«СибКом»

Непубличное Акционерное Общество

АО «СибКом»

ПРО

Программирование ПЛК

СИСТЕМНОЕ РУКОВОДСТВО

Программное обеспечение «СКПро»

РОФ.СБПУ.425200.641.РС-01

2023/03

Справочная и правовая информация

Информация о руководстве

Данное руководство содержит указания и положения, которые необходимо соблюдать для обеспечения персональной безопасности и предотвращения материального ущерба.

Опасность
Указывает что несоблюдение надлежащих мер безопасности приведет к существенному вреду здоровью или серьезному материальному ущербу

Предостережение
Указывает что несоблюдение надлежащих мер безопасности может привести к существенному вреду здоровью или серьезному материальному ущербу

Предупреждение
Указывает что несоблюдение надлежащих мер безопасности может привести к материальному ущербу

Указание
Указывает на информацию, которая обязательно должна быть принята к сведению.

Квалифицированный персонал

Работать с продуктом или системой, описываемой в данной документации, должен только квалифицированный персонал, допущенный для выполнения поставленных задач и соблюдающий указания документации, в частности, указания и предупреждения по технике безопасности. Квалифицированный персонал в силу своих знаний и опыта в состоянии распознать риски при обращении с данными системами и избежать возникающие угрозы.

Использования продукта или системы

Продукт или систему разрешается использовать только для целей, указанных в соответствующей технической документации.

Исходными условиями для безошибочной и надёжной работы продукта или системы являются надлежащие транспортировка, хранение, размещение, монтаж, оснащение, ввод в эксплуатацию, обслуживание и поддержание в исправном состоянии. Обязательно учитывайте указания в соответствующей документации.

Товарные знаки

Товарные знаки «СК-1000», «СК-4000», «СКПро» являются собственностью АО «СибКом». Все другие товарные знаки, используемые или упоминаемые в данном руководстве, являются собственностью их уважаемых владельцев. Никакая информация, содержащаяся в этом руководстве, не должна быть истолкована как предоставление тем или иным образом лицензии или права на использование любого товарного знака.

Исключение ответственности

Содержимое данного руководство проверено на соответствие с описанным программным и аппаратным обеспечением. Тем не менее, разночтения, либо отклонения не могут быть исключены, в связи с чем мы не гарантируем полное соответствие. Данные в данном руководстве подвергаются регулярной проверке и соответствующие изменения вносятся в последующие редакции данного руководства.

СОДЕРЖАНИЕ

1 Обзор

Данное руководство описывает процесс создания, редактирования и отладки пользовательских программ для контроллеров СК-4000 с помощью программного обеспечения «СКПро».

СК-4000 — это новое поколение программируемых логических контроллеров, поставляемых компанией «СибКом». При разработке СК-4000 применены новейшие достижения в области промышленной автоматизации и использованы совершенно новые программные и аппаратные платформы. Семейство СК-4000 обладает высокой производительностью, мощной защитой от помех и гибкими возможностями по расширению. Его просто применять в любых сложных средах и с учетом любых требований к обработке.

Программное обеспечение «СКПро» является важной составляющей экосистемы контроллеров СК-1000 и СК-4000 и представляет собой интегрированную среду программирования для ПЛК, включающую в себя редактор, компилятор, отладчик, эмулятор и инструмент графического пользовательского интерфейса, и используемую для выполнения конфигурирования оборудования, настройки каналов ввода-вывода, программирования прикладного ПО, моделирования, отладки и загрузки. Данное программное обеспечение предоставляет персоналу инженерного и технического подразделений простой и практичный инструмент для программирования и онлайн-отладки программ ПЛК.

ПО «СКПро» поддерживает такие языки, как язык релейно-контактных схем (LD), язык функциональных блоковых диаграмм (FBD), список инструкций (IL) и структурированный текст (ST) в соответствии со стандартом МЭК 61131-3, а также поддерживает специфический язык последовательных управляющих схем (SCC).

1.1 Особенности программного обеспечения

1.1.1 Совместимость с операционными системами

Сегодня графический пользовательский интерфейс является базовым функционалом для ряда задач. По этой причине ПО «СКПро» разработано, как программа для Microsoft Windows и совместимо со всеми современными версиями Windows. ПО «СКПро» разработано на языке VC++6.0 для Windows, потому его интерфейс полностью совместим с дизайном окон данной системы. Это включает в себя стандарты построения меню, использования горячих клавиш, мыши и инструментальной панели. Применение данного решения снижает время на обучение персонала и производственные затраты за счет простоты использования.

1.1.2 Поддержка международного стандарта МЭК

Из-за различий в системах инструкций у производителей ПЛК и различий в требованиях конечных пользователей к методам программирования, Международная электротехническая комиссия (МЭК) разработала стандарт языков программирования

МЭК 61131-3 для систем в среде Windows (в 1993, МЭК обнародовала международный стандарт на ПЛК – МЭК 61131), который определяет пять языков программирования: список инструкций (IL), структурированный текст (ST), релейно-контактные схемы (LD), функциональные блоковые диаграммы (FBD), последовательная функциональная схема (SFC). Стандарт включает в себя как текстовое (IL, ST), так и графическое (LD, FBD) программирование, и SFC, который можно отнести к обеим группам. Это стандартизованный список, с которым программируемый логический контроллер – устройство, созданное на основе цифровых технологий – работает на высоком уровне, что является определяющим трендом развития ПЛК.

Программное обеспечение «СКПро» обеспечивает унифицированную и эффективную среду для конфигурирования системы, в соответствии с международным стандартом МЭК 61131-3, позволяя специалисту «выучив раз, применять везде».

1.1.3 Управление проектами – древовидная структура управления

Программное обеспечение «СКПро» применяет концепции управления проектами, отображая их в виде древовидной структуры в интегрированной среде разработки и визуально отображает содержимое программы в виде нескольких документов. Таким образом, всё соответствующее содержимое, а также разработка или внесение изменений в программу становятся интуитивно понятными.

1.1.4 Языки программирования

Как решение для промышленной автоматизации, программное обеспечение «СКПро» поддерживает стандартные языки, совместимые со стандартом МЭК 61131-3: список инструкций (IL), язык релейно-контактных схем (LD), язык функциональных блоковых диаграмм (FBD) и структурированный текст (ST), и предоставляет новый тип языка программирования последовательных управляющих схем (SCC - Sequential Control Chart). Программа, созданная на вышеперечисленных языках, может быть перекрестно вызвана, что делает программирование более гибким и соответствующим требованиям различных сложных условий. При этом LD, FBD и SCC используют графический редактор и являются гибкими, удобными и быстрыми. Все языки поддерживают функции быстрого доступа *Cut* («Вырезать»), *Copy* («Копировать»), *Paste* («Вставить»), *Delete* («Удалить»), *Undo* («Отменить»), *Redo* («Повторить»), *Find and Replace* («Найти и заменить») и т.д.

1.1.5 Режим программирования — Перекрестный вызов

Управляющие программы состоят из программ с логической структурой. В рамках одной программы применим только один язык программирования. Все программы объединяются для создания целостного проекта управления процессами. В программах могут вызываться подпрограммы, написанные на различных языках программирования МЭК (LD, FBD, IL, ST) и SCC.

1.1.6 Вычислительные возможности — широкий спектр управляющих вычислительных функций

В программное обеспечение «СКПро» встроен ряд стандартных операторов, блоков управления функциями и стандартных функций. Среди этих функциональных блоков есть такие, как импульсный выход, коммутаторы master/slave, сетевые функции и функции последовательных коммуникаций и т.п., что позволяет инженерам с легкостью решать комплексные задачи по управлению процессом и сокращает время цикла разработки проекта.

1.1.7 Функция отображения — интуитивно понятный онлайн-просмотр.

Система позволяет в режиме онлайн отслеживать рабочий статус схем на языке LD. Красный сигнал означает, что схема замкнута, зеленый – разомкнута. Всё интуитивно понятно с первого взгляда.

Для языка SCC отслеживается не только рабочий статус, но и шаги выполнения. Так же можно внедрять установку и сброс таймера, сигнал тревоги и т.п., что позволяет специалистам-инженерам с легкостью использовать различные функции.

1.1.8 Функция модифицирования — изменения в онлайн-режиме

Функция позволяет модифицировать параметры функциональных блоков, добавлять, удалять и перемещать функциональные блоки в режиме онлайн. Также возможно вносить модификации в программу, исполняемую в ПЛК, не останавливая его работу.

1.1.9 Функция отладки — мощный онлайн-отладчик

Все языки — список инструкций (IL), релейно-контактные схемы (LD), функциональные блоковые диаграммы (FBD) и структурированный текст (ST), поддерживают функции отладки в режиме онлайн, такие как точка останова, пошаговое выполнение, и т.п.

Для языка SCC предусмотрено три режима: автоматическое выполнение, наблюдаемое выполнение и отладочное выполнение. Когда программа на языке SCC находится в режиме онлайн-отладки, выполнение каждой функционального бокса обозначается тремя различными цветами: серым для не выполненного, красным для выполняемого и синим для выполненного. Поддерживаются такие функции, как точка останова и пошаговое выполнение, а также завершение выполнения или перезапуск выполнения в любое время. Инженерный персонал может легко отлаживать программы и находить ошибки.

1.1.10 Функция мониторинга — онлайн-мониторинг в реальном времени

В режиме онлайн все точки данных могут управляться (принудительно устанавливаться, назначаться, наблюдаться), все значения переменных могут отслеживаться в таблице переменных; все события SOE могут просматриваться в таблице событий SOE; все сигналы тревоги могут запрашиваться в таблице сигналов тревоги; все сообщения об ошибках можно проверить на вкладке debug (отладка) в

окне вывода. Данные могут отображаться в трех форматах: десятичном, двоичном или шестнадцатеричном.

1.1.11 Симулятор – моделирование без реального аппаратного обеспечения

Программы могут быть разработаны и отлажены в симуляторе без необходимости использовать аппаратный ПЛК. «СКПро» может идеально моделировать аппаратные функции, точно воспроизводить поведение целевой программы и эффективно сокращать период разработки программы.

1.1.12 Средство диагностики

ПО «СКПро» обладает комплексными функциями диагностики приложений. В окне вывода могут отображаться все сбои системы и приложений. Двойное нажатие кнопки мыши в этом окне открывает доступ к редактору, что позволяет исправить недочёты программы.

1.1.13 Режим передачи

Все операции сохранения, загрузки и выгрузки результатов программирования применяются в файловом режиме, таким образом, они могут поддерживать все конфигурации программы в соответствии.

1.1.14 Языковая поддержка

Для ввода имен переменных необходимо использовать стандартную латиницу. Для комментариев можно использовать языки, поддерживаемые операционной системой компьютера, на котором установлено ПО «СКПро».

1.1.15 Визуальный режим печати

Для задач документирования проекта «СКПро» поддерживает визуальный режим печати для распечатки конфигурации ПЛК, информации о точках данных, и программ на всех языках (LD, FBD, ST, IL и SCC) в том же виде, в каком пользователь видит их на экране.

1.1.16 Дружественный пользовательский интерфейс

«СКПро» в полной мере использует преимущества графических и контекстно-зависимых интерфейсов Windows. Эргономика улучшена за счет оптимизации использования экранного пространства, прямого доступа к инструментам и информации, а также использования комментариев на различных языках и т.д.

1.2 Системные требования

1.2.1 Требования к аппаратному обеспечению

32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше.

2 ГБ (для 32-разрядного процессора) или 4 ГБ (для 64-разрядного процессора) ОЗУ.

2 ГБ свободного места на жестком диске.

Графическое устройство с поддержкой разрешения SXGA (1280*1024).

Сетевая карта Ethernet 100 Мбит/с.

1.2.2 Требования к операционным системам

Поддерживается работа с операционными системами семейства Microsoft Windows, начиная с Windows 7.

1.3 Установка программного обеспечения

Запустите программу установки “setup.exe”, и следуйте инструкциям, чтобы завершить установку.

2 Работа с программным обеспечением «СКПро»

ПО «СКПро» включает в себя законченную систему для конфигурирования ПЛК и разработки прикладных приложений, которая соответствует принципам работы в системе Windows и которая проста в использовании.

2.1 Рабочее окно

2.1.1 Рабочий интерфейс

Интерфейс программного обеспечения «СКПро» приведен на рисунке 2.1. Среда разработки включает в себя следующие элементы: меню, панель инструментов, браузер проекта, окно вывода, строку состояния и окно программы, положение каждого элемента показано на рисунке 2.1.

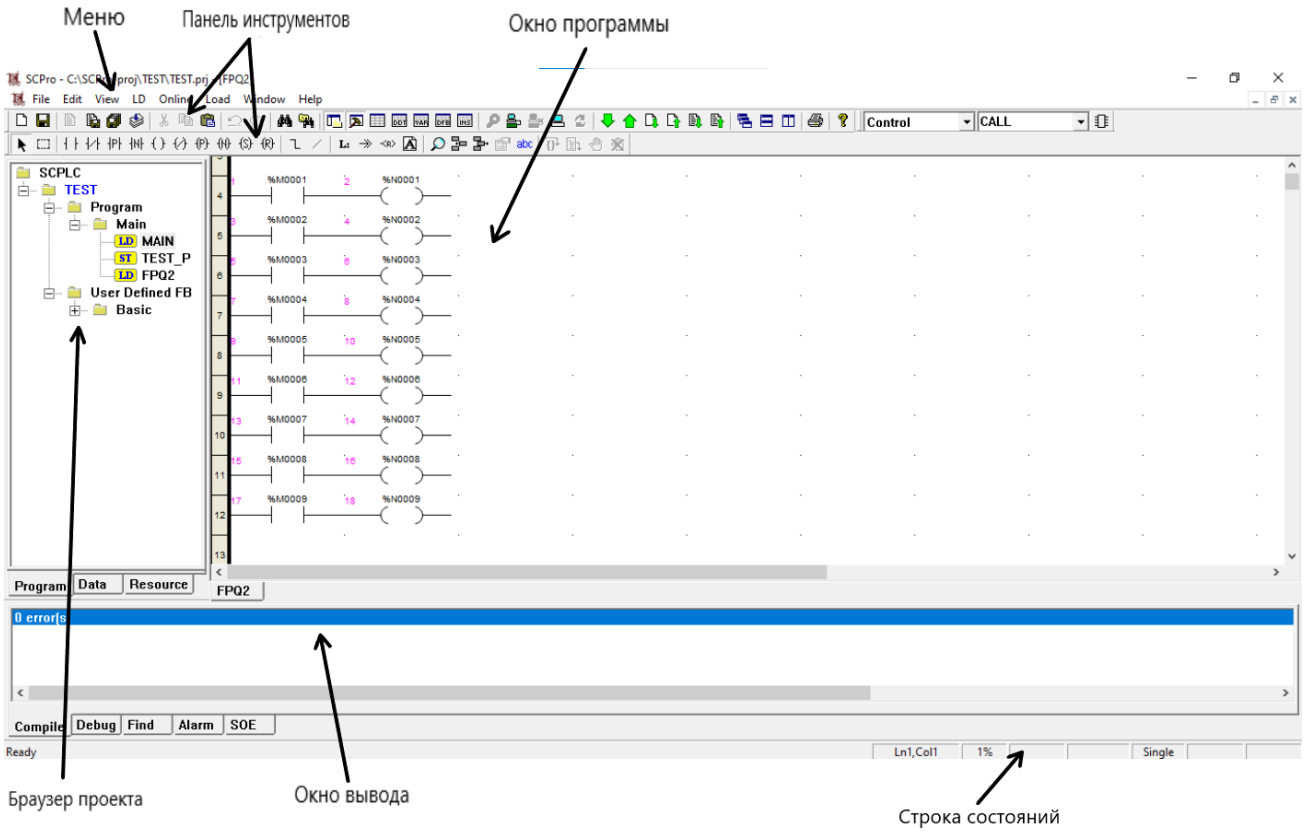


Рисунок 2.1 Рабочий интерфейс

2.1.2 Основной функционал каждого окна

Меню: Выполнение основных функций «СКПро».

Панель инструментов: Команды работы с файлами: "создать", "открыть" и "сохранить"; онлайн-операции: вход в систему, загрузка и выгрузка; а также вызов панели инструментов каждого языка программирования.

Строка состояния: Строка состояния находится в нижней части экрана. Справа от строки состояния находится информация о состоянии, такая как координаты

программы, онлайн / оффлайн, эмуляция, принудительная метка, процент свободной памяти и т.д. Содержимое каждой операции отображается слева от строки состояния.

Окно программы: Настройка системы, редактирование программы, отладка программы и т.д.

Браузер проекта: обеспечивает управление проектом.

Окно вывода: Отображение результатов операций поиска, компиляции и отладки программы.

2.2 Введение в структуру меню

2.2.1 Главное меню и выпадающее меню

Главное меню позволяет реализовывать основные функции ПО «СКПро», такие как **File** (Работа с файлами), **Edit** (Редактирование), **View** (Просмотр), **LD** (**FBD**, **SCC**, **IL**, **ST**, и т.д.), **Online** (Онлайн), **Load** (Загрузка), **Window** (Окно), и **Help** (Справка), а также другие, как показано на рисунке 2.2.

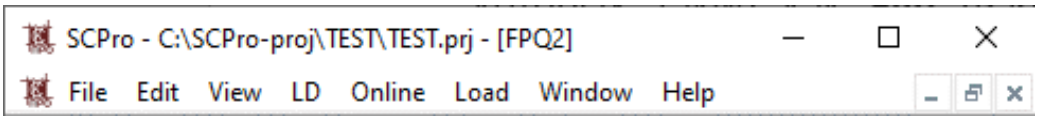


Рисунок 2.2 Главное меню

Чтобы вызвать выпадающее меню, щелкните левой кнопкой мыши по любому пункту главного меню, фон пункта меню визуально изменится, одновременно отобразится выпадающее меню. Наведите мышь на любой пункт выпадающего меню, пункт станет синим, это означает, что при нажатии кнопки мыши, выбрана будет эта операция; щелкните любое место любой клавишей мыши за пределами меню или нажмите клавишу **ESC**, чтобы отключить меню, как показано на рисунке 2.3:

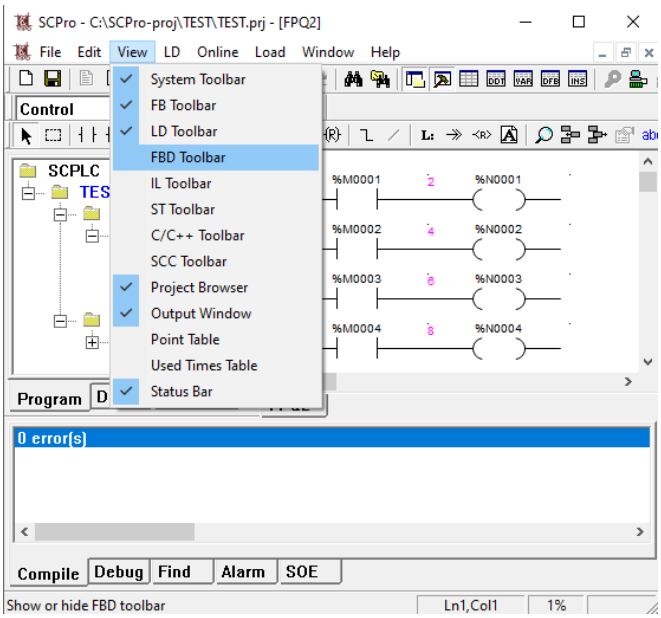


Рисунок 2.3 Выпадающее меню

2.2.2 Подпункты меню

Каждый подпункт меню отображается в выпадающем меню. Наведите курсор мыши на любой подпункт меню, он станет синим, это означает, что при нажатии левой кнопки мыши, выбрана будет эта операция; щелкните любое место за пределами меню или нажмите клавишу **ESC**, чтобы отключить меню, как показано на рисунке 2.4.

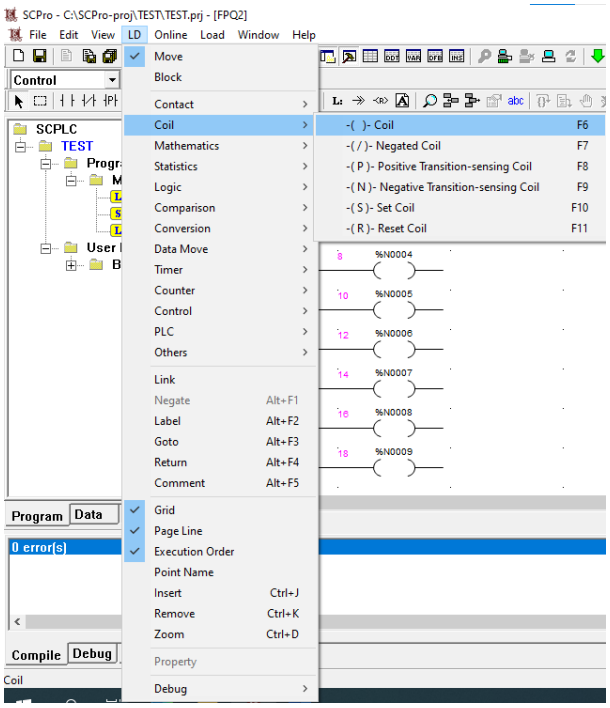


Рисунок 2.4 Подпункты меню

2.2.3 Контекстное или всплывающее меню

Щелкните объект правой кнопкой мыши, чтобы открыть контекстное меню; при выборе нескольких объектов также можно вызвать контекстное меню, но в этом случае меню будет содержать только те пункты, которые подходят для каждого из

объектов. Щелкните в любом месте за пределами меню или нажмите клавишу ESC, чтобы отключить меню, как показано на рисунке 2.5.

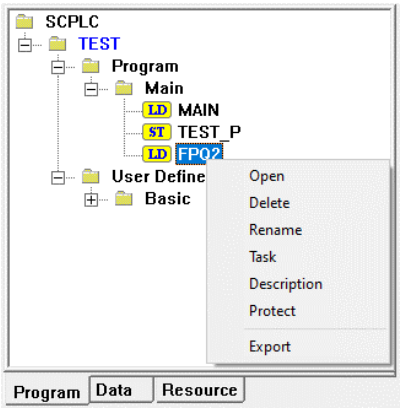


Рисунок 2.5 Контекстное или всплывающее меню

2.3 Функционал меню

2.3.1 Меню File

2.3.1.1 Команды меню File

Меню **File** используется для управления файлами, включая следующие **New** (Новый), **Open** (Открыть), **Save** (Сохранить), **User Management** (Управление пользователями), **Close** (Закрыть), **New Project** (Новый проект), **Save Project** (Сохранить проект), **New Program** (Новая программа), **Save Program** (Сохранить программу), **Save All Programs** (Сохранить все программы), **Compile Program** (Компилировать программу), **Compile All Programs** (Компилировать все программы), **Password** (Пароль), **Used Times** (Количество обращений), **Export Point** (Экспорт точек данных), **Import Point** (Импорт точек данных), **Print** (Печать), **Print Preview** (Предварительный просмотр), **Print Setup** (Настройка), и **Exit** (Выход), как показано на рисунке 2.6.

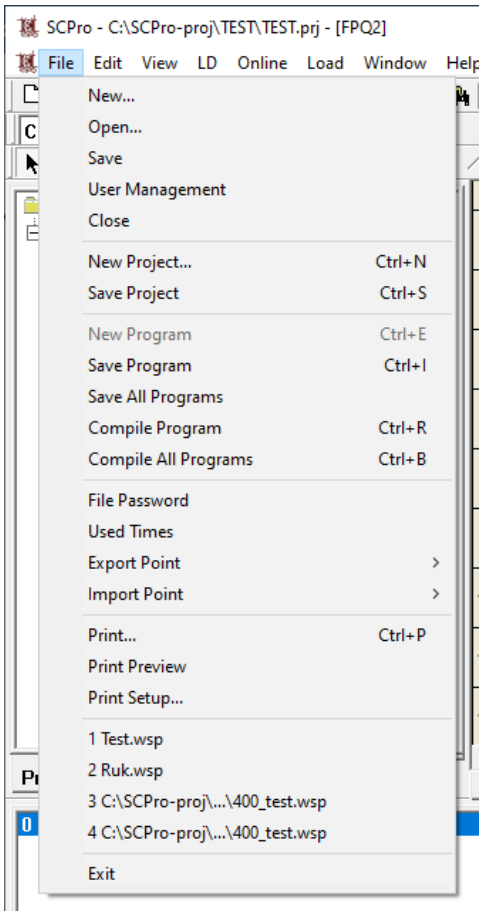


Рисунок 2.6 Меню File

2.3.1.2 Функционал меню File

【**New**】: Создать новый файл проекта, который будет содержать базу данных, программы LD, FBD, SCC, IL и ST и т.д.

