время для модуля функции ПИД для одного расчета. Рекомендуемый параметр по умолчанию — 2 с.

[Y]: PID вычисляет выходное значение. Поскольку кодовое значение модуля аналогового вывода составляет 0-20000, результат Y преобразуется в 0-20000 с помощью линейного преобразования, а затем выводится на канал %QW с помощью блока AOUT.

ПРИМЕР:



Описание: sta[0]~sta[8] — определяемые пользователем переменные с плавающей запятой таблицы регистров.

Открыть файл - FOPEN

Описание функции:

- Функция FOPEN используется для открытия файла для последующих операций чтения или записи. Результат работы FOPEN определяет, удалось ли открыть файл и получить к нему доступ.

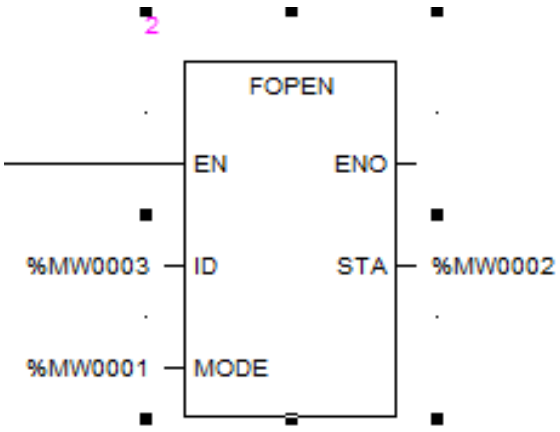
Параметры (могут варьироваться в зависимости от среды):

- **FileName** (или Path): Путь к файлу, который необходимо открыть. Может быть абсолютный (полный путь) или относительный (относительно текущего каталога).
- **Mode:** Режим открытия файла, определяющий, какие операции разрешены:
- **«r»** (или «rt»): Только для чтения (файл должен существовать).
- **«r+»** (или «rt+»): Для чтения и записи (файл должен существовать).
- **«w»** (или «wt»): Только для записи (создаёт новый файл или перезаписывает существующий).

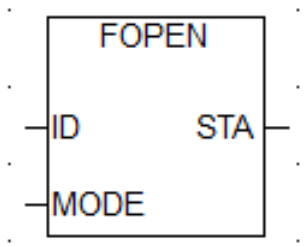
- **«w+»** (или «wt+»): Для чтения и записи (создаёт новый файл или перезаписывает существующий).
- **«a»** (или «at»): Только для добавления (дозапись в конец существующего файла; создаёт новый, если файла нет).
- **«a+»** (или «at+»): Для чтения и добавления (дозапись в конец существующего файла; создаёт новый, если файла нет).
- Могут быть также бинарные режимы: «rb», «rb+», «wb», «wb+», «ab», «ab+». В бинарном режиме данные читаются и записываются как последовательности байтов, без преобразования в текстовые строки.
- **File Handle** (или ID): Выходной параметр. Это идентификатор (номер) открытого файла, который используется в последующих функциях для работы с этим файлом (FREAD, FWRITE и т.д.). Если файл не открыт, File Handle будет иметь специальное значение (например, -1, NULL, 0), указывающее на ошибку.
- **Status** (или Return Value): Выходной параметр. Содержит код состояния, указывающий на успех или неудачу операции открытия файла (например, 0 – успешно, -1 – ошибка). Код ошибки может быть более детальным (например, файл не найден, нет прав доступа и т.д.).

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в ST:

FOPEN(ID:=Id, MODE:=Mode, STA=>Status);

Параметры:

Диаграмма	Параметр	Описание
ID	Id	порядковый номер имени открываемого файла, например, ID=1 означает открытие файла file1.dat (операции чтения и записи выполняются с файлами с именами filexx.dat, где xx находится в диапазоне от 1 до 32)
MODE	Mode	Режим работы, например, MODE=15 соответствует режиму a+b
STA	Status	Состояние выполнения функционального блока

Закрыть файл - FCLOSE

- Функциональный блок FCLOSE предназначен для закрытия файла, который был ранее открыт с помощью блока FOPEN (или аналогичной функции открытия файла).

Почему важно закрывать файлы:

Освобождение ресурсов: Когда файл открыт, операционная система (или контроллер) выделяет для него ресурсы, такие как память, дескриптор файла (идентификатор), буферы. Если файлы не закрывать, эти ресурсы остаются занятыми, даже когда вы больше не работаете с файлом, и это может привести к:

Утечкам памяти (Memory leaks): Постепенно всё больше памяти выделяется под открытые файлы и может закончиться.

Ограничению на количество открытых файлов: Операционная система может иметь ограничение на максимальное количество одновременно открытых файлов, и вы не сможете открыть новые файлы.

Снижению производительности: Занятые ресурсы могут замедлять работу всей системы.

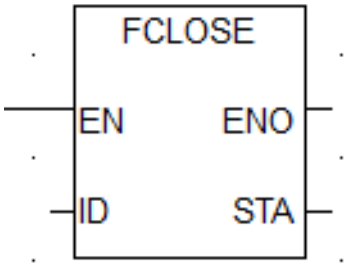
Сохранение данных: При записи данных в файл, они часто сначала сохраняются во внутренний буфер операционной системы. Закрытие файла гарантирует, что все данные из этого буфера будут записаны на физический диск или другое постоянное хранилище. Если файл не закрыть, часть или все данные могут быть потеряны.

Разблокировка файла: В некоторых случаях, файл может быть “заблокирован” процессом, который его открыл, что может помешать другим процессам или программам получить к нему доступ. Закрытие файла снимает эту блокировку.

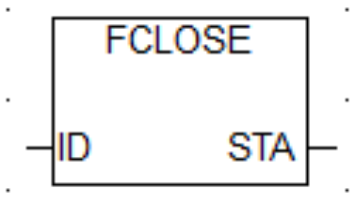
Согласованность данных: Закрытие файла является частью процесса обеспечения целостности данных. Это гарантирует, что все изменения внесенные в файл корректно сохранены.

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма в ST:

FCLOSE(ID:=Id, STA=>Status);

Параметры:

Диа-грамма	Параметр	Описание	Тип данных	Тип регистра
ID	ID	Порядковый номер имени открываемого файла, например, ID=1 означает закрытие файла file1.dat (операции чтения и записи выполняются с файлами с именами filexx.dat, где xx находится в диапазоне от 1 до 32)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
STA	Status	Состояние выполнения функционального блока	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Прочитать из файла - FREAD

- Функциональный блок FREAD предназначен для чтения данных из открытого файла. Он принимает идентификатор (handle) открытого файла и считывает данные из него в указанную область памяти (буфер).

Основные особенности FREAD:

Режим доступа: Данные читаются в соответствии с режимом, в котором открыт файл (MODE в FOPEN). Если файл открыт только для записи, чтение будет невозможно (или вызовет ошибку).

Тип данных: Тип данных, в которые считываются данные, должен соответствовать режиму доступа и формату файла (текстовый или бинарный).

Буферизация: Обычно чтение данных происходит через буфер операционной системы, поэтому не все данные могут быть сразу доступны.

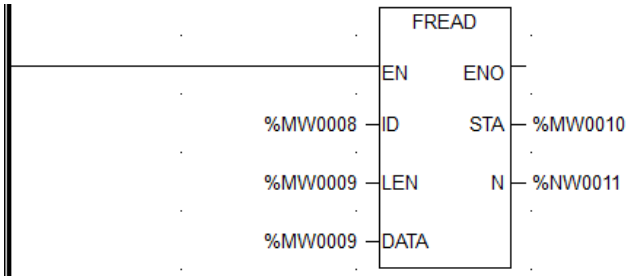
Размер чтения: FREAD читает не более, чем было указано в параметре Size или Length, но может прочитать и меньше, если достигнет конца файла.

Обработка ошибок: Необходимо проверять статус выполнения FREAD (STA) и обрабатывать ошибки.

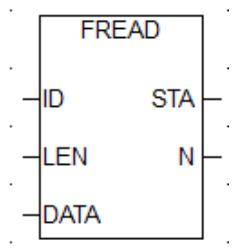
Позиция: Чтение происходит с текущей позиции в файле (если не указана другая позиция).

ВЫЗОВ:

Форма в LD:



Форма в FBD:



Форма в ST:

FREAD(ID:=Id, LEN:=Length, DATA:=DataBuffer, STA=>Status, N=>Number);

Параметры:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра
ID	Id	Порядковый номер имени открываемого файла, например, ID=1 означает закрытие файла file1.dat (операции чтения и записи выполняются с файлами с именами filexx.dat, где xx находится в диапазоне от 1 до 32)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR

LEN	Length	Количество байтов, которые нужно считать из буфера данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	DataBuffer	Область данных чтения из файла	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
N	Number	Количество регистров, которые необходимо считать из файла, (единицей измерения является тип входных данных.)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
STA	Status	Состояние выполнения функционального блока	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Записать в файл - FWRITE

- FWRITE является важным элементом в системах, работающих с файлами, и часто используется в сочетании с FOPEN и FCLOSE.

Основное назначение:

- Функциональный блок FWRITE предназначен для записи данных в открытый файл. Он принимает данные, которые нужно записать, и помещает их в файл, на который ссылается переданный ему идентификатор

Основные особенности:

Соответствие FOPEN: Файл должен быть предварительно открыт с помощью FOPEN и ему должен соответствовать ID, который передается в FWRITE.

Режим доступа: Данные записываются в файл в соответствии с режимом, в котором он был открыт (MODE в FOPEN). Если файл открыт только для чтения, запись будет невозможна (или вызовет ошибку).

Тип данных: Тип данных, которые нужно записать, должен соответствовать режиму доступа и формату файла (текстовый или бинарный).

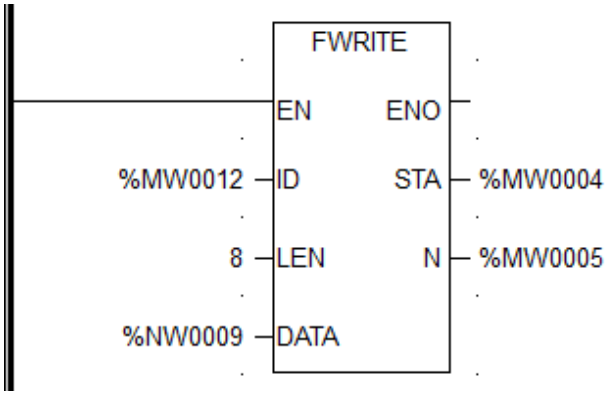
Буферизация: Обычно запись данных происходит через буфер операционной системы, поэтому данные могут быть не сразу записаны на диск. FCLOSE отвечает за сброс буфера на диск.

Обработка ошибок: Необходимо проверять статус выполнения FWRITE (STA) и обрабатывать ошибки.

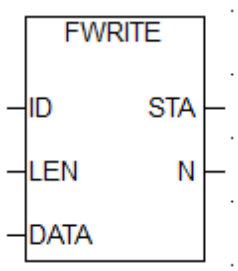
Размер записи: FWRITE может записывать как все, так и только часть данных, содержащихся в буфере Data.

ВЫЗОВ:

Форма в LD:



Форма в FBD:



Форма в ST:

FWRITE(ID:=Id, LEN:=Length, DATA:=DaataBuffer, STA=>Status, N=>Number);

Параметры:

Диаграмма	Параметр	Описание	Тип данных	Тип регистра

ID	ID	Порядковый номер имени открываемого файла, например, ID=1 означает закрытие файла file1.dat (операции чтения и записи выполняются с файлами с именами filexx.dat, где xx находится в диапазоне от 1 до 32)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
LEN	Length	Количество байтов, которые нужно записать из буфера данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	DataBuffer	Область данных записи в файл	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR
N	Number	Количество регистров, которые необходимо записать в файл, (единицей измерения является тип входных данных.)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
STA	Status	Состояние выполнения функционального блока	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, VAR

Новый функциональный блок

Во вкладке [Вид] / [Таблица пользовательских ФБ] щелкните по «Таблице типов функциональных модулей», как показано на рисунке 5.3 ниже.

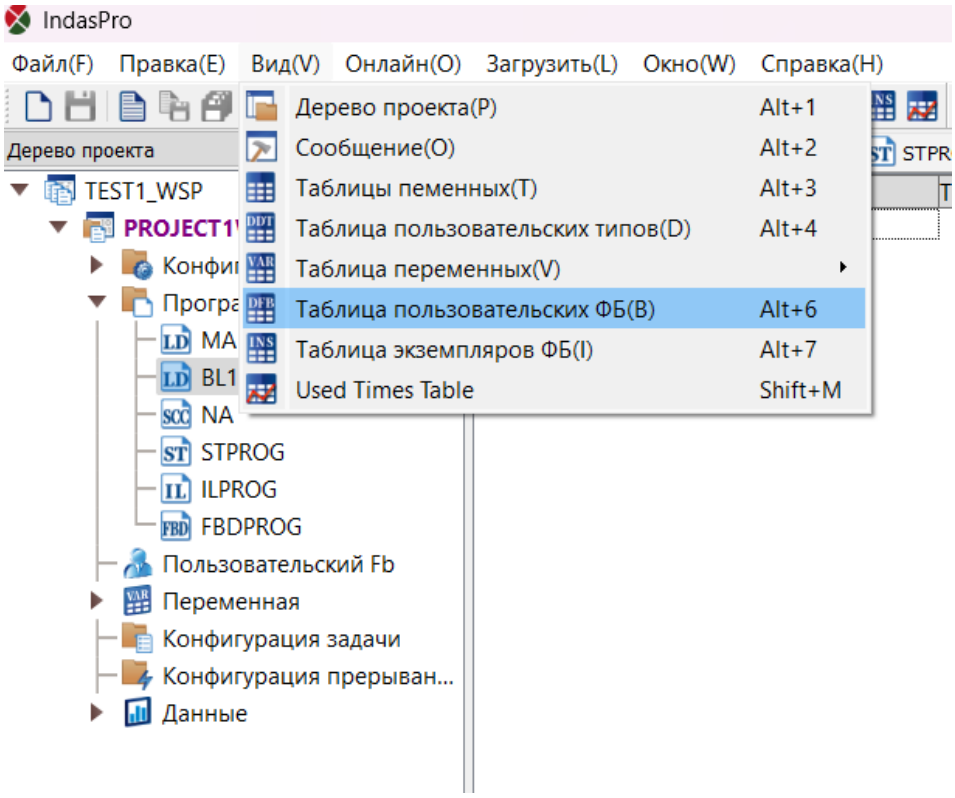


Рисунок 5.3 Открытие таблицы пользовательских ФБ

В открывшемся окне заполните имя функционального модуля, описание, входы, выходы и внутренние переменные. Как показано на рисунке 5.4.

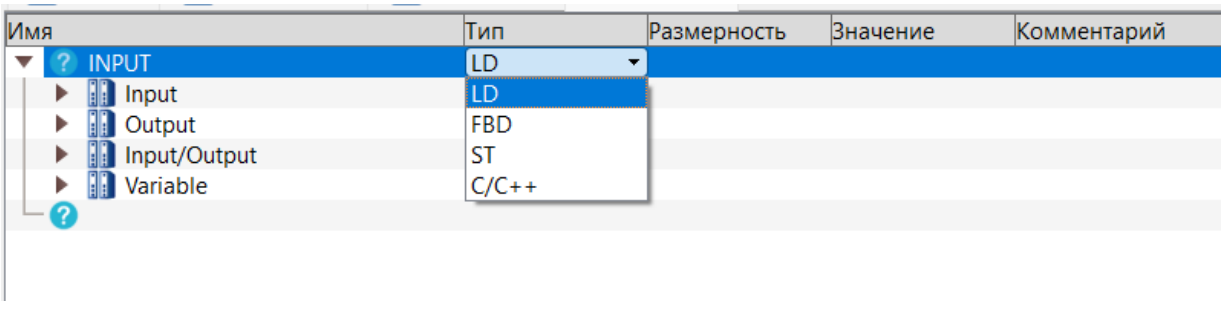


Рисунок 5.4 Схема определения функциональных блоков

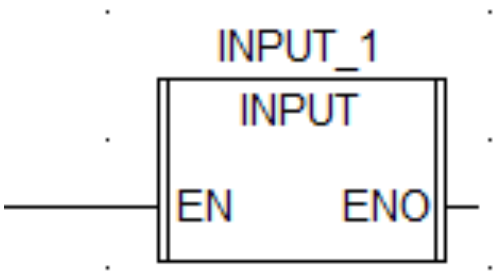


Рисунок 5.5 Пользовательский функциональный блок в программе

Инкапсулированные функциональные модули можно найти в [Панели инструментов] / [Настройка пользователя] и использовать их так же, как и другие общие функциональные модули. Это показано на рисунке 5.6.

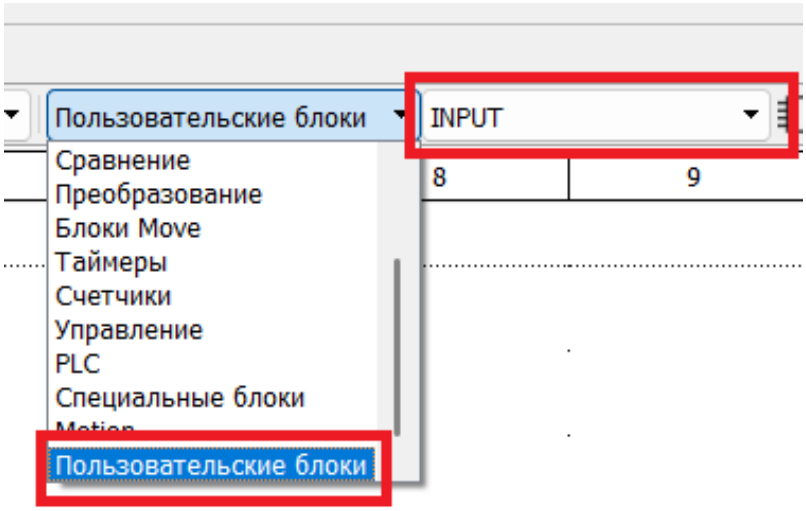


Рисунок 5.6 Вызов пользовательского функционального блока из контекстного меню

Программирование функциональных блоков

После создания нового функционального модуля значок функционального модуля будет , щелкните по нему правой кнопкой мыши, выберите тип анализа и, когда анализ будет успешным (значок изменится на) программа функционального модуля будет создана автоматически в дереве каталогов программ.

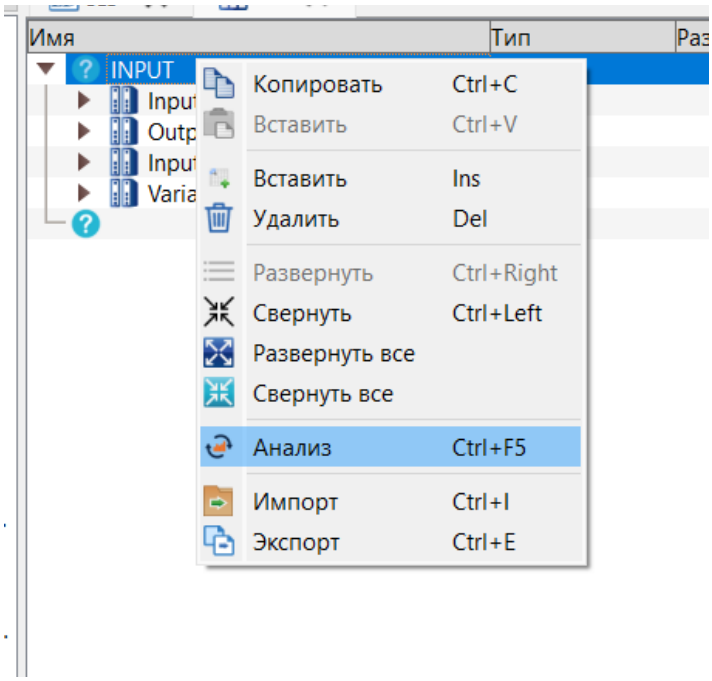


Рисунок 5.7 Анализ функционального блока

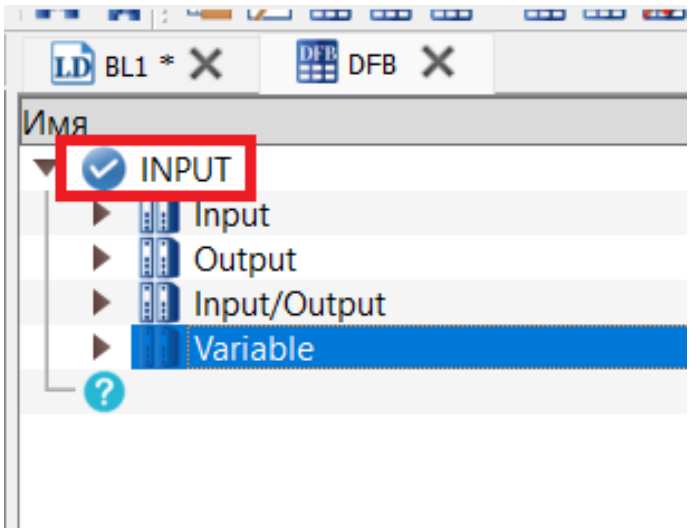


Рисунок 5.8 Автоматическое создание программы функционального блока после анализа

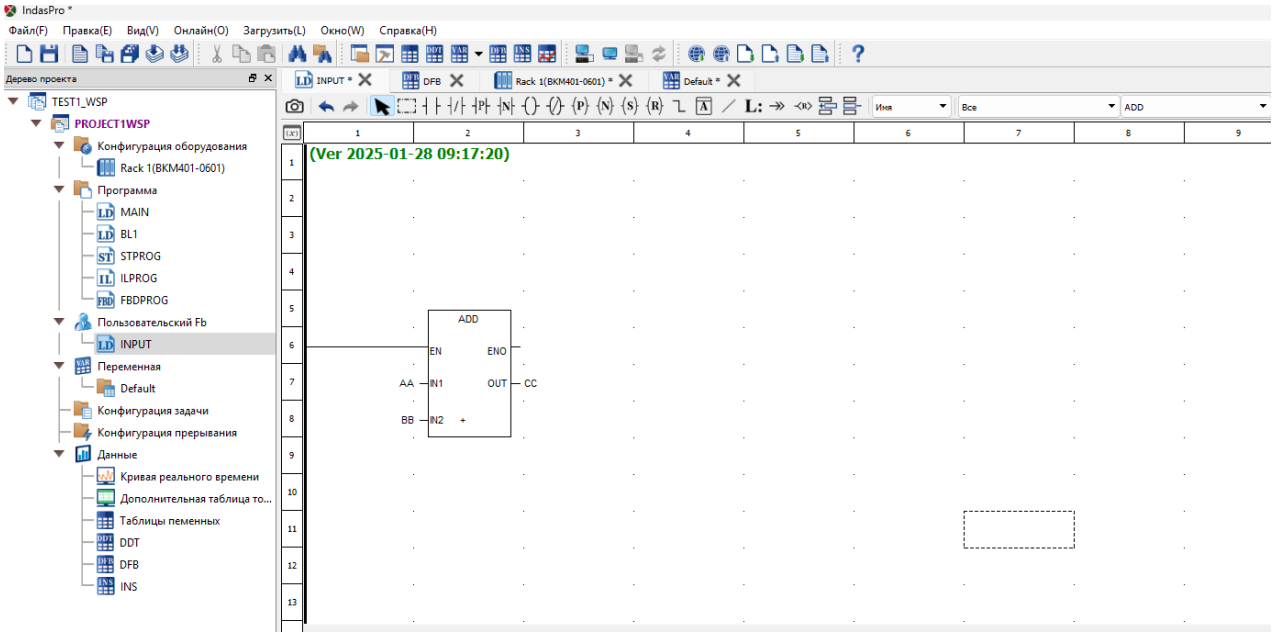


Рисунок 5.9 Интерфейс редактирования программы функционального модуля

Таблица типов функциональных модулей (DFB)

Каждый узел DFB по умолчанию имеет четыре дочерних узла, в каждом из которых можно редактировать и добавлять новые дочерние узлы, как показано ниже.

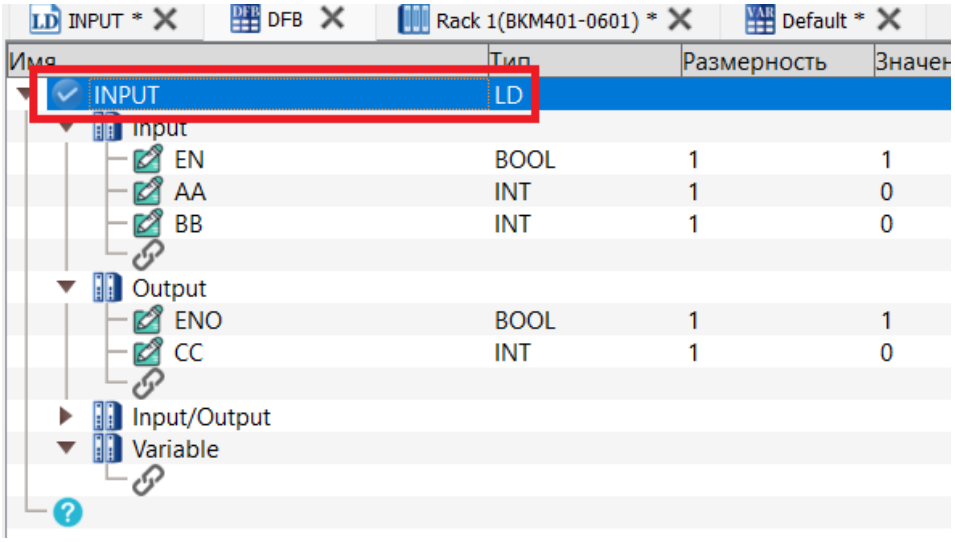


Рисунок 5.10 Интерфейс редактирования узла функционального модуля

Оба этих входных и выходных подузла имеют узлы включения по умолчанию (EN/ENO), которые не подлежат редактированию.

1. Имя: DFB Имена узлов первого уровня и их потомков по умолчанию пишутся с заглавной буквы и состоят из букв, подчеркиваний и цифр. Узлы первого уровня (имена функциональных модулей) не должны превышать 8 в длину, для узлов одного уровня не допускаются повторяющиеся имена, и они не могут

дублироваться с узлами первого уровня в таблице экземпляров функциональных модулей.

2. Типы данных: Типами узлов первого уровня могут быть: лестничная диаграмма, функциональная блок-схема и Структурированный текст; типами других подузлов могут быть базовый тип и тип анализируемого узла DDT.
3. Всего: Максимальное количество функциональных модулей 256, максимальное количество входов, выходов, входов/выходов, внутренних переменных контактов 64.

Описание функции контекстного меню

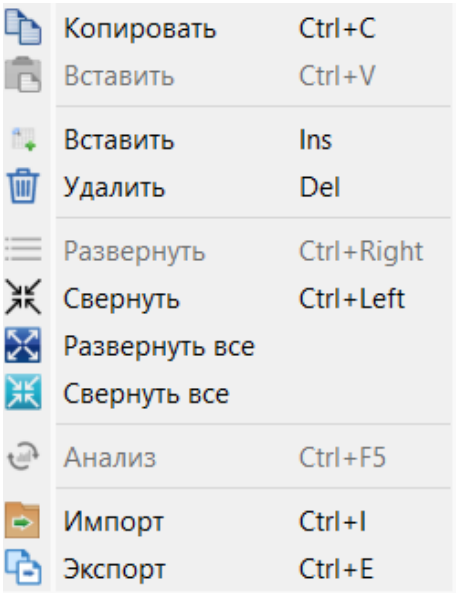


Рисунок 5.11 Контекстное меню таблицы типов функциональных модулей

[Копировать]: Вы можете копировать узлы первого уровня.

[**Вставить**]: Щелкните правой кнопкой мыши по пустой строке, чтобы вставить скопированный узел уровня 1, имя будет увеличиваться с 0 в соответствии с количеством вставок. Например, если имя скопированного узла уровня 1 — aa, имя первого вставленного узла — aa_0, имя второго вставленного узла — aa_1, ..., а имя узла, вставленного в N-й раз, — aa_N-1. Дочерние элементы вставленного узла такие же, как у скопированного узла.

[**Вставить**]: Щелкните существующий узел, которому должен предшествовать новый узел, и выберите Вставить, чтобы вставить пустую строку перед выбранным узлом с содержимым, введенным пользователем двойным щелчком. Вставка может быть выполнена на узлах уровня 1 и уровня 3; эта функция недоступна для дочерних узлов пользовательского типа.