【**Open**】: Открыть существующий файл проекта. Выберите **Open**, и программа **откроет** диалоговое окно "Открыть", тип файла - “SCPro Workspace Files“, расширение - “.wsp”. Если файл защищен паролем, появится диалоговое окно ввода **пароля**, как показано на рисунке 2.7.

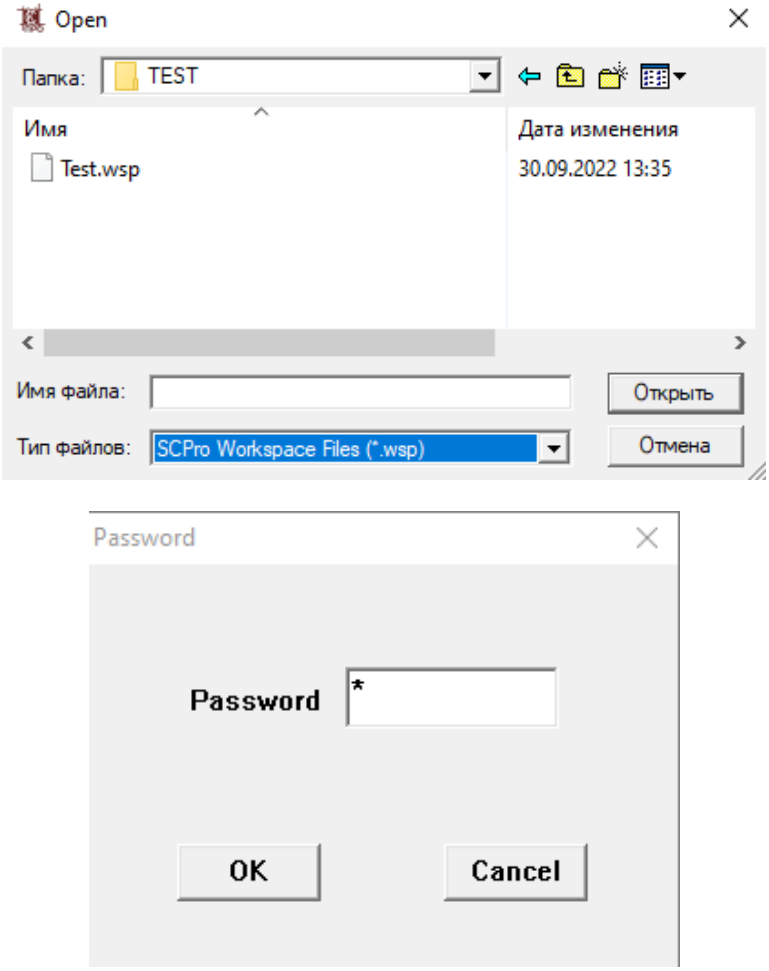


Рисунок 2.7 Открыть проект

【Save】: Сохранить редактируемый файл проекта. Если редактируемый файл не является новым, то исходный файл будет перезаписан, как показано на рис.2.8:

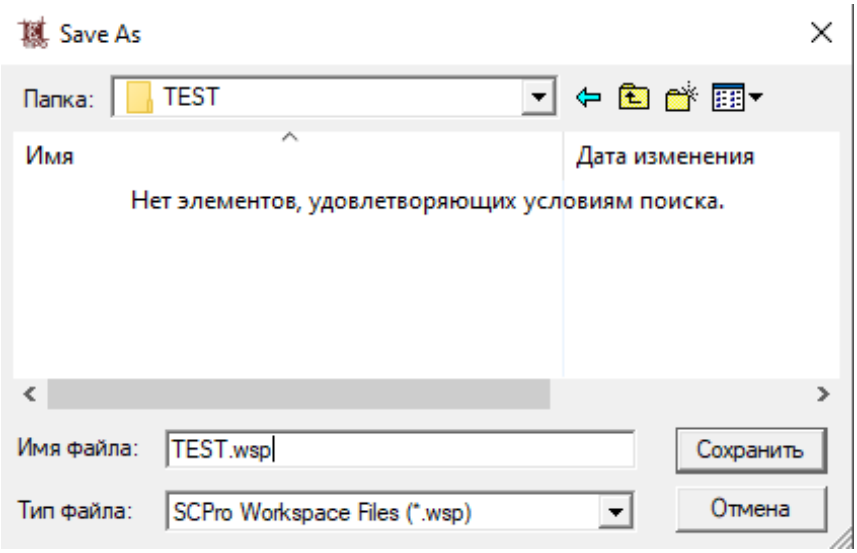


Рисунок 2.8 Сохранить проект

※ Изменение одного файла проекта повлияет на каждую программу в нем, потому обязательно запускайте компиляцию всех программ через команду *Compile All Programs* перед загрузкой.

【**New Program**】: Создать новую программу. Программа может быть написана на языке LD, FBD, SCC, IL или ST и т.д. Новой программе необходимо дать название, также можно добавить описание, содержимое которого будет отображаться в области редактирования программы, как показано на рисунке 2.9.

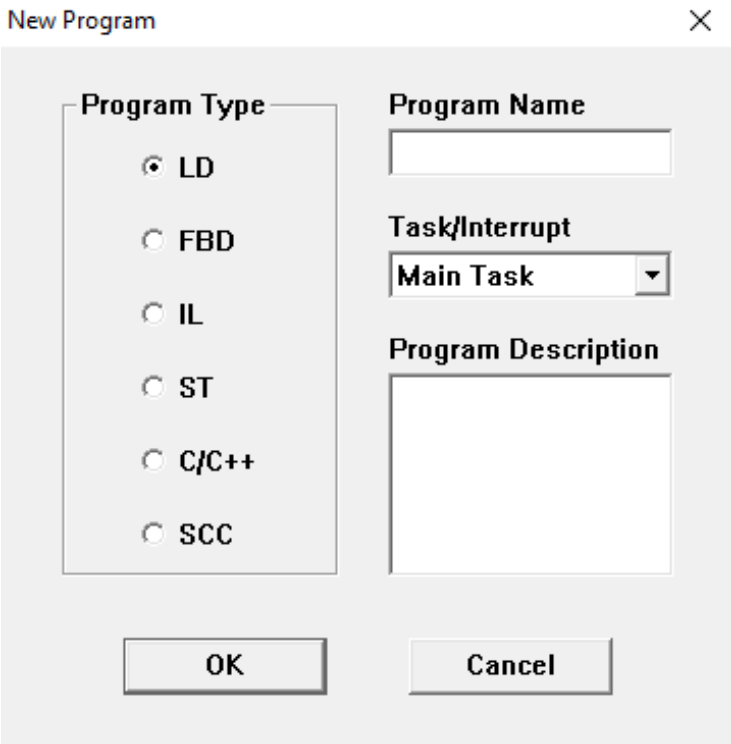


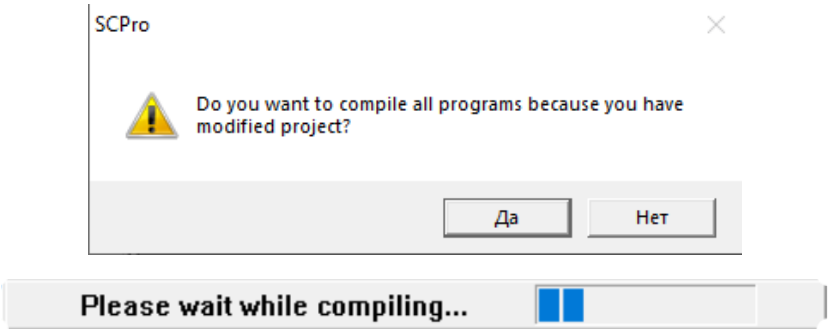
Рисунок 2.9 Новая программа

Программа может быть основной задачей, нумерованной задачей, либо прерыванием. Более подробно - см. раздел 3.3 настоящего руководства.

【**Save Program**】: Сохранить редактируемую программу. При выборе команды **Сохранить программу**, «СКПро» автоматически сохраняет программу, переписывая существующий файл или создавая новый.

【**Save All Programs**】: Сохранить все открытые программы.

【**Compile**】: Скомпилировать текущую программу. Если программа изменена, «СКПро» предложит сохранить её. После этого «СКПро» автоматически проверит текущую программу на предмет ошибок, при наличии таких ошибок, она не сможет быть скомпилирована; вкладка **Compile** окна вывода демонстрирует место, тип и количество ошибок, как показано на рисунке 2.10:



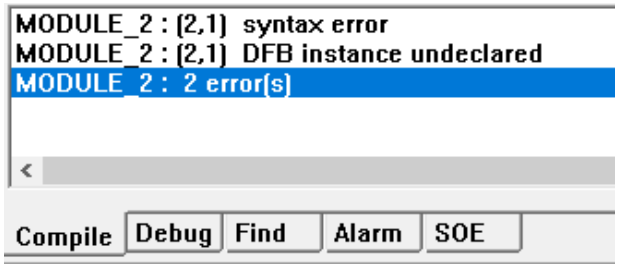


Рисунок 2.10 Скомпилировать программу

【Compile All Programs】: Скомпилировать все программы текущего проекта. Команда выполняет всё то же самое, что и команда **Compile Program**, «СКПро» предложит сохранить программу (если были внесены изменения). После этого «СКПро» автоматически проверит все программы на предмет ошибок, при наличии таких ошибок, они не смогут быть скомпилированы; также в отдельной вкладке отображаются место, тип и количество ошибок.

【Пароль】: Изменение пароля к файлу и пароля для входа в систему. Новым проектам не назначаются пароли к файлам, как показано на рис.2.11:

Рисунок 2.11 Пароль

【Used Times】: Статистический расчет применения каждой точки данных в программах показан на рисунке 2.12:

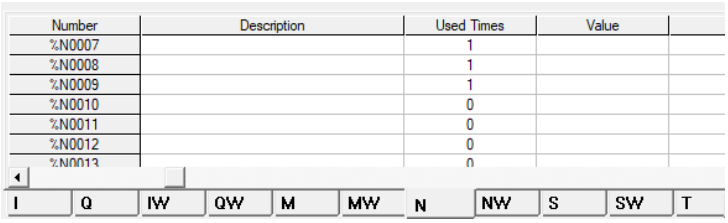


Рисунок 2.12 Количество применений

【Print】: Распечатать конфигурацию ПЛК, информацию о точках, LD, FBD, SCC, IL и ST. При выборе команды Print откроется стандартное диалоговое окно печати, в котором можно выбрать принтер, диапазон печати, копии и т.д. Как показано на рис.2.13:

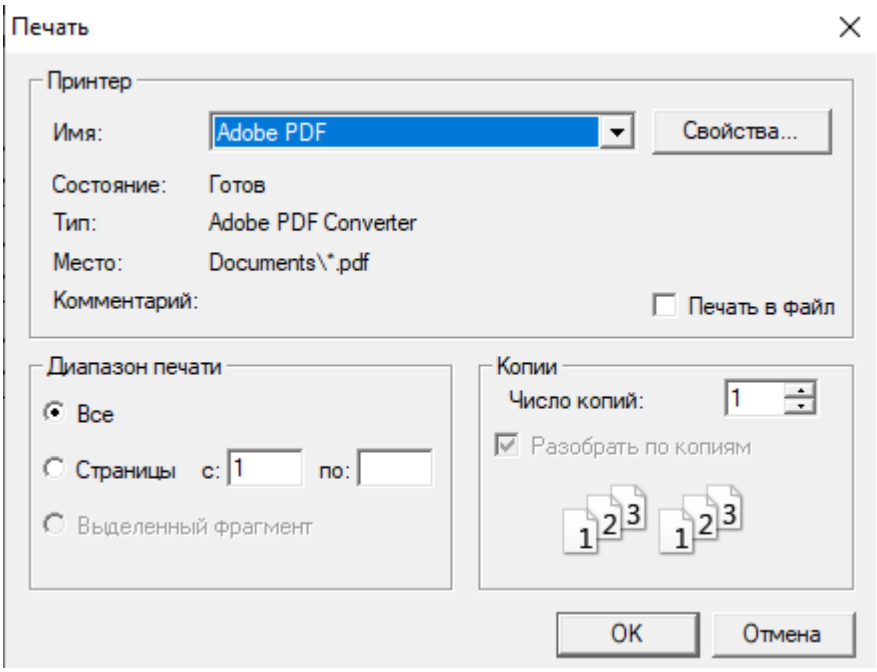


Рисунок 2.13 Печать

【**Print Preview**】: Предварительный просмотр результатов печати конфигурации ПЛК, сообщения о точках данных, LD, FBD, SCC, IL и ST.

【**Print Setup**】: Сбросить настройки печати. При выборе команды **Print Setup** откроется диалоговое окно **Print Setup**. Чтобы наилучшим образом отобразить конфигурацию на печати, для LD и FBD обычно устанавливается альбомная ориентация; для других программ ориентация – портретная. Как показано на рисунке 2.14:

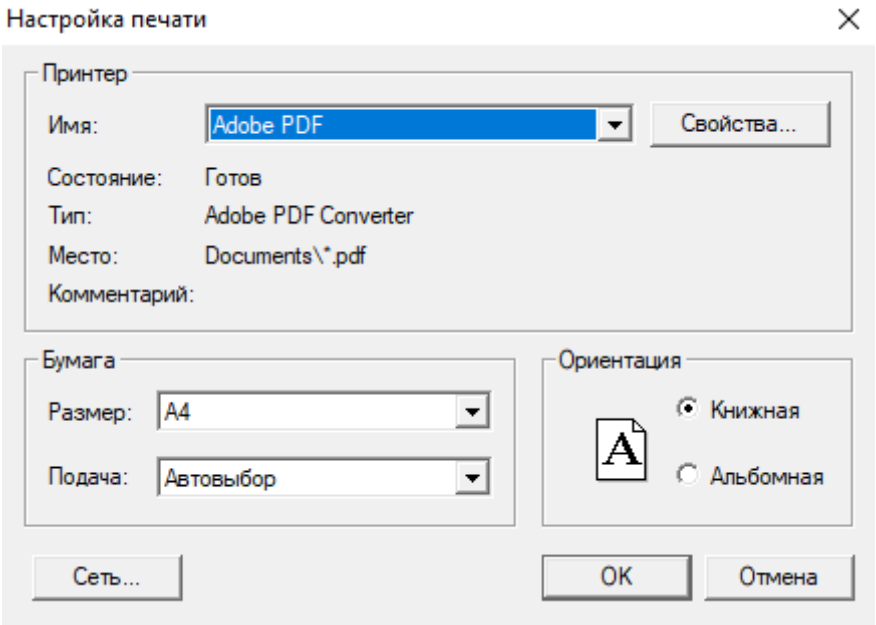


Рисунок 2.14 Настройка печати

【**Exit**】: Выйти из программы «СКПро».

Кроме того, в меню **File** отображаются несколько недавно открытых файлов проекта. Файлы могут быть открыты щелчком левой кнопки мыши. Как показано на рисунке 2.15:

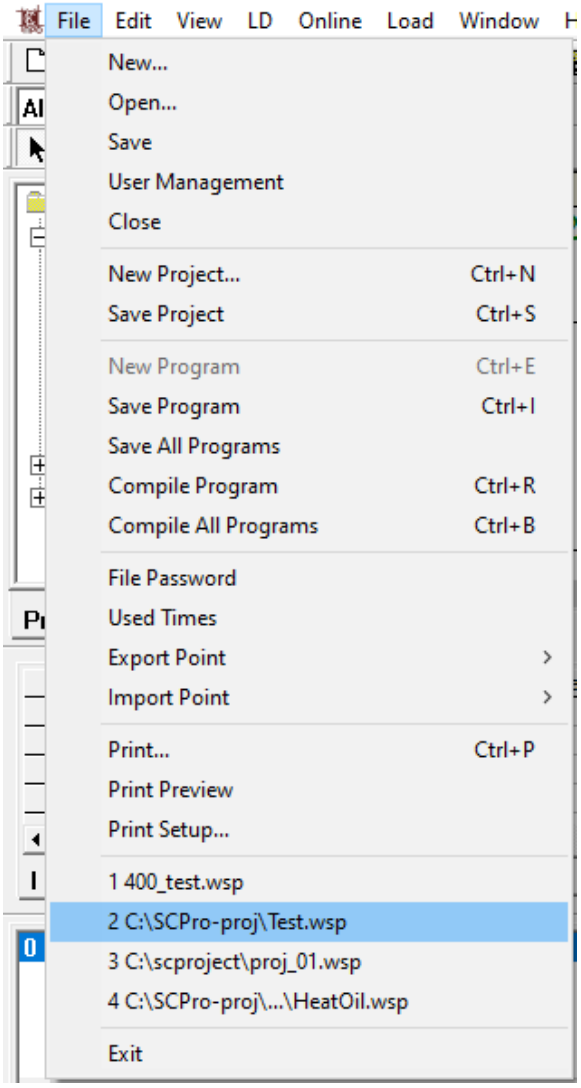


Рисунок 2.15 Открыть последние открытые файлы

2.3.2 Меню Edit

2.3.2.1 Команды меню Edit

В состав меню ***Edit*** входят команды, часто применяемые при редактировании программы, такие как ***Undo***, ***Redo***, ***Cut***, ***Copy***, ***Paste***, ***Delete***, ***Select All***, ***Find***, ***Replace*** и ***Global Find*** и т.п., как показано на рисунке 2.16:

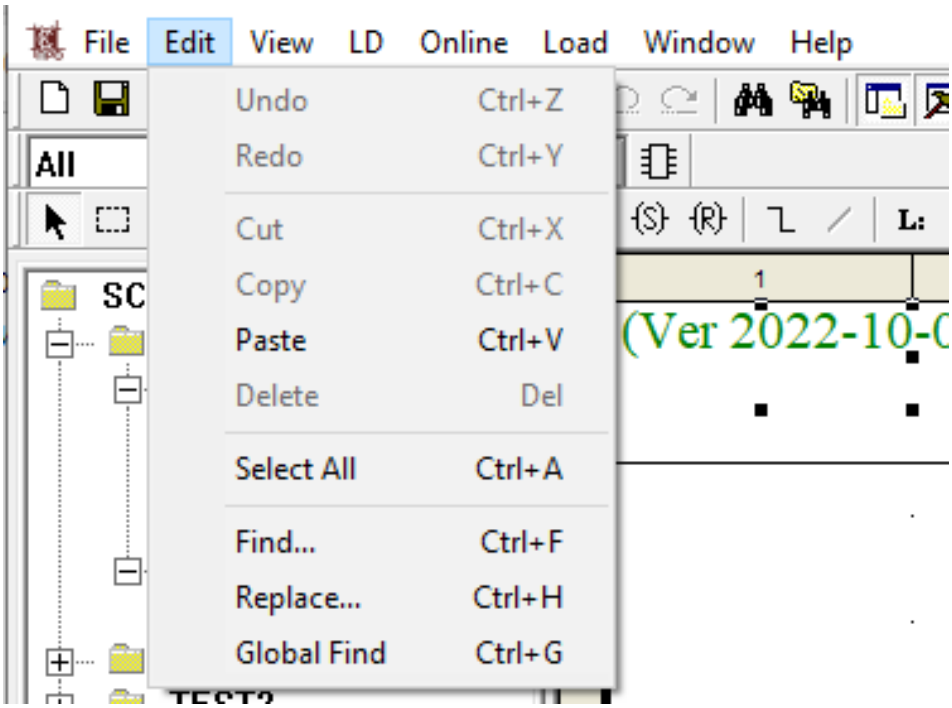


Рисунок 2.16 Меню Edit

2.3.2.2 Функционал меню Edit («Редактирование»)

【Undo】: Отменить последнюю операцию. При помощи команды ***Undo*** можно отменить такие операции, как разместить / удалить функцию блок или функциональное поле, вставить / вырезать раздел программы, переместить функцию или функциональное поле и т.д. Изменение параметров функции или функционального поля не воспринимается программой в качестве операции, поэтому не может быть отменено.

【Redo】: Повторить последнюю операцию. Команда ***Redo*** используется только в отношении операции, которая была отменена командой ***Undo***, любую другую операцию эта команда не отменяет.

【Cut】: Удалить выбранное содержимое и поместить его в буфер обмена, содержимое из буфера обмена можно затем вставить в другой участок. Вырезать командой ***Cut*** можно такое содержимое, как функцию (контакт, катушка и специальный функциональный блок) в LD и FBD, функциональное поле в SCC, инструкцию в IL или оператор в ST, а также содержимое области, выбранное с помощью операции блока.

【Copy】: Скопировать выбранное содержимое в буфер обмена, но не удалять его.

【Paste】: Вставить содержимое из буфера обмена в месте, где находится курсор.

【Delete】: Удалить выбранное содержимое, не перемещая его в буфер обмена.

Примечание: Операции ***Cut***, ***Copy***, ***Paste*** и ***Delete*** также можно выполнить, щелкнув по области редактирования правой кнопкой мыши. После выбора функции или функционального поля, могут быть выполнены такие операции, как ***Cut***, ***Copy*** и ***Delete***, тогда, как операция ***Paste*** может быть выполнена на пустом месте. Примеры приведены на рисунке 2.17:

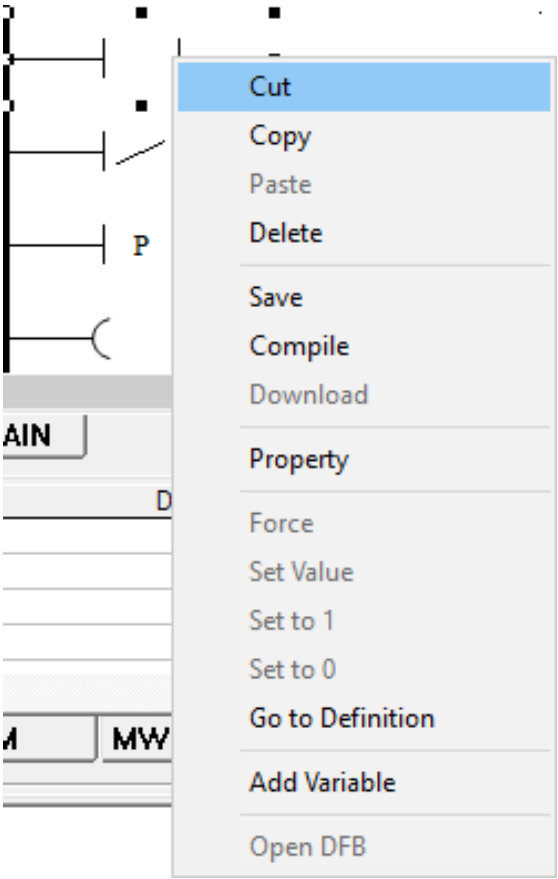


Рисунок 2.17 Операции вырезания (Cut), копирования (Copy), вставки (Paste), и удаления (Delete)

Операции вырезания (***Cut***), копирования (***Copy***), вставки (***Paste***), и удаления (***Delete***)

【Select All】: Выбрать все содержимое в текущей области редактирования. Как показано на рисунке 2.18:

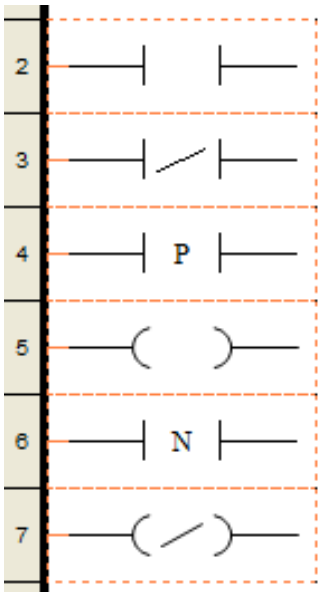


Рисунок 2.18 Операция Select all

【Find】: Найти подходящие функцию, функциональное поле, инструкцию или оператор. Операция поиска ***Find*** может быть выполнена только в текущей рабочей области.