[**Удалить**]: Щелкните правой кнопкой мыши по опции удаления узла уровня 1 или уровня 3, чтобы удалить узел. При удалении узла уровня 1 его нельзя удалить, если он уже используется в таблице экземпляров функциональных модулей. Узел включения по умолчанию нельзя удалить.

[**Развернуть**]: Щелкните по узлу, который необходимо развернуть, чтобы выбрать функцию развертывания и развернуть дочерние элементы узла.

[**Свернуть**]: Щелкните узел, дочерние узлы которого необходимо свернуть, чтобы выбрать функцию развертывания для сворачивания дочерних узлов этого узла.

[**Развернуть все**]: Щелкните правой кнопкой мыши в любом месте таблицы типов данных и выберите функцию «Развернуть все», чтобы развернуть все узлы всей таблицы типов данных.

[**Свернуть все**]: Щелкните правой кнопкой мыши в любом месте таблицы типов данных и выберите функцию «Свернуть все», чтобы свернуть все дочерние узлы всей таблицы типов данных.

[**Анализ**]: После редактирования узла щелкните правой кнопкой мыши и выберите параметр анализа.

Если узел проанализирован успешно, значок изменится на  ;

Если тип функционального модуля используется в таблице экземпляров функционального модуля, он будет обновлен одновременно. Возможные причины сбоя анализа.

- **(a)** Имя узла, который в данный момент будет проанализирован, является дубликатом имени уже проанализированного узла.
- **(b)** Узел DDT, на который ссылается текущий узел, был изменен и еще не проанализирован.

[**Экспорт**]: Функция экспорта доступна для узлов DFB уровня 1, которые были проанализированы. При экспорте вы можете переименовать экспортированные узлы. Соглашение об именовании начинается с буквы, состоит из буквы, цифры и подчеркивания и не должно превышать 8. Расширение файла экспорта — .dfb.

[**Импорт**]: Импортируйте узлы DFB из файла, вы можете изменить имя узла при импорте, имя узла не может дублироваться с существующим узлом в таблице DFB. Если импортируемый узел ссылается на существующий узел в таблице DDT, он не будет добавляться повторно, если содержимое одинаковое, но будет выведено сообщение об ошибке, если содержимое отличается.

Нажмите на  и появится таблица INS.

Функция экземпляров функциональных блоков: когда в программе упаковки функционального блока есть таймеры, счетчики, обнаружение нарастающего фронта, обнаружение падающего фронта, в случае повторного использования таймеров и счетчиков должны быть определены различные экземпляры функциональных блоков, то есть, если этот функциональный блок должен использоваться много раз в других программах, необходимо определить несколько экземпляров функциональных модулей. Если в упакованном функциональном модуле нет таймера или счетчика, нет необходимости вручную добавлять экземпляры функциональных модулей, система автоматически добавит один, который можно использовать повторно.

Если вы вручную размещаете пользовательские функциональные модули в программе, экземпляры добавляются автоматически; Если пользовательский функциональный модуль, который использовался в исходной программе, копируется для модификации, экземпляр не добавляется автоматически. При необходимости вручную измените экземпляр функционального модуля.

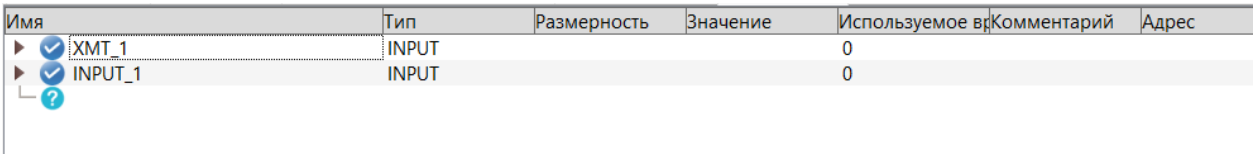


Рисунок 5.12 Редактирование экземпляров функциональных блоков

Для экземпляра, добавленного в экземпляр функционального блока, номер экземпляра должен быть вручную изменен во время использования функционального модуля. Номер экземпляра вновь добавленного функционального модуля по умолчанию — XXX_1. Чтобы изменить номер экземпляра, выберите функциональный модуль и щелкните правой кнопкой, как показано на рисунке 5.13.

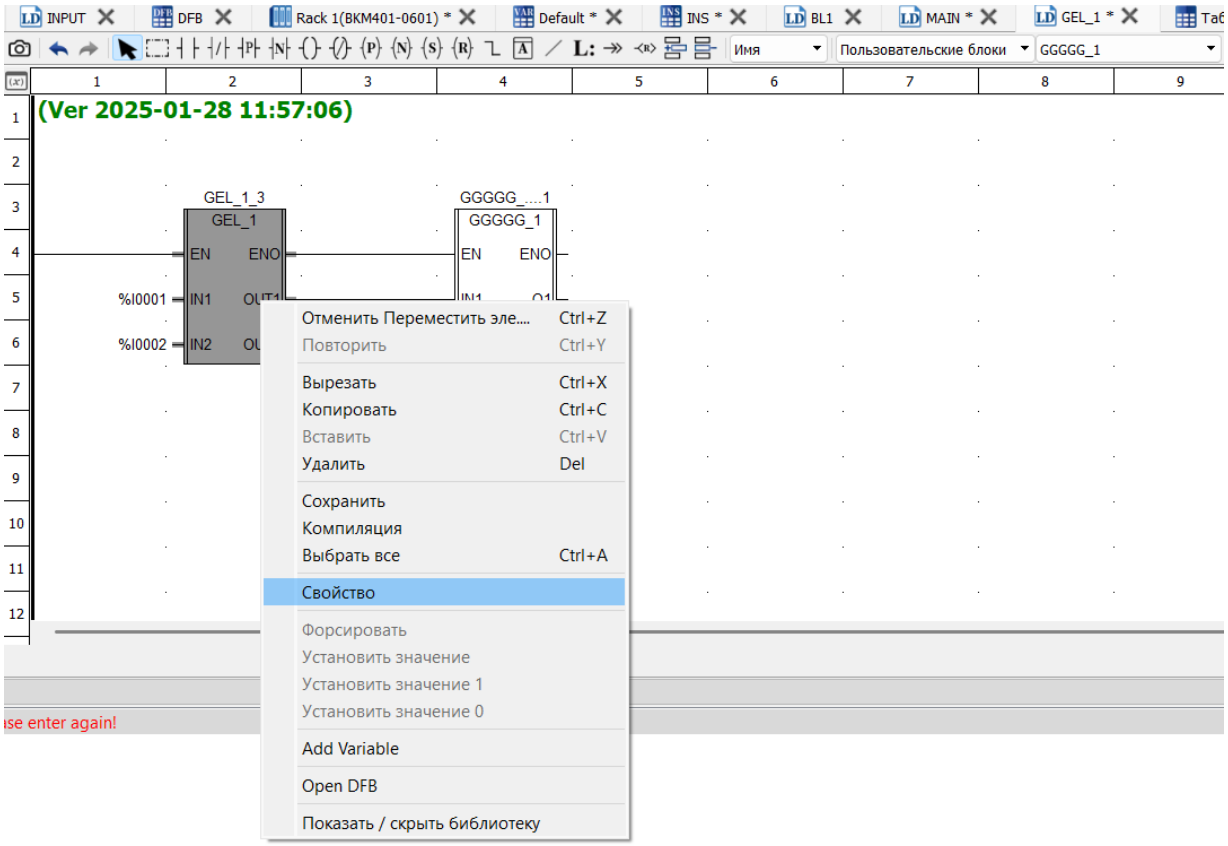


Рисунок 5.13 Свойство функционального блока

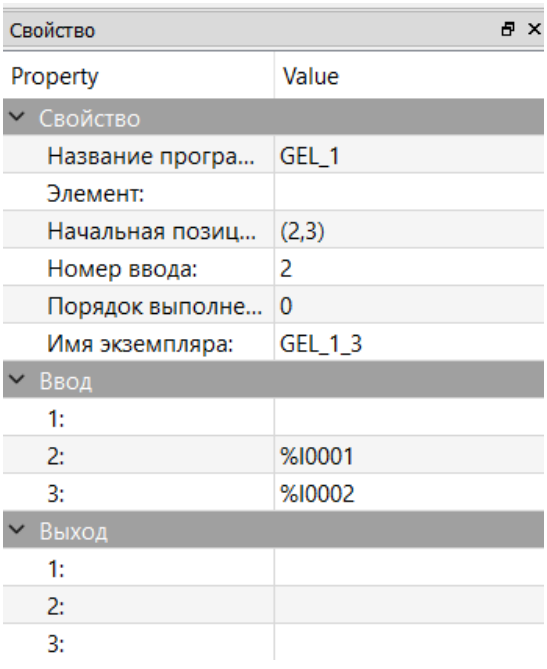


Рисунок 5.14 Модификация экземпляра функционального блока

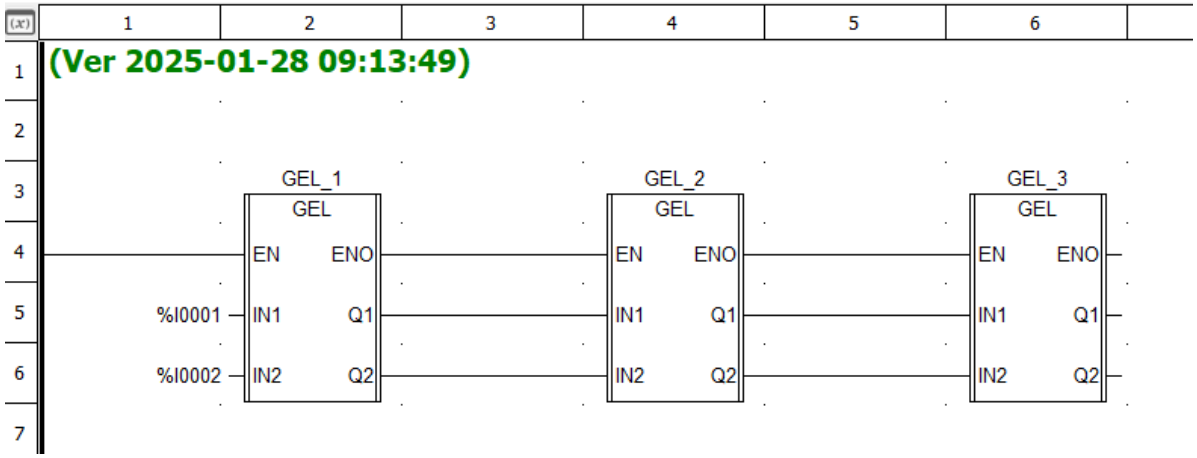
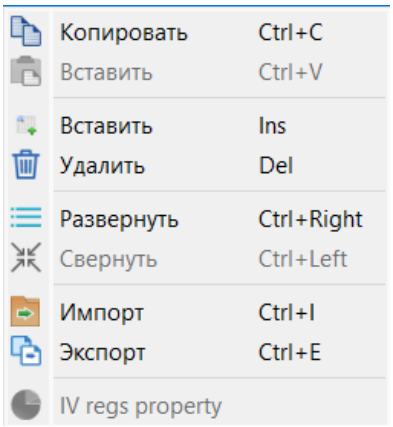


Рисунок 5.15 Пример использования экземпляров функциональных блоков

Каждый узел INS на самом деле является копией узла DFB, за исключением того, что внутренние переменные узла INS могут вычислять адрес. Адрес вычисляется таким же образом, как и таблица переменных.

1) **Имя:** Имена узлов первого уровня INS и его дочерних узлов по умолчанию пишутся с заглавной буквы и состоят из букв, подчеркиваний и цифр, длина имени узла первого уровня временно не ограничена, одно и то же имя однорангового узла не допускается, и не может совпадать с именем узла первого уровня в таблице типов функциональных модулей.
2) **Тип данных:** типом узла уровня 1 может быть имя анализируемого типа DFB.
3) **Диапазон внутренних переменных адресов:** до 16K.

Описание функции контекстного меню



[**Копировать**]: Вы можете копировать узлы первого уровня.

[**Вставить**]: Щелкните правой кнопкой мыши по пустой строке, чтобы вставить скопированный узел уровня 1, имя будет увеличиваться с 0 в соответствии с количеством вставок. Например, если имя скопированного узла уровня 1 — aa, имя первого

вставленного узла — aa_0, имя второго вставленного узла — aa_1, ..., а имя узла, вставленного в N-й раз, — aa_N-1. Дочерние элементы вставленного узла такие же, как у скопированного узла.

[**Вставить**]: Щелкните существующий узел, которому должен предшествовать новый узел, и выберите «Вставить», чтобы вставить пустую строку перед выбранным узлом с содержимым, введенным пользователем двойным щелчком.

[**Удалить**]: Щелкните правой кнопкой мыши по опции удаления, чтобы удалить узел.

[**Развернуть**]: Щелкните по узлу, который необходимо развернуть, чтобы выбрать функцию развертывания и развернуть дочерние элементы узла.

[**Свернуть**]: Щелкните по узлу, дочерние узлы которого необходимо свернуть, чтобы выбрать функцию сворачивания для сворачивания дочерних узлов этого узла.

[**Экспорт**]: используется для экспорта узлов INS. При экспорте можно переименовывать экспортированные узлы. Соглашение об именовании начинается с буквы, состоит из буквы, цифры и подчеркивания и не должно превышать 8. Расширение файла экспорта — .ins.

[**Импорт**]: импорт узлов INS из файла, вы можете изменить имя узла при импорте, имя узла не может дублироваться с существующим узлом в таблице INS.

Программирование LD

LD — это язык программирования, представленный графами, инструкции которых похожи на принципиальные схемы. С помощью LD можно отслеживать данные и поток тока между инструкциями в режиме онлайн.

Создание языка программирования LD должно соответствовать привычкам проектировщиков традиционных схем. Поэтому многие аспекты языков программирования LD похожи на цикл, составленный реле и другими электронными устройствами. Управляющие программы, написанные LD, очень интуитивны. После периода применения пользователи обнаружили, что язык программирования релейных схем прост в изучении и использовании, поэтому релейная схема получила широкое распространение. С непрерывным развитием языка программирования релейных схем и расширением таймеров, счетчиков, математических операций и других сложных операций язык программирования релейных схем стал одним из основных языков программирования ПЛК и других устройств автоматического управления.

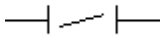
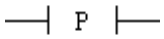
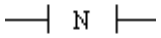
Многие инструкции и функциональные блоки в LD похожи на устройства управления для полевых контуров. Например, нормально открытый контакт в лестничной схеме (-| |-) эквивалентен нормально открытому контакту в поле, а нормально открытая катушка в лестничной схеме (-()-) эквивалентна катушке реле в поле.

Язык программирования LD имеет различные инструкции, функциональные блоки, богатые типы данных и удобные методы адресации, необходимые пользователям для программирования. Знание различных функциональных модулей и методов программирования языка LD является предпосылкой написания краткого и эфективного программирования релейных схем.

Контакты, катушки и функциональные блоки

Контакты

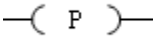
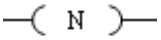
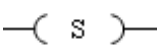
Контакты используются для обнаружения состояний назначенных регистров, типы которых могут быть только BOOL, т.е. ТОЛЬКО 0 или 1. Когда выполняются условия включения контакта, по контакту протекает ток. В состоянии онлайн это отражается на лестничной диаграмме тем, что контакт становится красным, в противном случае контакт становится зеленым. Условия размыкания контакта зависят от состояния заданной точки и типа контакта.

Тип контакта	Описание	ВЫЗОВ
Нормально открытый контакт	Если нормально открытый контакт равен 1, то контакт может проводить ток вправо.	
Нормально замкнутый контакт	Если нормально замкнутый контакт равен 0, то контакт может проводить ток вправо.	
Контакт, чувствительный к положительному переходу	Когда назначенная точка положительного контакта, считывающего переход, имеет изменение с 0 на 1, контакт будет проводить ток вправо и поддерживать в течение одного периода. Когда контакт снова сканируется в следующем цикле, ток больше не проводит, пока изменение с 0 на 1 не произойдет снова.	
Контакт, чувствительный к отрицательному переходу	Когда назначенная точка отрицательного переходного чувствительного контакта имеет изменение с 1 на 0, контакт будет проводить ток вправо и поддерживать в течение одного периода. Когда контакт снова сканируется в следующем цикле, ток больше не проводит, пока изменение с 1 на 0 не произойдет снова.	

Катушки

Катушка используется для управления заданной точкой, которая также должна быть данными типа BOOL. Катушка часто управляется условной логикой и срабатывает только тогда, когда условия на левой стороне катушки удовлетворены и логическая цепь замкнута. Существует также множество типов катушек.

Тип катушки	Описание	ВЫЗОВ
Катушка	Когда ток в катушке включен, установите заданную точку на 1. Катушка не удерживающая, поэтому, когда ток не проводит, катушка больше не будет установлена.	

Инвертированная катушка	Когда ток обратной катушки не проводит, заданная точка устанавливается; Сброс заданной точки при включении тока. Отрицательная катушка также не удерживает.	
Катушка, чувствительная к положительному переходу	При изменении текущего входного сигнала с 0 на 1катушка положительного перехода, заданная точка катушки устанавливается на 1 и сохраняется в течение одного периода. Когда катушка сканируется в следующем периоде, заданная точка возвращается к 0.	
Катушка, чувствительная к отрицательному переходу	При изменении текущего входа катушки, считывающей отрицательный переход, с 1 на 0 заданная точка катушки устанавливается на 1 и сохраняется в течение одного периода. При сканировании катушки в следующем периоде заданная точка возвращается к 0.	
Катушка установки	Когда ток установленной катушки проводит, заданная точка будет установлена в 0. Установленная катушка удерживает, т.е. только когда есть сбросная катушка той же точки, заданная точка изменится с 1 на 0. В противном случае, независимо от того, проводит ток или нет, заданная точка будет поддерживать 1.	
Катушка сброса	Когда ток катушки сброса проводит, заданная точка будет установлена в 1. Катушка сброса удерживает, т.е. только когда есть установленная катушка той же точки, заданная точка будет изменяться с 0 на 1. В противном случае, независимо от того, проводит ток или нет, заданная точка будет поддерживать 0.	

Функциональный блок

Функции и описания параметров функциональных блоков рассматриваются в Главе «Базовый функциональный блок».

Операция

Дважды щелкните левой кнопкой мыши по названию программы LD, после чего содержимое программы LD отобразится в правой области редактирования, и пользователи смогут программировать в ней.

В редакторе LD фон окна — логическая сетка. Слева находится левая шина. Эта шина соответствует активной линии. Во время программирования LD обрабатываются только объекты LD, подключенные к источнику питания, левой шине. Выход внутренней катушки и функционального модуля может быть подключен или не подключен к правой шине и фактически считается подключенным, тем самым устанавливая путь тока.

Различные функциональные блоки занимают разное количество сеток. Контакты и катушки занимают только одну сетку. Пользователи могут переворачивать страницы с помощью PgUp или PgDn, а также могут использовать мышь, чтобы перетаскивать полосу прокрутки справа от области редактирования, чтобы быстро перейти на любую нужную страницу.

При размещении функционального модуля в области редактирования вам нужно только выбрать функциональный блок в «лестничной диаграмме» или панели инструментов лестничной диаграммы строки меню. Щелкните область редактирования левой кнопкой мыши, и функциональный модуль будет помещен в место, на которое нажала мышь; При перемещении функционального блока просто выберите блок и удерживайте кнопку мыши, чтобы переместить его в указанное положение. Операции вырезания, копирования, вставки и удаления функционального блока можно реализовать с помощью опций меню. Операции с несколькими функциональными блоками такие же, как и с одним функциональным блоком, за исключением того, что сначала выбираются несколько функциональных блоков с помощью операций с блоками.

Дважды щелкните на контакте или катушке, и над контактом или катушкой появится поле ввода для ввода требуемых параметров. Параметрический вывод специального функционального модуля также нужно просто дважды щелкнуть, чтобы появилось поле ввода параметров. Как показано на рисунке 6.1

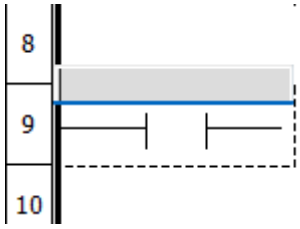


Рисунок 6.1 Входной параметр

Входной параметр может быть в форме Тип + Число, например %I00001, %MW00001. Также может использоваться определенное имя в таблице регистров памяти. Но вход переменной V должен использовать имена.

Связь

Связь – соединение между контактами, катушками, входами и выходами функциональных блоков.

Функция «Магнит»

Когда два функциональных блока находятся на определенном расстоянии, соединение будет построено автоматически, и будет добавлена линия, представляющая связь между двумя функциональными блоками. Когда один из функциональных блоков перемещается, направление линии будет рассчитано автоматически, и связь между этими двумя функциональными блоками будет существовать всегда. Когда один из функциональных блоков удаляется, все линии, подключенные к нему, будут удалены одновременно.

Связать вручную

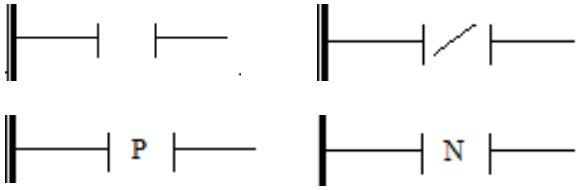
После выбора значка , переместите мышь в начало строки, индикатор мыши примет вид S, щелкните один раз, чтобы выбрать начало; затем перейдите в конец, индикатор изменится на E, щелкните один раз мышью, чтобы завершить подключение.

Реверс

: Реверс – переключение контактов или катушек в цикле.

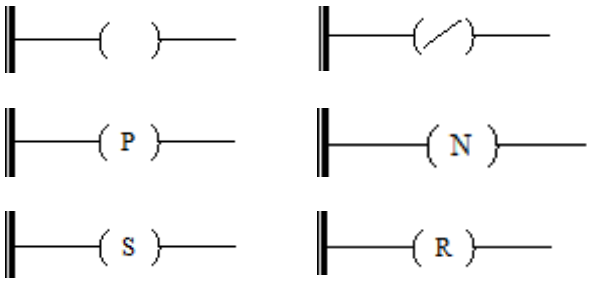
Переключение между контактами

Выберите контакт и один раз щелкните значок реверса . Контакт будет переключаться между следующими четырьмя типами в цикле.



Переключение между катушками

Выберите катушку и щелкните один раз по значку. Катушка будет переключаться между следующими шестью типами в цикле.



Метка

L:: Метка.

Разместите метку

Выберите значок L: и щелкните мышью в окне редактирования, чтобы разместить метку. Метка должна занимать одну строку в LD, как показано на рисунке 6.2

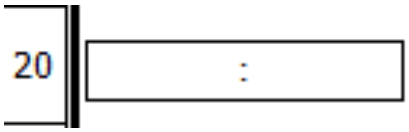


Рисунок 6.2 Метка в окне программы

Добавьте название метки.

Дважды щелкните значок и введите имя в сером окне, например, LABLE.

Рисунок 6.3 Ввод названия метки

: Перейти

Разместите значок

Выберите значок «Перейти» и поместите его в окно редактирования, как показано на рисунке 6.4.

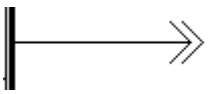


Рисунок 6.4 Элемент «Перейти» в окне программы

Дважды щелкните значок и введите имя в сером окне, например LABLE, что означает, что при изменении с 0 на 1 в точке %I00001 программа переходит в место LABLE.

Рисунок 6.5 Использование элемента «Перейти»

Блок RETURN

: Возвращать.

Когда LD выполняет функциональный блок RETURN, программа возвращается в место вызова подпрограммы и продолжает выполнение. Все последующие программы функционального блока RETURN не будут сканироваться и выполняться. Значок показан на следующем рисунке.

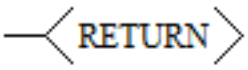


Рисунок 6.6 RETURN в окне программы

Комментарий

: Комментарий — введите комментарий в любом месте окна редактирования.

Вставить одну строку

: Вставить одну строку

Строка — это единица редактора LD. Если вы хотите вставить новые программы LD между верхним и нижним двумя функциональными блоками, вы можете использовать функцию «Вставить строку», чтобы создать достаточно места для новых программ.

Удалить одну строку

: Удалить одну строку

Если между функциональными блоками слишком много пустого пространства, чтобы сделать лестничную диаграмму аккуратной и компактной, лишнее пространство можно удалить с помощью функции «Удалить одну строку».

Программирование FBD

Использование языка программирования FBD для создания программ

Свойства программы FBD

- Сетки в программе FBD представляют собой наименьшее пространство между двумя объектами в FBD.
- Последовательность выполнения зависит от положения FBD (слева направо и сверху вниз). Если FBD подключается к сети с помощью графической связи, последовательность выполнения определяется потоком сигнала.

Создание программ FBD

Выберите [Файл]/[Новая программа], и появится окно, показанное на рисунке 7.1. Выберите FBD и введите имя.

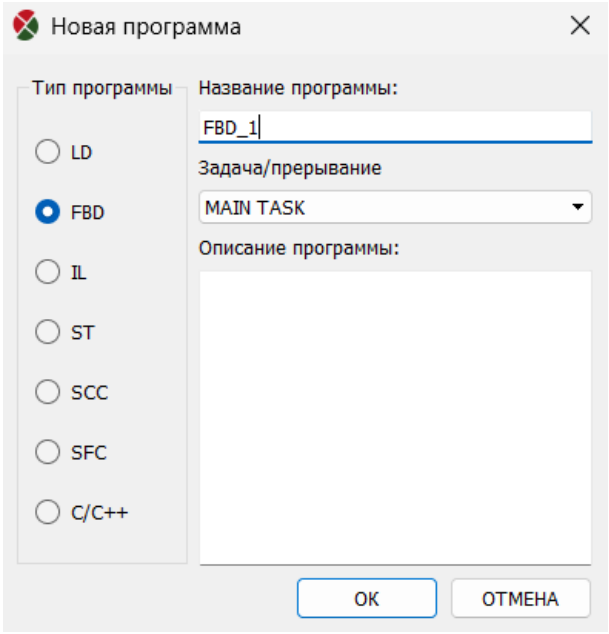


Рисунок 7.1 Схема новой программы

Размещение функциональных блоков

Выберите нужные функциональные блоки через меню. В зависимости от различных функций основные функциональные блоки можно классифицировать на математические, статические, логические, сравнение, преобразование, перемещение, таймер, счетчик, управление, ПЛК, специальные и определяемые пользователем. На панели инструментов, как показано на рисунке 7.2, сначала выберите тип, а затем выберите конкретные функциональные блоки, как показано на рисунке 7.3. Конкретные функции и параметры всех функциональных блоков указаны в главе 5.

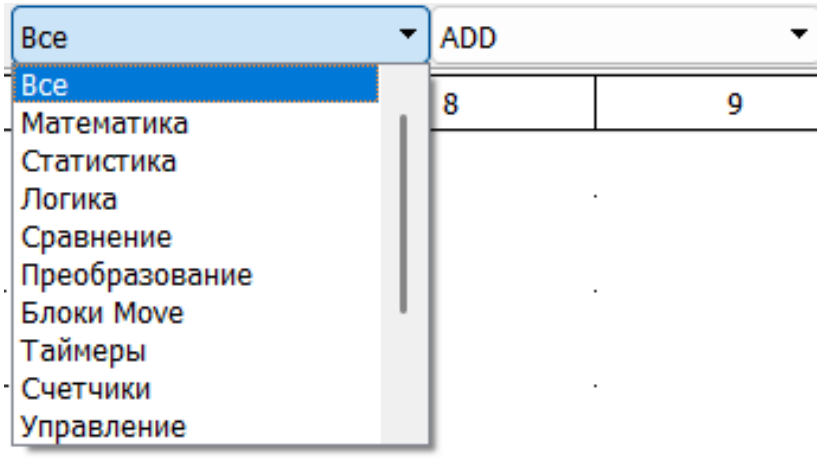


Рисунок 7.2 Классификация функциональных блоков

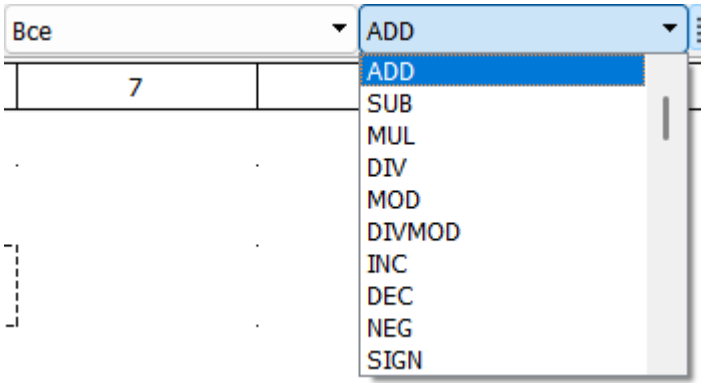


Рисунок 7.3 Выбор функционального блока

Изменение свойства функционального блока

Свойства всех функциональных блоков могут быть изменены. В FBD EN/ENO могут быть отображены/скрыты, а некоторые функциональные блоки могут увеличить количество входных параметров. Щелкните правой кнопкой мыши по выбранному функциональному блоку и выберите свойство, как показано на рисунке 7.4. Появится

диалоговое окно, показанное на рисунке 7.5, и пользователи смогут изменить соответствующие параметры.

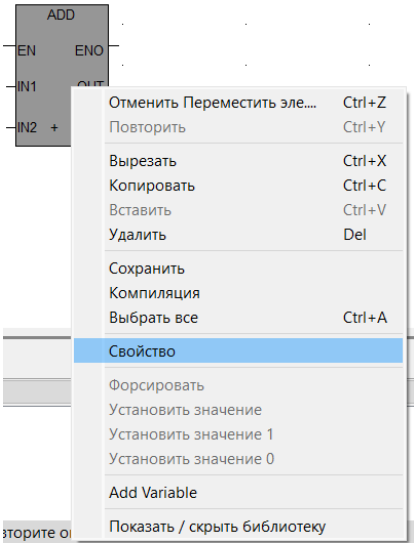


Рисунок 7.4 Открытия свойств функционального блока

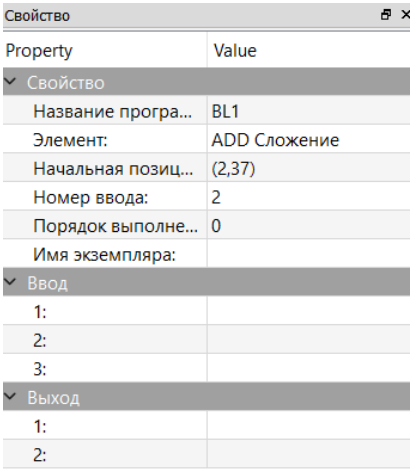


Рисунок 7.5 Свойство функционального блока

Операция

Связь

: Связь – соединение между контактами, катушками и входами и выходами функциональных блоков.

Функция «Магнит»

Когда два функциональных блока находятся на определенном расстоянии, соединение будет построено автоматически, и будет добавлена линия, представляющая связь между двумя функциональными блоками. Когда один из функциональных блоков перемещается, направление линии будет рассчитано автоматически, и связь между этими двумя функциональными блоками будет существовать всегда. Когда один из функциональных блоков удаляется, все линии, подключенные к нему, будут удалены одновременно.

Связать вручную

После выбора значка , переместите мышь в начало строки, индикатор мыши примет вид S, щелкните один раз, чтобы выбрать начало; затем перейдите в конец, индикатор изменится на E, щелкните один раз мышью, чтобы завершить подключение.

Отрицание

: Отрицание – Отрицание выходов и входов.

Как показано на рисунке 7.6, контакты IN1, IN2 функционального блока AND действительны для входа 0 после инвертирования, а IN3 по-прежнему действителен для входа 1.

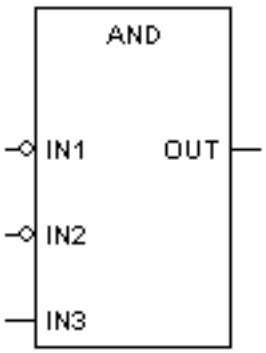


Рисунок 7.6 Отрицание в функциональном блоке

Метка

L: : Метка – знак перехода к.

Поместите метку

Выберите значок L: и щелкните мышью в окне редактирования, чтобы разместить метку. Метка должна занимать одну строку в LD, как показано на рисунке 7.7

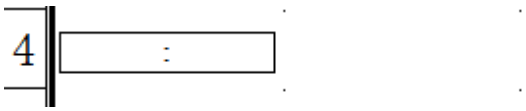


Рисунок 7.7 Метка в окне программы LD/FBD

Добавьте название метки.

Дважды щелкните значок и введите имя в сером окне, например, LABLE.

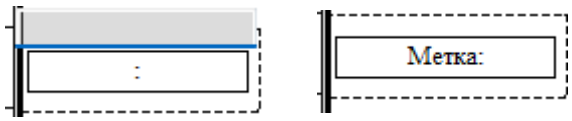


Рисунок 7.8 Ввод названия метки

» : Перейти

Разместите значок

Выберите значок «Перейти» » и поместите его в окне редактирования, как показано на рисунке 7.9.

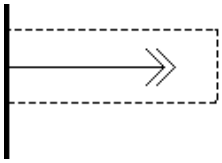


Рисунок 7.9 Элемент «Перейти»

Добавьте имя «Перейти»

Дважды щелкните значок и введите имя в сером окне, например LABLE, что означает, что при изменении с 0 на 1 в точке %I00001 программа переходит в место LABLE.

Рисунок 7.10 Использование элемента «Перейти»

Вернуться

: Вернуться

Когда LD выполняет функциональный блок RETURN (вернуться), программа возвращается в место вызова подпрограммы и продолжает выполнение. Все последующие программы функционального блока RETURN не будут сканироваться и выполняться. Значок показан на следующем рисунке.

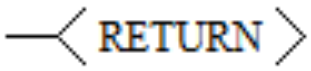


Рисунок 7.11 Операция «Вернуться»

Комментарий

: Комментарий — введите комментарий в любом месте окна редактирования.

Вставить одну строку

: Вставить одну строку

Строка — это единица редактора LD. Если вы хотите вставить новые программы LD между верхним и нижним двумя функциональными блоками, вы можете использовать функцию «Вставить одну строку», чтобы создать достаточно места для новых программ.

Удалить одну строку

: Удалить одну строку

Если между функциональными блоками слишком много пустого пространства, чтобы сделать лестничную диаграмму аккуратной и компактной, лишнее пространство можно удалить с помощью функции «Удалить одну строку».

Программирование на языке IL

Язык программирования IL позволяет вызывать функциональные блоки и функции, назначать значения и осуществлять переключение в различных зонах.

Язык IL включает в себя ряд инструкций. Каждая инструкция включает следующее содержание:

- Оператор
- При необходимости — один определитель.
- При необходимости один или несколько рабочих номеров.

Если используются несколько рабочих номеров, они должны быть разделены общим.

Перед инструкцией можно использовать метку. За этой меткой будет следовать двоеточие. Комментарии можно добавлять в любом месте редактора IL.

Структура языка программирования

IL — это так называемый язык, разработанный для аккумулятора, т. е. каждая инструкция будет использовать или изменять текущее значение аккумулятора. Поэтому IL всегда начинается с оператора LD, который является инструкцией загрузки аккумулятора.
Пример добавления:

Инструкция	Описание
LD 10	Загрузить 10 в аккумулятор
ADD 25	Значение аккумулятора увеличивается на 25.
ST A	Сохранить результат в ‘A’. Значение в ‘A’ и аккумуляторе равно 35. Если следующая инструкция не начинается с LD, то значение аккумулятора равно 35.

Оператор сравнения также часто использует аккумулятор. Результат сравнения типа BOOL всегда сохраняется в аккумуляторе как текущее значение аккумулятора.
Пример сравнения:

Инструкция	Описание
LD B	Загрузить «B» в аккумулятор.
GT 10	Сравнить B с 10.
ST A	Сохранить результат в ‘A’. Если ‘B’ меньше или равно 10, значение ‘A’ и аккумулятора равно 0 (ЛОЖЬ). Если ‘B’ больше 10, значение ‘A’ и аккумулятора равно 1 (ИСТИНА).

Последовательность выполнения

Инструкции выполняются сверху вниз.

Если значения «A», «B», «C» и «D» равны 1, 2, 3 и 4, расчет выполняется следующим образом:

LD	A
ADD	B
SUB	C
MUL	D
ST	E

Результат в «E» — 0.

Если вычислено следующим образом:

LD	A
ADD	B
SUB	C
MUL	D
ST	E

Тогда результат в «E» равен -9.

Описание инструкции

Рабочее число

Рабочее число может быть константой, регистрами, переменными и т. д.

Определитель

Определитель N используется для побитового изменения значения операндов на противоположное.

Пример определителя N: в следующих примерах, если «A» равно 1, а «B» равно 0, то «C» равно 1.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор.
ANDN B	Выполнить операцию «И» между значением аккумулятора и инвертированным значением ‘B’.
ST C	Сохранить результат в ‘C’.

Определитель C используется для выполнения инструкции, когда значение аккумулятора равно 1 (ИСТИНА).

Пример определителя C: в следующем примере, только когда «A» и «B» оба равны 1, выполняется инструкция «Jump to НАЧАЛО».

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «И» между значением аккумулятора и инвертированным значением ‘B’.
JUMP НАЧАЛО	Когда значение аккумулятора равно 1, тогда перейти к метке НАЧАЛО для выполнения инструкции.

Определитель CN используется для выполнения инструкции, когда значение аккумулятора равно 0 (ЛОЖЬ).

Пример определителя CN: в следующем примере, только когда «A» равно 0 или «B» равно 0, выполняется инструкция «Jump to НАЧАЛО».

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор.

AND B	Выполнить операцию «И» между значением аккумулятора и инвертированным значением ‘B’.
JUMP НАЧАЛО	Когда значение аккумулятора равно 0, тогда перейти к метке НАЧАЛО для выполнения инструкции.

Определитель ‘(’ and ‘)’. Скобки могут быть вложены друг в друга. Количество правых скобок должно быть равно количеству левых скобок.

Пример ‘(’ and ‘)’: в следующем примере, если «C» или «D» равно 1, то «E» равно 1 только тогда, когда «A» и «B» оба равны 1.

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор
AND B	выполнить побитовую операцию AND (И) между значением аккумулятора и «B», но отложить выполнение этой операции.
AND (	Операция AND будет выполнена позже, при наступлении определенного события, обозначенного как «правая скобка».
LD C	Загрузить значение «C» в аккумулятор.
OR D	Выполнить операцию «ИЛИ» между значением «D» и аккумулятором.
)	Выполняется отложенная операция AND (И), между результатом отложенной операции AND (между «A» и «B») и текущим значением аккумулятора.
ST E	Сохранить значение в «E».

Оператор

Оператор	**Описание оператора**	**Тип оператора**	**Тип операнда**
LD	Загрузить значение операнда в аккумулятор	N	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V
ST	Сохранить значение аккумулятора в операнде	N	Q, QW, M, MW, N, NW, V
S	Если значение аккумулятора равно 1, то установить операнд равным 1.		Q, M, N, V
R	Если значение аккумулятора равно 1, то установите операнд равным 0.		Q, M, N, V
AND	Логическое И	N, N(, (	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V
OR	Логическое ИЛИ	N, N(, (	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V
XOR	Исключающее ИЛИ	N, N(, (	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V

ADD	Сложение	(	CONSTANT, IW, QW, MW, NW, SW, V
SUB	Вычитание	(	CONSTANT, IW, QW, MW, NW, SW, V
MUL	Умножение	(	CONSTANT, IW, QW, MW, NW, SW, V
DIV	Деление	(	CONSTANT, IW, QW, MW, NW, SW, V
MOD	Остаток от деления	(	CONSTANT, IW, QW, MW, NW, SW, V
GT	Больше чем: >	(	CONSTANT, IW, QW, MW, NW, SW, V
GE	Больше или равно: >=	(	CONSTANT, IW, QW, MW, NW, SW, V
EQ	Равен: =	(	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V
NE	Не равен: <>	(	CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V
LE	Меньше или равно: <=	(	CONSTANT, IW, QW, MW, NW, SW, V
LT	Менее чем: <	(	CONSTANT, IW, QW, MW, NW, SW, V
JMP	Перейти к	C, CN	Метка
CAL	Вызов	C, CN	Функциональный блок, программа
RET	Вернуться	C, CN	Отсутствует

Метка

Метка используется для перехода к указанной цели, и метка должна быть первым элементом одной строки; метка должна быть уникальной во всей программе и не изменяться в зависимости от условий; Метки могут быть длиной до 24 символов; имена меток должны соответствовать соглашению об именах IEC; Метка должна быть отделена от следующей инструкции двоеточием «:»; Метка может появляться только в начале выражения, в противном случае в аккумуляторе появится неопределенное значение.

Комментарий

В редакторе IL комментарии всегда заканчиваются строками символов, начинающимися с '(*' и заканчивающимися на '*)'. Пользователи могут вставлять любые строки символов между двумя строками символов, а комментарии окрашиваются и отображаются.

Или закомментировать одну строку строкой символов '//'.

Оператор

Загрузить (LD и LDN)

Используйте LD для загрузки операнда в аккумулятор. Количество бит данных в аккумуляторе будет автоматически скорректировано в соответствии с типом данных операнда. Вышеуказанная операция может также применяться к типу производного типа данных.

Пример LD выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить «A» в аккумулятор.
ADD B	Сложить значение аккумулятора со значением 'B'
ST C	Сохранить результат в 'C'

Нагруженный операнд может быть инвертирован с помощью определителя N.

Пример LDN выглядит следующим образом:

Инструкция	Описание
LDN A	Побитово инвертируйте значение A, а затем загрузите в аккумулятор.
ADD B	Прибавить значение аккумулятора к 'B'
ST C	Сохранить результат в 'C'

Сохранить (ST и STN)

Используйте ST для сохранения текущего значения аккумулятора в операнте. Поэтому тип данных операнта должен соответствовать типу данных аккумулятора. Использовать старый результат в последующих вычислениях или нет, зависит от того, есть ли LD.

Примеры ST следующие:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор.
ADD B	Прибавить значение аккумулятора к «B».
ST C	Сохранить результат в «C»
ADD D	Добавить значение «C» к «D»

ST E	Сохранить значение в «E»
LD F	Загрузить значение «F» в аккумулятор.
SUB 2	Вычесть 2 из значения аккумулятора
ST G	Сохранить результат в «G»

Сохраненный операнд может быть инвертирован с помощью определителя N.

Пример STN выглядит следующим образом:

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор
ADD B	Сложить значение аккумулятора со значением «B»
STN C	Выполнить побитовую инверсию полученного результата и сохранить его в «C»

Установка(S), сброс (R)

Если текущее значение аккумулятора равно BOOL 1, а S устанавливает операнд в 1.

Пример S выглядит следующим образом:

Инструкция	**Описание**
LD A	Загрузить «A» в аккумулятор
S B	Если значение аккумулятора (т.е. значение «A») равно 1, то установить «B» равным 1.

Этот операнд всегда используется с операндом сброса R.

Пример триггера RS:

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор.
S B	Если значение аккумулятора (т.е. значение «A») равно 1, то установить «B» равным 1.
LD C	Загрузить значение «C» в аккумулятор.
R B	Если значение аккумулятора (т.е. значение «C») равно 1, то установить «B» равным 0.

Если текущее значение аккумулятора равно BOOL 1, а R устанавливает оперант в 0.

Пример R выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить «A» в аккумулятор
R B	Если значение аккумулятора (т.е. значение «A») равно 1, то установить «B» равным 0.

Этот операнд всегда используется с операндом множества S.

Пример триггера SR:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор.
R B	Если значение аккумулятора (т.е. значение «A») равно 1, то установить «B» равным 0.
LD C	Загрузить значение «C» в аккумулятор.
S B	Если значение аккумулятора (т.е. значение «C») равно 1, то установить «B» равным 1.

Логическая операция

AND (AND, AND(), ANDN, ANDN())

Операция И выполняется между значением аккумулятора и операнда. Для целочисленного типа данных операцию «И» следует выполнять побитно.

В следующем примере, если «A», «B» и «C» равны 1, то «D» равен 1.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением аккумулятора и «B»
AND C	Выполнить операцию «И» между значением аккумулятора и «C»
ST D	Сохранить результат в «D»

AND можно использовать одновременно с левой скобкой.

В следующем примере, если «A» равно 1 и один из «B» и «C» равен 1, то «D» равно 1.

Инструкция	Описание

LD A	Загрузить значение «A» в аккумулятор.
AND (	AND задерживается до тех пор, пока не появится правая скобка.
LD B	Загрузить значение «B» в аккумулятор.
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором.
)	Выполнить операцию «И» между значением аккумулятора и «A»
ST D	Сохранить результат в 'D'

AND можно использовать с определителем N одновременно.

В следующем примере, если «A» равно 1, а «B» и «C» равны 0, то «D» равно 1.

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор.
ANDN B	Инвертировать значение «B» на отрицательное и выполнить операцию «И» между ним и значением аккумулятора.
ANDN C	Инвертировать значение «C» на отрицательное и выполнить операцию «И» между ним и значением аккумулятора.
ST D	Сохранить результат в «D»

AND можно использовать с определителем N и левой скобкой '(' одновременно.

В следующем примере, если «A» равно 1, «B» равно 0 и «C» равно 1, то «D» равно 1.

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор.
ANDN (	И задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор.
ORN C	После отрицания значения «C» Выполнить операцию «ИЛИ» со значением аккумулятора.
)	Запустите операцию И, Выполнить операцию «И» между «A» и аккумулятором.
ST D	Сохранить результат в «D»

OR (OR, OR(), ORN, ORN())

Операция ИЛИ выполняется между значением аккумулятора и операнда. Для целочисленного типа данных операцию ИЛИ выполняют побитно.

В следующем примере, если «A» или «B» равно 1 и «C» равно 1, то «D» равно 1.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор.
OR B	Выполнить операцию «ИЛИ» между значением аккумулятора и «B»
AND C	Выполнить операцию «И» между значением аккумулятора и «C»
ST D	Сохранить результат в «D»

OR можно использовать одновременно с левой скобкой '('.

В следующем примере, если «A» равно 1 или «B» и «C» оба равны 1, то «D» равно 1.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
OR (	ИЛИ задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
AND C	Выполнить операцию «И» между значением «C» и аккумулятором.
)	Выполнить операцию «ИЛИ» между значением аккумулятора и «A»
ST D	Сохранить результат в 'D'

OR можно использовать с определителем N одновременно.

В следующем примере, если «A» равно 1 или если «B» и «C» оба равны 1, то «D» равно 1.

Инструкция	Описание
LD A	Загрузить «A» в аккумулятор
ORN B	Сделать инверсию значения «B» на отрицательное и выполнить операцию «ИЛИ» со значением аккумулятора.
AND C	Выполнить операцию «И» между значением аккумулятора и «C»
ST D	Сохранить результат в 'D'

OR можно использовать с определителем N и левой скобкой одновременно.

В следующем примере, если «A» равно 1 или один из «B» и «C» равен 0, то «D» равно 1.

Инструкция	Описание

LD A	Загрузить значение «A» в аккумулятор
ORN (	Операция «ИЛИ» задерживается до тех пор, пока не появится правая скобка.
LD B	Загрузить значение «B» в аккумулятор
AND C	Выполнить операцию «И» между значением аккумулятора и «C»
)	Вычислить значение аккумулятора и выполнить операцию «ИЛИ» со значением «A»
ST D	Сохранить результат в 'D'

XOR (XOR, XOR(), XORN, XORN())

Операция XOR выполняется между значением аккумулятора и операнда. Для целочисленного типа данных операцию XOR выполняют побитно.

В следующем примере, если один из «A» и «B» равен 1, а другой равен 0, то «D» равен 1. Если «A» и «B» одинаковы, то «D» равен 0.

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор.
XOR B	Выполнить операцию «Исключающее ИЛИ» между значением аккумулятора и «B»
ST D	Сохранить результат в «D»

XOR можно использовать одновременно с левой скобкой.

В следующем примере, если одно из «A» и «B AND C» равно 1, а другое равно 0, то «D» равно 1. Если «A» и «B AND C» одинаковы, то «D» равно 0.

Инструкция	**Описание**
LD A	Загрузить «A» в аккумулятор.
XOR (	«Исключающее ИЛИ» откладывается до тех пор, пока не появится правая скобка.
LD B	Загрузить «B» в аккумулятор.
AND C	Выполнить операцию «И» между значением аккумулятора и «C»
)	Выполнить операцию «Исключающее ИЛИ» между значением аккумулятора и «A»
ST D	Сохранить результат в 'D'

XOR можно использовать одновременно с определителем N.

В следующем примере, если «A» и «B» одинаковы, то «C» равен 1. Если «A» и «B» различны, то «C» равен 0.

Инструкция	Описание
LD A	Загрузить «A» в аккумулятор.
XORN B	Выполнить операцию XOR между отрицательным значением «B» и значением аккумулятора.
ST C	Сохранить результат в 'C'

XOR можно использовать с определителем N и левой скобкой '(' одновременно.

В следующем примере, если «A» и «B AND C» одинаковы, то «D» равен 1. Если «A» и «B AND C» различны, то «D» равен 0.

Инструкция	Описание
LD A	Загрузить «A» в аккумулятор.
XORN (	«Исключающее ИЛИ» откладывается до тех пор, пока не появится правая скобка.
LD B	Загрузить «B» в аккумулятор.
AND C	Выполнить операцию «И» между значением «C» и аккумулятором.
)	Выполнить операцию «Исключающее ИЛИ» между значением аккумулятора и «A»
ST D	Сохранить результат в «D»

Арифметическая операция

Сложение (ADD и ADD())

Сложить значение операнда со значением аккумулятора.

В следующем примере соблюдается уравнение D=A+B+C.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
ADD B	Сложить значение «B» со значением аккумулятора.
ADD C	Сложить значение «C» со значением аккумулятора.
ST D	Сохранить результат в 'D'

ADD можно использовать одновременно с левой скобкой '('.

В следующем примере соблюдается уравнение D=A+(BC)

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
ADD (	ADD откладывается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Вычесть значение «C» из аккумулятора.
)	Сложить значение «A» со значением аккумулятора
ST D	Сохранить результат в 'D'

Вычитание (SUB и SUB())

Значение аккумулятора вычитается из значения операнда с помощью SUB.

В следующем примере соблюдается уравнение D=A-B-C.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
SUB B	Значение аккумулятора вычитается из значения «B».
SUB C	Значение аккумулятора вычитается из значения «C».
ST D	Сохранить результат в 'D'

SUB можно использовать одновременно с левой скобкой '('.

В следующем примере соблюдается уравнение D=A-(BC).

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
SUB (	SUB задерживается до тех пор, пока не появится правая скобка.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Значение аккумулятора вычитается из значения «C».
)	Значение «A» вычитается из значения аккумулятора.
ST D	Сохранить результат в 'D'

Умножение (MUL и MUL())

Умножить значение аккумулятора на операнд.

В следующем примере соблюдается уравнение D=A×B×C.

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
MUL B	Умножить значение аккумулятора на «B»
MUL C	Умножить значение аккумулятора на «C»
ST D	Сохранить значение в «D»

MUL можно использовать одновременно с левой скобкой.

В следующем примере соблюдается уравнение D=A×(B-C).

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
MUL (	MUL задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Значение аккумулятора вычитается из значения «C».
)	Умножить значение A и аккумулятора
ST D	Сохранить результат в 'D'

Деление (DIV и DIV())

Значение аккумулятора делится на операнд.

В следующем примере соблюдается уравнение D=A/B/C

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
DIV B	Значение аккумулятора, деленное на «B»
DIV C	Значение аккумулятора, деленное на «C»
ST D	Сохранить значение в 'D'

DIV можно использовать одновременно с левой скобкой.

В следующем примере соблюдается уравнение D=A/(B-C)

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор

DIV (	DIV задерживается, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Значение аккумулятора вычитается из «C».
)	Значение «A», деленное на значение аккумулятора
ST D	Сохранить значение в 'D'

Модуль (MOD и MOD())

По MOD содержимое аккумулятора делится на операнд. Значение остатка — модуль.

В следующем примере соблюдается уравнение C=A-(A/B)*B). (A/B) — целая часть результата операции деления.

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор
МОД Б	Значение остатка получается путем деления содержимого аккумулятора на «B».
CT C	Сохранить результат в «C»

MOD можно использовать одновременно с левой скобкой.

В следующем примере соблюдается уравнение D=A-(A/(BC))*(BC).

Инструкция	**Описание**
LD A	Загрузить значение «A» в аккумулятор
MOD (	MOD задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Значение аккумулятора вычитается из «C».
)	Значение остатка получается путем деления содержимого аккумулятора на «A».
ST D	Сохранить результат в 'D'

Операция отношения

Больше (GT и GT())

С помощью GT сравнить значение аккумулятора и операнда. Если значение аккумулятора больше операнда, то результат в BOOL равен 1; если значение аккумулятора меньше операнда, то результат в BOOL равен 0.

Пример GT выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
GT 10	Сравнить значение аккумулятора с «10»
ST B	Если значение «A» больше «10», сохранить значение «1» в «B». Если значение «A» меньше или равно «10», сохранить значение «0» в «B»

GT можно использовать одновременно с левой скобкой.

Пример GT() выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
GT (	GT задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Значение аккумулятора вычитается из «C».
)	Сравнить значение «A» с аккумулятором
ST D	Если значение «A» больше, чем «B-C», сохранить значение «1» в «D». если значение «A» меньше или равно «B-C», сохранить значение от «0» до «D»

Больше или равно (GE и GE())

С помощью GE сравнить значение аккумулятора и операнта. Если значение аккумулятора больше или равно операнту, то результат равен 1 в BOOL; если значение аккумулятора меньше операнта, то результат равен 0 в BOOL.

Пример GE выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
GE 10	Сравнить значение аккумулятора с «10»
ST B	Если значение «A» больше или равно «10», сохранить значение «1» в «B». если значение «A» меньше «10», сохранить значение «0» в «B»

GE можно использовать одновременно с левой скобкой.

Пример GE() выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
GE (	GE задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Вычесть значение «C» из аккумулятора
)	Сравнить значение «A» с аккумулятором
ST D	Если значение «A» больше или равно «B-C», сохранить значение «1» в «D». если значение «A» меньше, чем «B-C», сохранить значение «0» в «D»

Равно (EQ и EQ())

С помощью EQ сравнить значение аккумулятора и операнда. Если значение аккумулятора равно операнду, то результат в BOOL равен 1; если значение аккумулятора НЕ равно операнду, то результат в BOOL равен 0.

Пример отношения «EQ» выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
EQ 10	Сравнить значение аккумулятора с «10»
ST B	Если значение «A» равно «10», сохранить значение «1» в «B». Если значение «A» НЕ равно «10», сохранить значение «0» в «B»

EQ можно использовать одновременно с левой скобкой.

Пример EQ() выглядит следующим образом:

Инструкция	Описание
LD A	Перезагрузить значение «A» в аккумулятор.
EQ (	EQ задерживается до тех пор, пока не появится правая скобка.
LD B	Загрузить значение «B» в аккумулятор
SUB C	Вычесть значение «C» из аккумулятора.
)	Сравнить значение «A» и аккумулятора
ST D	Если значение «A» равно «B-C», сохранить значение «1» в «D». Если значение «A» НЕ равно «B-C», сохранить значение «0» в «D»

Не равно (NE и NE())

По NE, сравнить значение аккумулятора и операнда. Если значение аккумулятора НЕ равно операнду, то результат равен 1 в BOOL; если значение аккумулятора равно операнду, то результат равен 0 в BOOL.

Пример NE выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
NE 10	Сравнить значение аккумулятора с «10»
ST B	Если значение «A» равно «10», сохранить значение «0» в «B». Если значение «A» НЕ равно «10», сохранить значение «1» в «B»

NE можно использовать одновременно с левой скобкой.

Пример NE() выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
NE (	NE задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Вычесть значение «C» из аккумулятора
)	Сравнить значение «A» с аккумулятором
ST D	Если значение «A» равно «B-C», сохранить значение «0» в «D». если значение «A» НЕ равно «B-C», сохранить значение «1» в «D»

Меньше или равно (LE и LE())

С помощью LE сравнить значение аккумулятора и операнда. Если значение аккумулятора меньше или равно операнду, то результат равен 1 в BOOL; если значение аккумулятора больше операнда, то результат равен 0 в BOOL.

Пример LE выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
LE 10	Сравнить значение аккумулятора с «10»
ST B	Если значение «A» больше «10», сохранить значение «0» в «B». Если значение «A» меньше или равно «10», сохранить значение «1» в «B».

LE можно использовать одновременно с левой скобкой.