Если текущая рабочая область – LD (или FBD), то поиск будет осуществлен только в текущем LD (или FBD). Например, если вы хотите найти функциональный блок с параметром “%M0036”, выберите **【Edit】 /【Find】**, после чего откроется диалоговое окно поиска ***Find***. Введите параметр “%M0036” в строке “*Find What*” и “All” в строке “*Find Limit*”, затем нажмите кнопку ***Find Next***. Программа автоматически перейдет к первой подходящей функции и отобразит её в виртуальном окне. Чтобы найти другие подходящие функции, необходимо несколько раз нажать кнопку “***Find Next***”, затем перейти к следующей в очереди функции с "%M0036", пока не будет найдено все необходимое. По завершении, «СКПро» откроет окно завершения *Finished*. Функция поиска может быть выполнена для всех функций, а также для функций указанного типа. Например, при установке значения “Нормально разомкнутый контакт” в параметре “*Find Limit*”, поиск функций будет осуществляться только среди нормально разомкнутых контактов. Это показано на рисунке 2.19:

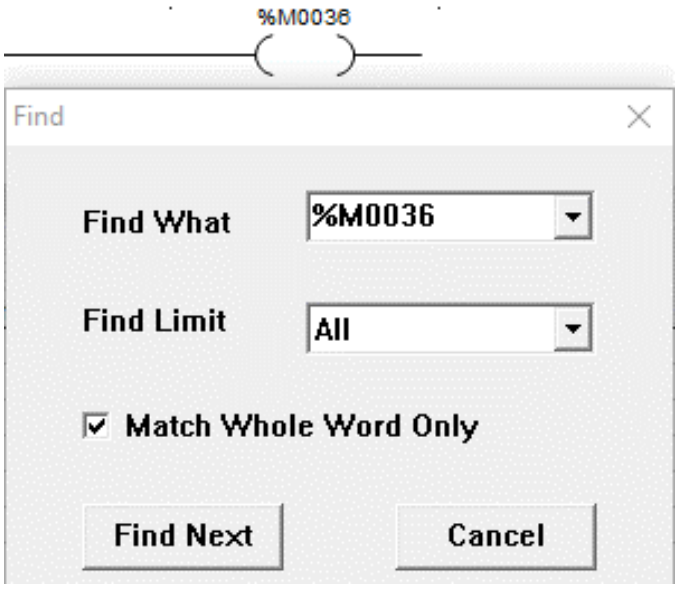


Рисунок 2.19 Операция "Найти" в LD

После того, как найдено будет всё, «СКПро» откроет окно “Finished”, как показано на рисунке 2.20:

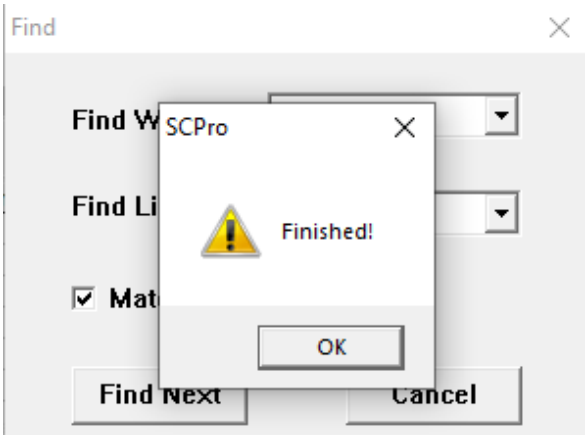


Рисунок 2.20 Поиск окончен

Если текущая рабочая область – это SCC, то поиск будет осуществляться в текущем SCC. Функция поиска для SCC осуществляет поиск не только в операторах выполнения функциональных полей SCC, но и в описаниях. Так же, как и в LD, функция поиска может выполняться во всех типах функциональных полей, а также в функциональных полях указанного типа. Например, если изменить ограничение поиска “*Find Limit*” на “*Execution Box*”, то поиск подходящих функциональных полей будет осуществляться только в полях выполнения. Это показано на рисунке 2.21:

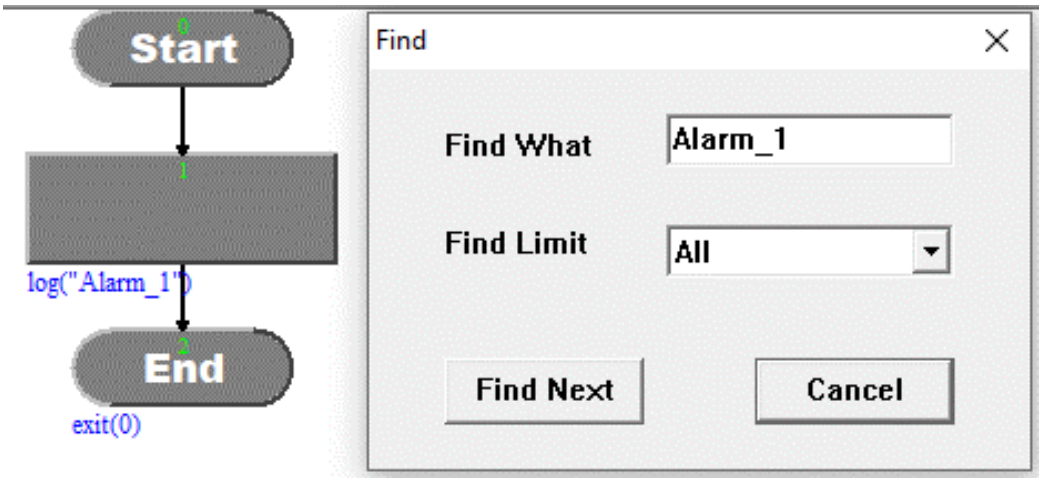


Рисунок 2.21 Операция поиска в SCC

Если текущая рабочая область – IL, то поиск будет осуществляться в текущей программе IL. В диалоговом окне поиска IL (см. рисунок 2.22) есть такие опции, как “Match Whole Word Only” («Найти только полное слово») и “Match Case” («С учетом регистра»). Например, если необходимо найти “R1”, но ввести “R1” в поле “Поиск” и не выбрать “С учетом регистра”, то поиск выдаст значения как “R1”, так и “r1”.

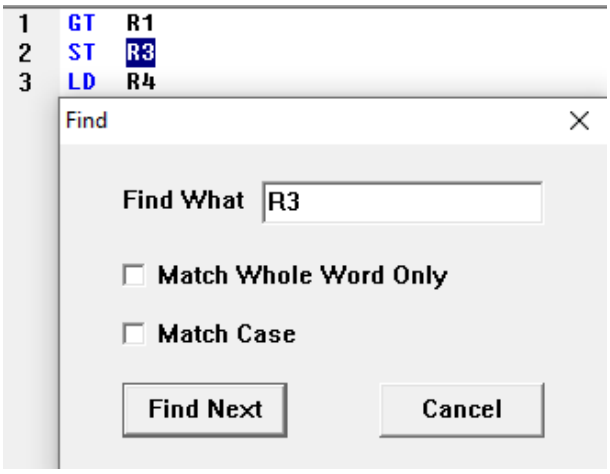


Рисунок 2.22 Операция поиска для IL

Если текущая рабочая область – ST, то поиск будет осуществляться в текущей программе ST. В диалоговом окне поиска ST (см. рисунок 2.23) есть такие опции, как “Match Whole Word Only” («Найти только полное слово») и “Match Case” («С учетом регистра»). Например, если необходимо найти “R1”, но ввести “R1” в поле “Поиск” и не выбрать “С учетом регистра”, то поиск выдаст значения как “R1”, так и “r1”.

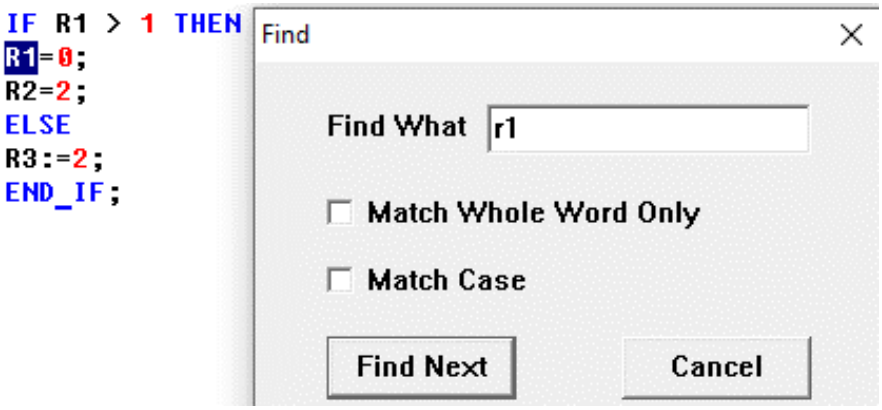


Рисунок 2.23 Операция поиска для ST

【**Replace**】: Найти подходящий блок функции, функциональное поле, инструкцию или оператор в текущей программе и заменить их. Операция ***Replace*** аналогична операции ***Find***, только имеет дополнительно функцию замены. Иными словами, операция ***Replace*** содержит в себе операцию ***Find***. В рамках операции ***Replace*** может быть выполнена либо выборочная (частичная) замена, либо полная замена. Замена может быть выполнена только в текущей рабочей области.

【**Global Find**】: Глобальный поиск подходящего блока функции, функционального поля, инструкции или оператора. Эта функция выполняется во всех программах (LD, FBD, SCC, IL и ST). После завершения глобального поиска результат отображается во вкладке "**Find**" окна вывода; также там отображаются место и номер результата, как показано на рисунке 2.24:

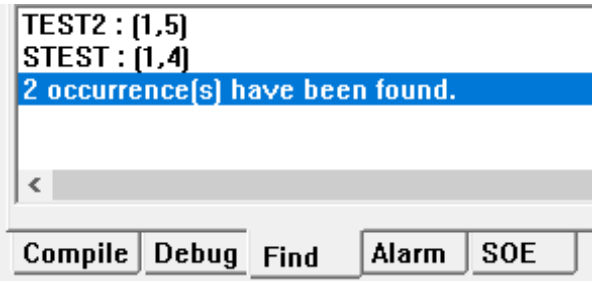


Рисунок 2.24 Глобальный поиск

2.3.3 Меню View

2.3.3.1 Команды меню View

В меню ***View*** («Вид») содержатся: ***System Toolbar*** («Системная панель инструментов»), ***FB Toolbar*** («Панель инструментов FB»), ***LD Toolbar*** («Панель инструментов LD»), ***FBD Toolbar*** («Панель инструментов FBD»), ***IL Toolbar*** («Панель инструментов IL»), ***ST Toolbar*** («Панель инструментов ST»), ***SCC Toolbar*** («Панель инструментов SCC»), ***Project Browser*** («Браузер проекта»), ***Output Window*** («Окно вывода»), ***Point Table*** (Таблица точек данных), и ***Status Bar*** («Строка состояния») и т.п., как показано на рисунке 2.25:

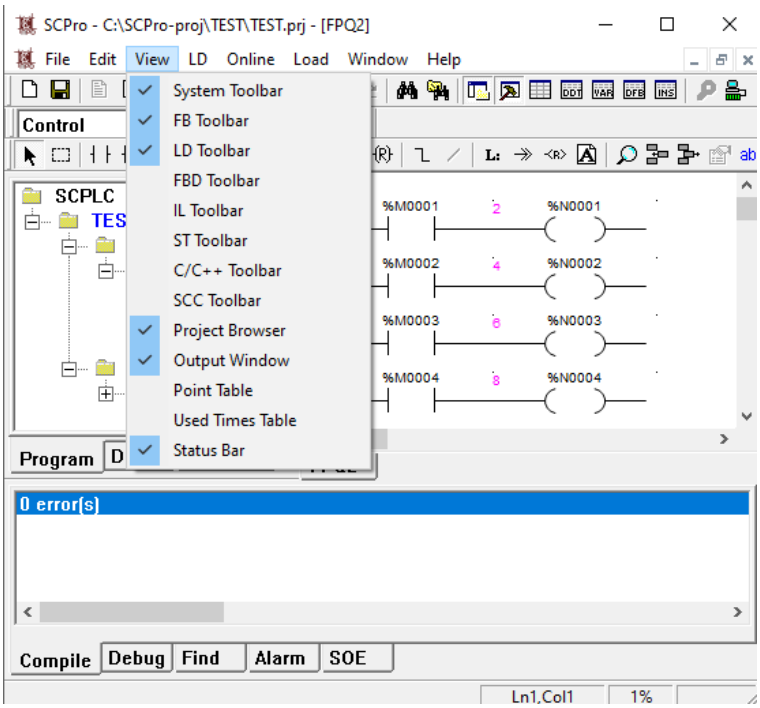


Рисунок 2.25 Меню View

2.3.3.2 Функционал меню View («Вид»)

【System Toolbar】: Показать или скрыть системную панель инструментов.

【FB Toolbar】: Показать или скрыть панель инструментов функций.

【LD Toolbar】: Показать или скрыть панель инструментов LD.

【FBD Toolbar】: Показать или скрыть панель инструментов FBD.

【IL Toolbar】: Показать или скрыть панель инструментов IL.

【ST Toolbar】: Показать или скрыть панель инструментов ST.

【SCC Toolbar】: Показать или скрыть панель инструментов SCC.

【Project Browser】: Показать или скрыть браузер проекта.

【Output Window】: Показать или скрыть окно вывода.

【Point Table】: Показать или скрыть таблицу точек данных.

【Used Time Table】: Показать таблицу применения точек данных.

【Строка состояния】: Показать или скрыть строку состояния.

2.3.4 Меню LD

2.3.4.1 Команды меню LD

В меню ***LD*** содержатся такие команды, как: ***Move*** («Переместить»), ***Block*** («Блокировка»), ***Contact*** («Контакт»), ***Coil*** («Катушка»), ***Mathematics*** («Математика»), ***Statistics*** («Статистика»), ***Logic*** («Логика»), ***Comparison*** («Сравнение»), ***Conversion*** («Преобразование»), ***Data Move*** («Перемещение данных»), ***Timer*** («Таймеры»), ***Counter***

(«Счетчики»), ***Control*** («Управление»), ***PLC*** («ПЛК»), ***Others*** («Прочие»), ***Link*** («Связать»), ***Negate*** («Присвоить отрицательное значение»), ***Label*** («Метка»), ***Goto*** («Перейти»), ***Return*** («Возврат»), ***Comment*** («Комментарий»), ***Grid*** («Сетка»), ***Page Line*** («Строка страницы»), ***Execution Order*** («Порядок исполнения»), ***Point Name*** («Название точки данных»), ***Insert*** («Вставка»), ***Remove*** («Удалить»), ***Zoom*** («Масштаб»), ***Property*** («Свойства») и ***Debug*** («Отладка»), и т.д., как показано на рисунке 2.26:

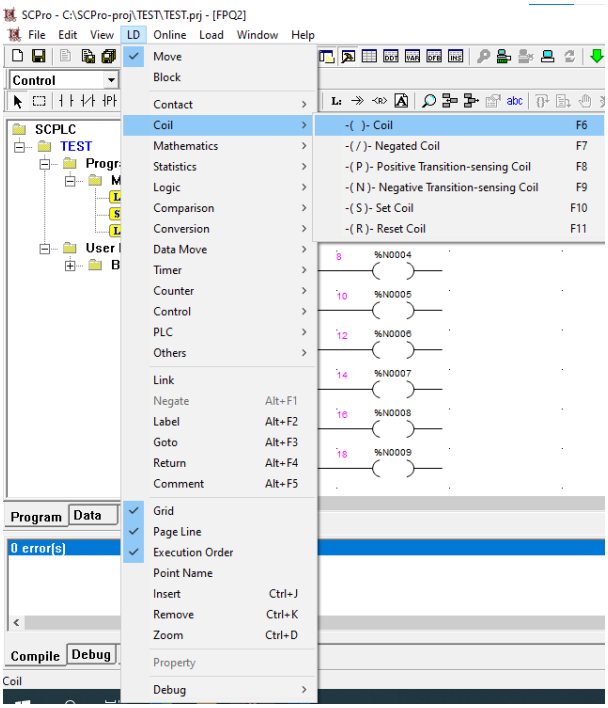


Рисунок 2.26 Меню LD

2.3.4.2 Функционал меню LD

【Move】: Если в меню ***LD*** не выбрана функция, то область редактирования по умолчанию находится в состоянии перемещения. Выбранную в данный момент позицию функции можно изменить движением мыши. Для перемещения функции, наведите на неё курсор мыши и, удерживая левую кнопку, переместите её в указанное место, затем отпустите кнопку.

【Block】: Операция ***Block*** используется для выбора всех функций и связующих элементов в одной области. Используйте мышь, чтобы выделить нужную область, все элементы в этой области будут выбраны. Операции перемещения, вырезания, копирования, удаления и другие для нескольких элементов должны быть завершены операцией ***Block***.

【Link】: Операция ***Link*** используется для создания связи между двумя точками-коннекторами двух функций. Наведите курсор мыши на первую точку-коннектор, которую необходимо связать, и щелкните левой кнопкой, чтобы выбрать её; затем наведите курсор мыши на вторую точку-коннектор и щелкните левой кнопкой на ней. Таким образом, между двумя точками-коннекторами функций возникает связь. Если точки-коннекторы не соответствуют принципу связи, то появится диалоговое окно с сообщением об этом, как показано на рисунке 2.27:

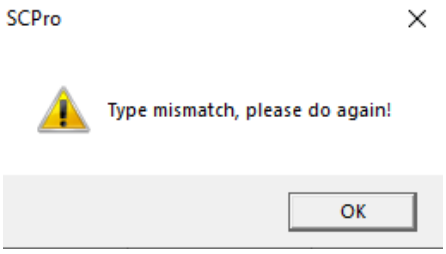


Рисунок 2.27 Ошибка связи

【**Negate**】: Когда выбранная функция является контактом, с помощью операции ***Negate*** можно циклически переключаться между нормально разомкнутым контактом, нормально замкнутым контактом, контактом с положительным переходом, контактом с отрицательным переходом.

Когда выбранная функция является катушкой, с помощью операции ***Negate*** можно циклически переключаться между катушкой, инверсной катушкой, катушкой положительного перехода, катушкой отрицательного перехода, катушкой установки и катушкой сброса.

【**Grid**】: Если для редактирования требуются визуальные направляющие, на фоне области редактирования LD может быть установлена точечная сетка.

【**Page Line**】: Для удобства навигации на фоне области редактирования LD могут отображаться строка и номер страницы.

【**Execution Order**】: Выделение порядка выполнения всех функций в состоянии сканирования, как показано на рис.Рисунок 2.28:

【**Point Name**】: Выделение названия точек данных всех функций. Каждой точке данных может быть присвоено имя в таблице точек; при отображении названия точки данных, в нижней части функции также будет отображаться присвоенное имя, как показано на рисунке 2.28:

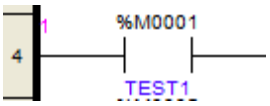


Рисунок 2.28 Порядок выполнения и название точки

【**Insert**】: Выберите место для вставки строки, щелкните левой кнопкой мыши в этом месте и выберите команду ***Insert***, чтобы добавить одну строку. В основном эта команда используется, когда во время редактирования программы LD необходимо добавить программный сегмент между существующими сегментами.

【**Remove**】: Выберите строку, которую необходимо удалить, щелкните левой кнопкой мыши в этом месте и выберите команду ***Remove***, чтобы удалить одну строку. В основном эта команда используется, когда во время редактирования программы LD необходимо удалить пустой сегмент программы, а на месте удаленной строки не должно быть никаких функциональных блоков или связок.

【**Zoom**】: Операция масштабирования Zoom может использоваться для изменения масштаба отображения области редактирования LD. После выбора команды ***Zoom***, «СКПро» откроет диалоговое окно **Zoom**. Можно ввести в поле ввода желаемое значение масштабирования, нажать кнопки вверх и вниз рядом с полем ввода или перетащить курсор для изменения масштаба; заданный масштаб всегда отображается в поле ввода, как показано на рисунке 2.29:

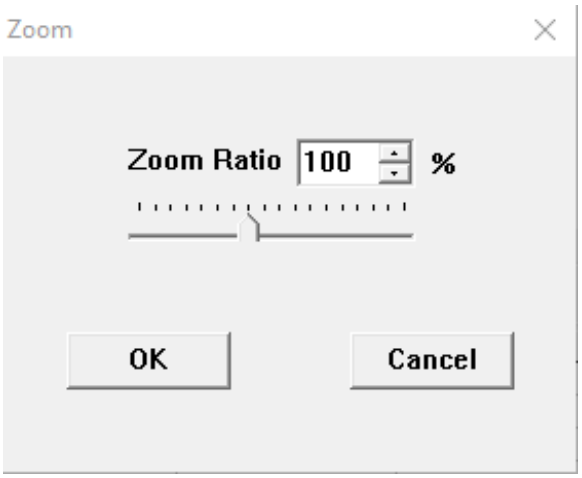


Рисунок 2.29 Масштабирование с командой Zoom

【**Property**】: Отобразить свойства выбранного объекта.

Для функции LD отображаются такие свойства, как тип, название программы, порядок выполнения, положение, определение параметров ввода/вывода и т.д. Тип функции отображается в верхней части диалогового окна. В примере далее контакт отображается следующим образом: название программы LD, в которой находится этот функция, “FPQ2”; Порядок выполнения 1-ый; Позиция находится в первом столбце и 4-ой строке; Параметр “Display EN/ENO” в редакторе FBD может использоваться в качестве альтернативы; Параметр “Input Number” («Номер ввода») может быть изменен в некоторых функциональных блоках LD и FBD; Параметр каждой точки-коннектора указан последовательно, как показано на рис.2.30:

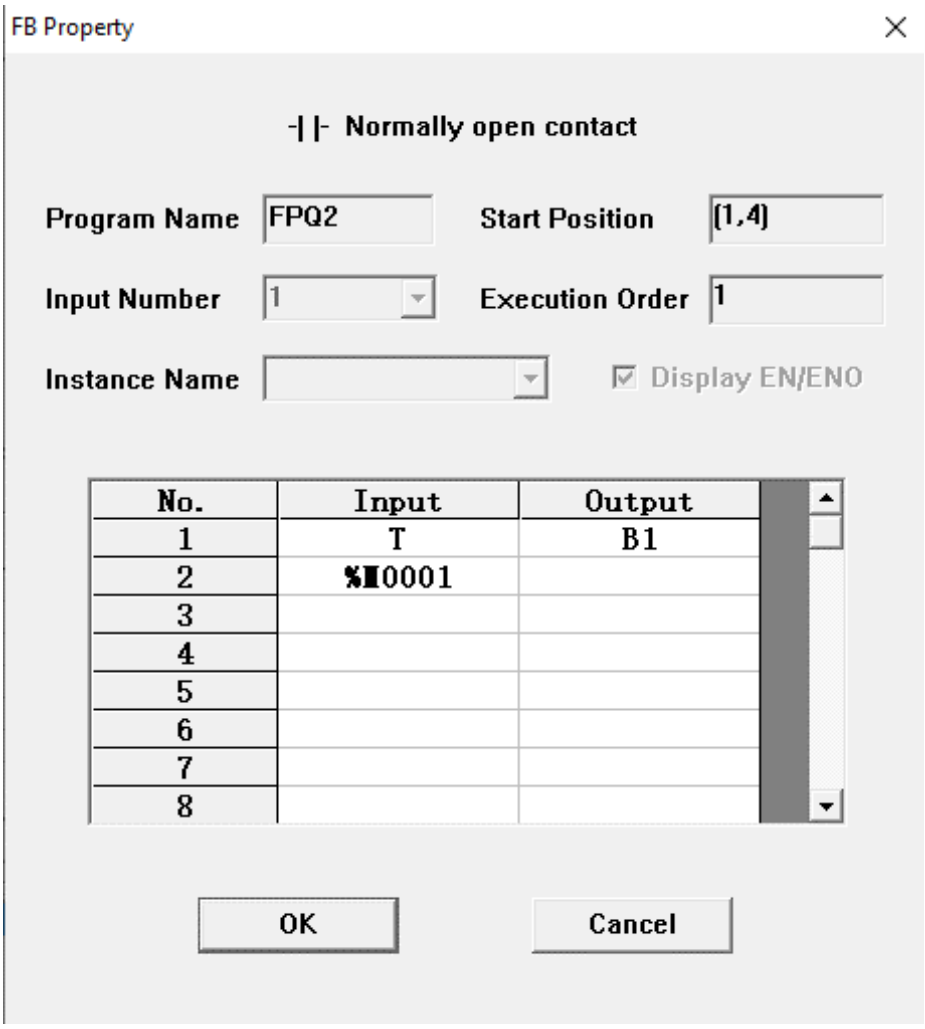


Рисунок 2.30 Свойство функционального блока

【Debug】: Отладка программы LD; включает команды ***Step*** («Шаг»), ***Continue*** («Продолжение»), ***Insert/Remove Breakpoint*** («Вставить/удалить точку останова») и ***Remove All Breakpoints*** («Удалить все точки останова»), и т.п.