Пример LE() выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
LE (	LE задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Вычесть значение «C» из аккумулятора
)	Сравнить значение «A» с аккумулятором
ST D	Если значение «A» больше, чем «B-C», сохранить значение «0» в «D». если значение «A» меньше или равно «B-C», сохранить значение «1» в «D»

Меньше чем (LT и LT())

С помощью LT сравнить значение аккумулятора и операнда. Если значение аккумулятора меньше операнда, то результат равен 1 в BOOL; если значение аккумулятора больше или равно операнду, то результат равен 0 в BOOL.

Пример LT выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
LT 10	Сравнить значение аккумулятора с «10»
ST B	Если значение «A» меньше «10», сохранить значение «1» в «B». Если значение «A» больше или равно «10», сохранить значение «0» в «B»

LT можно использовать одновременно с левой скобкой.

Пример LT() выглядит следующим образом:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
LT (	LT задерживается до тех пор, пока не появится правая скобка
LD B	Загрузить значение «B» в аккумулятор
SUB C	Вычесть значение «C» из аккумулятора
)	Сравнить значение «A» с аккумулятором
ST D	Если значение «A» больше или равно «B-C», сохранить значение от «0» до «D».

	если значение «A» меньше, чем «B-C», сохранить значение «1» в «D»

Перейти к (JMP, JMPC и JMPCN)

JMP

С помощью JMP реализуем безусловный переход к метке.

Пример следующий:

Инструкция	Описание
начало: LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между аккумулятором и значением «B»
OR C	Выполнить операцию «ИЛИ» между аккумулятором и значением «C»
ST D	Сохранить результат в 'D'
JMP начало	Независимо от значения аккумулятора, перейти к метке НАЧАЛО.

JMPC

По JUMPC, если значение условия равно 1, то переходим к метке.

Пример следующий:

Инструкция	Описание
начало: LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
Начало JMPC	Только когда значение аккумулятора равно 1, перейти к метке НАЧАЛО.

JMPN

По JUMPN, если значение условия равно 0, то переходим к метке.

Пример следующий:

Инструкция	Описание
начало: LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором

Начало JMPN	Только когда значение аккумулятора равно 0, перейти к метке НАЧАЛО.

Вызов (CAL, CALC и CALCN)

CAL

С помощью CAL реализуйте безусловный вызов функционального блока и подпрограмм.

Пример следующий:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
ST D	Сохранить результат в «D»
CAL IL1	Независимо от значения аккумулятора завершить вызов подпрограммы IL1.

CALC

По CALC, если значение условия равно 1, то реализуется вызов функционального блока и подпрограммы.

Пример следующий:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
CALC IL1	Только когда значение аккумулятора равно 1, завершить вызов подпрограммы IL1.

CALCN

По CALCN, если значение условия равно 0, то реализуется вызов функционального блока и подпрограммы.

Пример следующий:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор

AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
CALC IL1	Только когда значение аккумулятора равно 0, завершить вызов подпрограммы IL1.

Вернуться (RET, RETC и RETCN)

RET

Посредством RET реализуется безусловный возврат в подпрограмму.

Пример следующий:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
ST D	Сохранить результат в 'D'
RET	В зависимости от значения аккумулятора выполнить безусловный возврат из подпрограммы в основную программу.

RETC

По RETC, если значение условия равно 1, реализуем возврат в подпрограмму.

Пример следующий:

Инструкция	Описание
LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
ST D	Сохранить результат в 'D'
RETC	Только когда значение аккумулятора равно 1, завершить возврат из подпрограммы в основную программу.

RETCN

По RETCN, если значение условия равно 1, реализуем возврат в подпрограмму.

Пример следующий:

Инструкция	Описание

LD A	Загрузить значение «A» в аккумулятор
AND B	Выполнить операцию «И» между значением «B» и аккумулятором
OR C	Выполнить операцию «ИЛИ» между значением «C» и аккумулятором
ST D	Сохранить результат в 'D'
RETCN	Только когда значение аккумулятора равно 0, завершить возврат из подпрограммы в основную программу.

Функциональный блок

В зависимости от различных функций основные функциональные блоки можно классифицировать на математические, статические, логические, сравнение, преобразование, перемещение, таймер, счетчик, управление, ПЛК, специальные и определяемые пользователем. На панели инструментов, как показано на рисунке 8.1, сначала выберите тип, а затем выберите конкретные функциональные блоки, как показано на рисунке 8.2. Конкретные функции и параметры всех функциональных блоков рассматриваются в главе 5.

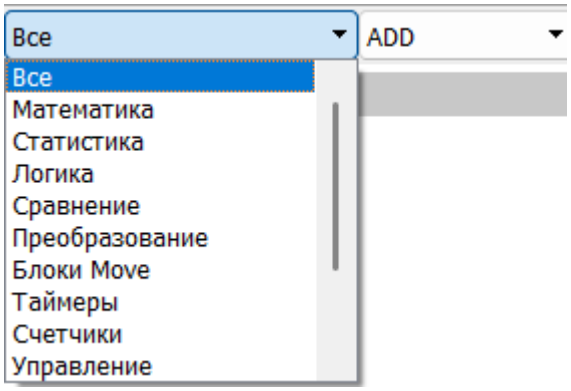


Рисунок 8.1 Диаграмма классификаций

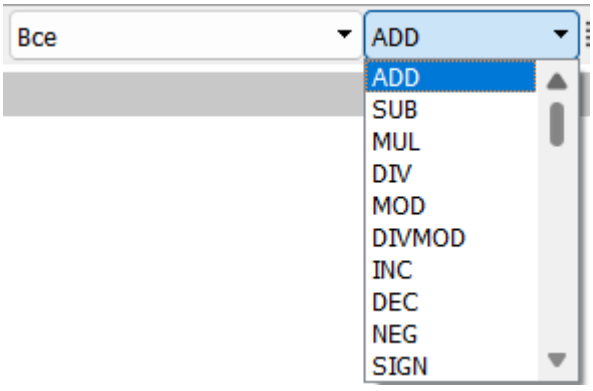


Рисунок 8.2 Схема выбора функционального блока

Программирование на языке ST

Язык ST похож на язык PASCAL, но это язык программирования, предназначенный для разработки приложений промышленного управления с сильными возможностями программирования, используемый для назначения значений переменным и функциональным блокам, создания выражений и написания условных операторов и т. д. ST подходит для приложений со сложными алгоритмическими операциями.

ST, имеющий свободную форму программирования, может вставлять табуляции, символы новой строки и комментарии в любом месте между ключевыми словами и идентификаторами. Для разработчиков, знакомых с компьютерным языком высокого уровня, ST прост в изучении и использовании. Кроме того, язык ST также прост для чтения и понимания, особенно при аннотировании значимыми идентификаторами и комментариями.

Выражение

Выражение состоит из операторов, функций и операндов.

Операнд

Операндами могут быть константы, точки, переменные и т. д.

Таблица операторов

Оператор	Описание	Тип	Уро-вень
()	Использовать параметр	Выражение	1
NOT	НЕ	Выражение, CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V	2
*	Умножение	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	3
/	Деление	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	3
MOD	Остаток от деления (модуль)	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	3
+	Сложение	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	4
-	Вычитание	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	4
<	Меньше чем	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	5
>	Больше, чем	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	5
<=	Меньше или равно	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	5
>=	Больше или равно	Выражение, CONSTANT, IW, QW, MW, NW, SW, V	5
=	Равно	Выражение, CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V	6
<>	Не равно	Выражение, CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V	6
AND	Логическое И	Выражение, CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V	7
XOR	Исключающее ИЛИ	Выражение, CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V	8
OR	Логическое ИЛИ	Выражение, CONSTANT, I, Q, IW, QW, M, MW, N, NW, S, SW, V	9
:=	Присвоить значение	Q, QW, M, MW, N, NW, V	10

Операторы

Скобки – ( )

Используйте скобки для изменения последовательности выполнения операторов.

В следующем примере, если A равно 1, B равно 2, C равно 3, D равно -4, то:

Инструкция	Результат работы
E:=A+BC*D	E=15
E:=(A+BC)*D	E=0

НЕ – NOT

Используйте операцию НЕ, чтобы инвертировать операнд побитно.

В следующем примере, если «A» — это «11011011», то:

Инструкция	Операционный результат
B:=NOT A	B=00100100

Умножение – '*'

С помощью умножения перемножьте значение первого операнта и второго.

В следующем примере, если A равно 2, а B равно 3, то:

Инструкция	Операционный результат
C:=A*B	C=6

Деление - '/'

При делении значение первого операнта делится на второе.

В следующем примере, если A равно 6, а B равно 3, то:

Инструкция	Операционный результат
C:=A/B	C=2

Остаток от деления (модуль) – MOD

При использовании MOD первое значение операнда делится на второе, а остаток отображается в результате.

В следующем примере, если A равно 6, а B равно 4, то:

Инструкция	Операционный результат
C:=A MOD B	C=2

Сложение – '+'

При сложении к значению первого операнда прибавляется значение второго.

В следующем примере, если A равно 6, а B равно 4, то:

Инструкция	Операционный результат
C:=A+B	C=10

Вычитание – '-'

При вычитании значение первого операнда вычитается из второго.

В следующем примере, если A равно 6, а B равно 4, то:

Инструкция	Операционный результат
C:=A-B	C=2

Больше чем - '>'

Используя '>', сравнить значение первого операнда и второго. Если первый операнд больше второго, то результат будет 1 в BOOL. Если первый операнд меньше второго, то результат будет 0 в BOOL.

Пример следующий:

Инструкция	Операционный результат
C:=A>B	If A=6, B=4, THEN C=1.
C:=A>B	If A=2, B=4, THEN C=0
C:=A>B	If A=2, B=2, THEN C=0

Больше или равно – '>='

Используя '>=', сравнить значение первого операнда и второго. Если первый операнд больше или равен второму, то результат равен 1 в BOOL. Если первый операнд меньше второго, то результат равен 0 в BOOL.

Пример следующий:

Инструкция	Операционный результат
C:=A>=B	If A=6, B=4, THEN C=1.
C:=A>=B	If A=2, B=4, THEN C=0
C:=A>=B	If A=2, B=2, THEN C=1

Равен – '='

Используя '=', сравнить значение первого операнда и второго. Если первый равен второму, результат равен 1 в BOOL. Если нет, результат равен 0 в BOOL.

Инструкция	Операционный результат
C:=A=B	If A=6, B=4, THEN C=0
C:=A=B	If A=2, B=4, THEN C=0
C:=A=B	If A=2, B=2, THEN C=1

Не равен – '<>'

С помощью '<>' сравнить значение первого операнда со вторым. Если первый НЕ равен второму, результат равен 1 в BOOL. Если равен, результат равен 0 в BOOL.

Пример следующий:

Инструкция	Операционный результат
C:=A<>B	If A=6, B=4, THEN C=1
C:=A<>B	If A=2, B=4, THEN C=1
C:=A<>B	If A=2, B=2, THEN C=0

Меньше чем – '<'

Используя '<', сравнить значение операнда со вторым. Если первое меньше второго, то результат равен 1 в BOOL. Если первое больше или равно второму, то результат равен 0 в BOOL.

Пример следующий:

Инструкция	Операционный результат
C:=A<Б	IF A=6, B=4, TO C=0

C:=A<Б	IF A=2, B=4, TO C=1
C:=A<Б	IF A=2, B=2, TO C=0

Меньше или равно – '<='

Используя '<=', сравнить значение первого операнда со вторым. Если первый операнд меньше или равен второму, то результат в BOOL равен 1. Если первый операнд больше второго, то результат в BOOL равен 0.

Пример следующий:

Инструкция	Операционный результат
C:=A<=Б	IF A=6, B=4, TO C=0
C:=A<=Б	IF A=2, B=4, TO C=1
C:=A<=Б	IF A=2, B=2, TO C=1

Логическое «И» - AND

Операция «AND» выполняется над целочисленными данными побитно.

Пример следующий:

Инструкция	Операционный результат
D:=A AND B AND C	Если одно из значений A, B, C равно 0, то D равно 0, в противном случае D равно 1.

Логическое «ИЛИ» - OR

Операция И выполняется над целочисленными данными побитно.

Пример следующий:

Инструкция	Операционный результат
D:=A AND B AND C	Если одно из значений A, B, C равно 1, то D равно 1, в противном случае D равно 0.

Исключающее «ИЛИ» - XOR

Операция XOR выполняется над целочисленными данными побитно.

В следующем примере, если A отличается от B, то C равен 1. Если они одинаковы, то C равен 0.

Инструкция	Операционный результат
C:=A XOR B	IF A=1, B=0, THEN C=1
C:=A XOR B	IF A=0, B=1, THEN C=1
C:=A XOR B	IF A=1, B=1, THEN C=0
C:=A XOR B	IF A=0, B=0, THEN C=0

Присвоить значение – ':='

При использовании ':=' текущее значение переменной заменяется присвоенным значением выражения. Присвоение значения включает описание спецификации переменной слева, оператор присваивания и выражение присваивания. Две переменные должны быть одного типа данных.

В следующем примере «:=« используется для присвоения одного значения переменной другому.

Инструкция	Описание
A:=B	Заменить значение переменной A на текущее значение переменной B

В следующем примере присваивание используется для немедленного присвоения числа переменной.

Инструкция	Описание
A:=10	Присвоить значение 10 переменной A.

В следующем примере присваивание используется для присвоения результата функционального блока переменной.

Инструкция	Описание
A:=ABS(B)	Присвоить результат функционального блока ABS переменной A.

В следующем примере присваивание используется для выделения обработанного результата переменной.

Инструкция	Описание
A:=(B+C-D)*E	Присвоить результат (B+CD)*E переменной A.

Инструкции

Инструкция должна заканчиваться точкой с запятой. Пользователи могут вводить несколько инструкций в одну строку. (Инструкции разделяются точкой с запятой).

IF…THEN…ELSE…END_IF

Когда bool номер инструкции, следующей за IF, равен 1 (TRUE), выполняется инструкция или группа инструкций, следующих за THEN. Когда bool номер инструкции, следующей за IF, равен 0 (FALSE), выполняется инструкция или группа инструкций, следующих за ELSE. Инструкция END_IF используется для обозначения конца инструкции.

Пример одиночного IF выглядит следующим образом:

Инструкция	Описание
IF %M1 = 1	Проверить, равно ли значение %M единице
THEN %MW1 := 1;	%M=1, выполнить инструкцию
ELSE %MW1 := 2;	%M=0, выполнить инструкцию
END_IF;	Обозначение конца инструкции

Пример множественных IF выглядит следующим образом:

Инструкция	Описание
IF %M1 = 1	Определить, соответствует ли %M1 критериям
THEN %MW1 := 1;	Если да, выполнить эту инструкцию.
ELSE %MW1 := 2;	Если нет, выполнить эту инструкцию.
IF %M2 = 1	Определить, соответствует ли %M2 критериям?
THEN %MW2 := 1;	Если да, выполнить эту инструкцию.
ELSE %MW2 := 2;	Если нет, выполнить эту инструкцию.
END_IF;	Обозначение конца инструкции.
END_IF;	Обозначение конца инструкции.

ELSEIF Инструкция

Когда bool номер инструкции, следующей за IF, равен 1 (TRUE), выполняется инструкция или группа инструкций, следующих за THEN. Когда bool номер инструкции, следующей за IF, равен 0 (FALSE), выполняется инструкция или группа инструкций,

следующих за ELSE. Когда bool номер инструкции, следующей за ELSEIF, равен 1 (TRUE), выполняется инструкция. Когда bool номер инструкции, следующей за ELSEIF, равен 0 (FALSE), определяется следующая инструкция, следующая за ELSE. Инструкция END_IF используется для обозначения конца инструкции.

Инструкция	**Описание**
IF %M1 = 1	Определить, соответствует ли %M1 критериям?
THEN %MW1 := 1;	Если да, выполнить эту инструкцию и прекратите выполнение следующего оператора IF.
ELSIF %M2 = 1	если нет, определить, является ли %M2=1.
THEN %MW2 := 1;	Если да, выполнить эту инструкцию.
END_IF;	Обозначение конца инструкции

CASE…OF… ELSE… END_CASE

Инструкция CASE состоит из выражений целочисленного типа данных и группы инструкций. Каждая группа имеет метку, состоящую из одного или нескольких целых чисел. Если метки содержат значение вычисленного селектора, инструкция выполняется; Если это значение не включено, инструкция не может быть выполнена. Инструкция OF отмечает начало. Инструкция ELSE может быть выполнена в инструкции CASE, инструкция которой выполняется только в том случае, если тег не содержит никакого значения селектора. END_CASE используется для обозначения конца инструкции.

Пример следующий:

Инструкция	**Описание**
CASE A+B OF	Каким условиям удовлетворяют значения (A+B) в операторе с несколькими ветвями?
1,5: C:=SIN(A) * COS(B);	Если значения равны 1 и 5, данная инструкция выполняется.
2: B:=C-A;	Если значение равно 2, Данная инструкция выполняется.
3,4,6: C:=C*A;	Если значения равны 3, 4 и 6, Данная инструкция выполняется.
ELSE B:=C*A; C:=A / B;	Если все условия не выполнены, то выполняется данная инструкция.
END_CASE;	Обозначение конца инструкции.

FOR…TO…BY…DO…END_FOR

Инструкция FOR используется в ситуации, когда пользователи могут убедиться в количестве совпадающих элементов. Если это невозможно, пользователи могут

использовать только инструкции WHILE или REPEAT. Инструкция FOR используется для повторения последовательности инструкций до тех пор, пока не появится инструкция END_FOR. Количество совпадающих элементов определяется начальным значением, конечным значением и управляющими переменными, которые должны быть того же типа данных и не могут быть изменены повторяющимися инструкциями. Инструкция FOR добавляет начальное значение управляющей переменной к конечному значению. Значение приращения по умолчанию равно 1. Вам следует проверять значения переменных перед запуском каждого нового обновленного цикла. Если значение переменной выходит за пределы диапазона между начальным и конечным, цикл завершится. Перед запуском первого цикла пользователи должны проверить, чтобы убедиться в приращении управляющей переменной. Если нет движения от начального до конечного, цикл не может быть выполнен. Используйте инструкцию FOR для обычного выполнения цикла. Инструкция DO используется для определения конца повторяющегося определения и начала инструкции. Вы можете использовать EXIT для досрочного завершения. END_FOR используется для обозначения конца инструкции.

Пример инструкции FOR с приращением 1: в следующем примере 1 — начальное значение, а 3 — конечное значение.

Инструкция	**Описание**
FOR i:= 1 TO 5 DO	i увеличивается от значения по умолчанию 1, выполнять следующую инструкцию в цикле. Когда i больше 5, закончить инструкцию FOR.
C:= C+4;	Группа инструкций, которые выполняются в цикле.
END_FOR;	Обозначение конца инструкции.

Если приращение не равно 1, приращение можно определить через BY. Направление действия (вперед или назад) определяется знаком константы, следующей за BY. Если знак положительный, цикл будет выполняться вперед, если отрицательный, цикл будет выполняться назад.

Например, FOR, в котором приращение не равно 1. В следующем примере i — управляющая переменная, 1 — начальное значение, а 3 — конечное значение.

Инструкция	**Описание**
FOR i:= 1 TO 5 BY 2 DO	i прибавляется с шагом 2. Когда i больше 5, завершаем инструкцию FOR.
C:= C+4;	Группа инструкций, которые выполняются в цикле.
END_FOR;	Обозначение конца инструкции.

WHILE…DO…END_WHILE

Результатом выполнения инструкции WHILE является то, что последовательность инструкций будет повторяться до тех пор, пока соответствующее выражение BOOL не станет равным 0 (ЛОЖЬ). Если начальное значение выражения равно 0, то группа инструкций не будет выполнена. Инструкция DO используется для идентификации конца повторяющегося определения и начала инструкции. Вы можете использовать EXIT для досрочного завершения. END_WHILE используется для обозначения конца инструкции.

Пример следующий:

Инструкция	Описание
A := 1;	Присвоить константу 1 переменной A.
WHILE A<= 100 DO A := A + 4;	IF A<=100, то выполнить инструкцию цикла A:=A+4; Если A>100, то прекратить выполнение A:=A+4.
END_WHILE;	Обозначение конца инструкции.

REPEAT…UNTIL…END_REPEAT

Результатом выполнения REPAET является то, что последовательность инструкций будет выполняться в цикле (по крайней мере один раз) до тех пор, пока соответствующее выражение BOOL не станет равным 1 (ИСТИНА). Инструкция UNTIL используется для обозначения конца условия. Вы можете использовать EXIT для досрочного завершения. END_REPEAT используется для обозначения конца инструкции.

Пример следующий:

Инструкция	Описание
A := 1;	Присвоить константу 1 переменной A
REPEAT	
A := A + 2	Выполнить инструкцию цикла A:=A+2
UNTIL A >= 100	IF A<100, то выполнить инструкцию цикла A:=A+2; Если A>=100, то прекратить выполнение A:=A+2.
END_REPEAT;	Обозначение конца инструкции.

EXIT

Инструкция EXIT предназначена для досрочного выхода из циклов FOR, WHILE или REPEAT, то есть, до наступления их обычного условия завершения. Если инструкция

EXIT находится внутри одного из этих циклов, то выполнение текущего цикла немедленно прерывается в месте расположения инструкции EXIT. После чего, программа переходит к выполнению первой инструкции, расположенной сразу после окончания цикла (то есть, после END_FOR, END_WHILE или END_REPEAT).

RETURN

Инструкция RETURN используется для прерывания сканирования подпрограммы и возврата к основной программе. После выполнения инструкции RETURN система прервет подпрограмму и немедленно вернется к основной программе.

COMMENT

В редакторе ST комментарий начинается с '(*' и заканчивается на '*)'. Пользователи могут ввести любой комментарий между двумя строками символов, а также в любом месте редактора. Комментарий отображается разными цветами.

Или закомментируйте строку, добавив «//» в начале одной строки.

GOTO

В редакторе ST метка заканчивается на «:». Например, «AA:». Оператор GOTO используется для перехода к назначенной метке, например «GOTO AA;».

Функциональный блок

В зависимости от различных функций основные функциональные блоки можно классифицировать на математические, статические, логические, сравнение, преобразование, перемещение, таймер, счетчик, управление, ПЛК, специальные и определяемые пользователем. На панели инструментов, как показано на рисунке 9.1, сначала выберите тип, а затем выберите конкретные функциональные блоки, как показано на рисунке 9.2. Конкретные функции и параметры всех функциональных блоков рассматриваются в главе 5.

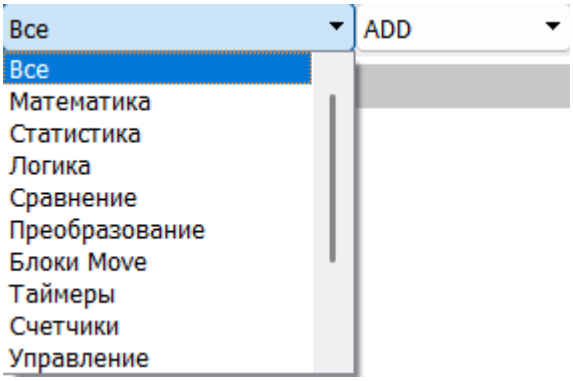


Рисунок 9.1 Диаграмма классификаций

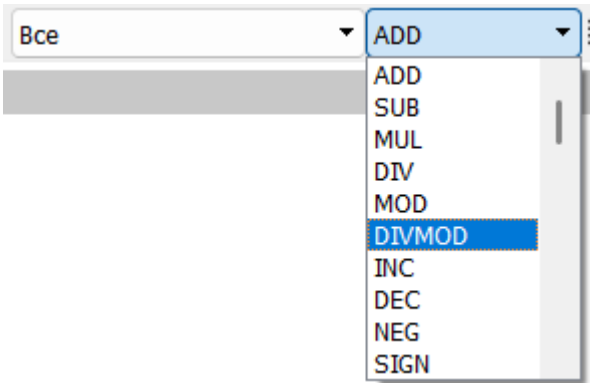


Рисунок 9.2 Схема выбора функционального блока

Программирование Последовательных Управляющих Диаграмм (SCC)

Современное промышленное управление в основном включает последовательное управление, управление процессами и управление движением. Традиционный способ процесса SCC заключается в том, чтобы сначала нарисовать схему управления последовательностью, а затем достичь процесса управления последовательностью с помощью LD, ST и некоторых других языков программирования, что неудобно в эксплуатации и поиске проблем в процессе. Программирование SCC разработано для решения вышеуказанных проблем. Оно описывает процесс управления последовательностью с помощью предписанных графиков, которые легко принимаются пользователями. SCC является наиболее подходящим языком для управления последовательностью, предоставляя простое графическое описание процесса и завершая соответствующую программу управления путем настройки всего процесса управления.

Дважды щелкните имя SCC, и содержимое SCC отобразится в правой области редактирования. Область редактирования SCC управляется на основе страниц, и каждая контрольная карта порядка поддерживает максимум 16 страниц. Размещение функциональных блоков не ограничивается страницами, что означает, что пользователи могут размещать блоки на любой странице. Однако при печати контрольной карты последовательности она печатается с первой по шестнадцатую страницу, поэтому при написании контрольной карты последовательности она должна быть написана как можно аккуратнее.

При размещении функциональных блоков в области редактирования пользователям нужно только выбрать нужные функциональные блоки на панели инструментов и щелкнуть место в области редактирования, куда вы хотите их поместить.

Двойной щелчок по полю функции позволяет редактировать ее свойства. Точки могут быть использованы в форме типа + серийный номер, например, %I00001 (ввод может быть сокращен до %I1), %MW0001 и т. д., а также в имени, определенном в таблице регистров памяти. Однако ввод V должен использовать имена.

Когда два функциональных блока находятся на определенном расстоянии (вверх и вниз), соединение между контактами, соответствующими типу, будет создано автоматически или вручную с помощью пункта «Соединение» в строке меню; Когда относительное положение двух функциональных блоков изменяется, соединение автоматически меняет направление; После удаления функционального блока все подключенные к нему соединения автоматически удаляются.

Диаграмма управления последовательностью представляет собой графическое описание процесса управления. При фактическом использовании реализация языка управления последовательностью должна быть достигнута посредством группы данных.

Операторы

Система допускает следующие операторы: Операции с более высоким приоритетом выполняются первыми, а операции с более низким приоритетом выполняются позже. Операции с одинаковым приоритетом выполняются слева направо. Если операцию с низким приоритетом необходимо выполнить первой, к ней можно добавить скобки.

Оператор	Описание	Приоритет
( )	Скобки	1
!	Отрицание	2
~	Побитовое отрицание	3
*	Умножить	4
/	Разделить	4
+	Сложить	5
-	Вычесть	5
《	Сдвиг влево	6
》	Сдвиг вправо	6
<	Меньше чем	7
<=	Меньше или равно	7
>	Больше чем	7
>=	Больше или равно	7
= =	Равно	8
! =	Не равно	8
&	Побитовое И	9
^	Побитовое Исключающее ИЛИ	10
|	Побитовое ИЛИ	11
&&	Логическое И	12
||	Логическое ИЛИ	13
=	Присваивание	14

Функции

x,y — константы, переменные или выражения.

Функция	Описание
max(x, y)	Максимальное значение
min(x, y)	Минимальное значение
sin(x)	Синус
asin(x)	Арксинус
cos(x)	Косинус
acos(x)	Арккосинус
tan(x)	Тангенс
atan(x)	Арктангенс
ln(x)	Натуральный логарифм
exp(x)	Экспоненциальное уравнение
lg(x)	Логарифм
expt(x, y)	Возведение в степень
abs(x)	Модуль
sqrt(x)	Корень

Выражения

Выражение может быть группой из одной константы, переменных или операторов и операнда. Например, все следующие выражения являются допустимыми.

m_i1 = %MW0001

m_i2 = 5 + var (Примечание: var — это определенное имя %IW0001)

m_i3 = m_i4 * 3

%Q0001 = 1

Переменные

Все непостоянные данные, используемые в контрольной карте последовательности, можно назвать переменными. Переменные состоят из локальных переменных и глобальных переменных.

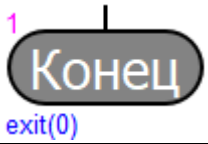
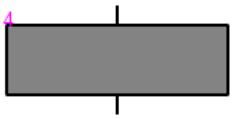
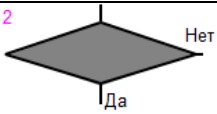
Локальные переменные имеют два типа: int и float. Для любой новой программы система начинает с пяти предопределенных целых чисел m_i1, m_i2, m_i3, m_i4, m_i5; пятью плавающими переменными m_f1, m_f2, m_f3, m_f4, m_f5. Локальные переменные действительны только в текущей программе, а локальные переменные с одинаковыми именами между программами не влияют друг на друга.

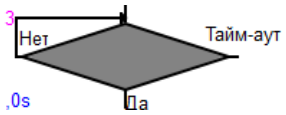
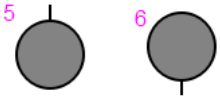
Глобальные переменные — это определенные точки в файле проекта, например %I00001, %MW0001 и т. д. Глобальные переменные также могут использовать имя точки, определенное в таблице регистров памяти, например, %I00001 определяется как DL_ON, который может использоваться в SCC напрямую. Глобальная переменная действительна во всех программах релейных диаграмм и других программах, и, если значение контрольной точки изменяется в любой контрольной диаграмме последовательности, значение контрольной точки изменится во всех программах.

Функциональные блок-схемы

Перед программированием в SCC пользователи должны быть знакомы с различными функциональными блоками. При размещении функциональных блоков в области редактирования пользователям нужно только выбрать нужные функциональные блоки на панели инструментов и щелкнуть место в области редактирования, куда вы хотите их поместить. Двойной щелчок по функциональным блокам позволяет редактировать их свойства.

Функциональные блоки показаны в следующей таблице:

Блок-схема функции	Описание	Блок-схема
Блок «Начало»	Блок «Начало» должен присутствовать и быть уникальным в каждой диаграмме последовательного управления.	
Блок «Конец»	Общая диаграмма последовательного управления должна иметь блок «Конец», и может быть несколько блоков «Конец».	
Блок «Выполнение»	Когда вам нужно использовать операцию, вы можете выбрать блок «Выполнение».	
Блок «Условие»	Используется для определения, истинно ли условие.	

Блок «Условие с ограничением по времени»	Используется для определения, истинно ли условие в течение определенного времени.	
Соединитель	Используйте соединение, чтобы связать несколько частей программы вместе или для перехода между различными частями программы.	

Блок «Начало»

Начальный блок должен присутствовать и быть уникальным в каждой контрольной карте последовательности, это начало контрольной карты последовательности, в противном случае программное обеспечение INDAS PRO выдаст сообщение об ошибке во время компиляции. Вам не нужно задавать параметры для начального блока. На рисунке 10.1 показано:

Рисунок 10.1 Блок «Начало»

Блок «Конец»

В общем случае, диаграмма контроля последовательности должна иметь конечный блок, может быть несколько конечных блоков, и диаграмма контроля последовательности выходит из конца разных ветвей. Для конечного блока не требуются никакие параметры. Отдельные диаграммы контроля последовательности, которые должны всегда зацикливаться, могут не иметь конечного блока, как показано на рисунке 10.2.

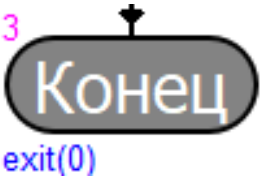


Рисунок 10.2 Блок «Конец»

Блок «Выполнение»

Когда вам необходимо выполнить операцию, вы можете выбрать поле выполнения.

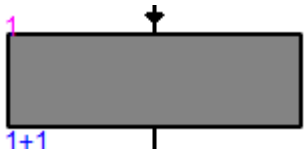


Рисунок 10.3 Блок «Выполнение»

В поле выполнения есть 13 основных операций, которые можно выбрать в соответствии с вашими потребностями. После отображения поля выполнения вы можете выбрать тип операции из списка основных операций и дважды щелкнуть, чтобы задать параметры для этой операции.

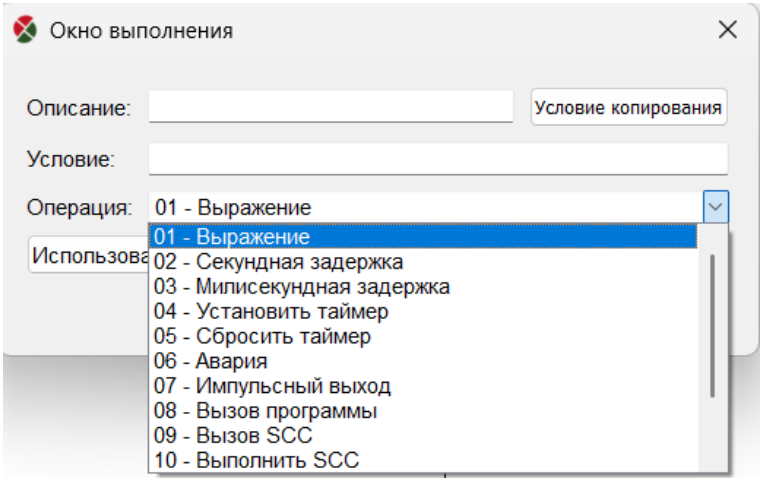


Рисунок 10.4 Настройка свойств поля выполнения

Выражение

Выражение может завершить назначение, чтение данных, операцию «И» т. д. Пользователи могут использовать различные операторы, представленные слева, и переменные, определенные справа, или они могут использовать переменной и константы напрямую, как показано на рисунке 10.5.

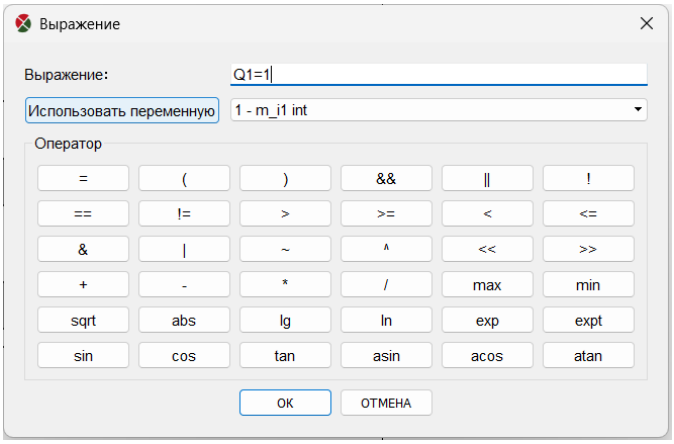


Рисунок 10.5 Выражение

Наиболее часто используемые операции выражений:

- [**Назначение**]: присвоить значение переменной и назначенной точке. Например, %Q0001=1, %MW0001=100.
- [**Прочитайте значение балла**]: прочитать значение точки и сохранить в переменной. Например, v_1=%IW0001.
- [**Операция**]: оперировать данными регистров. Например, v_1=sin%MW0001.

Задержка операции

После некоторых операций управления (например, действия цифрового выхода) необходимо выждать некоторое время для выполнения других операций, или после обратной связи по сигналу необходимо выждать некоторое время для выполнения определенной операции, что может быть достигнуто с помощью операции задержки. Введите количество секунд задержки (положительное целое число) и нажмите кнопку «ОК», как показано на рисунке 10.6.

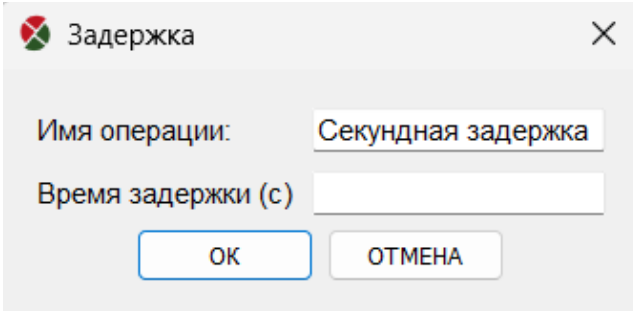


Рисунок 10.6 Секундная задержка

Задержка в миллисекундах

Задержка в миллисекундах похожа на операцию задержки, за исключением того, что ее единицей является миллисекунда. Минимальное время задержки в миллисекундах — это период сканирования системы. Если установленное время меньше периода сканирования, система автоматически задерживает период сканирования.

Установить таймер

Управление последовательностью может иметь процесс или операцию, которые необходимо завершить за определенное время, для чего необходимо использовать таймер для ограничения времени. После того, как лимит времени истек, перейти к метке OUT для выполнения. В это время нет необходимости очищать таймер, и система автоматически очистит его. Если время ограничения не достигнуто и операция завершена, то

следует добавить операцию очистки таймера. Когда выбрано это действие, оно выглядит так, как показано на рисунке 10.7:

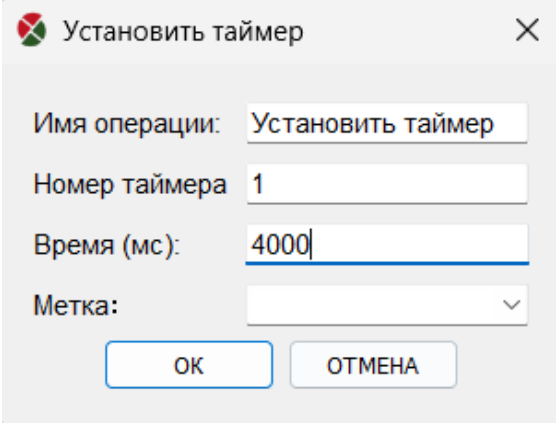


Рисунок 10.7 Установить таймер

Введите номер таймера (от 1 до 9), количество миллисекунд и выходную метку после срабатывания таймера. Рядом с входной меткой находится список, в котором перечислены все метки, используемые программой. Если выходная метка, которую вы хотите ввести, уже существует, вы можете выбрать ее.

Сброс таймера

Когда операция завершена, а время таймера не достигнуто, таймер необходимо очистить. Когда заданное время достигнуто, таймер будет очищен автоматически. Введите номер таймера, который необходимо сбросить, и нажмите «ОК», как показано на рисунке 10.8.

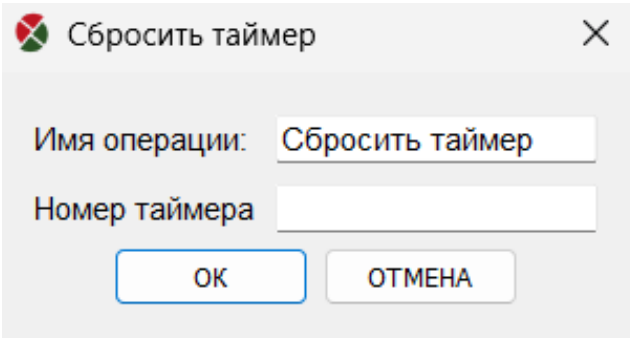


Рисунок 10.8 Сбросить таймер

Авария

Когда SCC выполняется до определенной стадии, состояние SCC должно быть сообщено операторам. Система устанавливает функцию тревоги, чтобы напомнить

операторам о некоторых неудачных операциях. Когда SCC выполняется до функционального поля тревоги, содержимое строки тревожных символов будет существовать в окне тревоги. Пользователи вводят строки тревожных символов и нажимают «ОК», как показано на рисунке 10.9.

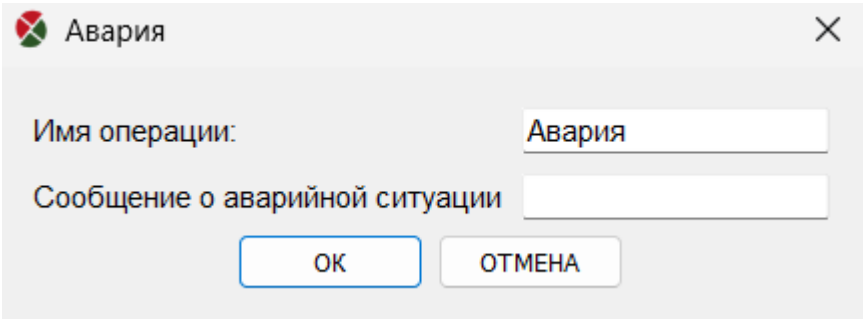


Рисунок 10.9 Авария

Импульсный цифровой выход

При последовательном управлении значения должны быть установлены и выведены на цифровые выходные точки. Если время вывода долгое или неточное, выражения могут использоваться для установки точки в 1 и очистки в 0 позже. Но более распространено сделать так, чтобы цифровая выходная точка выводила фиксированный импульс, в это время импульс может быть использован для открытия, как показано на рисунке 10.10.

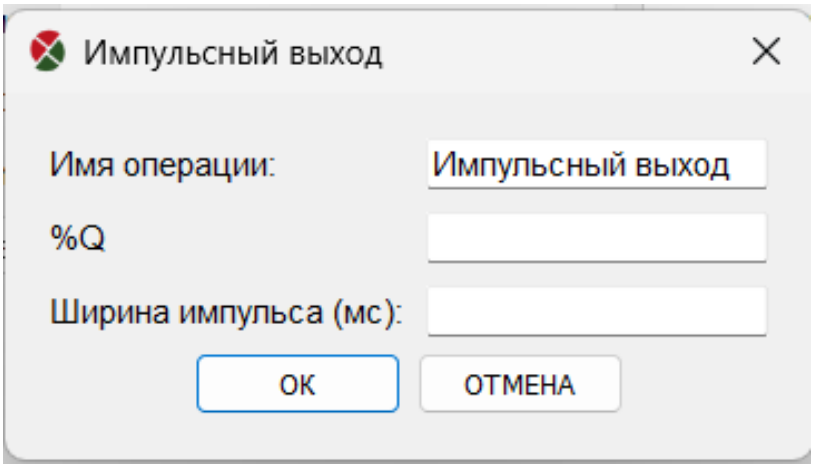


Рисунок 10.10 Импульсный цифровой выход

После выполнения функционального блока он выводит 1 на %Q0001 и удерживает его в течение трех секунд.

Вызов программы

SCC может вызывать программы LD, FBD, IL и ST напрямую. Когда SCC выполняет «вызов программы», он сканирует программу вызова один раз и возвращается в SCC, продолжая выполнение. В диалоговом окне вызова программы все программы в текущем файле проекта перечислены в списке, выберите программу для вызова и подтвердите ее, как показано на рисунке 10.11.

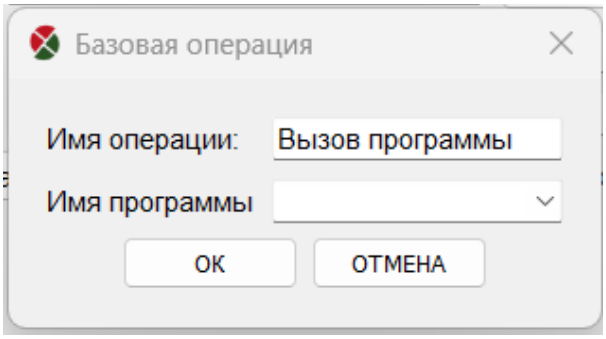


Рисунок 10.11 Вызов программы

Вызов SCC

Вызывающая диаграмма управления последовательностью передается на выполнение других программ диаграмм управления последовательностью во время выполнения программы диаграммы управления последовательностью. После завершения выполнения вызванной программы диаграммы управления последовательностью исходная программа управления последовательностью возвращается для продолжения нисходящего выполнения, как показано на рисунке 10.12.

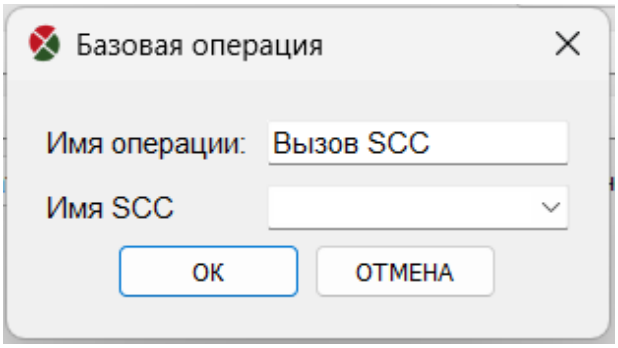


Рисунок 10.12 Вызов SCC

Выполнить SCC

Иногда необходимо выполнить программу диаграммы управления последовательностью, когда другая программа диаграммы управления последовательностью может быть выполнена в то же время, что требует выполнения этой команды операции. После

выполнения новой программы исходная программа диаграммы управления последовательностью продолжает выполняться, не будучи затронутой новой программой. В диалоговом окне действия Execute все программы диаграммы управления последовательностью в текущем файле проекта перечислены в списке программ, выберите программу для выполнения и подтвердите ее, как показано на рисунке 10.13.

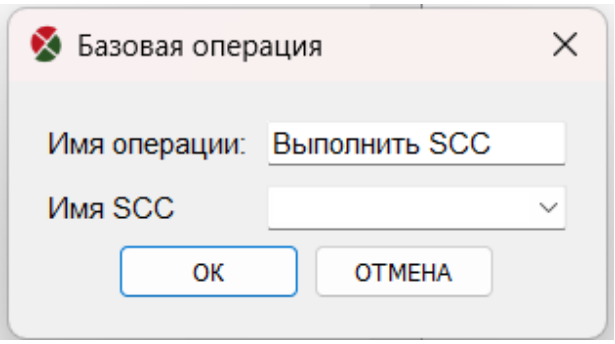


Рисунок 10.13 Выполнить SCC

Остановить SCC

При выполнении программы SCC иногда необходимо завершить другую программу SCC. Остановка программы SCC принудительно прерывает выполнение другой программы SCC, но если некоторые регистры были принудительно установлены в этой программе SCC, система не очистит их. Обратите внимание на очистку регистров при использовании операции остановки. Выберите имя SCC и нажмите «OK», как показано на рисунке 10.14.

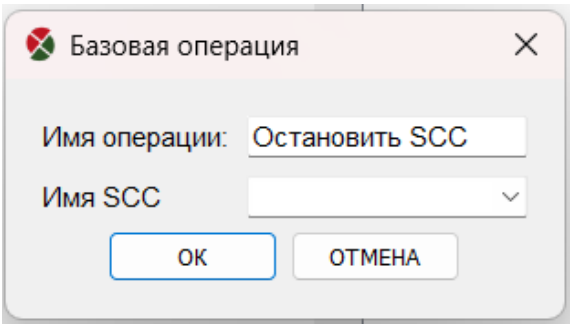


Рисунок 10.14 Остановить SCC

Блокировка и разблокировка SCC

При выполнении программы контрольной карты последовательности необходимо запретить (т. е. заблокировать) выполнение другой программы контрольной карты последовательности. При выполнении этой программы контрольной карты

последовательности она открывает (т. е. разблокирует) программу контрольной карты последовательности, которая только что была заблокирована.

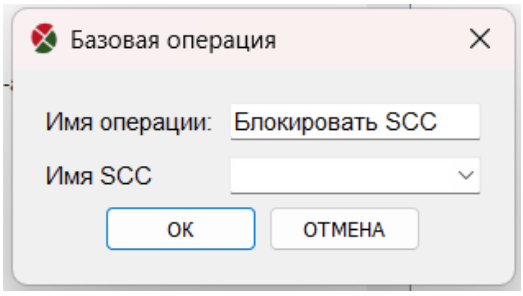


Рисунок 10.15 Блокировка SCC

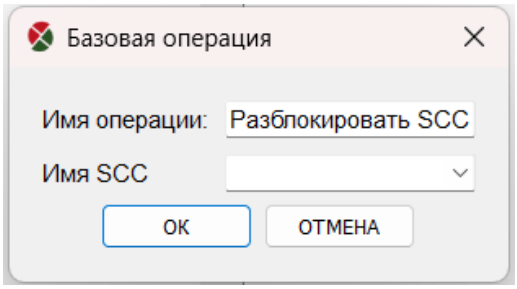


Рисунок 10.16 Разблокировать SCC

Блок «Условие»

Используется для определения истинности условия. Если условие истинно, перейдите в ветку «да», если нет, перейдите в ветку «нет». Дважды щелкните поле условия, чтобы изменить условие. Условия поддерживают все указанные выше операторы и переменные, как показано на рисунке 10.17.

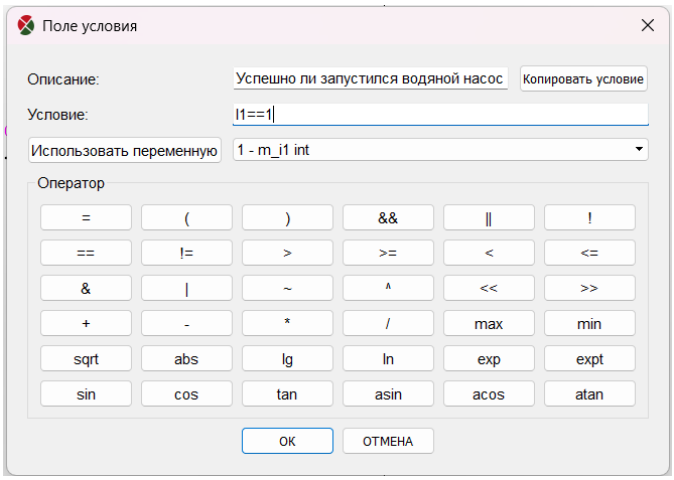


Рисунок 10.17 Поле условия

Блок «Условие с ограничением по времени»

Некоторые условия должны быть оценены в течение определенного периода времени. Если условие выполняется в течение ограниченного времени, перейдите в ветку «да»; Если условие не выполняется в течение ограниченного времени, оценка продолжается. Если предельное время истекло, а условие не выполняется, перейдите в ветку «тайм-аут». Для ограничения времени необходим таймер, а установка и очистка таймера автоматически выполняются системой без явного определения пользователем, как показано на рисунке 10.18.

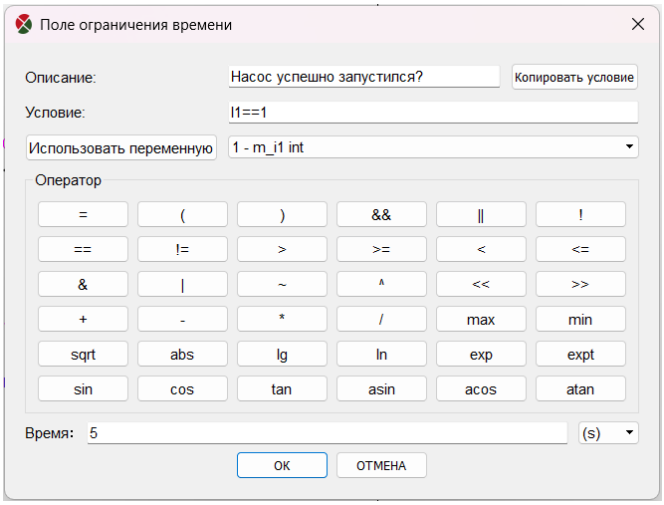


Рисунок 10.18 Условие с ограничением по времени

Соединитель

Когда схема управления последовательностью более сложная, программа может быть разделена на несколько частей, тогда вы можете использовать соединитель для соединения нескольких частей программы или использовать его для достижения перехода между различными частями программы. Введите имя соединителя и нажмите кнопку «ОК», как показано на рисунке 10.19.

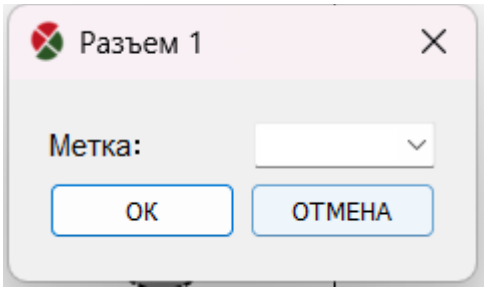


Рисунок 10.19 Соединитель

Комментарий

Добавьте комментарий, чтобы сделать программу более понятной. Выберите комментарий и щелкните место для добавления. Введите комментарий и щелкните «ОК». Перетащите комментарий, чтобы изменить его положение, или дважды щелкните комментарий, если вы хотите изменить его содержимое, как показано на рисунке 10.20.

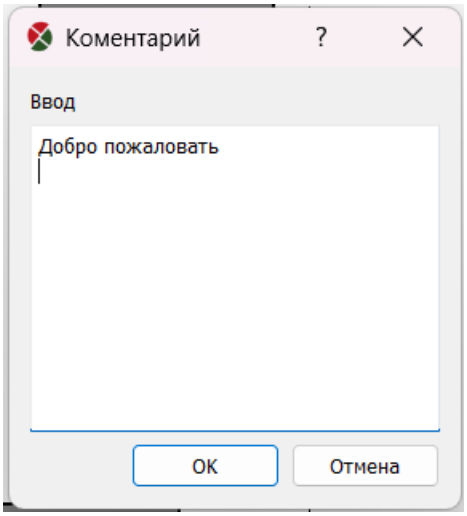


Рисунок 10.20 Комментарий

Соединение

Соединение между каждым функциональным блоком в контрольной схеме последовательности представляет собой порядок выполнения каждого шага в программе контрольной схемы последовательности. Вход и выход каждого шага должны быть соединены, в противном случае возникнет ошибка. Метод соединения следующий:

Функция «Магнит»

Когда два функциональных блока приближаются друг к другу на определенное расстояние, соединение будет построено автоматически. Когда любой из функциональных блоков перемещается, направление соединения автоматически вычисляется.

Ручное подключение

Используйте  для соединения. Соединение имеет принцип, что вход функционального блока может быть соединен только с выходом функционального блока, в противном случае возникнет ошибка. Выход одного функционального блока не может быть соединен с входом двух функциональных блоков одновременно. Однако вход одного функционального блока может быть соединен с выходом нескольких функциональных блоков. Направление линии автоматически рассчитывается на основе относительного положения двух функциональных блоков.

Удаление соединения

Нажмите левой кнопкой мыши на линии, которую вы хотите удалить, и линия станет синей. Нажмите правую кнопку мыши еще раз, появится плавающее меню, выберите пункт «Удалить», чтобы удалить соединение. Или выберите линию и нажмите клавишу «Удалить» на клавиатуре, чтобы удалить.

Перемещение соединения

Вы можете настроить направление соединения, когда соединение проходит через другие функциональные блоки и влияет на внешний вид диаграммы управления последовательностью.

Короче говоря, есть принцип: в линии сегмент линии, напрямую соединенный с блоком функции, не может быть перемещен. Другие сегменты могут быть перемещены. Метод перемещения следующий: панель инструментов диаграммы управления последовательностью должна находиться в состоянии «переместить», указывая на сегмент линии, который нужно переместить, нажмите левую кнопку, затем вся линия станет синей, в соответствии с направлением линии, около сегмента линии, который нужно переместить, удерживайте левую кнопку, а затем перемещайте мышь вверх и вниз или влево и вправо, сегмент линии будет перемещен соответствующим образом.

Отладка программы

Программное обеспечение INDAS PRO поддерживает различные языки программирования – LD, FBD, IL, ST, SCC. В соответствии с характеристиками каждого языка программирования INDAS PRO разработала множество методов отладки программ, что удобно для технических специалистов для отладки и изменения программы в ходе реализации проекта и настройки двух сред отладки: онлайн-отладка ПЛК и отладка симуляции. Онлайн-отладка ПЛК заключается в том, что INDAS PRO напрямую подключает ПЛК для отладки программы и загрузки модификации в ПЛК. Отладка симуляции заключается в моделировании среды, в которой соединены INDAS PRO и ПЛК, что является виртуальным соединением. Корректность программы может быть предварительно проверена с помощью настройки переменных, принудительного задания регистров измерения ввода-вывода и т. д., чтобы гарантировать успешность онлайн-отладки и безопасность отладки работы оборудования. Ниже приводится подробное описание шагов процедуры отладки для различных языков программирования.

Онлайн-модификация

Отладка лестничной диаграммы/функциональной блок-схемы интуитивно понятна. После подключения к сети все текущие токи красные (%M0008), а не текущие токи зеленые (%M0007), как показано на рисунке 11.1. LD обычно используется для реализации оценки состояния, поэтому состояние переменной можно просматривать напрямую.

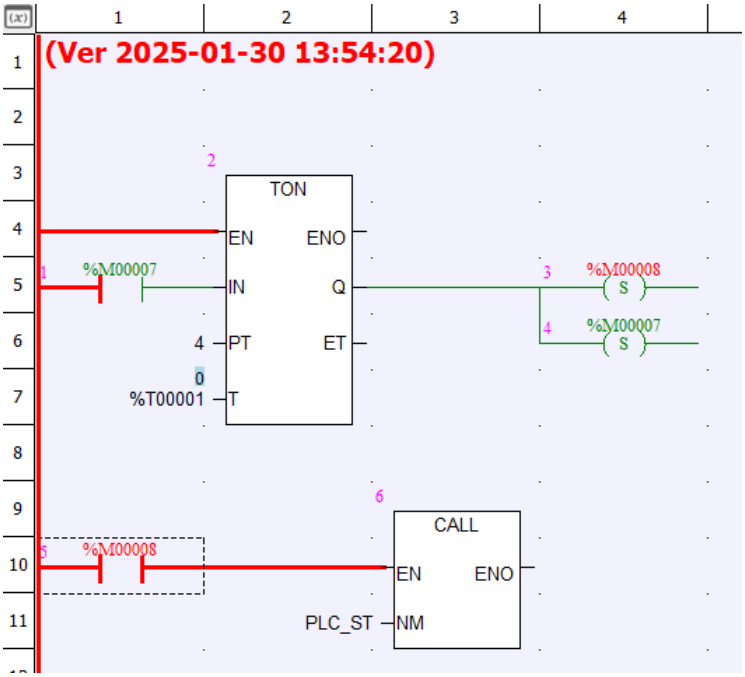


Рисунок 11.1 Протекание тока в LD

LD/FBD можно изменять в режиме онлайн, как показано на рисунке 11.2.