2.3.5 SCC

2.3.5.1 Команды меню SCC

В меню ***SCC*** содержатся такие команды, как: ***Move*** («Переместить»), ***Block*** («Блок»), ***Link*** («Связать»), ***Start Box*** («Начальное поле»), ***End Box*** («Конечное поле»), ***Execution Box*** («Поле выполнения»), ***Condition Box*** («Поле условий»), ***Time-limited Condition Box*** («Поле условий с ограничением по времени»), ***Connector 1*** («Коннектор 1»), ***Connector 2*** («Коннектор 2»), ***Comment*** («Комментарий»), ***Page Line*** («Строка страницы»), ***Execution Statement*** («Исполнение оператора»), ***Property*** («Свойство»), ***Debug*** («Отладка») и т.д., как показано на рис.2.31:

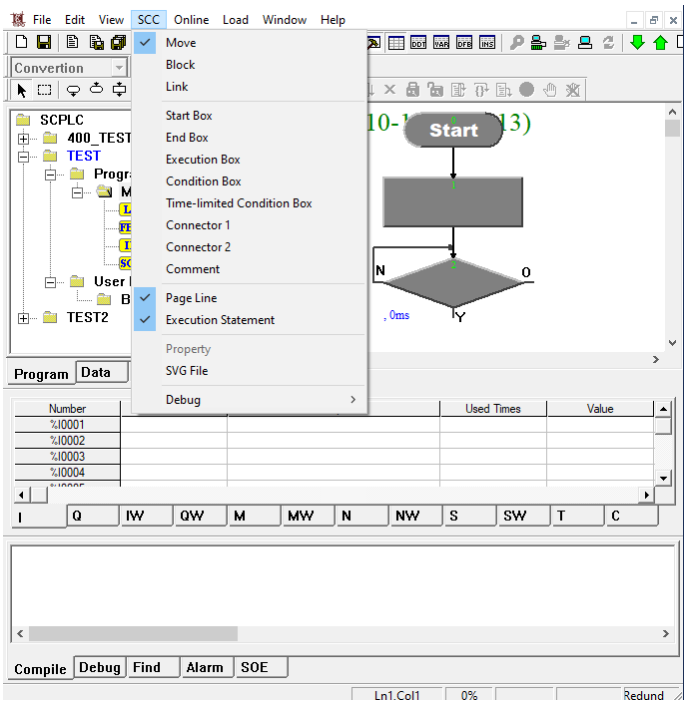


Рисунок 2.31 Меню SCC

2.3.5.2 Функционал меню SCC

【Move】: Если в меню ***SCC*** не выбрано функциональное поле, то область редактирования по умолчанию находится в состоянии перемещения. Выбранную в данный момент позицию функционального поля или связующего элемента можно изменить движением мыши. Для перемещения функционального поля или связи, наведите на неё курсор мыши и, удерживая левую кнопку, переместите её в указанное место, затем отпустите кнопку.

【Block】: Операция ***Block*** используется для выбора всех функциональных полей и связующих элементов в одной области. Используйте мышь, чтобы выделить нужную область, все элементы в этой области будут выбраны. Операции перемещения, вырезания, копирования, удаления и другие для нескольких элементов должны быть завершены операцией ***Block***.

【Link】: Операция ***Link*** используется для создания связи между двумя точками-коннекторами двух функциональных полей. Наведите курсор мыши на первую точку-коннектор, которую необходимо связать, и щелкните левой кнопкой, чтобы выбрать её; затем наведите курсор мыши на вторую точку-коннектор и щелкните левой кнопкой на ней. Таким образом, между двумя точками-коннекторами функций возникает связь. Если точки-коннекторы не соответствуют принципу связи, то появится диалоговое окно с сообщением об этом, как показано на рис.2.32.

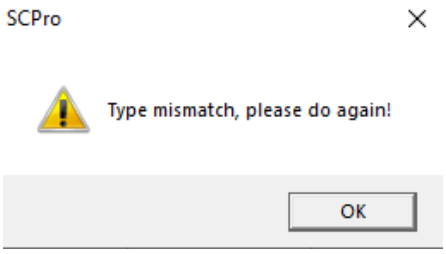


Рисунок 2.32 Ошибка связи SCC

【**Page Line**】: Для удобства навигации на фоне области редактирования SCC могут отображаться строка и номер страницы.

【**Execution Statement**】: Показать или скрыть исполнение оператора SCC. Если вам необходимо ознакомиться лишь с процессом выполнения SCC, отображать исполнение оператора не обязательно, достаточно лишь прочитать описание функционального поля SCC. Если необходимо узнать подробности выполнения определенного шага последовательного управления, необходимо отобразить выполнение оператора. Символ √ перед данным элементом указывает на то, что исполнение оператора находится в состоянии "показать", в противном случае - в состоянии "скрыть". При отображении исполнения оператора, в каждом функциональном поле отображается серийный номер, который указывает порядок размещения функционального поля вместо порядка выполнения, как в LD (FBD). Пример показан на рисунке 2.33

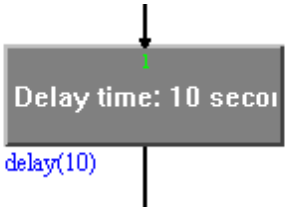


Рисунок 2.33 Оператор выполнения

【**Свойство**】: Отобразить свойства выбранного объекта.

Свойства функционального поля SCC включают в себя тип, имя SCC, серийный номер, положение, описание, оператор выполнения, название операции и относительные параметры. Тип функционального поля отображается в верхней части диалогового окна. Окно выполнения отображается следующим образом: имя SCC — SSS, серийный номер - 1, позиция - (163,75)-(303,125), операция выполнения - это выражение. Отображаются только действующие параметры, так как параметры функциональных полей отличаются в зависимости от их типа. Как показано на рисунке 2.34:

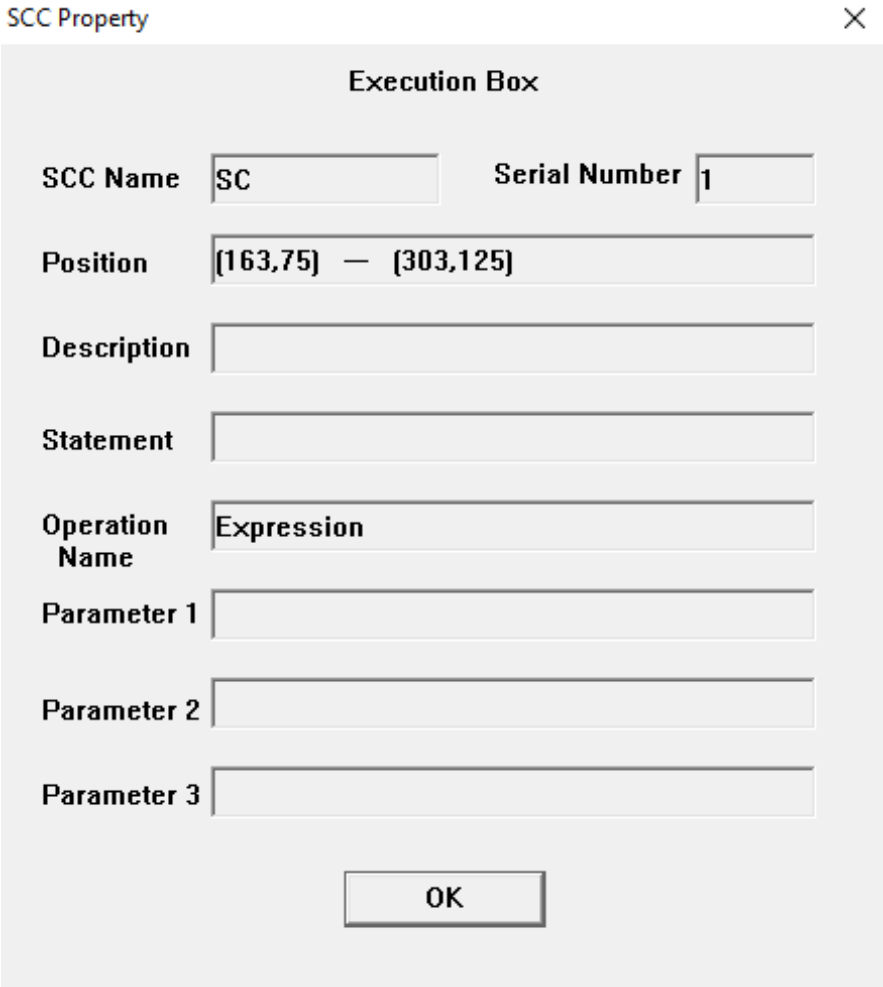


Рисунок 2.34 Свойство SCC

【**Debug**】: Отладка программы SCC; включает в себя следующие функции: ***Automatic*** («Автоматический»), ***Watching*** («Просмотр»), ***Debugging*** («Отладка»), ***Stop*** («Останов»), ***Lock*** («Блокировка»), ***Unlock*** («Разблокировка»), ***Restart*** («Перезапуск»), ***Step***, («Шаг»), ***Continue*** («Продолжение»), ***Stop Debugging*** («Остановка отладки»), ***Insert/Remove Breakpoint*** («Вставка/Удаление точки останова»), и ***Remove All Breakpoints*** («Удаление всех точек останова») и т.д.

2.3.6 Online

2.3.6.1 Команды меню Online

В меню ***Online*** содержатся такие команды, как: ***Connect/PLC*** («Подключение/ПЛК»), ***Connect/Simulator*** («Подключение/Симулятор»), **Disconnect** («Отключение»), ***Display Format*** («Формат отображения»), ***Refresh Program*** («Обновление программы»), ***Unforce*** («Отмена принудительного ввода»), ***Reset*** («Сброс»), ***Set Time*** («Установить время»), ***Master/Slave Switch*** («Переключатель Master/Slave») и т.д., как показано на рисунке 2.35:

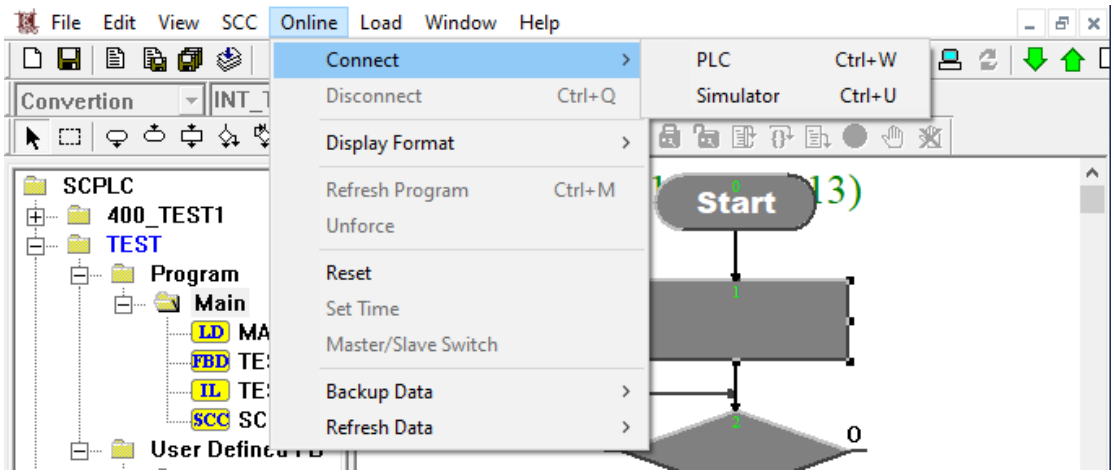


Рисунок 2.35 Меню Online

2.3.6.2 Функционал меню Online

【Connect】/【PLC】: Подключить текущий компьютер для отладки к ПЛК. Перед подключением нужно обязательно убедиться, что сеть физически доступна. В противном случае «СКПро» **откроет** диалоговое окно с ошибкой, как показано на рисунке 2.36:

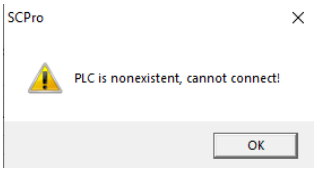


Рисунок 2.36 Сбой соединения с ПЛК

Условия необходимые для подключения к ПЛК

IP-адрес компьютера для отладки должен находиться в том же адресном пространстве, что и IP-адрес ПЛК, то есть, первые три сегмента IP-адреса должны совпадать. В противном случае соединение установить не удастся. Например: если IP-адрес ПЛК - 192.168.1.100, то IP-адрес компьютера для отладки должен начинаться с 192.168.1.***.

«СКПро» автоматически выполнит поиск ПЛК в сети в соответствии с IP-адресом Ethernet модуля ЦПУ в конфигурации ПЛК. Настраивать его заново необходимости нет.

Указание

ПЛК должен быть сконфигурирован согласно руководству по первому подключению.

Конфигурация после успешного подключения

После успешного подключения, цвет фона области редактирования программы становится бледно-лиловым. Информация о данных в ПЛК передается в «СКПро» по сети; параметры со значением 1 и проводящие связи отображаются красным цветом, а параметры со значением 0 и непроводящие связи отображаются зеленым цветом, как показано на рисунке 2.37:

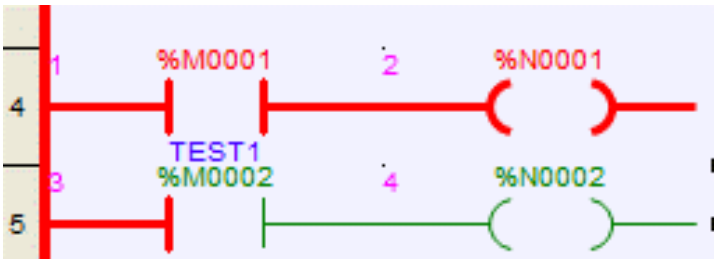


Рисунок 2.37 LD в состоянии онлайн

【Connect】/【Simulator】: Функция имитирует рабочее состояние ПЛК, принудительно вводит фактические значения точек и переменных, а также отлаживает программы.

【Disconnect】: Разрыв соединения компьютера для отладки с ПЛК или симулятором.

【Display Format】: Функция в режиме реального времени указывает форматы отображения целочисленных данных (не для значений типа BOOL). Есть три формата отображения: десятичный, шестнадцатеричный и двоичный; выбранный формат обозначается символом √. Например, если текущий регистр “%MW0001” - десятичный 2002, то в таблице точек три вида форматов отображения будут показаны следующим образом. Все приведенные ниже значения представляют "2002" в различных форматах отображения, за шестнадцатеричными и двоичными данными соответственно следуют буквы “H” и “B”, которые представляют текущий формат отображения, как показано на рисунке 2.38:

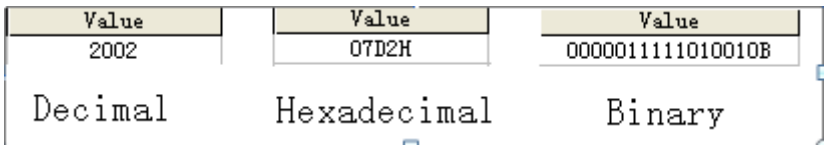


Рисунок 2.38 Формат отображения

【Refresh Program】: Функция обеспечивает удобную отладку для изменения программ без необходимости отключения ПЛК. Если при включенном ПЛК удалить, переместить функцию или изменить параметр функции, справа от названия программы в браузере проекта появится метка “*”; например, “MAIN*”. После внесения подобных изменений, необходимо выбрать команду ***Refresh Program***, чтобы загрузить изменения в ПЛК. После внесения изменений и перед выходом из «СКПро», сохраните измененные программы, в противном случае это приведет к несогласованности программы. Перед обновлением, «СКПро» автоматически скомпилирует программу. Если возникнет какая-либо ошибка, то программа не обновится, а место ошибки будет указано. После обновления, ПЛК будет запущен с учетом новых параметров загруженных программ. Перезапускать систему не нужно. Поскольку команда ***Refresh Program*** изменяет только исполняемую программу, не изменяя исходную программу, сохраненную в ПЛК, программа, загруженная в этот момент, не обновится, поэтому рекомендуется сохранять программы после внесения изменений и выполнять полную загрузку после того, как программа определена в финальном виде.

【**Unforce**】: При включенном ПЛК в таблице точек данных есть команда ***Force*** для принудительного ввода. После запуска функции ***Force*** отсканированные состояния сигналов дискретных и аналоговых входов/выходов не будут отправлены в соответствующие области памяти, они могут быть установлены в соответствии с требованиями отладки без учета их фактического состояния. Команда ***Unforce*** отменяет принудительный ввод точек и возвращается к стандартному сканированию.

【**Reset**】: Сброс модуля центрального процессора ПЛК по сети для перезагрузки. При работе с резервированной системой, команда одновременно делает сброс обоих модулей ЦП. Если модуль ЦП ПЛК, подлежащий сбросу, не находится в одной сети с инженерной станцией, то «СКПро» выдаст сигнал *"Reset failure"* («Сбой сброса»).

【**Set Time**】: Установка времени для ПЛК по сети в режиме online. При работе с резервированной системой, команда одновременно устанавливает время для обоих модулей процессора. Однако устанавливаемое время является временем инженерной станции, а не стандартным временем.

【**Master/Slave Switch**】: Команда переключения на резервный ЦПУ действительна только для резервированных систем. В системах, оснащённых двумя ЦПУ один выполняет роль ведущего (master), а второй – ведомого (slave). Команда **Master/Slave Switch**, переключает текущий ведущий ЦПУ в состояние ведомого, а текущий ведомый – в состояние ведущего.

2.3.7 Меню Download

См. главу 3. «Управление проектами».

2.3.8 Меню Window

2.3.8.1 Команды меню Window («Окно»)

В меню ***Window*** содержатся такие команды, как ***Close*** («Закрыть»), **Close All** («Закрыть все»"), ***Cascade*** («Окна каскадом»), ***Tile Horizontally*** («Окна плиткой по горизонтали») и ***Tile Vertically*** («Окна плиткой по вертикали») и т.д., как показано на рисунке 2.39:

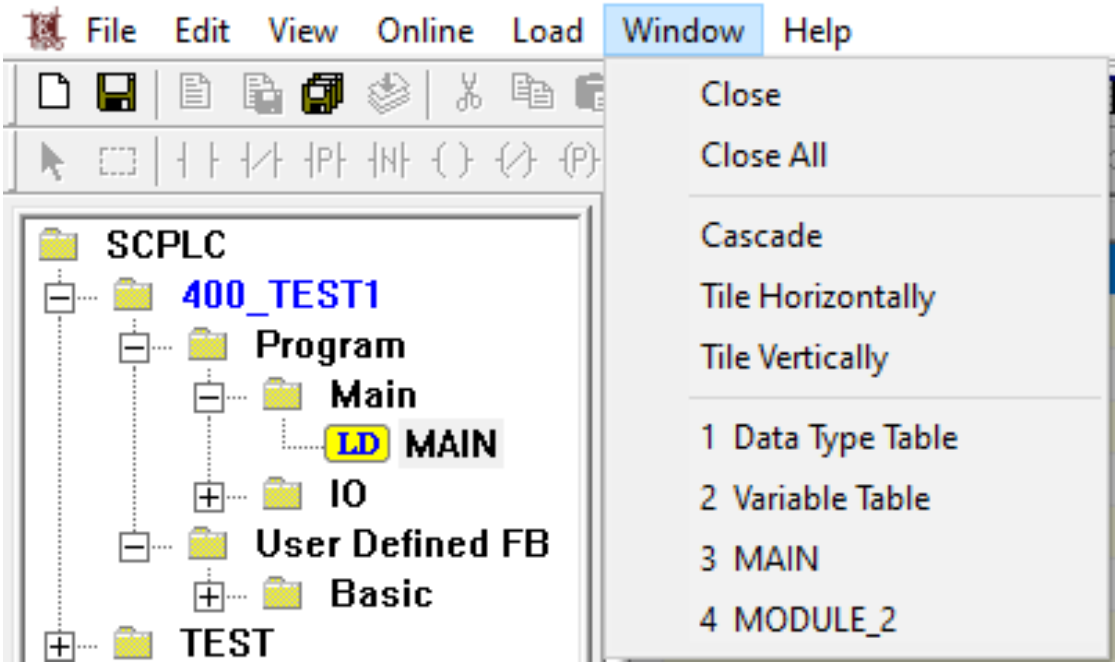


Рисунок 2.39 Меню Window

2.3.9 Меню Help («Справка»)

2.3.9.1 Команды меню Help («Справка»)

В меню ***Help*** («Справка») содержатся такие команды, как: ***Contents*** («Содержимое»), ***Index*** («Индекс»), ***Search*** («Поиск»), ***About...*** («О «СКПро») и т.д. Внешний вид приведён на рисунке 2.40:

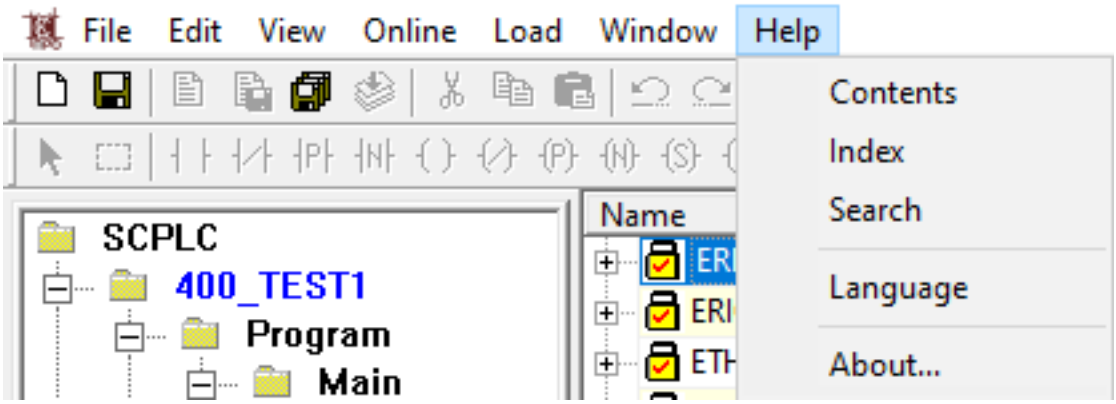


Рисунок 2.40 Меню Help («Справка»)

2.3.9.2 Функционал меню Help («Справка»)

【Contents】: Отобразить вкладку ***Contents*** («Содержимое») из меню «Справка».

【Index】: Отобразить вкладку ***Index*** («Индекс») из меню «Справка».

【Search】: Отобразить вкладку ***Search*** («Поиск») из меню «Справка».

【About ...】 Отобразить версию и авторские права на программное обеспечение. Как показано на рисунке – 2.41:

Рисунок 2.41 Информация о версии «СКПро»

2.4 Системная панель инструментов

Для удобства, системная панель инструментов может размещать в верхней части области редактирования часто применяемые функции операций и процессов редактирования в виде иконок. Все функции могут быть запущены с помощью операций меню, поэтому здесь приведены соответствующие операции меню. Подробная информация о функциях приведена в описании команд меню.

: **New**, соответствует команде меню【**File**】/【**New**】.

: **Open**, соответствует команде меню【**File**】/【**Open**】.

: **Save**, соответствует команде меню【**File**】/【**Save**】.

: **New Program**, соответствует команде меню【**File**】/【**New Program**】.

: **Save Program**, соответствует команде меню【**File**】/【**Save Program**】.

: **Save All Programs**, соответствует команде меню【**File**】/【**Save All Programs**】.

: **Compile Program**, соответствует команде меню【**File**】/【**Compile Program**】.

: **Cut**, соответствует команде меню【**Edit**】/【**Cut**】.

: **Copy**, соответствует команде меню【**Edit**】/【**Copy**】.

: **Paste**, соответствует команде меню【**Edit**】/【**Paste**】.

: **Undo**, соответствует команде меню 【**Edit**】/【**Undo**】.

: **Redo**, соответствует команде меню【**Edit**】/【**Redo**】.

: **Find**, соответствует команде меню【**Edit**】/【**Find**】.

: **Global Find**, соответствует команде меню【**Edit**】/【**Global Find**】.

: **Project Browser**, соответствует команде меню【**View**】/【**Project Browser**】.

: **Output Window**, соответствует команде меню【**View**】/【**Output Window**】.

: **Point Table**, соответствует команде меню【**View**】/【**Point Table**】.

: **Login**, соответствует команде меню【**Online**】/【**Login**】.

: **PLC Connect**, соответствует команде меню【**Online**】/【**Connect**】/【**PLC**】.

: **Disconnect**, соответствует команде меню【**Online**】/【**Disconnect**】.

: **Simulator Connect**, соответствует команде меню【**Online**】/【**Connect**】/【**Simulator**】.

: **Refresh Program**, соответствует команде меню【**Online**】/【**Refresh Program**】.

: **Download All**, соответствует команде меню【**Load**】/【**Download All**】.

: **Download Project**, соответствует команде меню【**Load**】/【**Download Project**】.

: **Upload Project**, соответствует команде меню【**Load**】/【**Upload Project**】.

: **Download Program**, соответствует команде меню【**Load**】/【**Download Program**】.

: **Upload Program**, соответствует команде меню【**Load**】/【**Upload Program**】.

: **Print**, соответствует команде меню【**File**】/【**Print**】.

: **About**, соответствует команде меню【**Help**】/【**About SCPro**】.

2.5 Панель инструментов LD

Для удобства редактирования программ LD, панель инструментов LD может размещать в верхней части меню **LD** операции и функциональные блоки в виде иконок.

: **Move**, Быстрая клавиша【**LD**】/【**Move**】.

: **Block**, Быстрая клавиша【**LD**】/【**Block**】.

: **Normally Open Contact**, соответствует команде меню 【**LD**】/【**Normally Open Contact**】. Быстрая клавиша: F2

: **Normally Close Contact**, соответствует команде меню【**LD**】/【**Normally Close Contact**】. Быстрая клавиша: F3

: **Positive Transition-sensing Contact**, соответствует команде меню【**LD**】/【**Positive Transition-sensing Contact**】. Быстрая клавиша: F4

: **Negative Transition-sensing Contact**, соответствует команде меню【**LD**】/【**Negative Transition-sensing Contact**】. Быстрая клавиша: F5

: **Coil**, соответствует команде меню【**LD**】/【**Coil**】. Быстрая клавиша: F6

: **Negated Coil**, соответствует команде меню【**LD**】/【**Negated Coil**】. Быстрая клавиша: F7

: **Positive Transition-sensing Coil**, соответствует команде меню【**LD**】/【**Positive Transition-sensing Coil**】. Быстрая клавиша: F8

: **Negative Transition-sensing Coil**, соответствует команде меню【**LD**】/【**Negative Transition-sensing Coil**】. Быстрая клавиша: F9

: **Set Coil**, соответствует команде меню【**LD**】/【**Set Coil**】. Быстрая клавиша: F10

: **Reset Coil**, соответствует команде меню【**LD**】/【**Reset Coil**】. Быстрая клавиша: F11

: **Link**, соответствует команде меню【**LD**】/【**Link**】.

: **Negate**, соответствует команде меню【**LD**】/【**Negate**】.

: **Label**, соответствует команде меню【**LD**】/【**Label**】.

: **Goto**, соответствует команде меню【**LD**】/【**Goto**】.

: **Return**, соответствует команде меню【**LD**】/【**Return**】.

: **Comment**, соответствует команде меню【**LD**】/【**Comment**】.

: **Zoom**, соответствует команде меню【**LD**】/【**Zoom**】.

: **Insert**, соответствует команде меню【**LD**】/【**Insert**】.

: **Remove**, соответствует команде меню【**LD**】/【**Remove**】.

: **Property**, соответствует команде меню【**LD**】/【**Property**】.

: **Step**, соответствует команде меню【**LD**】/【**Debug**】/【**Step**】.

: **Continue**, соответствует команде меню【**LD**】/【**Debug**】/【**Continue**】.

: **Insert/Remove Breakpoint**, соответствует команде меню【**LD**】/【**Debug**】/【**Insert/Remove Breakpoint**】.

: **Clear All Breakpoints**, соответствует команде меню【**LD**】/【**Debug**】/【**Clear All Breakpoints**】.

Существует большое количество типов базовых функций, потому необходимо выбирать одну функцию с помощью двух выпадающих списков. Первый выпадающий список используется для выбора группы, а второй - для выбора указанной функции. Например, текущая группа отображения – *Move*, что означает, что функции группы *Move* будут приведены во втором выпадающем списке. Выбрать базовую функцию

можно, нажав на обозначающую её иконку. Таким образом, базовая функция будет размещена непосредственно в области редактирования, как показано на рис.2.42:

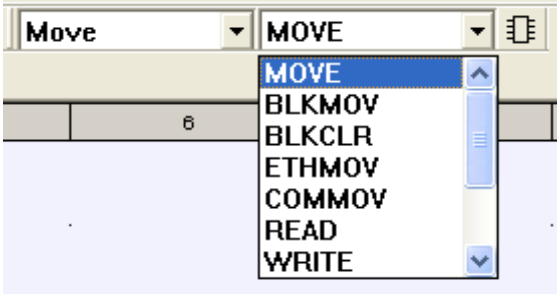


Рисунок 2.42 Базовый функциональный блок

2.6 Панель инструментов FBD

Для удобства редактирования программ FBD, панель инструментов FBD может размещать в верхней части меню FBD операции и функциональные блоки в виде иконок.

: **Move**, соответствует команде меню【**FBD**】/【**Move**】.

: **Block**, соответствует команде меню【**FBD**】/【**Block**】.

: **Link**, соответствует команде меню【**FBD**】/【**Link**】.

: **Negate**, соответствует команде меню【**FBD**】/【**Negate**】.

: **Label**, соответствует команде меню【**FBD**】/【**Label**】.

: **Goto**, соответствует команде меню【**FBD**】/【**Goto**】.

: **Return**, соответствует команде меню【**FBD**】/【**Return**】.

: **Comment**, соответствует команде меню【**FBD**】/【**Comment**】.

: **Zoom**, соответствует команде меню【**FBD**】/【**Zoom**】.

: **Insert**, соответствует команде меню【**FBD**】/【**Insert**】.

: **Remove**, соответствует команде меню【**FBD**】/【**Remove**】.

: **Property**, соответствует команде меню【**FBD**】/【**Property**】.

: **Step**, соответствует команде меню【**FBD**】/【**Debug**】/【**Step**】.

: **Continue**, соответствует команде меню【**FBD**】/ 【**Debug**】/【**Continue**】.

: **Insert/Remove Breakpoint**, соответствует команде меню【**FBD**】/【**Debug**】/【**Insert/Remove Breakpoint**】.

: **Clear All Breakpoints**, соответствует команде меню【**FBD**】/【**Debug**】/【**Clear All Breakpoints**】.

2.7 Панель инструментов IL

Для удобства редактирования программ IL, панель инструментов IL может размещать в верхней части меню **IL** операции и функциональные блоки в виде иконок.

LD :**LD** инструкция, соответствует команде меню 【**IL**】/【**Load**】/【**LD**】.

LDN :**LDN** инструкция, соответствует команде меню【**IL**】/【**Load**】/【**LDN**】.

ST :**ST** инструкция, соответствует команде меню【**IL**】/【**Store**】/【**ST**】.

STN :**STN** инструкция, соответствует команде меню【**IL**】/【**Store**】/【**STN**】.

S :**S** инструкция, соответствует команде меню【**IL**】/【**Store**】/【**S**】.

R :**R** инструкция, соответствует команде меню【**IL**】/【**Store**】/【**R**】.

AND :**AND** инструкция, соответствует команде меню【**IL**】/【**Logic**】/【**AND**】.

OR :**OR** инструкция, соответствует команде меню【**IL**】/【**Logic**】/【**OR**】.

XOR :**XOR** инструкция, соответствует команде меню【**IL**】/【**Logic**】/【**XOR**】.

GT :**GT** инструкция, соответствует команде меню【**IL**】/【**Comparison**】/【**GT**】.

GE :**GE** инструкция, соответствует команде меню【**IL**】/【**Comparison**】/【**GE**】.

LT :**LT** инструкция, соответствует команде меню【**IL**】/【**Comparison**】/【**LT**】.

LE :**LE** инструкция, соответствует команде меню【**IL**】/【**Comparison**】/【**LE**】.

EQ :**EQ** инструкция, соответствует команде меню【**IL**】/【**Comparison**】/【**EQ**】.

NE :**NE** инструкция, соответствует команде меню【**IL**】/【**Comparison**】/【**NE**】.

JMP :**JMP** инструкция, соответствует команде меню【**IL**】/【**Jump**】/【**JMP**】.

CAL :**CAL** инструкция, соответствует команде меню【**IL**】/【**Call**】/【**CAL**】.

RET :**RET** инструкция, соответствует команде меню【**IL**】/【**Call**】/【**RET**】.

: **Step**, соответствует команде меню【**IL**】/【**Debug**】/【**Step**】.

: **Continue**, соответствует команде меню【**IL**】/【**Debug**】/【**Continue**】.

: **Insert/Remove Breakpoint**, соответствует команде меню【**IL**】/【**Debug**】/【**Insert/Remove Breakpoint**】.

: **Clear All Breakpoints**, соответствует команде меню 【**IL**】/【**Debug**】/【**Clear All Breakpoints**】.

2.8 Панель инструментов ST

Для удобства редактирования программ ST, панель инструментов ST может размещать в верхней части меню **ST** операции и функциональные блоки в виде иконок.

:= : **Assignment**, соответствует команде меню 【**ST**】/【**Operator**】/【**Assignment**】.

IF :**IF** оператор, соответствует команде меню 【**ST**】/【**IF**】.

CASE :**CASE** оператор, соответствует команде меню 【**ST**】/【**CASE**】.

FOR :**FOR** оператор, соответствует команде меню 【**ST**】/【**FOR**】.

WHILE :**WHILE** оператор, соответствует команде меню 【**ST**】/【**WHILE**】.

REPEAT :**REPEAT** оператор, соответствует команде меню 【**ST**】/【**REPEAT**】.

EXIT :**EXIT** оператор, соответствует команде меню 【**ST**】/【**EXIT**】.

RETURN : **RETURN** оператор, соответствует команде меню 【**ST**】/【**RETURN**】.

: **Step**, соответствует команде меню 【**ST**】/【**Debug**】/【**Step**】.

: **Continue**, соответствует команде меню 【**ST**】/【**Debug**】/【**Continue**】.

: **Insert/Remove Breakpoint**, соответствует команде меню 【**ST**】/【**Debug**】/【**Insert/Remove Breakpoint**】.

: **Clear All Breakpoints**, соответствует команде меню 【**ST**】/【**Debug**】/【**Clear All Breakpoints**】.

2.9 Панель инструментов SCC

Для удобства редактирования программ SCC, панель инструментов SCC может размещать в верхней части меню **SCC** операции и функциональные блоки в виде иконок.

: **Move**, соответствует команде меню 【**SCC**】/【**Move**】.

: **Block**, соответствует команде меню 【**SCC**】/【**Block**】.

: **Start Box**, соответствует команде меню 【**SCC**】/【**Start Box**】.

: **End Box**, соответствует команде меню 【**SCC**】/【**End Box**】.

: **Execution Box**, соответствует команде меню 【**SCC**】/【**Execution Box**】.

: **Condition Box**, соответствует команде меню【**SCC**】/【**Condition Box**】.

: **Time-limited Condition Box**, соответствует команде меню【**SCC**】/【**Time-limited Condition Box**】.

: **Connector 1**, соответствует команде меню【**SCC**】/【**Connector 1**】.

: **Connector 2**, соответствует команде меню【**SCC**】/【**Connector 2**】.

: **Link**, соответствует команде меню【**SCC**】/【**Link**】.

: **Comment**, соответствует команде меню【**SCC**】/【**Comment**】.

: **Automatic**, соответствует команде меню【**SCC**】/【**Debug**】/【**Automatic**】.

: **Watching**, соответствует команде меню【**SCC**】/【**Debug**】/【**Watching**】.

: **Debugging**, соответствует команде меню【**SCC**】/【**Debug**】/【**Debugging**】.

: **Stop**, соответствует команде меню【**SCC**】/【**Debug**】/【**Stop**】.

: **Lock**, соответствует команде меню【**SCC**】/【**Debug**】/【**Lock**】.

: **Unlock**, соответствует команде меню【**SCC**】/【**Debug**】/【**Unlock**】.

: **Restart**, соответствует команде меню【**SCC**】/【**Debug**】/【**Restart**】.

: **Step**, соответствует команде меню【**SCC**】/【**Debug**】/【**Step**】.

: **Continue**, соответствует команде меню【**SCC**】/【**Debug**】/【**Continue**】.

: **Stop Debugging**, соответствует команде меню【**SCC**】/【**Debug**】/【**Stop Debugging**】.

: **Insert/Remove Breakpoint**, соответствует команде меню【**SCC**】/【**Debug**】/【**Insert/Remove Breakpoint**】.

: **Clear All Breakpoints**, соответствует команде меню【**SCC**】/【**Debug**】/【**Clear All Breakpoints**】.

2.10 Окно вывода

Для удобства компиляции, отладки программы и отслеживания статуса, в окне вывода отображаются результаты компиляции, статус отладки, результаты поиска, тревоги, события SOE и т.д.

【**Compile**】: Это окно вывода результатов компиляции программ.

Если дважды щелкнуть на сообщение об ошибке в этом окне, то система перейдет к соответствующему местоположению ошибки. Окно показано на рис.2.43:

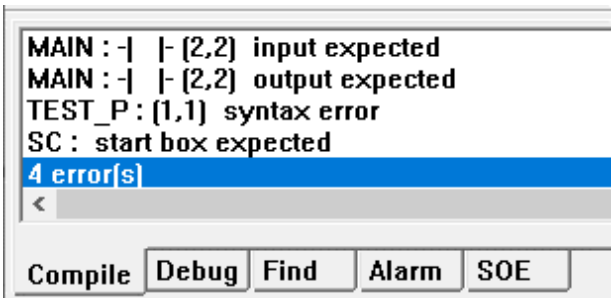


Рисунок 2.43 Окно вывода

【Debug】: Это окно вывода сообщений процесса отладки. В режиме онлайн-отладки, ошибки, возникающие в процессе выполнения программы, будут выводиться в этом окне.

【Find】: Это окно вывода результатов глобального поиска с помощью команды ***Global Find***. При глобальном поиске с помощью функции **【Edit】/【Global Find】** или кнопки на панели инструментов, местоположение и общее количество найденного содержимого будут отображаться в этом окне, как показано на рис.2.44:

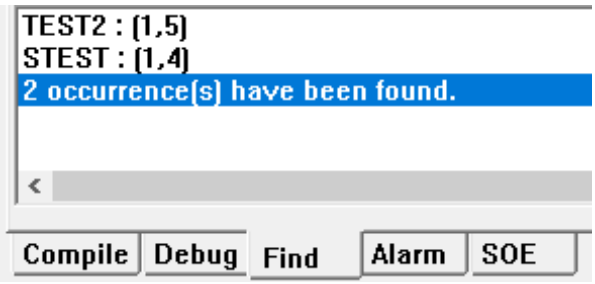


Рисунок 2.44 Результаты глобального поиска

【Alarm】: Это окно вывода сообщений о тревогах, возникающих в программе. В этом окне, в поле ***Execute Box/Alarm*** будут отображаться сообщение о тревогах, при редактировании программ с помощью редактора SCC в процессе выполнения программы, а также сообщения о состоянии запуска/завершения программы, как показано на рис.2.45:

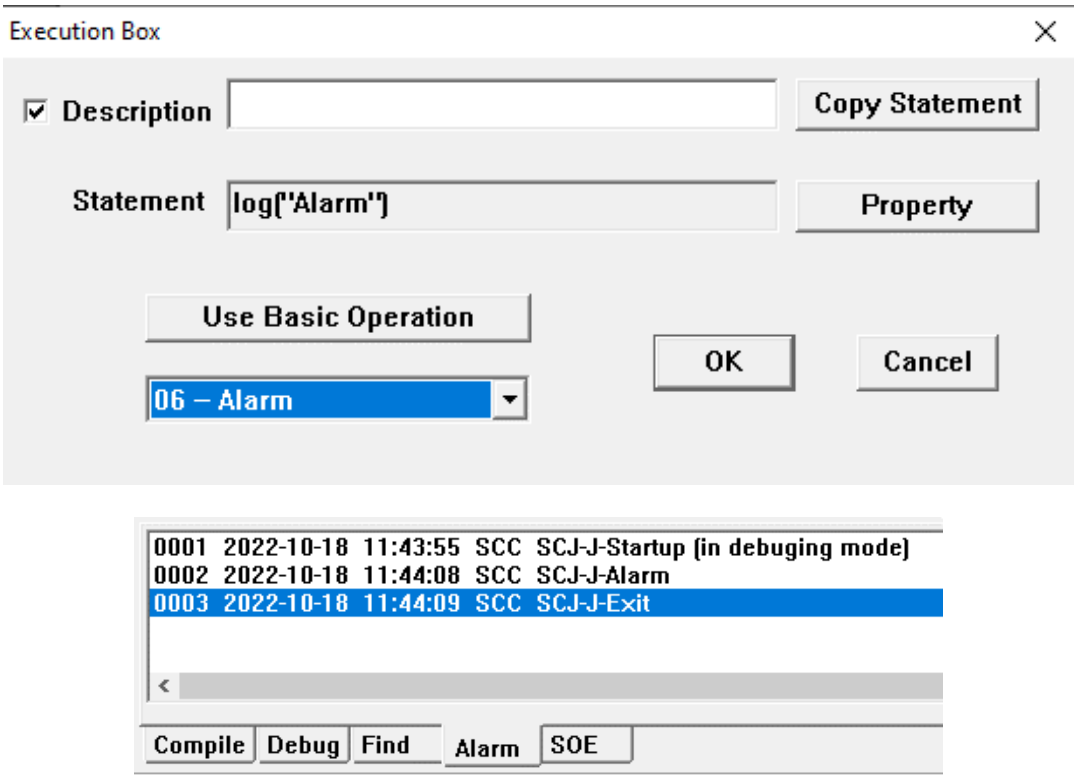


Рисунок 2.45 Сообщение о тревоге

【**SOE**】: Это окно вывода сообщений о событиях SOE.

Если ПЛК, оснащен модулем последовательности событий (SOE), в этом окне будут выводиться все сообщения о событиях SOE, включая номер точки данных, изменение 0->1 бит или 1->0 бит, сообщения о времени и т.д., как показано на рисунок 2.46:

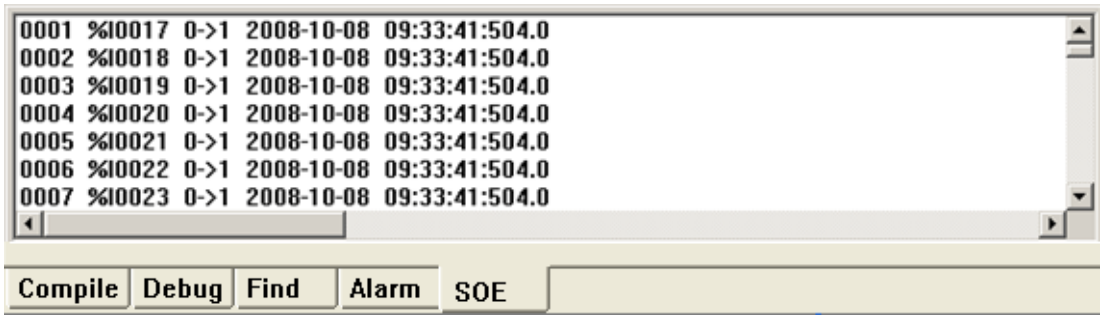


Рисунок 2.46 Сообщение о событии SOE

2.11 Справочник по горячим клавишам

File	Новый проект	Ctrl + N
	Новая программа	Ctrl + E
	Сохранить проект	Ctrl + S
	Сохранить программу	Ctrl + I
	Скомпилировать все программы	Ctrl + B

	Скомпилировать программу	Ctrl + R
	Открыть проект	Ctrl + O
	Печать	Ctrl + P
Edit	Отменить	Ctrl + Z
	Повторить	Ctrl + Y
	Вырезать	Ctrl + X
	Копировать	Ctrl + C
	Вставить	Ctrl + V
	Удалить	Del
	Выбрать все	Ctrl + A
	Поиск	Ctrl + F
	Заменить	Ctrl + H
	Глобальный поиск	Ctrl + G
	Вставка	Ctrl + J
	Удалить	Ctrl + K
	Масштабировать	Ctrl + D
Online	Подключение / ПЛК	Ctrl + W
	Подключение / Симулятор	Ctrl + U
	Отключение	Ctrl + Q
Load	Загрузить все	Ctrl + L
	Загрузить программу	Ctrl + T
	Обновить программу	Ctrl + M
Debug	Шаг	Ctrl+F9
	Продолжить	Ctrl+F10
	Вставка/Удаление точки останова	Ctrl+F11
	Удаление всех точек останова	Ctrl+F12

View	Окно программы	Alt+0
	Браузер проекта	Alt+1
	Окно вывода	Alt+2
	Таблица точек данных	Alt+3
Program	Нормально разомкнутый контакт	F2
	Нормально замкнутый контакт	F3
	Контакт с положительным переходом	F4
	Контакт с отрицательным переходом	F5
	Катушка	F6
	Инверсная катушка	F7
	Катушка положительного перехода	F8
	Катушка отрицательного перехода	F9
	Катушка установки	F10
	Катушка сброса	F11
	Присвоить отрицательное значение	Alt + F1
	Метка (Label)	Alt + F2
	Переход (Goto)	Alt + F3
	Возврат (Return)	Alt + F4
	Комментарий (Comment)	Alt + F5
	Блок функции (Function block (FB))	First letter of FB
Link	Левая вертикальная связь (Left vertical link)	Alt + F10
	Правая вертикальная связь (Right vertical link)	Alt + F11

	Горизонтальная связь (Horizontal link)	Alt + F12
Перемещение курсора	Вверх	↑
	Вниз	↓
	Влево	←
	Вправо	→
	Страница вверх	Page Up
	Страница вниз	Page Down
	Первая колонка	Home
	Последняя колонка	End
	Первая строка	Ctrl + Home
	Последняя строка	Ctrl + End
Переместить FB	Вверх	Ctrl +↑
	Вниз	Ctrl +↓
	Влево	Ctrl +←
	Вправо	Ctrl +→
Прочее	Помощь	F1
	Свойства FB	Shift + Enter
	Перейти в программу	Ctrl + Enter

3 Работа с проектами

После входа в среду разработки «СКПро», основной задачей пользователя становится преобразование имеющегося у него проекта в код, который может быть исполнен в ПЛК. В среде разработки проект управляет всеми элементами, такими как программы, данные, ресурсы и т.д. С технической точки зрения, управление проектом осуществляется с помощью файла конфигурации проекта и файлов программ проекта. Файл конфигурации проекта отвечает за конфигурацию оборудования, управление программами, настройку параметров ПЛК и т.д. Файлы программы проекта определяют программный набор процессов, которые должны быть выполнены, такие, как программа LD, программа FBD, программа IL, программа ST и программа SCC и т.д.