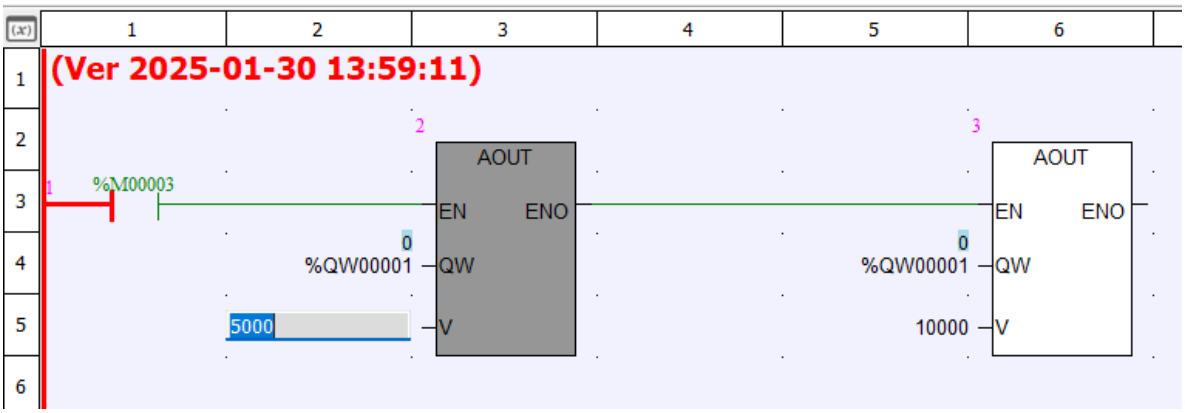


Рисунок 11.2 Изменение в режиме «Онлайн» в LD

После подключения к сети, для операции модификации программы, имя программы будет отмечено знаком *, как показано на рисунке 11.3 MAIN*, указывая на то, что текущая программа была изменена, но не загружена.

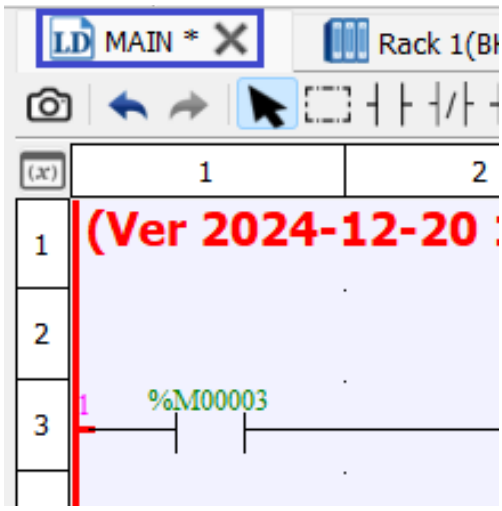


Рисунок 11.3 Измененная программа

После внесения изменений выберите чтобы сохранить файл и [online]/[Refresh Program] для загрузки программы в ПЛК, как показано на рисунке 11.4.

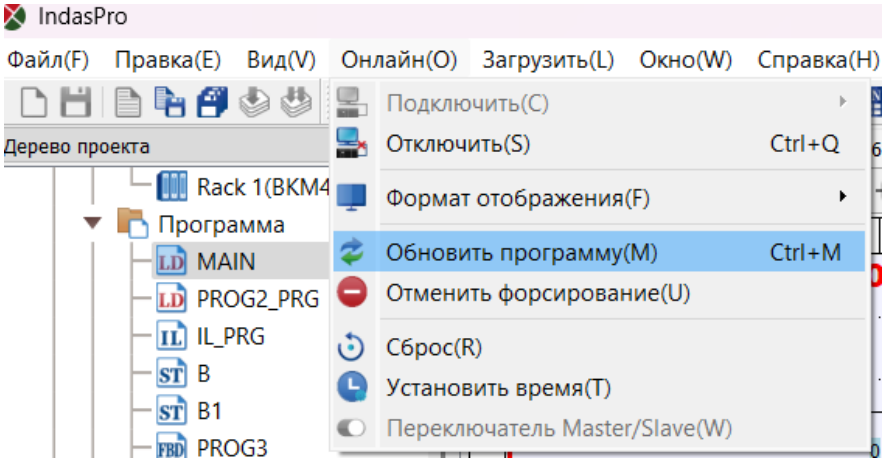


Рисунок 11.4 Обновление программы

Отладка в режиме онлайн

Онлайн-отладка в основном используется для пошаговой отладки программ. Пользователи могут устанавливать точки останова в LD/FBD. Когда программа достигает точки останова, она останавливает выполнение и ждет команду «Step». Если отладка программы завершена, очистите все точки останова и нажмите «Продолжить» для сканирования программы.

Шаг

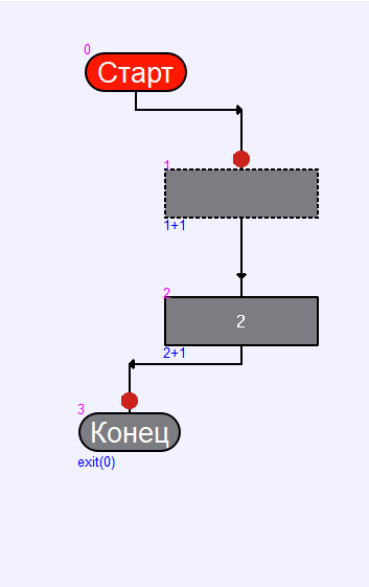
При отладке программы, после одного шага, программа останавливается на месте и ждет команду «Шаг». Нажмите на иконку  для выполнения следующего функционального блока.

Продолжать 

Когда программа остановится, нажмите значок «продолжить», чтобы программа выполнялась непрерывно. Если в программе есть точка останова, она остановится при достижении первой точки останова и будет ждать команды «Шаг».

Вставить/удалить точку останова 

Точки останова можно устанавливать на любом этапе программы. Способ установки точки останова следующий: выберите функциональный блок, на котором необходимо установить точку останова, и щелкните значок  . Щелкните еще раз, чтобы удалить точку останова. Программа может устанавливать точки останова в 10 разных местах одновременно, как показано на рисунке 11.5.



Удалить все точки останова 

Нажав на иконку  можно удалить все точки останова одновременно.

- Все установленные точки останова будут автоматически очищены после выхода из сети.
- Чтобы включить функцию установки и очистки точек останова, необходимо выбрать «Создать отладочную информацию».

Отладка SCC

Программа SCC должна быть отлажена после компиляции и загрузки. Перед отладкой необходимо сначала подключиться к ПЛК или симуляции онлайн. В это время панель инструментов SCC действительна, и вы можете отправлять команды отладки.

Автоматическое выполнение

Автоматическое выполнение заключается в том, что во время выполнения программы информация не будет передаваться на отладочный компьютер, т.е. отладочный компьютер не может отражать процесс выполнения программы в реальном времени. Способ запуска автоматического выполнения: нажмите на иконку  .

Наблюдение за выполнением

Наблюдая за выполнением  заключается в том, что ПЛК отправляет результат выполнения программы на отладочный компьютер в реальном времени во время выполнения программы. В это время отладочный компьютер может отражать процесс выполнения программы в реальном времени (только компьютер, который выдал команду отладки). При таком выполнении отладочный персонал может контролировать процесс выполнения управляющей программы в реальном времени и может вовремя улавливать работу различного оборудования (что может быть отражено в программе). Во время процесса выполнения красный цвет указывает на то, что программа выполняется, синий цвет указывает на то, что программа завершила выполнение, а серый цвет указывает на то, что программа еще не была выполнена. Щелкните значок  для запуска наблюдения за выполнением. Если необходимо вызвать подпрограмму, она автоматически переключается в окно подпрограммы и контролирует выполнение подпрограммы. После выполнения подпрограммы она автоматически возвращается в текущее окно программы, как показано на рисунке 11.6.

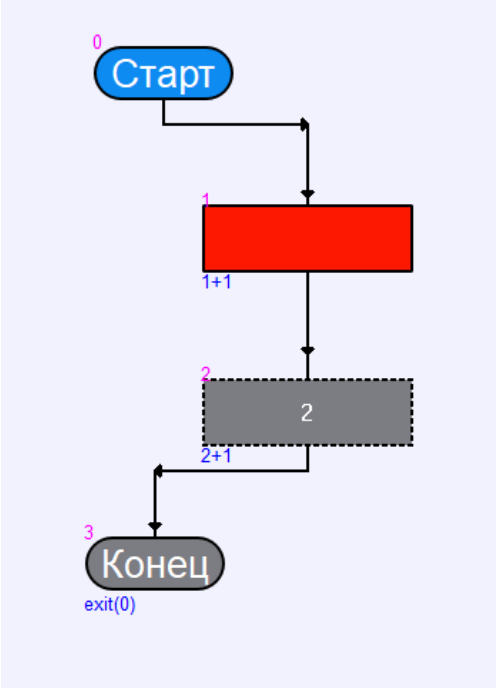


Рисунок 11.6 Наблюдение за выполнением SCC

Остановка выполнения

Если пользователь хочет прекратить выполнение программы в каком-либо месте, щелкните значок .

Выполнение в отладке

В отличие от мониторинга выполнения, отладочное выполнение больше подходит для отладки программ и поддерживает различные средства отладки программ. Метод запуска следующий: выберите кнопку «Отладка выполнения» на панели инструментов SCC, нажмите левую кнопку мыши, будет выдана команда отладки выполнения, и программа начнет выполняться. Однако программа находится в режиме пошагового выполнения. После выполнения первого шага программа останавливается и ждет команды «Выполнить один шаг».

Шаг

При отладке программы, после одного шага, программа останавливается на месте и ждет команду «Шаг». Нажмите на иконку для выполнения следующего функционального блока.

Продолжать

Когда программа остановится, нажмите значок «продолжить», чтобы программа выполнялась непрерывно. Если в программе есть точка останова, она остановится при достижении первой точки останова и будет ждать команды «Шаг».

Вставить/удалить точку останова

Точки останова можно устанавливать на любом этапе программы. Способ установки точки останова следующий: выберите функциональный блок, на котором необходимо установить точку останова, и щелкните значок . Щелкните еще раз, чтобы удалить точку останова. Программа может устанавливать точки останова в 10 разных местах одновременно, как показано на рисунке 11.7.

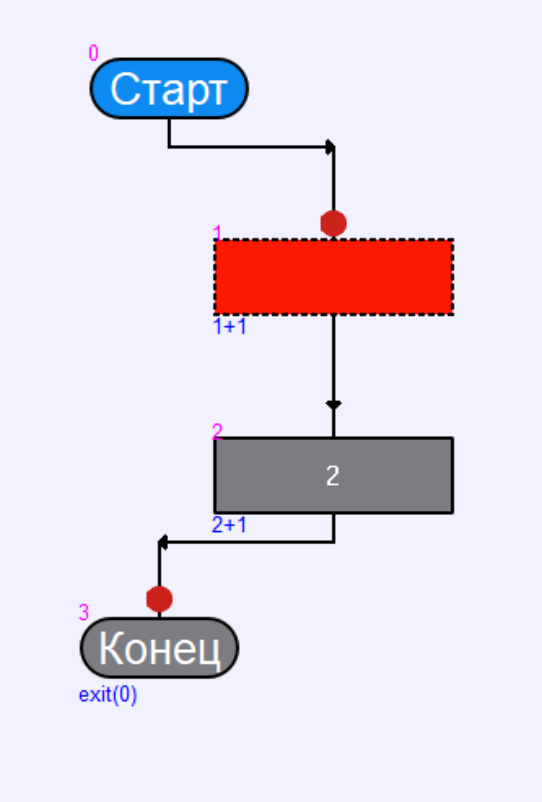


Рисунок 11.7 Вставить точку останова SCC

Удалить все точки останова

Нажав на иконку можно удалить все точки останова одновременно.

Примечание
Все установленные точки останова будут автоматически очищены после выхода из сети.

Перезапуск

В режиме пошагового выполнения, когда программа выполняется до середины, есть два способа перезапустить выполнение программы.

1. Нажмите на значок .
2. Нажмите на и затем нажмите кнопку для начала выполнения программы.

Остановить отладку

В режиме отладки выполнения нажмите кнопку «Остановить отладку», выполнение отладки программы будет остановлено.

Блокировка и разблокировка

Программу можно разблокировать и заблокировать вручную, заблокировав кнопку разблокировки. .

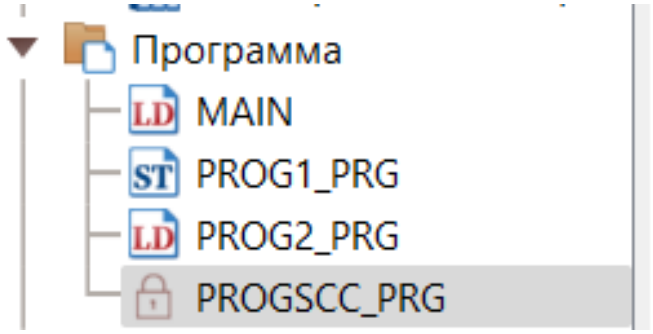


Рисунок 11.8 Блокировка программы SCC

Шаг

При отладке программы, после одного шага, программа останавливается на месте и ждет команду «Шаг». Нажмите на иконку для выполнения следующего функционального блока.

Продолжать

Когда программа остановится, нажмите значок «продолжить», чтобы программа выполнялась непрерывно. Если в программе есть точка останова, она остановится при достижении первой точки останова и будет ждать команды «Шаг».

Вставить/удалить точку останова

Точки останова можно устанавливать на любом этапе программы. Способ установки точки останова следующий: выберите функциональный блок, на котором необходимо установить точку останова, и щелкните значок . Щелкните еще раз, чтобы удалить точку останова. Программа может устанавливать точки останова в 10 разных местах одновременно, как показано на рисунке 11.7.

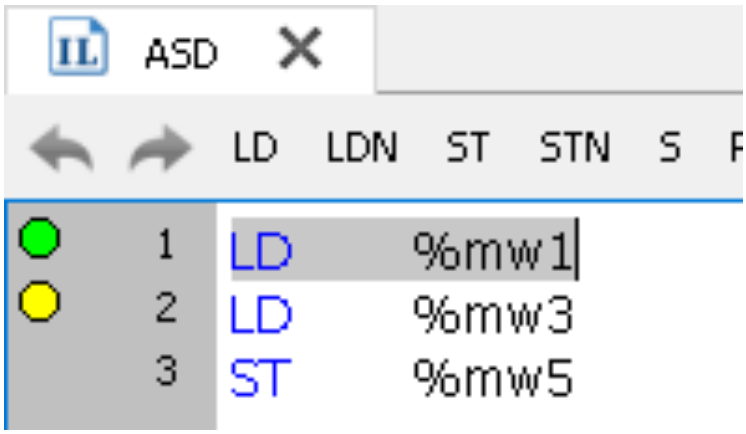


Рисунок 11.9 Вставить точку останова IL

Удалить все точки останова

Нажав на иконку можно удалить все точки останова одновременно.

Примечание
Все установленные точки останова будут автоматически очищены после выхода из сети.

Отладка ST

Шаг

При отладке программы, после одного шага, программа останавливается на месте и ждет команду «Шаг». Нажмите на иконку для выполнения следующего функционального блока.

Продолжать

Когда программа остановится, нажмите значок «продолжить», чтобы программа выполнялась непрерывно. Если в программе есть точка останова, она остановится при достижении первой точки останова и будет ждать команды «Шаг».

Вставить/удалить точку останова

Точки останова можно устанавливать на любом этапе программы. Способ установки точки останова следующий: выберите функциональный блок, на котором необходимо установить точку останова, и щелкните значок . Щелкните еще раз, чтобы удалить точку останова. Программа может устанавливать точки останова в 10 разных местах одновременно, как показано на рисунке 11.7.

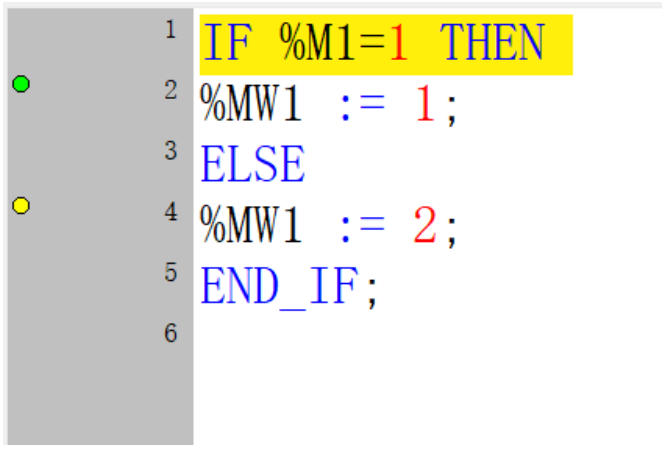


Рисунок 11.10 Вставить точку останова ST

Удалить все точки останова

Нажав на иконку можно удалить все точки останова одновременно.

> **Примечание**
> Все установленные точки останова будут автоматически очищены после выхода из сети.

Отладка в режиме «Симулятор»

Обзор

Программное обеспечение INDAS PRO поддерживает два режима отладки: онлайн-симуляция и онлайн-ПЛК для тестирования программ.

Функция имитации соединения, которая является виртуальным соединением без ПЛК, или виртуальным испытанием процесса производства на месте, для проверки написанной программы. Чтобы увидеть эфект программы или является ли существующая программа правильной.

Онлайн-функция ПЛК с использованием режима связи и мощной функции онлайн-мониторинга INDAS PRO, в соответствии с фактическим сигналом или аналоговым сигналом или процессом управления на месте, онлайн-мониторинг и отладка программы, чтобы увидеть эфект программы или правильность существующей программы.

Различия между подключением симулятора и подключением ПЛК заключаются в следующем:

В режиме «подключение симулятора» INDAS PRO фактически не подключен к ПЛК. В процессе симуляции все сигналы принудительно устанавливаются или изменяются только в компьютере, где находится INDAS PRO. Результаты или данные операции генерируются компьютером, и реальное управление выходом не может быть сгенерировано.

В режиме «Онлайн» INDAS PRO фактически подключен к ПЛК. Все сигналы принудительно устанавливаются или изменяются в ПЛК, а результаты или данные операции генерируются ПЛК, генерируя реальное управление выходом.

Оба режима отладки могут достигать цели проверки производительности программы или корректности существующей программы. При нормальных обстоятельствах онлайн-симуляция подходит для предпроектной стадии проекта, а ПЛК онлайн подходит для стадии заводской отладки и стадии отладки на месте после завода.

Основные функции режима онлайн-симуляции следующие:

- Все входные и выходные сигналы могут быть принудительно изменены для изменения значения.
- Он может полностью имитировать действие логических связей, инструкций приложений и программ в реальной работе.

Процесс отладки подключения симулятора

Блок-схема процесса отладки подключения симулятора представлена на рисунке 11.11.

Рисунок 11.11 Блок-схема процесса отладки подключения симулятора

Протоколы связи

Контроллеры NAPLC обычно конфигурируются с последовательным портом или портом Ethernet, при этом последовательный порт поддерживает протокол MODBUS, а порт Ethernet поддерживает протокол MODBUS TCP/IP.

Функции обмена

MODBUS — это Master/Slave протокол, который позволяет выполнять операции чтения/записи одного или нескольких слов (16 бит), но ни в коем случае не поддерживает операции чтения/записи байтов.

Обмен сообщениями осуществляется ведущим устройством, берущим на себя инициативу, т. е. ведущий инициирует обмен. Помимо широковещательных команд, каждый полный обмен состоит из двух сообщений, нисходящего и восходящего.

- Сообщение по нисходящей линии связи: запрос от мастера
- Сообщение по восходящей линии связи: ответ, отправленный обратно от ведомого устройства

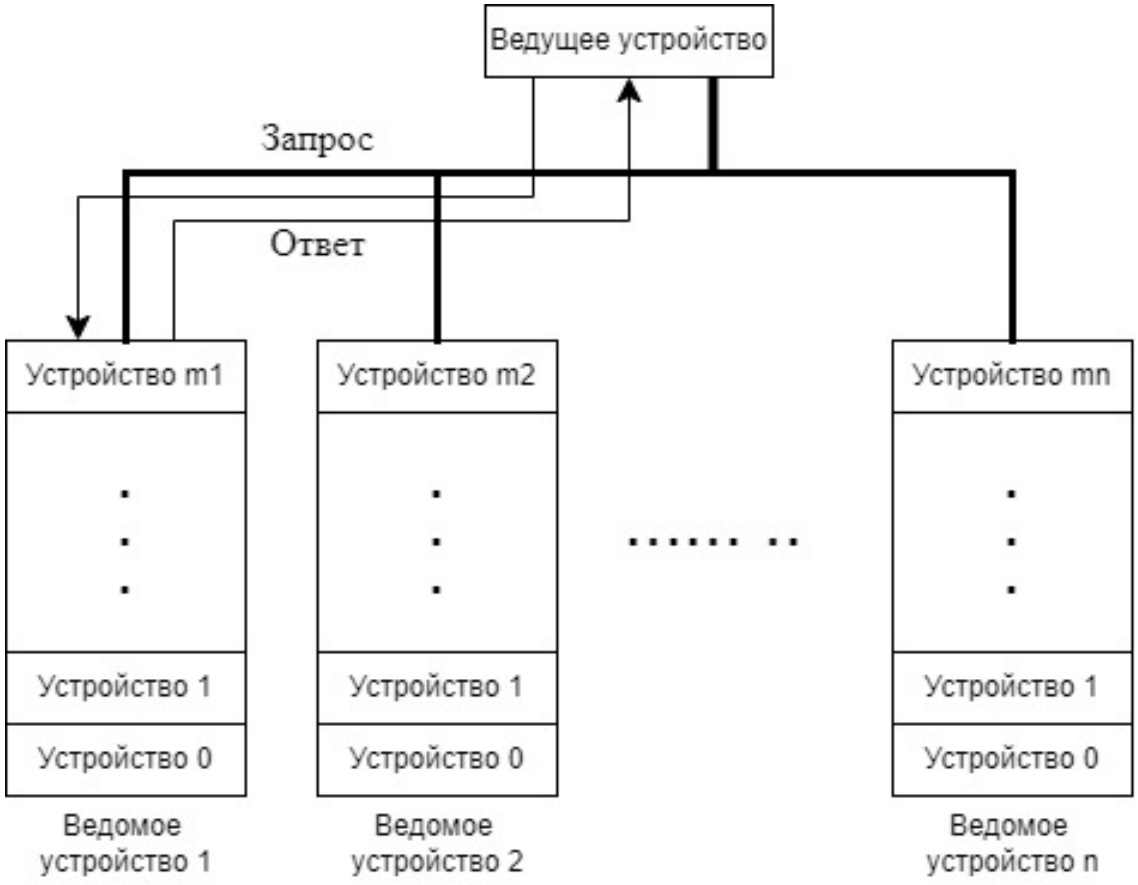


Рисунок 12.1 Обмен обычной информацией

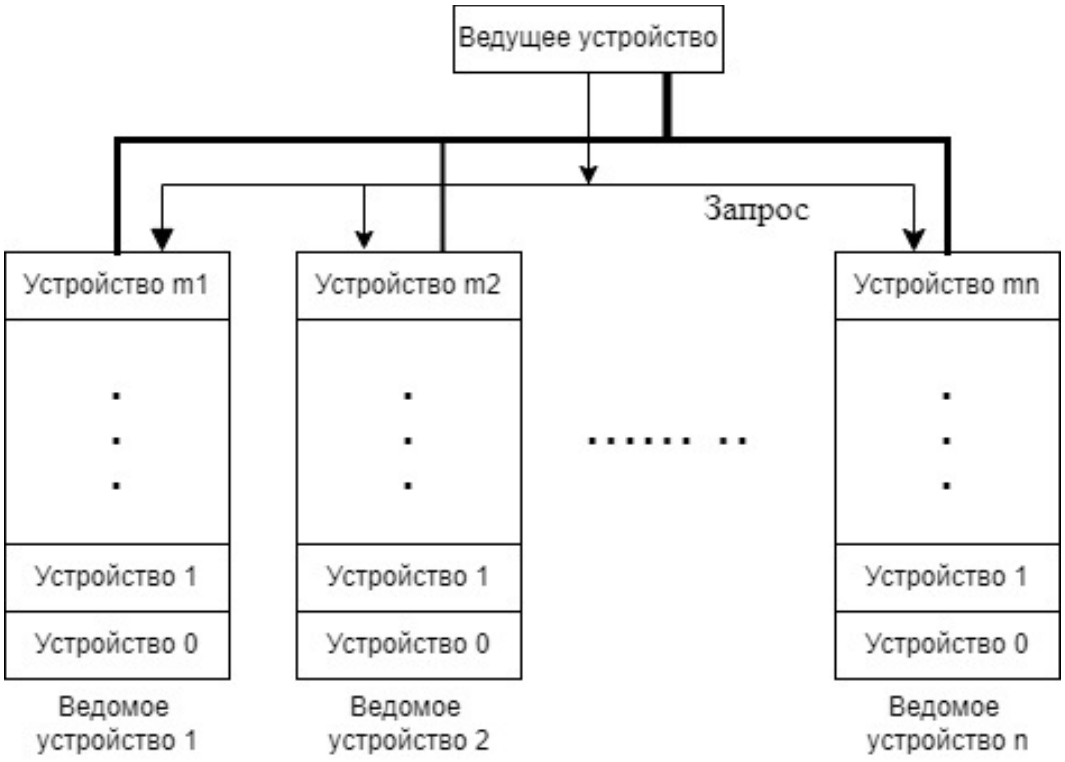


Рисунок 12.2 Обмен информацией в широковещательном режиме

Запрос от ведущего устройства обычно может быть отправлен только указанному ведомому устройству (идентифицируемому по номеру ведомого устройства, указанному в первом байте кадра запроса), как показано на рисунке 12.1. В широковещательном режиме (номер ведомого устройства 0) запрос отправляется всем ведомым устройствам. Широковещательная команда должна быть командой записи, и ведомые устройства не отправляют ответ, как показано на рисунке 12.2.

Формат сообщения

Все обмениваемые сообщения (кадры) типа RTU, независимо от восходящего/нисходящего потока, имеют одинаковую структуру.

От станции номер	Код функции	Область данных	Контрольная сумма CRC16
1 байт	1 байт	n байт	2 байта

Каждый кадр содержит 4 типа информации:

Номер Slave:

Номер ведомого устройства — 1 байт, диапазон от 0 до FFH. Исключением является то, что если это значение равно 0, оно используется как Id широковещательного

сообщения для ведущего устройства. Таким образом, номер ведомого устройства может быть использован физически только между 01H и FFH (т.е. между 1 и 255).

Код функции:

Код функции составляет 1 байт и используется для выбора команды (чтение, запись или ответ, если контрольная сумма верна и т. д.) Допустимые коды функций находятся в диапазоне от 1 до 255, а поддерживаемые в данном руководстве коды функций подробно описаны ниже.

Область данных:

Область данных представляет собой n байт, содержащих строку шестнадцатеричных данных, связанных с кодом функции.

Проверка CRC16:

Контрольная сумма CRC16 представляет собой 2-байтовую контрольную сумму, вычисляемую методом CRC16 от первого байта сообщения (ведомого номера) до последнего байта области данных.

Пример

Алгоритм контрольной суммы CRC16 реализован на языке C следующим образом.

unsigned short CRC16(unsigned char* buf, unsigned short len)

```
{
        unsigned short crc=0xffff;
        unsigned short i,j,k;
        for(i=0;i<len;i++)
        {
        crc =crc ^ buf[i];
        for(j=0;j<8;j++)
                {
                k=crc & 01;
                crc=crc >> 1;
                if (k==0) continue;
                crc =crc ^ 0xA001;
                }
        }
        return crc;
}
```

Метод адресации

Ведущее устройство может связаться с n ведомыми устройствами, 1 ≤ n ≤ 255, за исключением того, что n = 0 (т.е. номер ведомого устройства = 0) используется в качестве широковещательного сообщения мастера.