3.1 Браузер проекта

Используя браузер проекта, можно отобразить содержимое проекта «СКПро», а также переключаться между составляющими элементами проекта, такими, как программа, данные и ресурсы.

Браузер проекта отображается в виде дерева содержимого и предоставляет доступ к следующим командам:

- **Program**---**LD**, **FBD**, **IL**, **ST**, и **SCC** и т.д.;
- **Data**---**Point Table**, **Variable Table**, **Optional Point Table**;
- **Resource**---**PLC Configuration**, **Task Configuration**, **Interrupt Configuration** и т.д., как показано на рисунке 3.1:

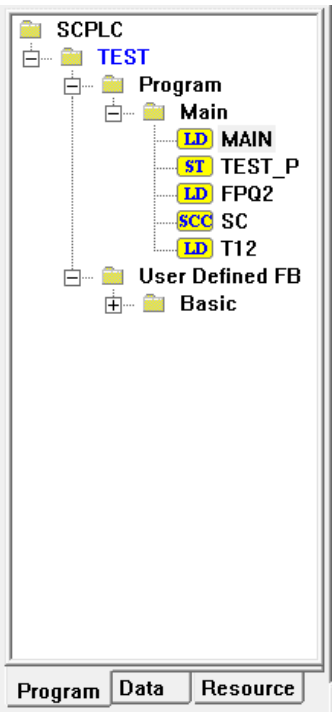


Рисунок 3.1 Браузер проекта

По умолчанию программы отображается на третьем уровне вкладки "Program" дерева, дважды щелкните узел, чтобы открыть соответствующее содержимое.

3.2 Создание нового проекта

Создание нового проекта включает в себя несколько следующих шагов:

Шаг	Операция
1	Создание нового проекта. См. раздел 3.2.1 «Создание проекта».
2	Настройка ПЛК. См. раздел 3.2.2 «Конфигурация аппаратного обеспечения ПЛК».
3	Создание пользовательских программ. См. раздел 3.3 «Управление программой».
4	Сохранение и компиляция программы.
5	Выгрузка проекта и программ. Смотрите 3.8 «Загрузка и выгрузка файла проекта» и 3.9 «Загрузка и выгрузка программных файлов».

3.2.1 Создание проекта

Для начала, создайте пустой проект. Откройте программу «СКПро», нажмите **【File】**/ **【New】** в главном меню или значок на системной панели инструментов, как показано на рисунке 3.2:

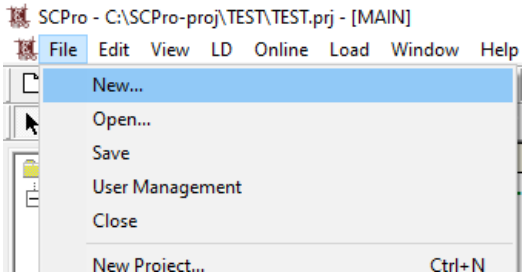


Рисунок 3.2 Создание нового проекта

В открывшемся диалоговом окне **Save As** будет предложено сохранить проект. Сохраните проект, например, как “test.prj”, как показано на рисунке 3.3. Этот файл можно открыть напрямую, если в дальнейшем понадобится внести какие-либо изменения.

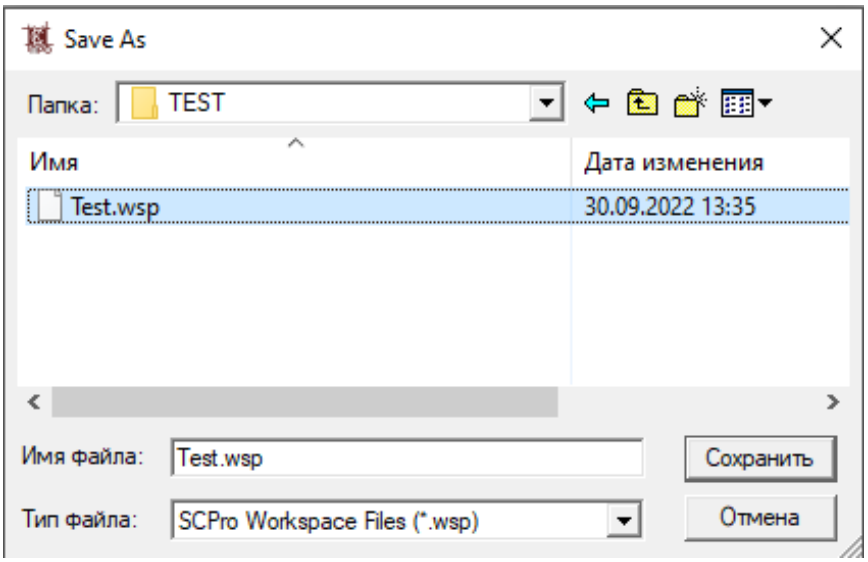


Рисунок 3.3 Сохранение существующего файла

«СКПро» выведет диалоговое окно с предупреждением: «Пожалуйста, настройте ПЛК!», как показано на рисунке 3.4, выберите **OK**, а затем создайте конфигурацию оборудования.

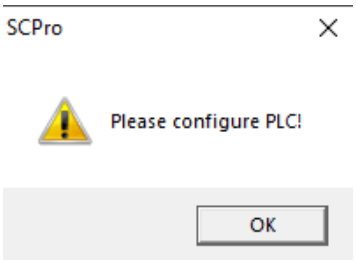
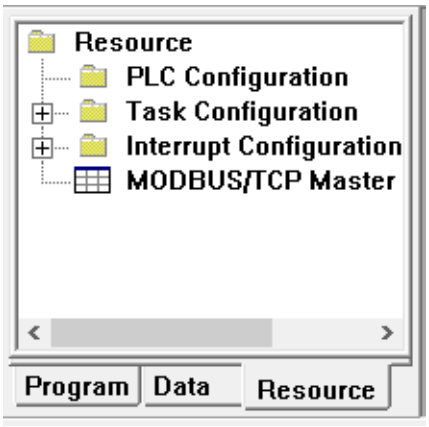


Рисунок 3.4 Диалоговое окно предупреждения

3.2.2 Конфигурация аппаратного обеспечения ПЛК

В браузере проекта дважды щелкните **【Resource】/【PLC Configuration】** левой кнопкой мыши, после чего появится диалоговое окно **Hardware Configuration**, как показано на рисунке ниже. Чтобы сохранить исходный файл проекта и исходные файлы программы в ПЛК, необходимо поставить галочку рядом с пунктом “Allow to upload”. Если галочка поставлена, то исполняемые и исходные файлы будут загружены в ПЛК с возможностью последующей выгрузки; если галочка не поставлена, то будут загружены только исполняемые файлы и последующая выгрузка с помощью «СКПро» завершится неудачей.



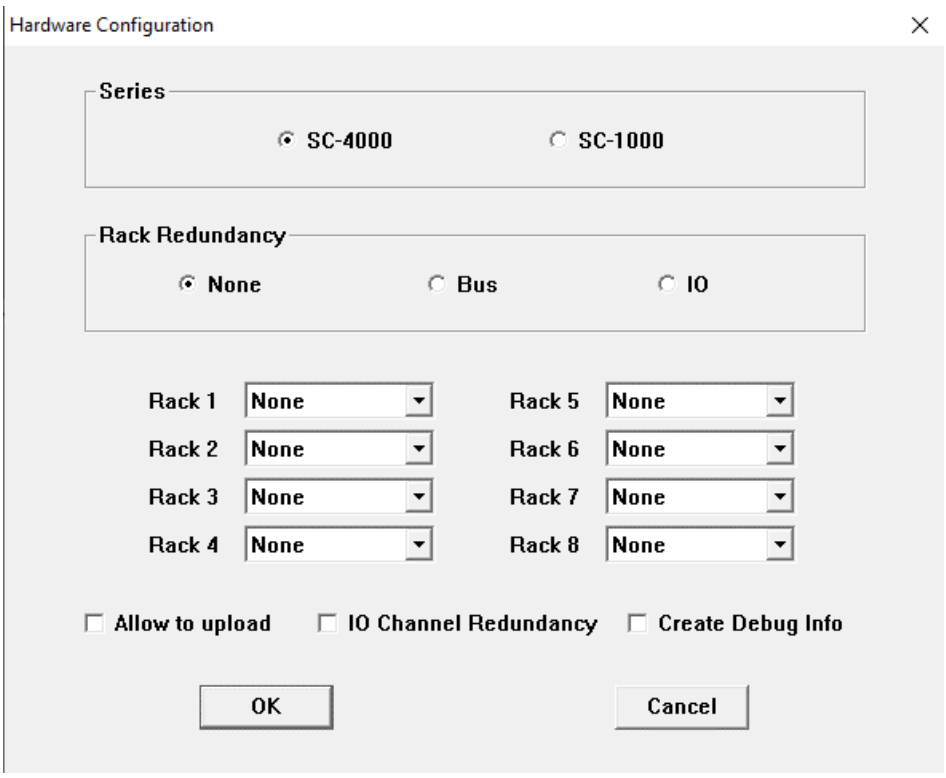


Рисунок 3.5 Конфигурация оборудования

Стойки (Rack) должны быть сконфигурированы в соответствии с техническими требованиями. Стойка 1 является основной стойкой по умолчанию, а остальные – стойки расширения. Для каждой стойки можно выбрать монтажную плату с 6 слотами, 9 слотами, 12 слотами или 15 слотами в соответствии с техническими требованиями. Нажмите кнопку **OK** после завершения настройки, как показано на рис.Рисунок 3.6:

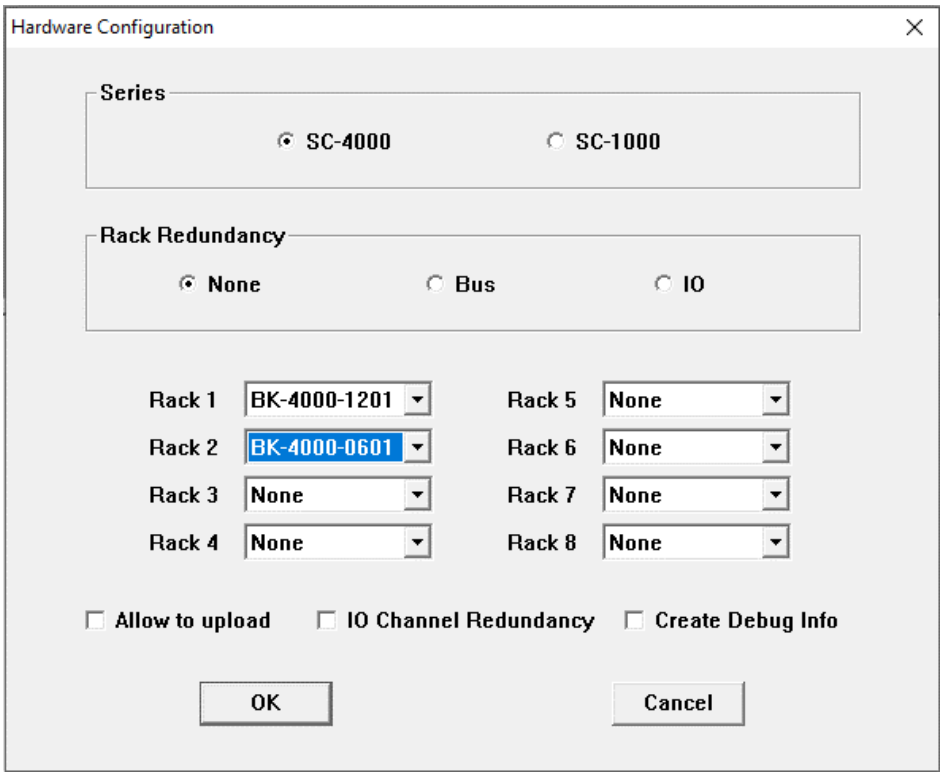


Рисунок 3.6 Конфигурация оборудования

После нажатия кнопки **OK**, аппаратная структура внутри браузера проекта будет выглядеть, как показано на рисунке 3.7

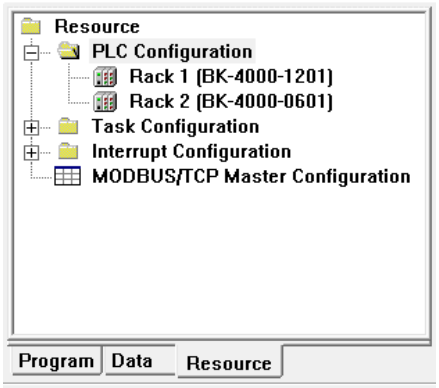


Рисунок 3.7 Аппаратная структура

Далее необходимо выбрать модули для каждой стойки. Дважды щелкните нужную стойку в браузере проекта, конфигурация модулей отобразится в правой области редактирования, все слоты в это время будут пусты, как показано на рисунке 3.8

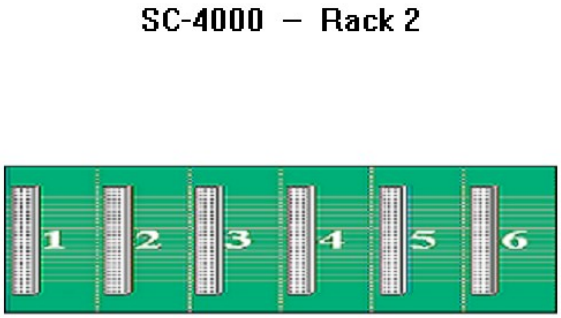


Рисунок 3.8 Пустая стойка

Указание

Любой слот стойки ПЛК серии СК-4000 не предъявляет особых требований к типу модуля, то есть почти любой модуль может быть сконфигурирован в любом слоте, если его можно выбрать. Исключение составляют системы с резервированными ПЛК.

Дважды щелкните модуль или пустой слот, появится диалоговое окно **Module**, а затем выберите группу модулей и тип модуля, как показано на рисунке 3.9:

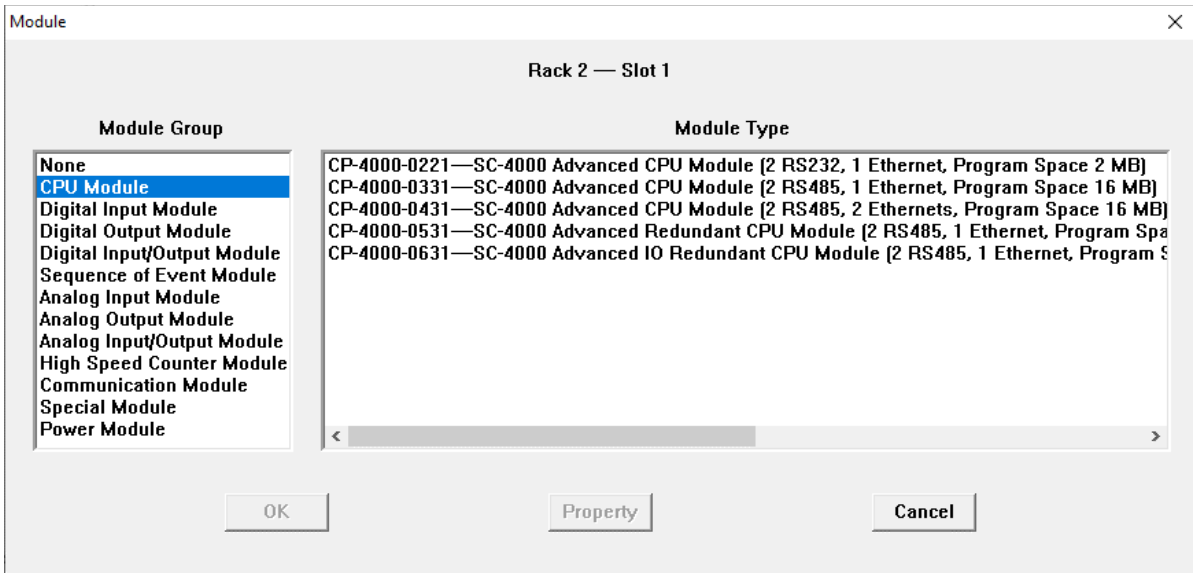


Рисунок 3.9 Конфигурация модуля

Группы включают в себя следующие модули: модули центрального процессора (CPU), модули дискретного ввода (digital input), модули дискретного вывода (digital output), модули дискретного ввода / вывода (digital input/output), модули последовательности событий (sequence of event), модули аналогового ввода (analog input), модули аналогового вывода (analog output), модули аналогового ввода / вывода (analog input/output), модули высокоскоростного счетчика (high speed counter), модули связи (communication), специальные модули (special) и блоки питания (power). "None" означает, что слот пуст, модуля нет.

После выбора модуля нажмите кнопку "*Property*", чтобы настроить свойства модуля, как показано на рисунке 3.10:

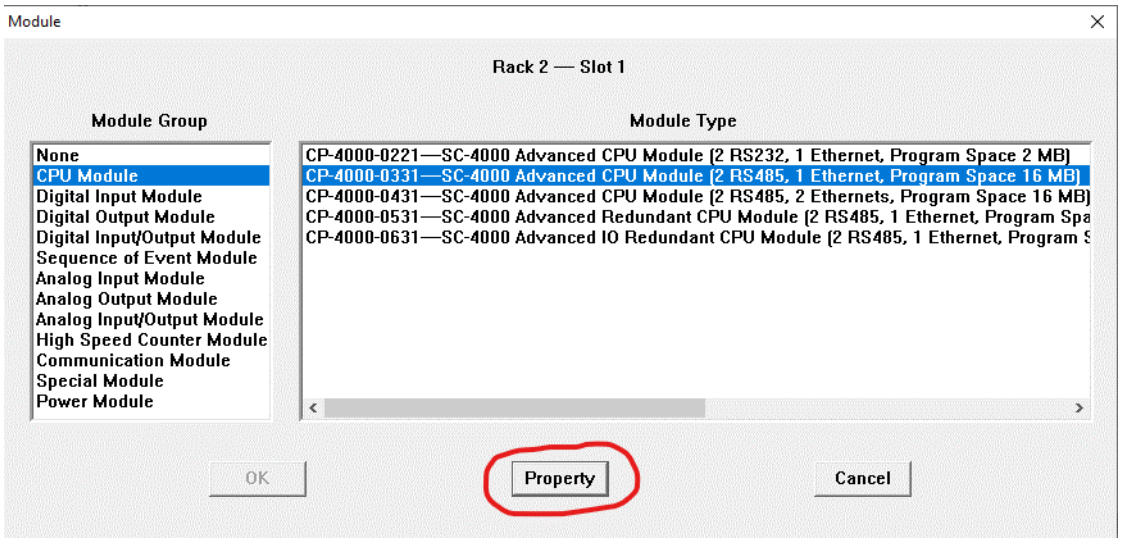


Рисунок 3.10 Свойства модуля

Модуль центрального процессорного устройства (CPU Module)

Модули ЦПУ включают в себя базовые, стандартные, продвинутые и резервируемые продвинутые модули. Для каждого такого типа модулей есть несколько вариантов, отличающихся количеством последовательных портов, интерфейсов Ethernet и памятью программы/данных. Резервируемые продвинутые модули ЦПУ могут быть использованы только в резервированных системах. Для различных модулей ЦПУ процедура настройки конфигурации встроенных последовательных портов одинакова: необходимо установить значения параметров «Скорость передачи данных» (*Baudrate*), «Бит данных» (*Data Bit*), «Стоповый бит» (*Stop Bit*), «Четность» (*Parity*) и «Протокол» (*Protocol*) из соответствующих раскрывающихся списков. Существуют требования к конфигурациям Ethernet-адреса двойных сетей. IP-адрес Ethernet состоит из четырех частей. Для двойной сети первая, вторая и четвертая части IP-адреса обеих сетей должны быть одинаковыми, а третья - отличаться. Например, двойные сетевые адреса могут быть настроены следующим образом 192.168.200.100 и 192.168.201.100, как показано на рис.3.11:

Для ПЛК с возможностью централизованной синхронизации времени по протоколу NTP доступно два поля "NTP Server 1 IP Address" и "NTP Server 1 IP Address". Синхронизация времени происходит раз в минуту, текущее время ПЛК далее можно проверить в онлайн-режиме в %SW1-%SW7.

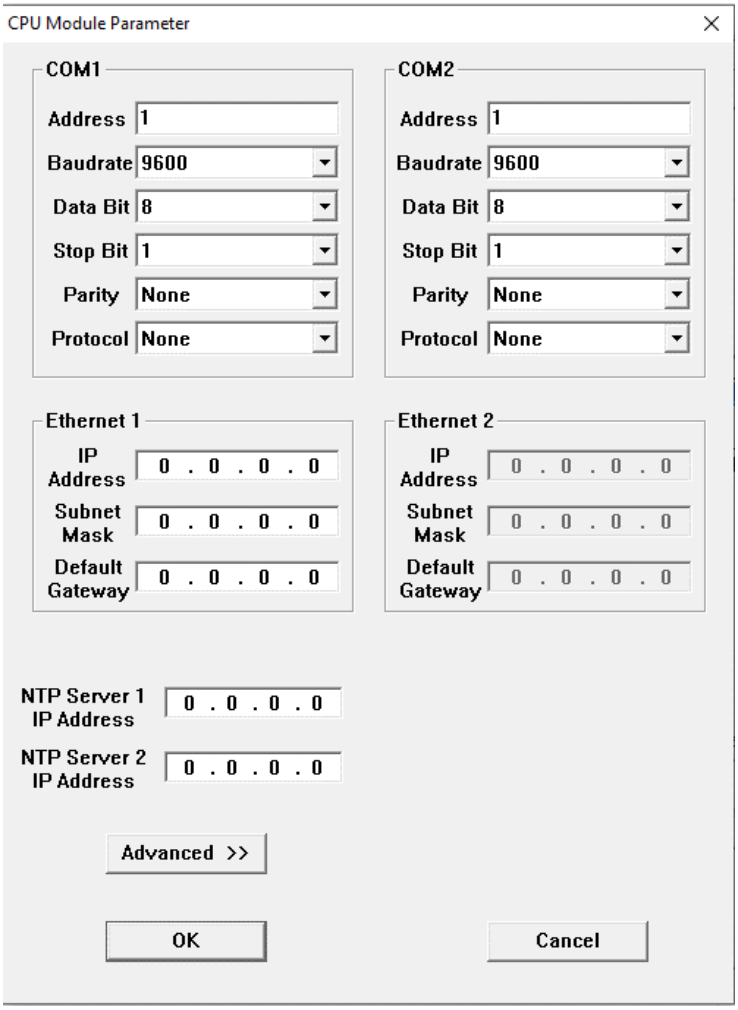


Рисунок 3.11 Параметры модуля процессора

Модули дискретного ввода

Существует несколько различных модулей дискретного ввода, однако принцип их настройки идентичен – как минимум, необходимо указать только стартовый номер бита для первого входа. После установки стартового номера, последующие будут присвоены автоматически, последовательным возрастанием, как показано на рисунке 3.12.

Все модули дискретного ввода имеют возможность фильтрации сигнала от дребезга, время фильтрации каналов настраивается в параметрах модуля. При неверной настройке времени фильтрации появится предупреждение, содержащее пределы (минимальный - 0, максимальный - 250) и единицу измерения (10 мс).

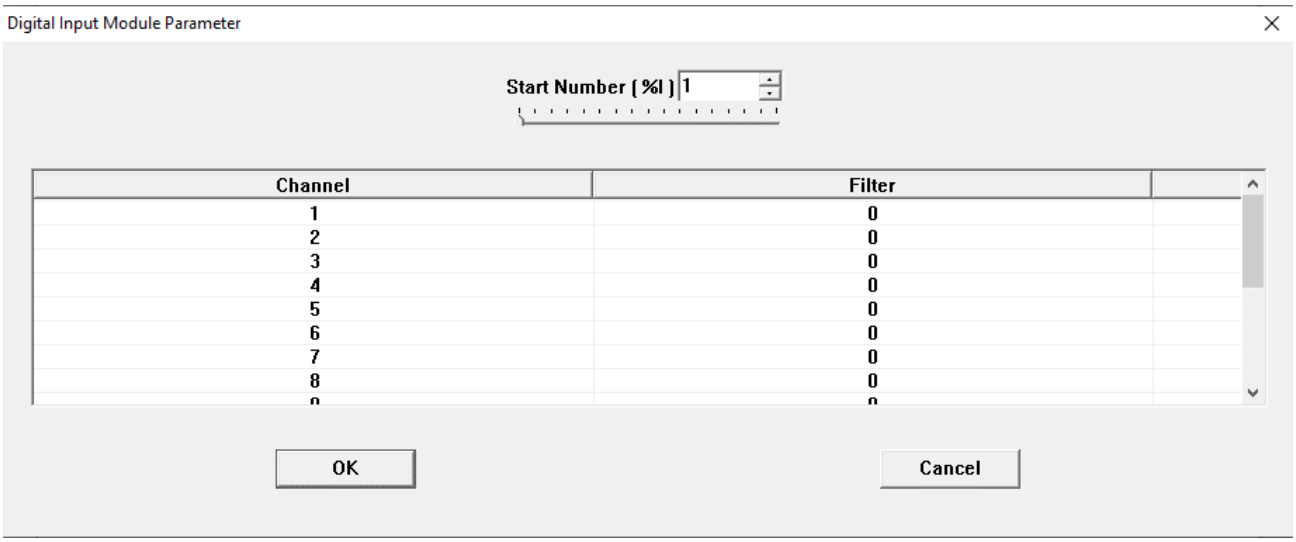


Рисунок 3.12 Параметры модуля цифрового ввода

Модули дискретного вывода

Существует несколько различных модулей дискретного вывода, однако принцип их настройки идентичен – как минимум, необходимо указать только стартовый номер бита для первого выхода. После установки стартового номера, последующие будут присвоены автоматически, последовательным возрастанием, как показано на рисунке 3.13. Все модули дискретного вывода настраиваются на безопасное состояние при ошибках вывода (ошибка самодиагностики модуля, ошибка связи с ПЛК), в выпадающем поле "Fault Output" имеется три режима настройки: "Hold" – значение канала останется тем же, что было до ошибки; "1" – значение канала установится в "Канал включен"; "0" – значение канала установится в "Канал отключен".

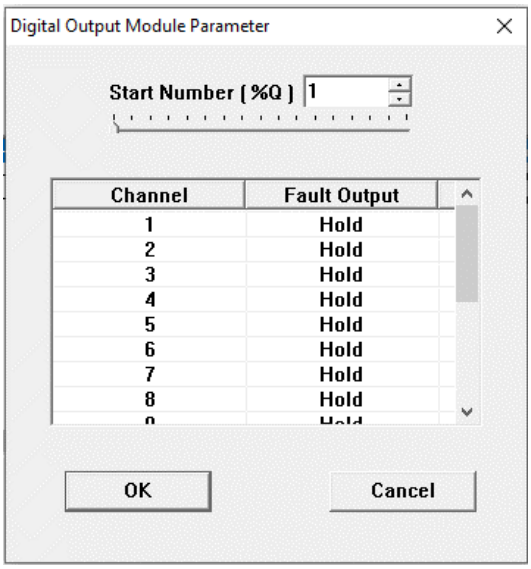


Рисунок 3.13 Параметры модуля цифрового вывода

Модули аналогового ввода

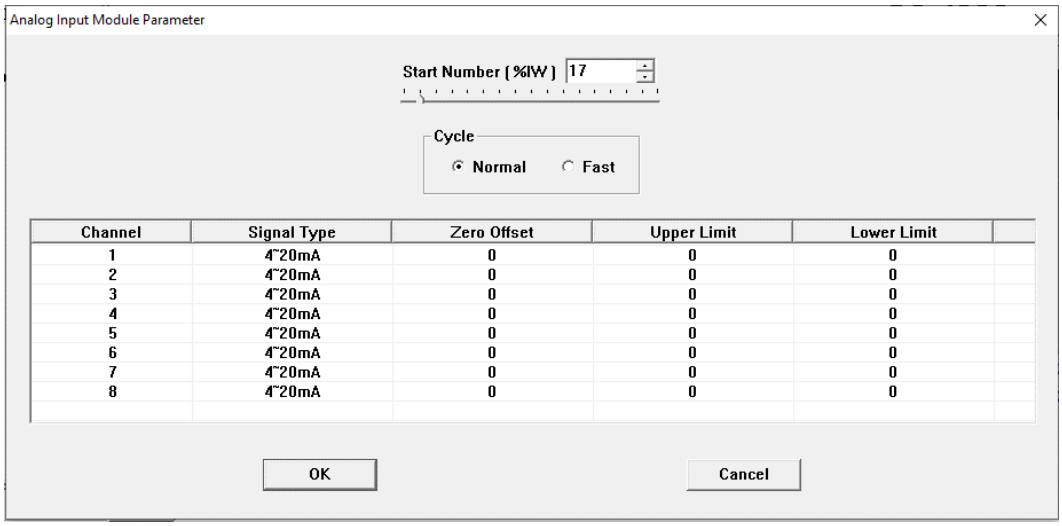
Существует несколько различных модулей аналогового ввода, однако принцип их настройки идентичен – как минимум, необходимо указать только стартовый номер слова для первого входного канала. После установки стартового номера, последующие будут присвоены автоматически, последовательным возрастанием, как показано на рисунке 3.14.

Кроме того, в зависимости от модуля, каждый канал модуля имеет настройку типа сигнала ("Signal Type"), смещение нуля ("Zero Offset"), а так же может иметь настройки верхнего предела ("Upper Limit") и нижнего предела ("Lower Limit"). Единицы измерения сигнала АЦП зависят от выбранного типа сигнала. Подробное описание предоставлено в соответствующем системном руководстве на линейку ПЛК или на модуль.

При настройке модулей AI-4000-0801, AI-4000-0802, AI-4000-1601, AI-4000-0804 можно также настроить цикл опроса. Для цикла опроса можно выбрать значение Normal, либо Fast, где Normal – опрос раз в 400 мс, Fast – опрос при каждом выполнении цикла сканирования (выбор данного значения увеличивает общее время цикла).

Указание

Настройки "Zero Offset", "Upper Limit" и "Lower Limit" не смещают измеряемый диапазон, а лишь пересчитывают выходное значение.



Channel	Signal Type	Zero Offset	Upper Limit	Lower Limit
1	4~20mA	0	0	0
2	4~20mA	0	0	0
3	4~20mA	0	0	0
4	4~20mA	0	0	0
5	4~20mA	0	0	0
6	4~20mA	0	0	0
7	4~20mA	0	0	0
8	4~20mA	0	0	0

Рисунок 3.14 Параметры модуля аналогового ввода

Модули аналогового вывода

Существует несколько различных модулей аналогового вывода, однако принцип их настройки идентичен – как минимум, необходимо указать только стартовый номер слова для первого выходного канала. После установки стартового номера, последующие будут присвоены автоматически, последовательным возрастанием, как показано на рисунке 3.15.

Кроме того, каждый канал модуля имеет настройку типа сигнала ("Signal Type"). Все модули аналогового вывода настраиваются на безопасное состояние при ошибках вывода (ошибка самодиагностики модуля, ошибка связи с ПЛК), в выпадающем поле "Fault Output" имеется два режима настройки: "Hold" – значение канала останется тем же, что было до ошибки; "Preset" – значение канала установится в предустановленное "Preset Value". Единицы измерения сигнала ЦАП зависят от выбранного типа сигнала. Подробное описание предоставлено в соответствующем системном руководстве на линейку ПЛК или на модуль.

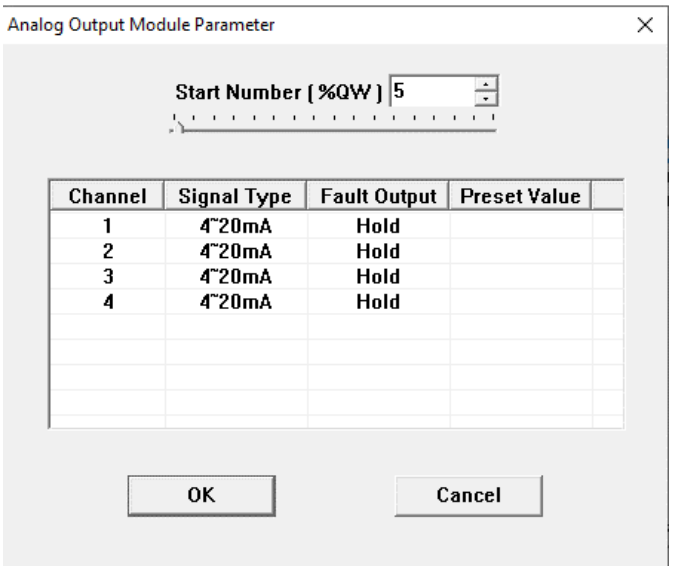


Рисунок 3.15 Параметры модуля аналогового вывода

3.3 Программа

В программе должна быть как минимум одна основная программа “MAIN”. Каждый цикл сканирования начинается с основной программы, а циклы сканирования других подпрограмм выполняются путем вызова подпрограммы из основной программы. Основная программа “MAIN” и все подпрограммы перечислены на вкладке ***Program*** браузера проекта. Дважды щелкните по названию программы, чтобы увидеть её содержимое. Программы разделены на типы по языкам, на которых они написаны: LD, FBD, IL, ST, SCC и т.д., как показано на рисунке 3.17:

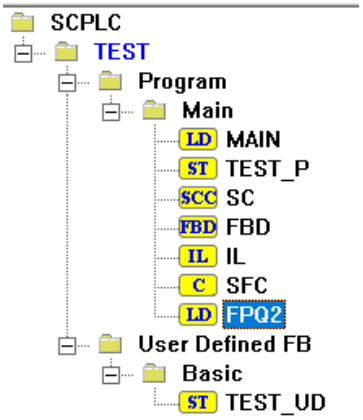


Рисунок 3.16 Структура программы

Вместо сканирования SCC выполняется методом последовательного управления, то есть пошагово, останавливаясь после выполнения последнего шага. Программа SCC никогда не выполняется автоматически, она может быть запущена только программой “MAIN” или другими подпрограммами. Выполнение SCC не влияет на выполнение программ LD или других программ.

3.3.1 Добавление программы

Если вы хотите добавить новую программу, в главном меню нужно запустить команду **【File】/【New Program】** или кнопку на панели инструментов. Затем выбрать тип программы и дать ей название. “Program Description” заполнять не обязательно, но данное поле может помочь пользователю понять предназначение программы. Программа может быть основной задачей (выполнение цикла), а также задачей 1 ~ 16 (они выполняются своевременно, приоритет от низкого до высокого), также может быть прерыванием (прерывание по таймеру 1 ~ 4, прерывание ввода-вывода, уровень прерывания от низкого до высокого, в любой из выполняемых программ необходимо включить обработку прерываний функциональным блоком "ENI"), как показано на рис.3.18:

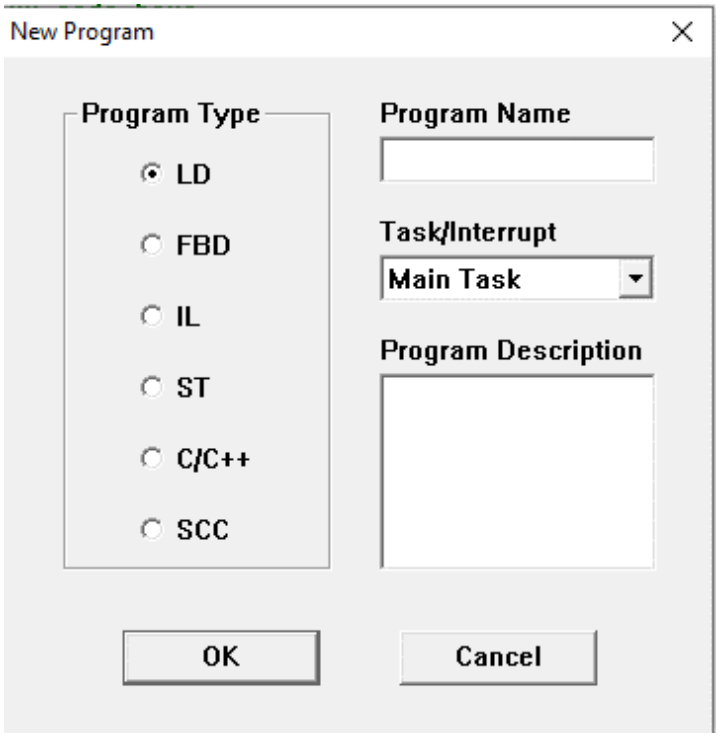


Рисунок 3.17 Добавление программы

3.3.2 Удаление программы

Если вы хотите удалить программу, щелкните на программе правой кнопкой мыши и выберите ***Delete*** (основная программа “MAIN” создается по умолчанию, и не может быть удалена), как показано на рисунке 3.19:

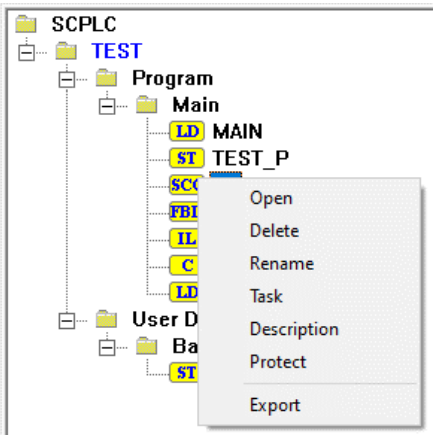


Рисунок 3.18 Удаление программы

3.3.3 Изменение названия программы

Если вы хотите переименовать программу, щелкните на программе правой кнопкой мыши и выберите ***Rename*** (основная программа “MAIN” не может быть переименована), как показано на рисунке 3.20:

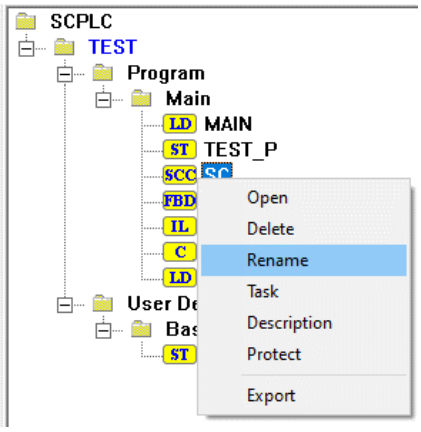


Рисунок 3.19 Изменение названия программы

3.3.4 Добавление описания программы

Если вы хотите добавить описание программы, щелкните на программе правой кнопкой мыши и выберите ***Description***. Содержимое описания отображается в области редактирования программы, как показано на рисунке 3.21:

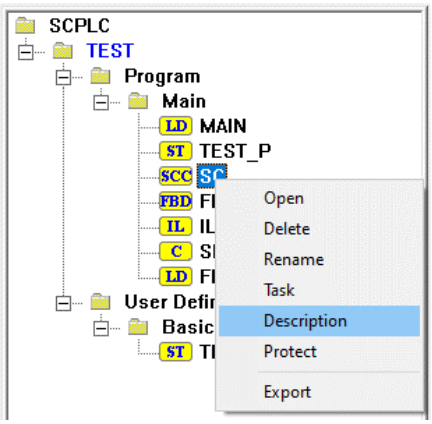


Рисунок 3.20 Добавление описания программы

3.3.5 Защита программы

Если вы хотите защитить программу, щелкните на программе правой кнопкой мыши и выберите ***Protect***, как показано на рисунке 3.22. Откроется диалоговое окно **Protect**, как показано на рисунке 3.23, «None» означает отсутствие защиты; «Read Only» означает, что после защиты программа может быть только просмотрена другими людьми, но не может быть изменена; «No Read or Write» означает, что после защиты программа не может быть просмотрена другими людьми, а также не может быть удалена или изменена. Если вы хотите изменить программу, вам необходимо ввести пароль и изменить параметр защиты программы на «None».

Предупреждение
После защиты запомните пароль, так как его невозможно восстановить без передачи копии разработанного ПО службе поддержки!

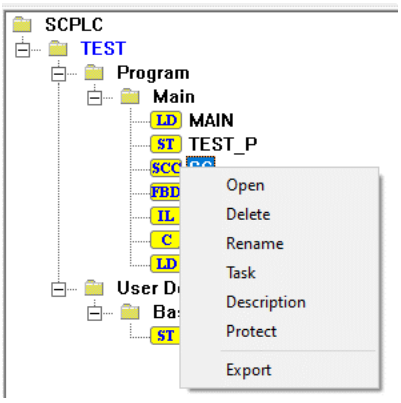


Рисунок 3.21 Защита программы

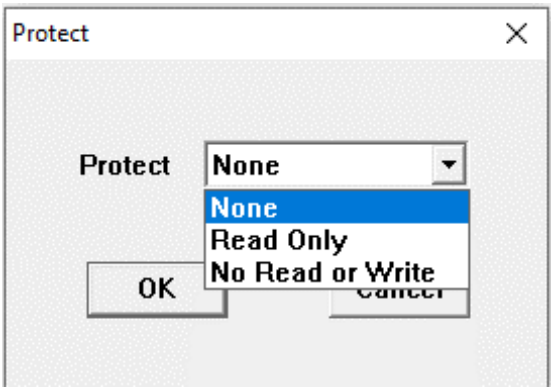


Рисунок 3.22 Свойства защиты

3.4 Задача (Task)

Проект может быть сконфигурирован таким образом, чтобы исполнять несколько задач. Вызов программ, в таком случае, будет зависеть от распределения задач.

В состоянии по умолчанию система выполняется как единая задача, «MAIN» — это основная программа, она выполняется автоматическим и самой первой. Затем она прямо или косвенно вызывает другие программы. В таком случае нет необходимости отдельно настраивать задачи. Как правило, ПЛК управляет средой выполнения одной задачи, что удовлетворяет требования к работе системы.

В рабочем окружении с несколькими задачами основная задача (автоматически выполняющая программу «MAIN») выполняется циклически, и её приоритет является самым низким. В рабочем окружении поддерживается до 16 задач, и приоритет задачи № 16 является самым высоким. Каждая программа должна относится к основной задаче или любой из задач № 1 - 16 (настроить можно при создании новой программы); программы, относящиеся к одной и той же задаче, могут вызываться между собой.

Все задачи и все программы каждой задачи перечислены в разделе **Task Configuration** вкладки 【**Resource**】 браузера проекта, как показано на рисунке 3.24:

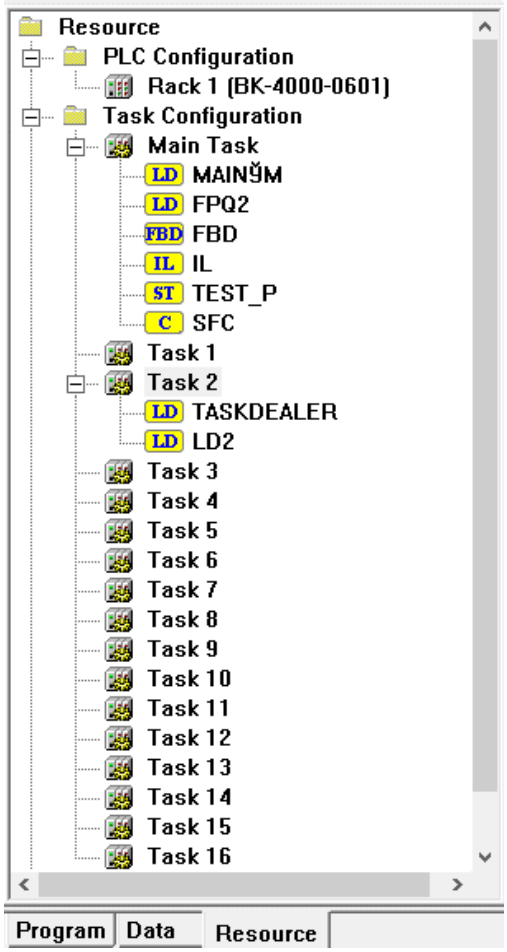


Рисунок 3.23 Список задач

Программы LD2 и TASKDEALER относятся к задаче 2, щелкните правой кнопкой мыши по задаче, чтобы выбрать **Configuration**, а затем откройте диалоговое окно **Task Configuration**, установите для параметра "Automatically Execute Program" (Автоматически выполнять программу) значение LD2 и установите "Cycle" (Цикл) равный 200 мс, как показано на рисунке 3.25. Временной цикл не может быть установлен менее, чем на 1 миллисекунду. Программа автоматического выполнения (LD2) начинает выполняться и вызывает другие программы (TASKDEALER), относящиеся к этой задаче.

Если цикл вызова программы Task1-Task16 менее общего цикла выполнения программ %SW32 "Scan Time", то программа Task1-Task16 будет вызвана в цикле %SW32, но не чаще!

Пример 1: Task1 сконфигурирована на цикл 2 мс, и программист сделал таймер на цикл 2 мс, но в результате Task1 выполняется после вызова Main Task, и время между его вызовами равно %SW32, в результате таймер сильно занижает свое значение и не достоверен.

Пример 2: Task1 сконфигурирована на цикл 200 мс, и программист сделал таймер на цикл 200 мс, но в результате Task1 выполняется после вызова Main Task, и время между его вызовами равно 200 ± %SW32 мс, в результате таймер недостоверен.

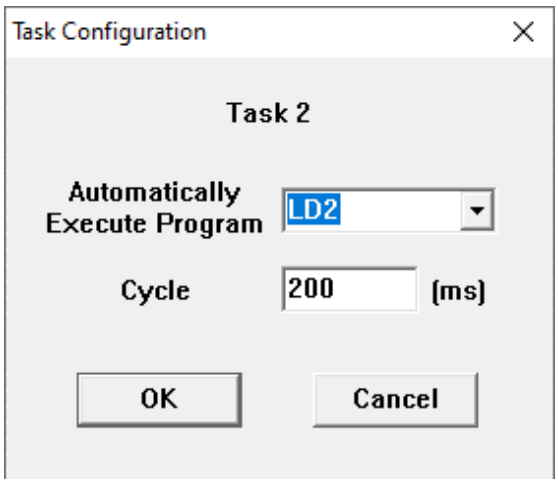


Рисунок 3.24 Конфигурация задач

3.5 Прерывание

Прерывание — это определённый тип реагирования ПЛК на внешние события или внутренние события и их обработка. Данный процесс включает в себя три части: событие прерывания, программу прерывания и управление прерыванием. Прерывание осуществляется в результате появления события прерывания, таковым может быть прерывание ввода-вывода или прерывание по таймеру. При возникновении события прерывания, ПЛК прерывает сканирование текущей программы, и право управления ПЛК переходит программе прерывания, запущенной этим событием прерывания. После выполнения программы прерывания право управления автоматически возвращается прерванной программе ПЛК, и она продолжает выполняться.

ПЛК СК-х000 поддерживают пять видов прерываний, а именно, прерывание по таймеру 1, прерывание по таймеру 2, прерывание по таймеру 3, прерывание по таймеру 4 и прерывание ввода-вывода. Уровень прерывания таймера 1 является самым низким, а уровень прерывания ввода-вывода - самым высоким. ПЛК обрабатывает события прерывания согласно уровням.

Программа прерывания настраивается при создании новой программы, и чем она короче, тем лучше.

Управление прерыванием осуществляется функциональными блоками **ENI** (Включение прерывания) и **DISI** (Отключение прерывания). При запуске ПЛК прерывания отключены. Прерывание осуществляется только когда программа прерывания включена, что позволяет вызвать её. Когда программа прерывания отключена, аппаратное обеспечение продолжает принимать ответ на прерывание, но вызывать программу прерывания оно не может.