Доступ ведущего устройства к ведомому устройству фактически сводится к доступу к ряду адресов устройства в ведомом устройстве, поэтому адрес, к которому должен получить доступ ведущий, определяется тремя элементами: номером ведомого устройства, номером устройства и адресом в устройстве.

Номер ведомого устройства указан в разделе «Формат сообщения».

Номер устройства и адрес в устройстве описываются в сообщении вместе как «словесный адрес».

Метод адресации 1

Номер устройства используется как старший байт «адреса слова» (часто называемого смещением номера устройства), а адрес в устройстве используется как младший байт «адреса слова». Очевидно, что ведущее устройство может получить доступ только к максимальному количеству адресов слов от 0 до 255 для конкретного устройства, и ведущее устройство может получить доступ только к максимальному количеству адресов слов от 0 до 255 для конкретного устройства. Длина непрерывно доступных адресов слов намного меньше 255, как описано в 5.2.3 этой главы.

При этом методе адресации, поскольку количество ведомых устройств может достигать 255, а каждое ведомое устройство может состоять из 256 устройств, к сети MODBUS можно подключить максимум 255 x 256 = 65280 устройств.

Метод адресации 2

В режиме 1 выше указано, что ведущее устройство имеет доступ к максимуму адресов слов от 0 до 255 для конкретного устройства. Во многих приложениях адрес устройства может быть больше этого, и в этом случае адрес устройства делится на страницы (часто эти страницы также называются смещениями), каждая из которых имеет диапазон адресов слов от 0 до 255. Таким образом, режим 2 имеет тот же формат, что и режим 1 с точки зрения доступа.

Разница между двумя методами адресации

Режим 1: на каждом ведомом устройстве можно установить до 255 устройств, но диапазон адресов каждого устройства составляет от 0 до 255.

Режим 2: к каждому ведомому устройству можно подключить меньше устройств, чем в режиме 1, но адрес устройства может быть больше диапазона 255. Если количество устройств на ведомой станции = 1, то теоретическое максимальное количество адресов слов, к которым можно получить доступ, = 256 x 256 = 65536, т. е. диапазон составляет от 0 до 65535, хотя ограничение длины количества последовательных адресов слов, к которым может получить доступ ведущий узел одновременно, по-прежнему такое же, как и для режима 1.

Специальное примечание по режиму адресации 2

Во многих приложениях, где к подчиненному устройству подключено только одно устройство, понятие «подчиненное устройство» больше не отличается от «устройства» на подчиненном устройстве, а «страница» и «адрес слова» страницы объединяются в широкий адрес слова. «Адрес слова» страницы объединяется в широкий адрес слова.

Ошибка ответа

Помимо широковещательного сообщения, ведомое устройство выполняет следующие нормальные или ненормальные ответы на запросы ведущего устройства.

- Подчиненное устройство взаимодействует нормально, и содержание сообщения верно, подчиненное устройство отправляет обратно соответствующее ответное сообщение.
- Подчиненное устройство не может получить сообщение-запрос от ведущего устройства обычным образом из-за сбоя связи, в этом случае ведущее устройство должно выполнить соответствующую ошибку тайм-аута.
- Подчиненное устройство получает сообщение-запрос от ведущего устройства в обычном режиме, но с ошибкой CRC16. В этом случае ведомое устройство не отправляет ответное сообщение, а ведущее устройство выполняет соответствующую ошибку тайм-аута.
- Подчиненное устройство получает сообщение-запрос от ведущего устройства в обычном режиме, но не может обработать сообщение-запрос в обычном режиме (например, ведущее устройство запрашивает чтение несуществующего диска или переменной), в этом случае ведомое устройство отправляет обратно сообщение-ответ об ошибке, чтобы предупредить ведущее устройство. В сообщении-ответе об ошибке.

От станции номер	Код функции | 80H	Код типа ошибки	Контрольная сумма CRC16
1 байт	1 байт	1 байт	2 байта

Код функции: код функции главного запроса | 80H.

Область данных: 1 байт, код типа ошибки, как указано ниже.

Код типа ошибки	Определение	Значение
01	ILLEGAL_FUNCTION	Недопустимые коды функций
02	ILLEGAL_DATA_ADDRESS	Запрошенный адрес данных является недопустимым
03	ILLEGAL_DATA_VALUE	Недопустимые данные в сообщении-запросе
04	SLAVE_DEVICE_FAILURE	Неустранимый сбой происходит, когда ведомое устройство отвечает на запрос.
05	ACKNOWLWDGE	Подчиненной станции требуется больше времени для обработки запроса от главной станции.
06	ILLEGAL_FUNCTION	ведомое устройство занято и не может ответить на запрос в данный момент. Пожалуйста, отправьте повторно позже

Пример

Письмо-ответ об ошибке ведомого устройства

Ведущее устройство запрашивает чтение состояния цифрового выхода подчиненной станции (адрес 10) по адресу 1234 (04A1H), но ведомое устройство возвращает сообщение об ошибке, если точка цифрового выхода не существует. Это показано ниже.

Сообщение от ведущего устройства.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	0A
2	Код функции	01
3	Стартовый адрес старшие 8 бит	04
4	Начальный адрес младшие 8 бит	A1
5	Количество старших 8 бит данных	00

6	Количество младших 8 бит данных	01
CRC16	-	-

Сообщение об ошибке ответа ведомого устройства.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	0A
2	Код функции	81
3	Код типа ошибки	02
CRC16	-	-

Протокол MODBUS

Обзор функционального кода

В данном руководстве MODBUS использует следующие коды функций.

Код функции (десятичный)	Значение
01	Чтение состояния катушки
02	Чтение состояния входа
03	Чтение регистра хранения
04	Чтение регистра типа ввода
05	Принудительное управление отдельными катушками
06	Запись одиночного регистра
15	Принудительное управление несколькими катушками
16	Запись нескольких регистров

Коды функций и классификация данных

Соответствие между кодами функций и соответствующими данными в данном руководстве показано ниже (в качестве примера приведен CPU401-0501, другие типы).

(Верхний предел адреса переменной ЦП отличается от уставного адреса).

Префикс	Тип регистра	Код функции операции чтения	Код функции операции записи	Адрес протокола	Примечание
0x	Q	01	05 / 15	00000～02047	Подробную информацию о диапазоне адресов см. в следующей таблице.
	M	01	05 / 15	10000～26383	
	N	01	05 / 15	30000～34095	
1x	I	02		00000～02047	
	S	02		10000～14095	
3x	IW	04		00000～00511	
	SW	04		05000～09095	
	Event	04		10000～16173	
4x	MW	03	06 / 16	00000～16383	
	QW	03	06 / 16	20000～20511	
	NW	03	06 / 16	21000～25095	
	Clock	03	16	30000～30004	

Префикс	Тип регистра	Код функции операции чтения	Код функции операции записи	Адрес протокола	Примечание
	Clock	03	06/ 16	19990～19996	
	V	03	06 / 16	31000～47383	

Таблица адресов связи MODBUS RTU/TCP NAPLC.

Префикс	Тип регистра	Код функции операции чтения	Код функции операции записи	Адрес протокола
0X	Q	01	05 / 15	00000 до 16384
	M	01	05 / 15	20000 по 36383
	N	01	05 / 15	40 000 до 44 095
1X	I	02		00000 до 16384
	S	02		20000 по 22047
3X	IW	04		00000 до 04096
	SW	04		05000 - 09095
	Events	04		10000 до 16173
4X	MW	03	06 / 16	00000 до 32767
	QW	03	06 / 16	60 000 до 64 095
	NW	03	06 / 16	50 000 до 54 095
	Clock	03	16	65000 ~ 65004
	Clock	03	06	65010 ~ 65016

В протоколах MODBUS адрес начинается с 0. Например, адрес протокола %Q0005 равен 4. Формат события

NAPLC открыл область хранения событий, которая включает события SOE, аварийные сообщения и сообщения связи, а конкретные события можно прочитать следующим образом.

Начальный адрес события	10000
Область адресов определения события	10000 - 10029 30 слов
Максимальное количество событий SOE	256

Длина сообщения события SOE	8 слов
Максимальное количество сообщений тревоги	64
Длина сообщения о тревоге	32 слова
Максимальное количество сообщений связи	64
Длина сообщения	32 слова

ГДЕ:

10003 — это начальный адрес хранения событий SOE.

10005 — текущий указатель хранения событий SOE.

10013 — начальный адрес области хранения сообщений о тревогах.

10015 — текущая область хранения указателей для сообщений о тревогах.

10023 — начальный адрес области хранения сообщений связи.

10025 — текущая область хранения указателей для сообщений связи.

Формат сообщения о событии SOE

Адрес (байты)	**Имя**	**Описание**
1	Идентификация события	1
2	Характеристика	1: ВЫКЛ→ВКЛ. 0: ВКЛ→ВЫКЛ
3	Год	Это значение после (год - 2000)
4	Месяц	
5	День	
6	Время	
7	Минуты	
8	Секунды	
9	Миллисекунды	
10		
11	Номер переменной	
12		
13	Измеренные значения	0/1
14		

15 16	Резервный	0

Формат сообщения о тревоге

Адрес (байты)	Имя	Описание
1	Серийный номер последовательной контрольной карты	
2 3	Год	
4	Месяц	
5	День	
6	Час	
7	Минуты	
8	Секунды	
9 | | 64	Строка тревоги	Максимум 56 байт

Формат системных часов

	Год	Месяц	День	Час	Минута	Секунды	Миллисекунды
Адрес	30,000	30001	30001	30002	30003	30003	30004
Байты	2	1	1	2	1	1	2

Подробное описание кодов функций

01 Чтение статуса катушки

1. **Описание**

   Позволяет выполнять запросы на чтение для статуса ВКЛ/ВЫКЛ регистров измерения Q, M и N (классифицируются с префиксом 0x) на подчиненной станции.

2. **Запрос ведущего устройства**

   В сообщении-запросе определяется начальный адрес и количество регистров для считывания.

Пример

Чтение %Q20 по %Q56 на подчиненной станции 17

Байты	Значение	Пример (HEX)
1	Адрес подчиненной станции	11
2	Код функции	01
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	13
5	Количество старших 8 бит данных	00
6	Количество младших 8 бит данных	25
CRC16	-	-

3. **Ответ ведомого устройства**

   - Возвращаемые данные о состоянии от ведомого устройства следуют следующему шаблону: состояние начального адреса представлено младшим битом (LSB) первого байта данных и хранится в побитовом непрерывном порядке «от младшего к старшему, от младшего к старшему байту».
   - Если число возвращаемых состояний не является целым числом, кратным 8, к самому старшему байту, превышающему указанное значение, добавляется «0».
   - Индикация состояния: 1 = ВКЛ; 0 = ВЫКЛ.

Пример

Ответные сообщения от ведомого устройства для приведенного выше примера следующие.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	01
3	Количество байтов	05
4	Данные (%Q27-%Q20)	CD
5	Данные (%Q35-%Q28)	6B
6	Данные (%Q43-%Q36)	B2

7	Данные (%Q51-%Q44)	0E
8	Данные (%Q56-%Q52)	1B
CRC16	-	-

В примере:

- Статус %Q27-%Q20 возвращается как CDH (1100 1101 B), тогда старший бит (MSB) CDH является статусом %Q27, а младший бит (LSB) является статусом %Q20. Фактическое состояние %Q27 - %Q20 - ON-ON-OFF-OFF-ON-ON-OFF-ON.
- Последний 1 байт 1BH указывает состояние %Q56-%Q52: ВКЛ-ВКЛ-ВЫКЛ-ВКЛ-ВКЛ. Старшие 3 бита этого байта дополняются «0».

02 Чтение статуса входа

1. **Описание**

   Позволяет выполнять запросы на чтение состояния ВКЛ/ВЫКЛ регистров измерения I и S (классифицируются с префиксом 1x) на подчиненной станции.

2. **Запрос от главной станции**

   В сообщении-запросе определяется начальный адрес и количество регистров для считывания.

Пример

Чтение %I20 по %I56 на подчиненной станции 17

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	02
3	Начальный адрес на 8 бит выше	00
4	Начальный адрес младшие 8 бит	13
5	Количество старших 8 бит данных	00
6	Количество младших 8 бит данных	25
CRC16	-	-

3. **Ответ ведомого устройства**

- Далее следуют данные о состоянии возврата от ведомого устройства: состояние начального адреса представлено младшим битом (LSB) первого байта данных и

хранится в побитовом непрерывном порядке «от младшего к старшему, от младшего к старшему байту».

- Если число возвращаемых состояний не является целым числом, кратным 8, к самому старшему байту, превышающему указанное значение, добавляется «0».
- Индикация состояния: 1 = ВКЛ; 0 = ВЫКЛ.

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	02
3	Количество байтов	05
4	Данные (%I27-%I20)	CD
5	Данные (%I35-%I28)	6B
6	Данные (%I43-%I36)	B2
7	Данные (%I51-%I44)	0E
8	Данные (%I56-%I52)	1B
CRC16	-	-

В примере:

- Статус %I27-%I20 возвращается как CDH (1100 1101 B), тогда старший бит (MSB) CDH является статусом %I27, а младший бит (LSB) является статусом %I20. Фактическое состояние %I27 - %I20 - ON-ON-OFF-OFF-ON-ON-OFF-ON.
- Последний 1 байт 1BH указывает состояние %I56-%I52: ВКЛ-ВКЛ-ВЫКЛ-ВКЛ-ВКЛ. Старшие 3 бита этого байта дополняются «0».

03 Чтение регистров хранения

1. **Описание**

   Позволяет выполнять запросы на чтение измеренных значений MW、NW、QW и времени подчиненной системы (классифицируется с префиксом 4x) на подчиненной станции.

2. **Запрос от главной станции**

В сообщении-запросе определяется начальный адрес и количество регистров для считывания.

Пример

Чтение %MW108 по %MW110 на подчиненной станции 17

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	03
3	Начальный адрес на 8 бит выше	00
4	Начальный адрес младшие 8 бит	6B
5	Количество старших 8 бит данных	00
6	Количество младших 8 бит данных	03
CRC16	-	-

3. **Ответ ведомого устройства**

   Данные о состоянии, возвращаемые ведомым устройством, сохраняются непрерывно: «старший байт сначала, младший байт затем».

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	03
3	Количество байтов	06
4	Старшие данные 8 бит (%MW108)	02
5	Низкие данные 8 бит (%MW108)	2B
6	Старшие данные 8 бит (%MW109)	00
7	Низкие данные 8 бит (%MW109)	00
8	Старшие данные 8 бит (%MW110)	00
9	Низкие данные 8 бит (%MW110)	64
CRC16	-	-

В этом примере измеренные значения для %MW108-%MW110 возвращаются как 022BH, 0000H, 0064H.

4. **Чтение системного времени**

   Адрес системного времени — от 30000 до 30004.

Пример

Чтение системного времени на подчиненном устройстве 17

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	03
3	Стартовый адрес старшие 8 бит	75
4	Начальный адрес младшие 8 бит	30
5	Количество старших 8 бит данных	00
6	Количество младших 8 бит данных	05
CRC16	-	-

[Пример] Текст письма-ответа Slave выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	03
3	Количество байтов	0A
4	Высшие 8 бит года	07
5	Нижние 8 бит года	D8
6	День	0A
7	Месяц	09
8	Высокие 8 бит часа	00
9	Низкие 8 бит часа	0C
10	секунды	0E
11	Минуты	0D
12	Миллисекунды высотой 8 бит	01
13	Миллисекунды младшие 8 бит	02

CRC16	-	-

В примере:

- Вернуться к: 2008 (07D8H) 9 (09H) месяц 10 (0AH) день 12 (0CH) час 13 (0DH) минуты 14 (0EH) секунды 258 (0102H) миллисекунд

04 Чтение входных регистров

1. **Описание**

   Позволяет выполнять запросы на чтение измеренных значений IW, SW и событий (классифицируемых с префиксом 3x) со станции.

2. **Запрос от главной станции**

   В сообщении-запросе определяется начальный адрес и количество регистров для считывания.

Пример

Чтение %IW108 по %IW110 на подчиненной станции 17

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	04
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	6B
5	Количество старших 8 бит данных	00
6	Количество младших 8 бит данных	03
CRC16	-	-

3. **Ответ ведомого устройства**

   Данные о состоянии, возвращаемые ведомым устройством, сохраняются непрерывно: «старший байт сначала, младший байт затем».

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	04
3	Количество байтов	06
4	Старшие 8 бит данных (%IW108)	02
5	Данные младшие 8 бит (%IW108)	2B
6	Старшие 8 бит данных (%IW109)	00
7	Данные младшие 8 бит (%IW109)	00
8	Старшие 8 бит данных (%IW110)	00
9	Данные младшие 8 бит (%IW110)	64
CRC16	-	-

В этом примере измеренные значения %IW108-%IW110 возвращаются как 022BH, 0000H, 0064H.

4. Читать события

Пример

В подчиненном устройстве 17 последнее значение указателя SOE было 10039 (1 событие SOE), а на этот раз значение равно 10047 (2 события SOE), что указывает на то, что было сгенерировано новое событие SOE. Второе событие SOE должно быть прочитано по адресу 10038, а длина должна быть равна 8.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	04
3	Стартовый адрес старшие 8 бит	27
4	Начальный адрес младшие 8 бит	36
5	Количество старших 8 бит данных	00
6	Количество младших 8 бит данных	08
CRC16	-	-

Пример

Текст письма-ответа Slave выглядит следующим образом.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	04
3	Количество байтов	10
4	Природа SOE (1: ВЫКЛ→ВКЛ. (0: ВКЛ→ВЫКЛ)	01
5	Идентификация события SOE	01
6	Месяц	09
7	Год (разница с 2000)	02
8	Час	0B
9	День	0C
10	Секунды	0E
11	Минуты	0D
12	Миллисекунды высотой 8 бит (в 0,1 мс)	01
13	Миллисекунды младшие 8 бит (в 0,1 мс)	02
14	Номер переменной, 8 цифр	00
15	Номер переменной, младшие 8 цифр	02
16	Зарезервированные данные	00
17	Измеренные значения	01
18	Зарезервированные данные	00
19	Зарезервированные данные	00
CRC16	-	-

В примере

- Событие SOE: 12 сентября 2002 г. 11:13:14 25,8 мс Действие SOE2 (ВЫКЛ→ВКЛ).

05 Запись одной катушки

1. **Описание**

   Позволяет принудительно включать и выключать запросы на запись для одной переменной Q, N или M (классифицируемой префиксом 0x) на подчиненной станции.

2. **Запрос от главной станции**

- В сообщении-запросе определяются адрес и статус обязательной переменной.
- Принудительные данные: FF00H принудительно включен; 0000H принудительно выключен.

Пример

Принуждение%Q173 в положение ВКЛ на подчиненном устройстве 17

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	05
3	Начальный адрес на 8 бит выше	00
4	Начальный адрес младшие 8 бит	AC
5	Принудительные данные старшие 8 бит	F
6	Принудительные данные, младшие 8 бит	00
CRC16	-	-

3. **Ответ ведомого устройства**

   Если ответ правильный, ведомое устройство возвращает сообщение-запрос в его первоначальной форме.

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	05
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	AC
5	Принудительные данные старшие 8 бит	F
6	Принудительные данные, младшие 8 бит	00
CRC16	-	-

06 Запись одного регистра

1. **Описание**

   Позволяет записывать запросы на отдельные измерения MW, NW или QW (классифицируются с префиксом 4x) на подчиненной станции.

2. **Запрос от главной станции**

   В сообщении запроса определяются адрес и значение переменной записи.

Пример

Принудительно установить %MW2 на 3 на подчиненном устройстве 17 (адрес %MW2 — 0001H)

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	06
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	01
5	Запись значения старших 8 бит	00
6	Записать значение младших 8 бит	03
CRC16	-	-

3. **Ответ ведомого устройства**

   Если ответ правильный, ведомое устройство возвращает сообщение-запрос в его первоначальной форме.

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	06
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	01
5	Запись значения старших 8 бит	00
6	Записать значение младших 8 бит	03

CRC16	-	-

07 Запись нескольких катушек

1. **Описание**

   Позволяет принудительно включать и выключать запросы на запись для нескольких регистров измерения Q, N или M (классифицируемых префиксом 0x) на подчиненной станции одновременно.

2. **Запрос от главной станции**

   - В сообщении-запросе определяются начальный адрес, количество и обязательный статус регистров.
   - Укажите принудительное состояние побитно, как указано ниже: состояние начального адреса представлено младшим битом (LSB) первого байта данных и хранится непрерывно побитно, «от младшего бита к старшему и от младшего байта к старшему».
   - Если число возвращаемых состояний не является целым числом, кратным 8, к самому старшему байту, превышающему указанное значение, добавляется «0».
   - Принудительная индикация состояния: 1 = ВКЛ; 0 = ВЫКЛ.

Пример

Принудительное изменение %Q29 на %Q20 на подчиненной станции 17

ВЫКЛ-ВКЛ-ВКЛ-ВКЛ-ВЫКЛ-ВКЛ-ВКЛ-ВКЛ-ВКЛ-ВЫКЛ-ВКЛ, затем данные CDH, 01H

Bit:	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1
Q	27	26	25	24	23	22	21	20	-	-	-	-	-	-	29	28

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	0F
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	13
5	Принудительное число старших 8 бит	00
6	Принудительное число младших 8 бит	0A

7	Количество обязательных байтов данных	02
8	принудительное значение байта данных 1	CD
9	принудительное значение байта данных 2	01
CRC16	-	-

3. **Ответ ведомого устройства**

   Правильно, ведомое устройство возвращает начальный адрес и количество обязательных регистров измерения.

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	0F
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	13
5	Принудительное число старших 8 бит	00
6	Принудительное число младших 8 бит	0A
CRC16	-	-

10 Запись нескольких регистров

1. **Описание**

   Позволяет подчиненному устройству настроить запрос на запись для нескольких регистров измерения MW, NW, QW или системного времени (классифицируется с префиксом 4x) на подчиненной станции.

2. **Запрос от главной станции**

   Сообщение-запрос определяет начальный адрес, номер и значение переменной для запрошенной операции записи.

Пример

Запись %MW2 в %MW3 как 000AH, 0102H на подчиненной станции 17

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	10
3	Начальный адрес на 8 бит выше	00
4	Начальный адрес младшие 8 бит	01
5	Количество регистров записи старшие 8 бит	00
6	Количество регистров записи младшие 8 бит	02
7	Количество записанных байтов данных	04
8	Регистр 1 данные старшие 8 бит	00
9	Регистр 1 Данные младшие 8 бит	0F
10	Регистр 2, старшие 8 бит данных	01
11	Данные регистра 2 младшие 8 бит	02
CRC16	-	-

3. **Ответ ведомого устройства**

   Правильно, ведомое устройство возвращает начальный адрес и количество регистров измерения для записи.

Пример

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	Значение	Пример (HEX)
1	Адрес ведомого устройства	11
2	Код функции	10
3	Стартовый адрес старшие 8 бит	00
4	Начальный адрес младшие 8 бит	01
5	Количество регистров записи старшие 8 бит	00
6	Количество регистров записи младшие 8 бит	02
CRC16	-	-

4. **Установка системного времени**

Адрес системного времени — от 30000 до 30004.

Пример

Установите системное время ведомого устройства 17 на: 2008 (07D8H) год 9 (09H) месяц 10 (0AH) день 12 (0CH) час 13 (0DH) минуты 14 (0EH) секунды 0 (0000H) миллисекунд.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	10
3	Стартовый адрес старшие 8 бит	75
4	Начальный адрес младшие 8 бит	30
5	Количество регистров записи старшие 8 бит	00
6	Количество регистров записи младшие 8 бит	05
7	Количество записанных байтов данных	0A
8	Старшие 8 бит года	07
9	Младшие 8 бит года	D8
10	День	0A
11	Месяц	09
12	Высокие 8 бит часа	00
13	Низкие 8 бит часа	0C
14	секунды	0E
15	Минута	0D
16	Миллисекунды высотой 8 бит	00
17	Миллисекунды младшие 8 бит	00
CRC16	-	-

Пример

Сообщение-ответ ведомого устройства выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Адрес ведомого устройства	11
2	Код функции	10

3	Стартовый адрес старшие 8 бит	75
4	Начальный адрес младшие 8 бит	30
5	Количество регистров записи старшие 8 бит	00
6	Количество регистров записи младшие 8 бит	05
CRC16	-	-

Спецификация MODBUS/TCP

- MODBUS/TCP — сетевой протокол TCP для MODBUS, номер сетевого порта TCP 502.

MODBUS/TCP добавляет 6-байтовый заголовок протокола к ранее описанному протоколу MODBUS и удаляет обычную контрольную сумму MODBUS CRC16. Формат следующий.

Байты	Значение	Value
1	Протокол идентификации	00H
2	Протокол идентификации	00H
3	Протокол идентификации	00H
4	Протокол идентификации	00H
5	Длина сообщения MODBUS высокая 8 бит	00H
6	Длина сообщения MODBUS не менее 8 бит	Байт 7 и последующие байты
7	Адрес ведомого устройства	0 до FFH
8	Код функции	См. 12.4.
9 ... н	Данные	См. 12.4.

Пример

Чтение %IW108 по %IW110 на подчиненном устройстве 17. Главное устройство отправляет следующее сообщение.

Байты	Значение	Пример (HEX)
1	Протокол идентификации	00
2	Протокол идентификации	00
3	Протокол идентификации	00
4	Протокол идентификации	00
5	Длина сообщения MODBUS высокая 8 бит	00
6	Длина сообщения MODBUS не менее 8 бит	06
7	Адрес ведомого устройства	11
8	Код функции	04

9	Стартовый адрес старшие 8 бит	00
10	Начальный адрес младшие 8 бит	6B
11	Количество старших 8 бит данных	00
12	Количество младших 8 бит данных	03

Ответное сообщение от Slave на приведенный выше пример выглядит следующим образом.

Байты	**Значение**	**Пример (HEX)**
1	Протокол идентификации	00
2	Протокол идентификации	00
3	Протокол идентификации	00
4	Протокол идентификации	00
5	Длина сообщения MODBUS высокая 8 бит	00
6	Длина сообщения MODBUS не менее 8 бит	09
7	Адрес ведомого устройства	11
8	Код функции	04
9	Количество байтов	06
10	Старшие 8 бит данных (%IW108)	02
11	Данные младшие 8 бит (%IW108)	2B
12	Старшие 8 бит данных (%IW109)	00
13	Данные младшие 8 бит (%IW109)	00
14	Старшие 8 бит данных (%IW110)	00
15	Данные младшие 8 бит (%IW110)	64

В этом примере измеренные значения %IW108 – %IW110 возвращаются как 022BH, 0000H и 0064H.

Руководство по программированию MODBUS/TCP

Подчиненным устройством является MODBUS/TCP Server, а пользовательской программой является Client. Ниже описываются спецификации программирования для клиентской программы.

Процедура клиента MODBUS/TCP состоит из следующих шагов.

1. установить потоковый сокет (TCP) с помощью вызова сокета ( ).
2. подключить сокет s к ведомому устройству с помощью вызова connect ( ) (порт 502).
3. подготовка сообщений MODBUS/TCP.
4. отправить сообщение на сокет s с помощью вызова send().
5. используйте вызов select(), чтобы определить, доступны ли данные на сокете s, с таймаутом 10 с.
6. Вызов recv() используется для получения сообщения на сокете s. Сначала принимается 6-байтовый заголовок MODBUS/TCP, а затем принимается само сообщение в соответствии с длиной в заголовке.
7. обработка сообщений.
8. повторите шаги 3–7.

Настройки связи по MODBUS

Подчиненное устройство MODBUS RTU

Процессоры NAPLC интегрированы с интерфейсами RS232 или RS485. Выбор MODBUS RTU через протокол на рисунке ниже означает, что последовательный порт работает как подчиненное устройство RTU, а сенсорный экран, конфигурационное программное обеспечение и т. д. могут выступать в качестве главного устройства для считывания адресов регистров ПЛК. Это показано на рисунке 13.1 ниже.

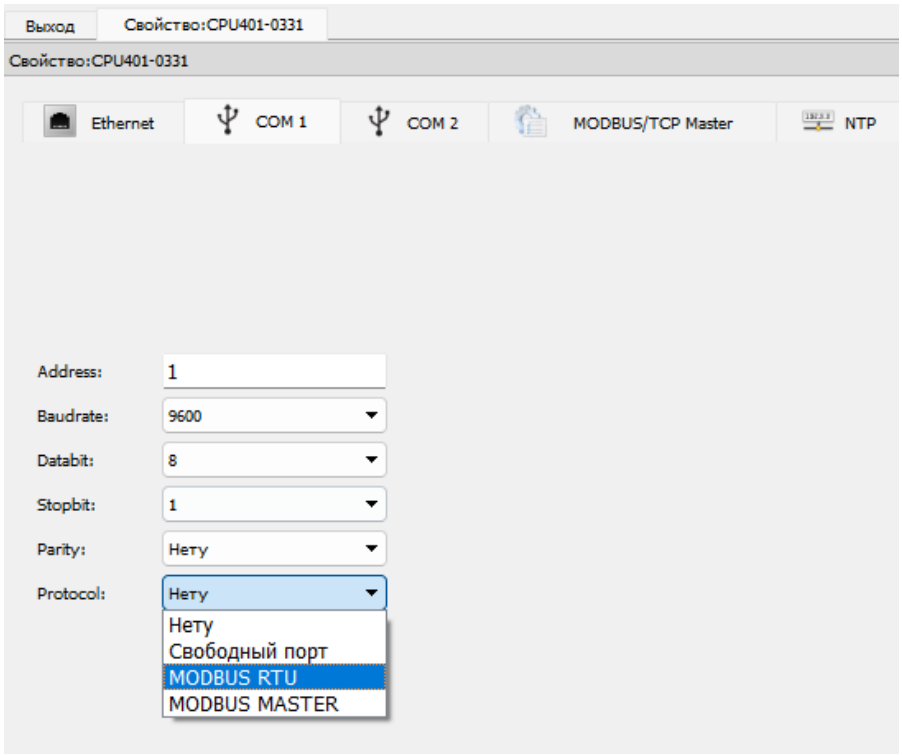


Рисунок 13.1 Настройка протокола связи

Таблица соответствия между кодами функций и категориями данных

Соответствие между кодами функций и соответствующими данными в данном руководстве показано ниже (в качестве примера приведен CPU401-0501, другие типы).

(Верхний предел адреса переменной ЦП отличается от уставного адреса).

Префикс	Тип регистра	Код функции операции чтения	Код функции операции записи	Адрес протокола	Примечание
0X	Q	01	05 / 15	00000 to 02047	Диапазон адресов варьируется в зависимости от процессора,
	M	01	05 / 15	10000 to 26383	
	N	01	05 / 15	30000 to 34095	

1X	I	02		00000 to 02047	см. таблицу ниже.
	S	02		10000 to 14095	
3X	IW	04		00000 to 00511	
	SW	04		05000 to 09095	
	Events	04		10000 to 16173	
4X	MW	03	06 / 16	00000 to 16383	
	QW	03	06 / 16	20000 to 20511	
	NW	03	06 / 16	21000 to 25095	
	Clock	03	16	30000 to 30004	
	Clock	03	06/ 16	19990 - 19996	
	V	03	06 / 16	31000 to 47383	

Таблица адресов связи MODBUS RTU/TCP для NAPLC (CPU401-0701).

Пре-фикс	Тип ре-гистра	Код функ-ции опера-ции чтения	Код функ-ции опера-ции записи	Адрес протокола	Примечание
0X	Q	01	05 / 15	00000 to 16384	Подробную информацию о диапазоне адресов см. в следующей таблице.
	M	01	05 / 15	20000 to 36383	
	N	01	05 / 15	40,000 to 44095	
1X	I	02		00000 to 16384	
	S	02		20000 to 22047	
3X	IW	04		00000 to 04096	
	SW	04		05000 to 09095	
	Events	04		10000 to 16173	
4X	MW	03	06 / 16	00000 to 32767	
	QW	03	06 / 16	60,000 to 64095	
	NW	03	06 / 16	50,000 to 54095	
	Clock	03	16	65000~65004	
	Clock	03	06	65010~65016	

Примечание
В протоколах MODBUS, адрес начинается с 0. Например, адрес протокола %Q0005 является 4.

Протокол ведомого устройства MODBUS TCP

Стандартный Ethernet-интерфейс NAPLC использует протокол MODBUSTCP, где ПЛК выступает в роли сервера, а сенсорная панель (тачскрин) или программное обеспечение для конфигурирования - в роли клиента для доступа. Номер порта по умолчанию - 502 (не подлежит изменению). ПЛК в роли сервера имеет ограничение на максимальное количество клиентов, которые могут подключаться одновременно. Для NA300/NA400 это ограничение составляет 16 подключений. Это проиллюстрировано на рисунке 13.2 ниже.

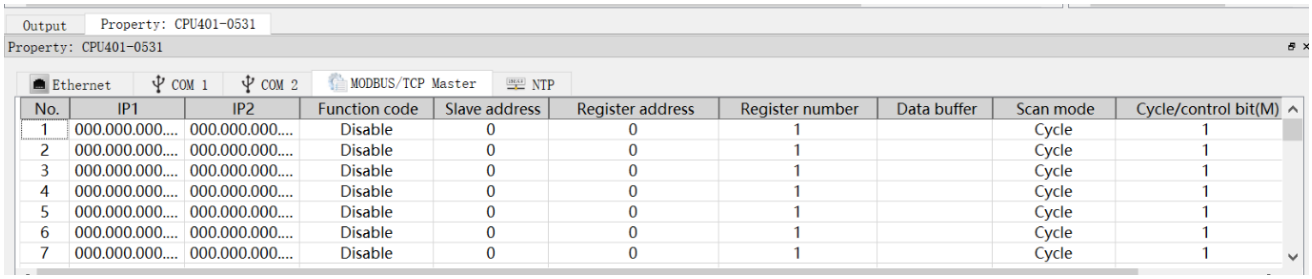


Рисунок 13.2 Схема настройки протокола MODBUS TCP

Таблица адресов переменных протокола MODBUS TCP

В данном руководстве ниже показано соответствие между кодами функций и соответствующими данными.

Префикс	Тип регистра	Код функции операции чтения	Код функции операции записи	Адрес протокола	Примечание
0X	Q	01	05 / 15	00000 to 02047	Диапазон адресов варьируется в зависимости от процессора, см. таблицу ниже.
	M	01	05 / 15	10000 to 26383	
	N	01	05 / 15	30000 to 34095	
1X	I	02		00000 to 02047	
	S	02		10000 to 14095	
3X	IW	04		00000 to 00511	
	SW	04		05000 to 09095	
	Events	04		10000 to 16173	
4X	MW	03	06 / 16	00000 to 16383	
	QW	03	06 / 16	20000 to 20511	
	NW	03	06 / 16	21000 to 25095	

	Clock	03	16	30000 to 30004	
	Clock	03	06/ 16	19990 - 19996	
	V	03	06 / 16	31000 to 47383	

Примечание
В протоколах MODBUS адрес начинается с 0. Например, адрес протокола %Q0005 равен 4.

Настройки MODBUS для связи с ведущим устройством

Конфигурация MODBUS RTU Master

Выберите ПЛК NA200H, последовательные порты которого могут поддерживать протокол Modbus Master, как показано ниже:

[Выбор последовательного порта]: Настройте параметры последовательных портов. Выберите протокол MODBUS MASTER, и появится интерфейс конфигурации. (Рисунок 13.3)

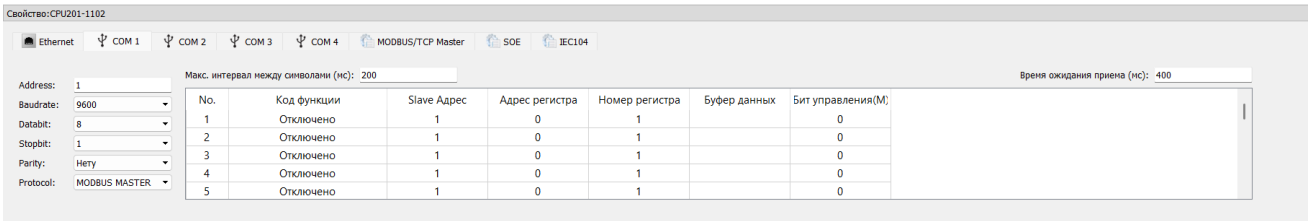


Рисунок 13.3 Схема конфигурации протокола

[**Макс. внутреннее значение для символов (10–1000, по умолчанию 200 мс)**]: После отправки команды полученное сообщение не может быть передано за один раз, а максимальный интервал — это интервал между любыми двумя символами при нескольких передачах сообщения.

[**Таймаут приема (10–10000, по умолчанию 400 мс)**]: После того, как ведущий отправит команду, дождитесь времени ответа ведомого устройства.

[**Число**]: можно настроить максимум 64 команды кода функции.

[**Код функции**]: Если функция Modbus не настроена, функция Disable. Конфигурацию можно определить в соответствии с кодом функции Modbus. Modbus поддерживает следующие коды функций:

Код функции (десятичный)	**Определение**
01	Чтение статуса катушки
02	Чтение статуса входа
03	Чтение регистров хранения
04	Чтение входных регистров
05	Запись одной катушки
06	Запись одного регистра
15	Запись нескольких катушек
16	Запись нескольких регистров

[**Адрес ведомого устройства (1-255)**]: адрес связи ведомых устройств последовательных портов Modbus.

[**Адрес регистра (0-65535)**]: Ведущее устройство выполняет операцию чтения и записи Modbus в регистры подчиненных устройств. Максимальное значение зависит от доступного диапазона регистра подчиненных устройств.

- Примечание: Количество регистров в кодах функций 05 и 06 может быть установлено только равным 1, что определяется определением самого кода функции.
- Количество регистров в функциональных кодах 01, 02 и 15 варьируется от 1 до 1920.
- Количество регистров кодов функций 03, 04 и 16 варьируется от 1 до 120.

[**Буфер данных**]: может быть установлен на имя переменной %M, %MW, %N, %NW, V. Когда ведущее устройство считывает и записывает данные в ведомое устройство, буфер данных является начальным адресом области хранения данных, а количество данных — это количество регистров, настроенных ранее.

[**Бит управления**]: Установив регистр управления позицией %M, вы можете запустить время отправки этой инструкции чтения и записи в соответствии с необходимостью, когда %M переходит от 0 к 1, команда отправляется один раз, и команда автоматически очищается после успешной передачи. Если управление не требуется, эта команда устанавливается в 0. В этом случае команда чтения/записи выполняется каждые 500 мс. Например, бит управления устанавливается в 2, то есть операция контролируется %M2, и после того, как %M2 устанавливается в 1, команда выполняется один раз, а затем %M2 автоматически возвращается в 0.

[**Маркерные биты для общения**]:

Номер последовательного порта	**Системные регистры**	**Номер**	**Статус**
COM1	%SW513-%SW516	1-64	1 указывает на ошибку связи. Каждый бит соответствует серийному номеру.
COM2	%SW517-%SW520	1-64	1 указывает на ошибку связи. Каждый бит соответствует серийному номеру.
COM3	%SW521-%SW524	1-64	1 указывает на ошибку связи. Каждый бит соответствует серийному номеру.

COM4	%SW525-%SW528	1-64	1 указывает на ошибку связи. Каждый бит соответствует серийному номеру.

Реализация функции ведущего устройства MODBUS RTU с использованием функционального блока MODRW

Центральный процессор ПЛК NA300/400 не поддерживает функцию главной конфигурации, и пользователи могут реализовать функцию главной станции MODBUS RTU через предоставленный стандартный функциональный блок.

Описание функционального блока MODRW

Этот функциональный блок используется для реализации функции чтения и записи данных стандартного главного протокола MODBUS, для автоматического анализа сообщений протокола MODBUS и для проверки CRC и длины данных. Пользователям нужно только заполнить адрес чтения-записи данных и данные функции, чтобы реализовать функцию связи. Вызов функционального блока требует триггера синхронизации, а периодический интервал — это период для чтения и записи данных с минимальным значением 50 мс. Интервал времени связи должен быть скорректирован в соответствии со временем ответа ведомого устройства во время связи, в противном случае данные могут быть прочитаны неправильно или не могут быть прочитаны.

Показатели коммуникационных состояний:

CW21 (состояние отправки COM1), SW23 (состояние отправки COM2)

0: в процессе передачи

1: успешная передача

2: сбой передачи

SW22 (состояние приема COM1), SW24 (состояние приема COM2)

0: в приеме

1: успешно получено

2: неисправность последовательного порта

3: тайм-аут приема

4: характерный интервал слишком длинный

5: больше максимального количества символов в сообщениях

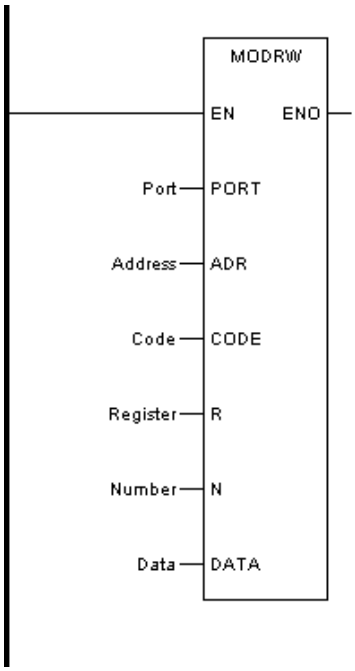
7: неправильное ответное сообщение

8: неправильное сообщение о запросе

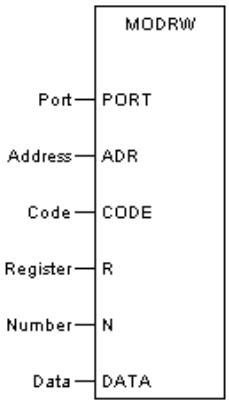
9: неправильная проверка

ВЫЗОВ

Форма в LD:



Форма в FBD:



Форма на языке IL:

CAL MODRW (PORT:=Порт, ADR:=Адрес, CODE:=Код, R:=Регистр, N:=Номер, DATA:=Данные)

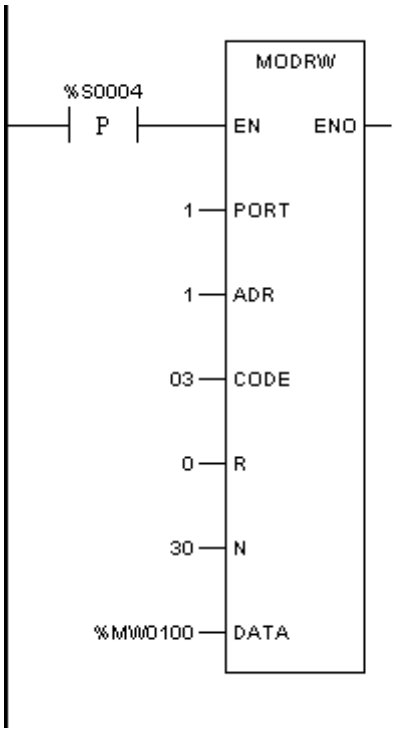
Форма в ST:

MODRW (PORT:=Порт, ADR:=Адрес, CODE:=Код, R:=Регистр, N:=Номер, DATA:=Данные);

Описание параметра:

Диа-грамма	Параметр	Описание	Тип данных	Тип регистра
PORT	PortNo	Номер последовательного порта (1 ИЛИ 2), конкретное определение параметра, относящегося к конфигурации ЦП.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
ADR	Address	Адрес ведомого устройства MODBUS в диапазоне от 1 до 255.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
CODE	Code	Стандартный функциональный код протокола MODBUS, в настоящее время поддерживающий следующие функциональные коды: 01, 02, 03, 04, 05, 06, 15, 16.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
R	Register	Адрес регистров данных.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
N	Number	Количество регистров чтения-записи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	CONSTANT, IW, QW, MW, NW, SW, VAR
DATA	Data	Буфер данных чтения-записи, хранящий данные для передачи и получения.	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	IW, QW, MW, NW, SW, I, Q, M, N, S, VAR

ПРИМЕР:



Описание: В приведенном выше примере показано, что каждую секунду последовательный порт 1 ЦП используется для считывания данных 30 последовательных регистров, начиная с 0, который поступает от ведомого устройства с адресом 1, с использованием кода функции 03. И считанные данные сохраняются в 30 последовательных регистрах, начиная с %MW100.

Примечания: этот функциональный блок не может использоваться в CPU401-1101 и CPU201-1101 и использовать интерфейс конфигурации MODBUS RTU master для завершения чтения и записи MODBUS. Между тем, все модули, которые могут конфигурировать MODBUS MASTER в модуле, не могут использовать этот функциональный блок.

Основная конфигурация MODBUS/TCP

Выберите [MODBUS/TCP Master] для настройки MODBUS/TCP Master.

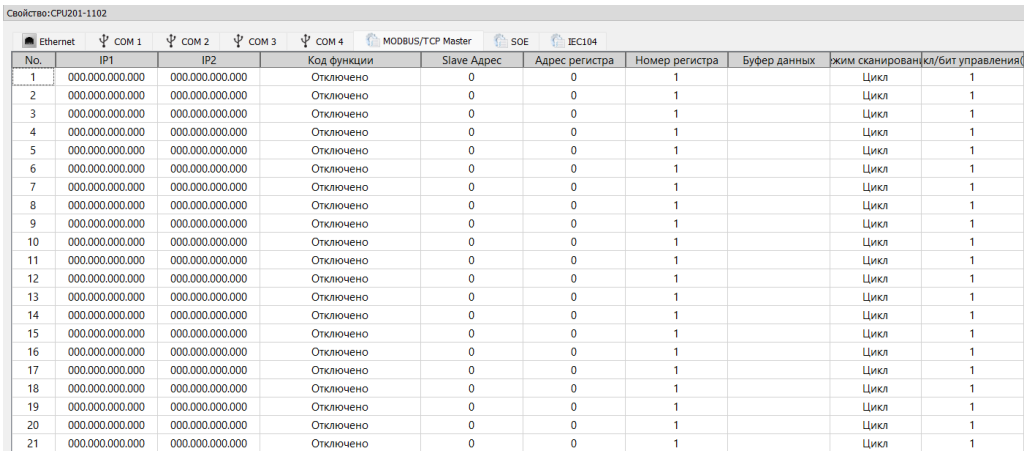
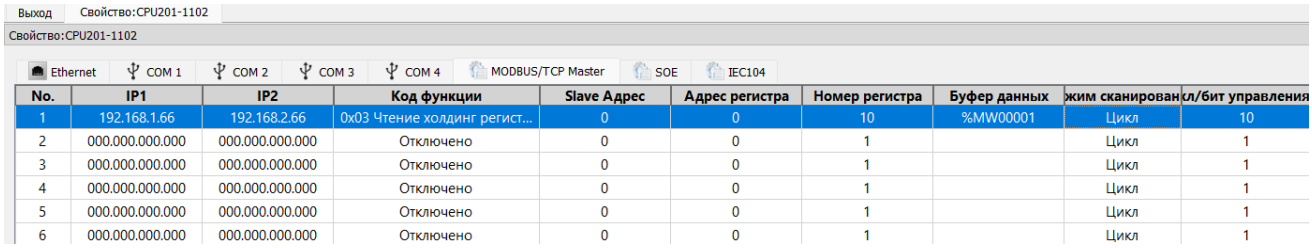


Рисунок 13.4 Интерфейс конфигурации MODBUS/TCP

[**No.**]: Можно настроить максимум 256 команд кода функции.

[**IP-адрес**]: IP-адрес ведомого устройства, с которым необходимо связаться.

Примечание: IP-адреса главного и подчиненного устройств должны находиться в одном сегменте сети.



[**Код функции**]: Если не настроено, функция отключена. Конфигурацию можно определить в соответствии с кодом функции Modbus/TCP. Modbus/TCP поддерживает следующие коды функций:

Код функции (десятичный)	**Определение**
01	Чтение статуса катушки
02	Чтение статуса входа
03	Чтение регистров хранения
04	Чтение входных регистров
05	Запись одной катушки
06	Запись одного регистра
15	Запись нескольких катушек
16	Запись нескольких регистров

[**Slave Адрес (1-255)**]: этот параметр необходим для самого протокола MODBUS, но для большинства продуктов связи MODBUS/TCP (таких как наш ПЛК серии NA) этот параметр не требуется, поэтому вы можете настроить его по своему усмотрению. Однако для продуктов с этим параметром адрес подчиненной станции должен быть настроен в строгом соответствии с требованиями.

[**Адрес регистра (0-65535)**]: Ведущее устройство выполняет операцию чтения и записи Modbus/TCP в регистры подчиненных. Здесь указан адрес регистра подчиненной станции. Для разных подчиненных станций адрес регистра также отличается.

[**Номер регистра**]: Количество подчиненных регистров, которые главный считывает и записывает через коды функций MODBUS/TCP. Максимальное значение зависит от доступного диапазона регистров в подчиненном устройстве.

- Примечание: Количество регистров в кодах функций 05 и 06 может быть установлено только равным 1, что определяется определением самого кода функции.
- Количество регистров в функциональных кодах 01, 02 и 15 варьируется от 1 до 1920.
- Количество регистров кодов функций 03, 04 и 16 варьируется от 1 до 120.

[**Буфер данных**]: может быть установлен на имя переменной %M, %MW, %N, %NW, V. Для кода функции записи вы можете установить его на имя переменной %I, %Q, %IW, %QW, %M, %MW, %N, %NW, %S, %SW и V. Это начальный адрес области, в которой хранятся данные, когда ведущее устройство считывает и записывает данные на ведомое устройство.

[**Режим сканирования**]:

1. **Cycle**: Периодически отправлять эту команду. Период варьируется от 1 до 36000. Минимальная единица — 100 мс. То есть, если здесь установлено значение 1, период устанавливается на 100 мс. Рекомендуемое значение — 10, т. е. 1 с.
2. **Управление битами**: установив регистр управления позицией %M, вы можете запустить время отправки этой инструкции чтения и записи в соответствии с необходимостью, когда %M переходит от 0 к 1, команда отправляется один раз, и команда автоматически очищается после успешной передачи. Если управление не требуется, эта команда устанавливается в 0. В этом случае команда чтения/записи выполняется каждые 500 мс. Например, бит управления устанавливается в 2, то есть операция контролируется %M2, и после того, как %M2

устанавливается в 1, команда выполняется один раз, а затем %M2 автоматически возвращается в 0.

[**Биты маркера для связи**]: %S0145-%S0399 соответствуют состоянию чтения и записи кодов функций серийных номеров 1-255. 1 указывает на то, что связь неисправна, а 0 указывает на то, что связь нормальная.

Краткое руководство пользователя по программированию

Быстрое руководство пользователя по программированию.

Инвентаризация предметов

Пожалуйста, проверьте, что номер вашей детали соответствует номеру вашего заказа и что упаковка полная. Если упаковка повреждена или отсутствует какой-либо продукт, свяжитесь с поставщиком как можно скорее.

Монтаж оборудования, проводка

Сначала выберите соответствующий модуль ЦП и модуль расширения в соответствии с фактическими потребностями проекта; затем примите решение об установке модуля в соответствии с условиями на месте и изначально определить работу ПЛК; спланируйте и разработайте разумную схему электропроводки для подключения датчиков или исполнительных механизмов на месте к клеммам модуля ПЛК.

Подключение кабеля питания

Подключите кабель питания в соответствии с моделью и типом выбранного модуля ЦП.

Примечание: Выбирайте разные уровни напряжения для разных моделей, обращайте внимание на уровень напряжения, положительную и отрицательную полярность, а также уделяйте больше внимания безопасности источника питания при выборе модулей питания 220 В переменного тока, чтобы избежать ненужных травм или повреждения оборудования! После подключения кабеля питания не включайте питание первым. После проверки правильности подключения всех кабелей включите питание системы и убедитесь, что индикатор питания P на панели модуля ЦП горит и правильно отображает данные, чтобы обеспечить надежную работу ПЛК.

Установление связи с ПК

Модуль ЦП подключается к ПК через обычный кабель Ethernet для установления канала передачи данных. ЦП поставляется с двумя портами Ethernet, любой из двух портов Ethernet может быть подключен к ПК. IP-адрес ПЛК по умолчанию - «192.168.1.66».

Первая загрузка программы

При первой загрузке программы установите трехпозиционный переключатель в положение «Debug» и используйте ПЛК для работы в режиме DEBUG для ручной загрузки. После завершения ручной загрузки ПЛК можно запустить в положении «Run» с трехпозиционным переключателем, установленным в положение «Run». Ниже приведено краткое описание ручной загрузки.

Ручные загрузки используются для изменения сетевого IP-адреса ЦП при первой загрузке файлов проекта на ЦП или, когда IP-адрес Ethernet ЦП забыт. Ручная загрузка автоматически загрузит весь файл проекта и файлы программы на ЦП. Сначала поверните DIP-переключатель ЦП в положение «Debug», снова включите ЦП и после успешного запуска начнет моргать индикатор R на лицевой стороне модуля ЦП и красный индикатор «F» погаснет.

Измените IP-адрес в сетевом подключении ПК и IP-адрес по умолчанию ЦП так, чтобы они находились в одном сегменте сети (первые три сегмента одинаковы, но последний сегмент отличается), как показано на рисунке 14.1.

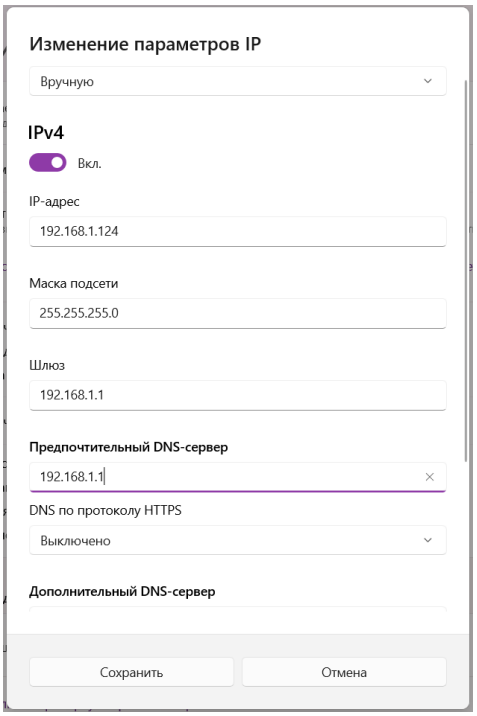


Рисунок 14.1 Настройка IP

Выбор меню INDAS PRO «Загрузка/Ручная загрузка» открывает следующее окно, показанное на рисунке 14.2.

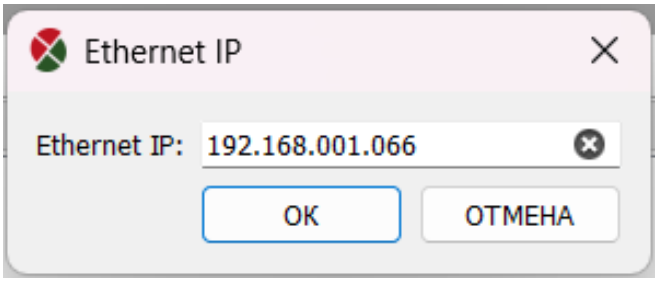


Рисунок 14.2 Ручная загрузка

Если сетевое оборудование не работает или адрес неверный, система выдаст окно с сообщением об ошибке подключения, пожалуйста, проверьте настройки сети в это время. Команда ping 192.168.1.66 может быть использована для проверки того, является ли состояние сети нормальным. Если время отклика при пинговании нормальное, но вы не можете загрузить программу, пожалуйста, проверьте настройки брандмауэра вашего компьютера, чтобы увидеть, заблокирована ли передача файлов или не занесен ли INDAS PRO в черный список и т. д. Для систем Win7, Win8 и Win10 необходимо изменить настройки брандмауэра или отключить его, в противном случае вы не сможете загрузить.

После успешной загрузки переведите DIP-переключатель на ЦП в положение RUN, и ЦП перезагрузится, индикатор R будет мигать один раз в секунду, а индикатор A будет гореть, указывая на то, что ЦП работает нормально.

После загрузки программы в ПЛК необходимо сбросить и перезапустить модуль ЦП, прежде чем загруженная программа сможет быть выполнена, в противном случае система все равно выполнит предварительно загруженную программу. Сброс можно выполнить с помощью команды сброса программного обеспечения для программирования.

Написание управляющих программ

Программное обеспечение INDAS PRO устанавливается на персональный компьютер (ПК) для установления связи с модулем ЦП, настройки соответствующих параметров

оборудования, а затем проектирования и разработки соответствующих инженерных приложений для реализации и удовлетворения ваших требований к управлению.

Оборудование в эксплуатации

После проверки и подтверждения правильности всех процессов программа управления загружается в ПЛК, и система управления на базе NaPLC готова к производству.

Версия записи

Дата	Описание	Версия
28.02.2025	Первоначальная версия	2025.1.2
07.03.2025	Исправление ошибок	2025.1.3

О программе

ПО: INDAS PRO 2025.1.3