Все прерывания и все программы каждого прерывания перечислены в разделе ***Interrupt Configuration*** вкладки 【**Resource**】 браузера проекта. В примере программа

прерывания таймера 1 (Timer INT 1) - это программа LD “TMR1”, программа прерывания таймера 2 (Timer INT 2) - программа FBD “TMR2”, программа прерывания таймера 3 ((Timer INT 3) - программа IL “TMR3”, программа прерывания таймера 4 (Timer INT 4) - программа ST “TMR4”, программа прерывания ввода-вывода (I/O INT)- - это программа LD “IO”, как показано на рисунке:

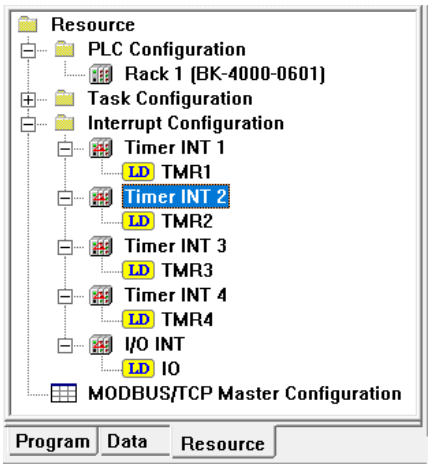


Рисунок 3.25 Список прерываний

Для срабатывания прерывания таймера 1...4 необходимо настроить время таймера. Щелкните правой кнопкой мыши на прерывании, которое необходимо настроить, чтобы выбрать ***Configuration***, затем появится диалоговое окно **Interrupt Configuration**, введите цикл, как показано на рисунке 3.27.

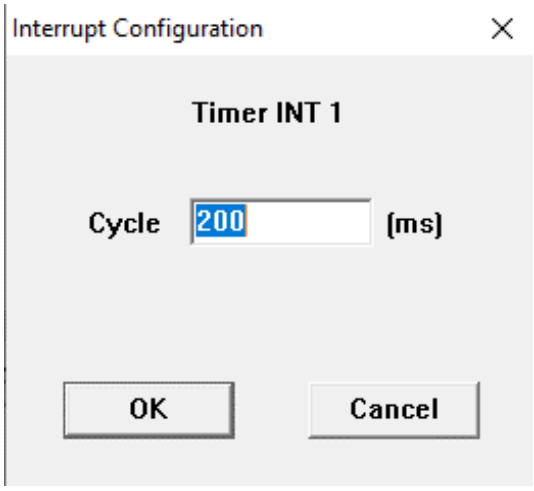


Рисунок 3.26 Конфигурация таймера прерывания

3.6 Защита проекта

Чтобы защитить конфигурацию проекта и программы от просмотра или изменения, для проекта может быть установлен пароль. Существует два уровня паролей: для файлов и для входа в систему. Пароль к файлу используется для открытия файла проекта, а пароль для входа в систему используется для подключения, загрузки и выгрузки. По умолчанию, файлу нового проекта пароль не назначается.

3.7 Подключение и отключение

Подключение означает, что «СКПро» подключено к ПЛК по сети Ethernet и обменивается с ним пакетами данных. Чтобы «СКПро» начало процесс подключения, щелкните на значок **【Connect】** на системной панели инструментов. После успешного подключения можно просмотреть статус выполнения программы, напрямую запросить статус каждого регистра, а также принудительно вызвать некоторые регистры при помощи программ. При неудачном подключении будет выдан сигнал о сбое подключения.

Отключение означает, что «СКПро» отключено от ПЛК, и в отключенном состоянии может быть изменена каждая составляющая проекта, а также могут быть загружены файлы проекта и программы.

3.8 Загрузка и выгрузка файла проекта

После создания или изменения файла проекта он должен быть загружен в ПЛК, в противном случае при подключении к ПЛК появится сообщение, показанное на рисунке 3.28:

Рисунок 3.27 Предупреждение о подключении

Загрузка означает, что отредактированные программы (файлы проекта и программы) выгружаются в ПЛК.

Выгрузка означает, что файлы проекта и программы ПЛК сохраняются на компьютере (они могут быть загружены только при отметке галочкой пункта “Allow to upload”.).

Download project (Загрузка проекта)

Загрузите файл проекта в ПЛК. При загрузке «СКПро» автоматически выполнит поиск узла в сети и продолжит загрузку в соответствии с адресом Ethernet в конфигурации оборудования. Во время процессов поиска и загрузки появится сообщение “Please wait while downloading project...” (“Пожалуйста, подождите пока идет выгрузка проекта...”). Если поиск узла ПЛК невозможен, об этом известит сигнал о сбое подключения. При работе с резервной системой, «СКПро» автоматически загрузится в оба ПЛК при наличии подключения к резервному ПЛК. После загрузки программ в ПЛК, модуль центрального процессора должен быть сброшен и перезапущен один раз, после чего выгруженные программы могут быть выполнены, в противном случае система выполнит программы, находившиеся в ПЛК до загрузки. Операция сброса может бытьвыполнена командой ***Reset***.

Manually download (Ручная загрузка)

Ручная загрузка часто используется для первой загрузки файла проекта и всех программных файлов. IP-адрес установлен по умолчанию, в случае если ПЛК не требуется загружать файл проекта. После выбора команды ***Manually Download*** откроется диалоговое окно, как показано на рисунке 3.29:

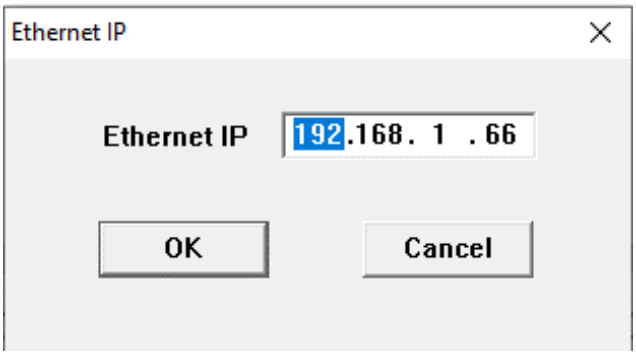


Рисунок 3.28 Ручная загрузка

Нажмите кнопку ***OK***, чтобы загрузить файлы проекта и программы, после чего система отобразит окно процесса, как показано на рисунке 3.30:

Рисунок 3.29 Процесс ручной загрузки

Если сетевое оборудование не подключено или адрес указан неверно, то «СКПро» откроет окно, как показано на рисунке 3.31:

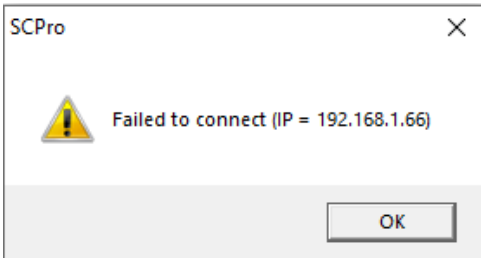


Рисунок 3.30 Сбой загрузки

В случае сбоя необходимо проверить сетевые конфигурации.

Примечание: IP-адрес компьютера для отладки должен находиться в том же адресном пространстве, что и IP-адрес ПЛК, то есть, первые три сегмента IP-адреса должны совпадать. В противном случае соединение установить не удастся. Например: если IP-адрес ПЛК - 192.168.1.66, то IP-адрес компьютера для отладки должен начинаться с 192.168.1.***. Более подробно – см. системное руководство по подключению к ПЛК.

Upload project (Выгрузка проекта)

Данная функция требуется для выгрузки проекта с ПЛК на компьютер для отладки. Отличие от загрузки заключается в том, что при выгрузке появляется диалоговое окно,

в котором будет предложено ввести IP-адрес Ethernet ПЛК, по которому требуется выгрузить файл. После ввода нажмите кнопку ***OK*** и «СКПро» выполнит поиск узла в сети в соответствии с адресом, как показано на рисунке 3.32:

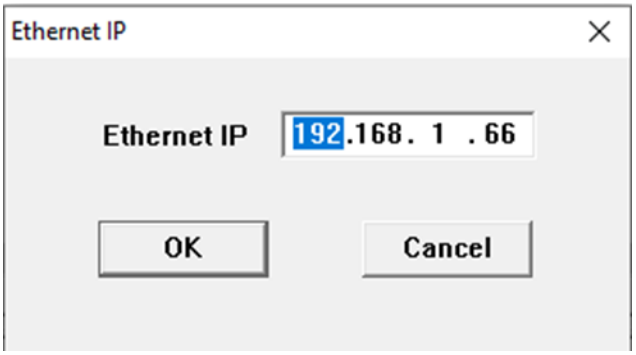


Рисунок 3.31 Выгрузка проекта

После выгрузки «СКПро» предложит ввести название для сохраненяемого проекта. Можно перезаписать файл проекта или другие файлы с теми же именами на текущем компьютере либо ввести новое имя для сохранения в новом файле.

3.9 Загрузка и выгрузка программ

Download program (Загрузка программы)

Если файл проекта не изменен, а изменен только файл программы, то файл программы в ПЛК может быть обновлен с помощью функции *Download Program*.

Upload program (Выгрузка программы)

Выгрузите все программные файлы с ПЛК, включая LD, FBD, ST, IL, SCC и т.д. Выгрузка программ аналогична выгрузке файла проекта, для этого также необходимо ввести IP-адрес ПЛК. После этого появится диалоговое окно, в котором необходимо выбрать название программы для выгрузки, также можно выбрать все программы, как показано на рисунке 3.33:

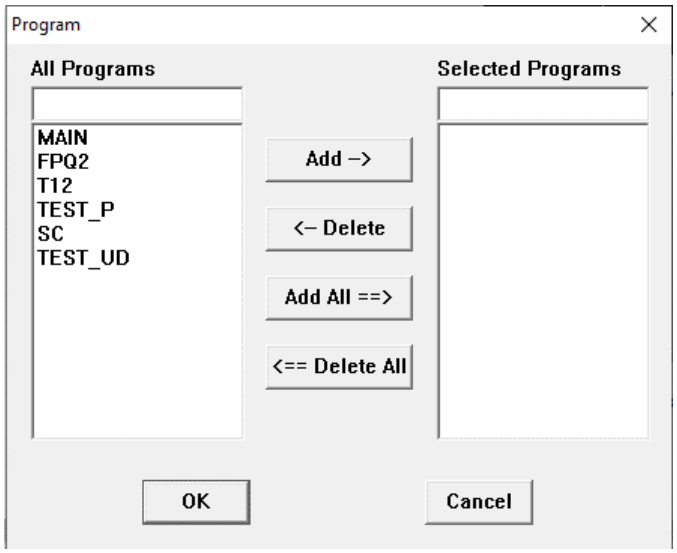


Рисунок 3.32 Выгрузка программы

Download All (Загрузить все)

Загрузка файла проекта и всех программных файлов в ПЛК одновременно.

Download All Programs (Загрузить все программы)

Загрузка всех программных файлов в ПЛК.

4 Управление данными

Стандарт IEC61131-3 определил наиболее часто используемые типы данных при программировании ПЛК, поэтому в отношении ПЛК эти типы данных определяются и применяются единообразно. Это упрощает конфигурирование и обслуживание систем, оснащенных ПЛК производства различных компаний, для производителей машин и установок и инженерно-технического персонала: унифицированные типы данных повышают читаемость и портируемость программ ПЛК.

4.1 Тип данных

ПЛК СК-4000 работает со следующими типами данных:

Ключевое слово	Тип	Бит	Допустимый диапазон	Описание
BOOL	Boolean	1	0 или 1	Хранится в единицах бита только в двух состояниях: 1 или 0.
BYTE	Byte	8	0~255	Используется 8 бит регистра данных, 8 бит данных могут быть независимыми и указывать только на состояние текущего бита: 0 или 1; также они могут быть целым числом без знака в диапазоне от 0 до 255.
WORD	Word	16	0~65535	Используется 16 бит регистра данных, 16 бит данных могут быть независимыми и указывать только на состояние текущего бита: 0 или 1; также они могут быть целым числом без знака в диапазоне 0~65535.
DWORD	Double word	32	0~4294967295	Используется 32 бита регистра данных, 32 бита данных могут быть независимыми и указывать только на состояние текущего бита: 0 или 1; также они могут быть целым числом без знака в диапазоне 0~4294967295.
SINT	Short integer	8	-128~+127	Используется 8 бит регистра данных, указывается целое число со знаком в диапазоне -128~+127.
INT	Integer	16	-32768~+32767	Используется 16 бит регистра данных, указывается целое число со знаком в диапазоне -32768~+32767.
DINT	Double integer	32	-2147483648~+2147483647	Используется 32 бита регистра данных, указывается целое число со знаком в диапазоне -2147483648 ~+2147483647.
REAL	Real	32	——	Указывается значение с плавающей запятой.

USINT	Unsigned short integer	8	0~255	Используется 8 бит регистра данных, указывается целое число без знака в диапазоне 0~255.
UINT	Unsigned integer	16	0~65535	Используется 16 бит регистра данных, указывается целое число без знака в диапазоне 0~65535.
UDINT	Unsigned double integer	32	0~4294967295	Используется 32 бита регистра данных, указывается целое число без знака в диапазоне 0~4294967295.

4.2 Управление данными

4.2.1 Вкладка "Data"

В браузере проекта вкладка 【**Data**】 включает в себя следующие элементы: **Point Table**, **Variable Table**, **Optional Point Table** и т.д., как показано на рис.4.1:

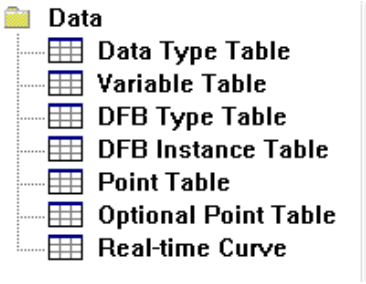


Рис.4.1 Вкладка "Data"

4.2.2 Point table (Таблица точек данных)

Тип точки

Тип	Наименование	Тип данных	Описание
I	Digital input (Дискретный вход)	BOOL, 0 или 1	Текущее состояние базовой точки дискретного входа.
Q	Digital output (Дискретный выход)	BOOL, 0 или 1	Текущее состояние базовой точки дискретного выхода.
IW	Analog input (Аналоговый вход)	INT, напряжение и ток: 0~20000 Температурный сигнал: -32768~+32767	Текущее значение базовой точки аналогового входа.

QW	Analog output (Аналоговый выход)	WORD, 0~20000	Текущее значение базовой точки аналогового выхода.
M	Bit register (Битовый регистр)	BOOL, 0 или 1	Логическая переменная область хранения, предлагаемая пользователю системой.
MW	Word register (Регистр слов)	WORD, 0~65535	Область хранения переменной Word, предлагаемая пользователю системой.
N	Non-volatile bit register (Энерго-независи-мый битовый регистр)	BOOL, 0 или 1	Отличие от регистра M заключается в энергонезависимости. После отключения питания ПЛК данные в регистре N не будут утеряны.
NW	Non-volatile word register (Энерго-независи-мый регистр слов)	WORD, 0~65535	Отличие от регистра MW заключается в энергонезависимости. После отключения питания ПЛК данные в регистре NW не будут утеряны.
S	System bit register (Регистр системных бит)	BOOL, 0 или 1	Логическая переменная, отражающая текущее состояние системы, определенное внутри системы. Каждый элемент имеет свое конкретное определение. Может быть прочитан, но не может быть переписан.
SW	System word register (Регистр системных слов)	WORD, 0~65535	Переменная в формате word, отражающая текущее состояние системы, определенное внутри системы. Каждый элемент имеет свое конкретное определение. Может быть прочитан, но не может быть переписан.

T	Timer (Таймер)	DWORD, 0~60000	Таймер, предлагаемый пользователю системой.
C	Counter (Счетчик)	DWORD, 0~4294967295	Счетчик, предлагаемый пользователю системой.

ПЛК СК-4000 предоставляет большое количество системных регистров для хранения состояния работы системы. Пользователю предоставляется удобный интерфейс для считывания и контроля состояния работы ПЛК в режиме реального времени.

Определения системного регистра:

Номер	Наименование	Описание
System bit register		
%S0001	FIRST_SCAN	Первое сканирование
%S0002	ALWAYS_ON	Всегда включен
%S0003	ALWAYS_OFF	Всегда выключен
%S0004	T_SECOND	Таймер на 1 секунду
%S0006	PRG_OVERRUN	Переполнение выполнения программы
%S0007	PRG_EXECERR	Ошибка выполнения программы
%S0010	BACKUP_OK	Резервное копирование устройства резервированной пары ЦПУ прошло успешно
%S0033	IO_COMERR	Ошибка связи модуля ввода-вывода
%S0034	IO_DIAGERR	Ошибка самодиагностики модуля ввода-вывода
%S0035	IO_CFGERR	Несоответствие типа модуля ввода-вывода
%S0036	COM_COMERR	Ошибка связи модуля связи
%S0037	COM_DIAGERR	Ошибка самодиагностики модуля связи
%S0038	COM_CFGERR	Несоответствие типа модуля связи
%S0039	VER_DISMATCH	Несоответствие версий ПО резервированной пары ЦПУ
%S0041	IO_DOWNLOAD	Инициализация модуля ввода-вывода прошла успешно

%S0042	COM_DOWNLOAD	Инициализация модуля связи прошла успешно
Состояние работы процессора		
%S0097	CPU1_MASTER	Мастер CPU1
%S0098	CPU1_FAULT	Ошибка CPU1
%S0099	CPU1_GPSLOST	Ошибка GPS CPU1
%S0100	CPU1_SECOND	Запуск CPU1
%S0101	CPU1_CAN1FLT	Ошибка CAN1 CPU1
%S0102	CPU1_CAN2FLT	Ошибка CAN2 CPU1
%S0103	CPU1_ETH1FLT	Ошибка ETH1 CPU1
%S0104	CPU1_ETH2FLT	Ошибка ETH2 CPU1
%S0105	CPU1_TASKFLT	Ошибка задачи CPU1
%S0106	CPU1_SELFON	CPU1 онлайн
%S0107	CPU1_FATAL	Фатальная ошибка CPU1
%S0114	CPU1_PEERON	CPU1 одноранговый узел онлайн
%S0115	CPU1_PEERMST	Одноранговый мастер CPU1
%S0116	CPU1_STOP	Состояние остановки CPU1
%S0117	CPU1_DEBUG	Состояние отладки CPU1
%S0118	CPU1_NVRAMFLT	Ошибка NVRAM CPU1
%S0120	CPU1_FIRST	CPU1 первый
%S0121	CPU2_MASTER	Мастер CPU2
%S0122	CPU2_FAULT	Ошибка CPU2
%S0123	CPU2_GPSLOST	Ошибка GPS CPU2
%S0124	CPU2_SECOND	Запуск CPU2
%S0125	CPU2_CAN1FLT	Ошибка CAN1 CPU2
%S0126	CPU2_CAN2FLT	Ошибка CAN2 CPU2
%S0127	CPU2_ETH1FLT	Ошибка ETH1 CPU2
%S0128	CPU2_ETH2FLT	Ошибка ETH2 CPU2
%S0129	CPU2_TASKFLT	Ошибка задачи CPU2
%S0130	CPU2_SELFON	CPU2 онлайн

%S0131	CPU2_FATAL	Фатальная ошибка CPU2
%S0138	CPU2_PEERON	CPU2 одноранговый узел онлайн
%S0139	CPU2_PEERMST	Одноранговый мастер CPU2
%S0140	CPU2_STOP	Состояние остановки CPU2
%S0141	CPU2_DEBUG	Состояние отладки CPU2
%S0142	CPU2_NVRAMFLT	Ошибка NVRAM CPU2
%S0144	CPU2_FIRST	CPU2 первый
Состояние основной связи MODBUS/TCP (1: Ошибка, 0: OK)		
%S0145		Запрос 1
……		……
%S0208		Запрос 64
……		……
%S0399		Запрос 255
Состояние работы модуля		
%S0513	MDU001_COMERR	Ошибка связи модуля (адрес 001)
%S0514	MDU001_DIAGERR	Ошибка самодиагностики модуля (адрес 001)
%S0515	MDU001_CFGERR	Несоответствие типа модуля (адрес 001)
%S0516	MDU001_RVS1	Резерв 1 модуля (адрес 001)
%S0517	MDU001_RVS2	Резерв 2 модуля (адрес 001)
%S0518	MDU001_RVS3	Резерв 3 модуля (адрес 001)
%S0519	MDU001_RVS4	Резерв 4 модуля (адрес 001)
%S0520	MDU001_RVS5	Резерв 5 модуля (адрес 001)
%S0521	MDU002_COMERR	Ошибка связи модуля (адрес 002)
%S0522	MDU002_DIAGERR	Ошибка самодиагностики модуля (адрес 002)
%S0523	MDU002_CFGERR	Несоответствие типа модуля (адрес 002)
%S0524	MDU002_RVS1	Резерв 1 модуля (адрес 002)
%S0525	MDU002_RVS2	Резерв 2 модуля (адрес 002)
%S0526	MDU002_RVS3	Резерв 3 модуля (адрес 002)

%S0527	MDU002_RVS4	Резерв 4 модуля (адрес 002)
%S0528	MDU002_RVS5	Резерв 5 модуля (адрес 002)
……	……	……
……	……	……
System word register		
%SW0001	TIME_YEAR	Часы: год
%SW0002	TIME_MONTH	Часы: месяц
%SW0003	TIME_DAY	Часы: день
%SW0004	TIME_HOUR	Часы: час
%SW0005	TIME_MINUTE	Часы: минута
%SW0006	TIME_SECOND	Часы: секунда
%SW0007	TIME_MS	Часы: миллисекунда
%SW0008	TIME_Week	Часы: День недели
%SW0009	ALARM_PTR	Указатель тревоги
%SW0010	SOE_PTR	Указатель SOE
%SW0011	OVERRUN_INFO	Место переполнения программы
%SW0014	EXERERR_INFO	Место ошибки программы
%SW0021	COM1_SEND	Состояние отправки COM1
%SW0022	COM1_RECV	Состояние приема COM1
%SW0023	COM2_SEND	Состояние отправки COM2
%SW0024	COM2_RECV	Состояние приема COM2
%SW0025	STTM_YEAR	Время начала: год
%SW0026	STTM _MONTH	Время начала: месяц
%SW0027	STTM _DAY	Время начала: день
%SW0028	STTM _HOUR	Время начала: час
%SW0029	STTM _MINUTE	Время начала: минута
%SW0030	STTM _SECOND	Время начала: секунда
%SW0031	STTM _MS	Время начала: миллисекунда
%SW0032	SCAN_TIME	Время сканирования

Свойство точки данных

Все точки данных имеют два общих вида свойства, и особые свойства, которыми обладают только некоторые точки.

Общие свойства

【Number】: Номер используется для того, чтобы различать точки данных. Он генерируется автоматически и не может быть изменен. Номер может быть непосредственно использован в качестве объекта операции блока функции или инструкции. Номер состоит из двух частей: тип точки и номер точки, например %I0001, %MW0100 и т.д. "Тип точки" указывает текущий тип точки данных, например, "%I" указывает точку дискретного входа, "%MW" указывает регистр слов. "Номер точки" указывает текущий последовательный номер точки, который не может превышать своего максимального значения. Для разных типов точек и разных типов процессоров максимальное значение отличается.

Тип	Наименование	CP-4000-0221	CP-4000-0331	CP-4000-0431	CP-4000-0531	CP-4000-0631
I	Дискретные входы	1024	4096	8192	16384	16384
Q	Дискретные выходы	1024	4096	8192	16384	16384
IW	Аналоговые входы	256	1024	2048	4096	4096
QW	Аналоговые выходы	256	1024	2048	4096	4096
M	Битовые регистры	4096	8192	16384	16384	16384
MW	Регистры слова	4096	16384	32768	32768	32768
N	Энергонезависимые битовые регистры - реманентные	1024	2048	4096	4096	4096
NW	Энергонезависимые регистры слова - реманентные	1024	2048	4096	4096	4096
S	Системные битовые регистры	1024	2048	4096	4096	4096
SW	Системные регистры слова	1024	2048	4096	4096	4096
T	Таймеры	256	512	1024	1024	1024
C	Счетчики	256	512	1024	1024	1024

Каждой точке данных может быть присвоено имя, которое может быть непосредственно использовано в качестве объекта операции при программировании LD и SCC и т.д., а также может быть отображено непосредственно в программе.

Например, точке "%I0001" присвоено имя “DL_ON”. В области редактирования LD поместите контакт и введите параметр “DL_ON”, после компиляции «СКПро» автоматически его идентифицирует. Если имя точки изменено, то точка “DL_ON” заменяется на вторую точку цифрового ввода, нужно только присвоить точке %I0002 имя “DL_ON”, и никаких других изменений в программе не потребуется.

【Description】: Каждой точке данных также может быть присвоено описание, которое должно подробно описывать точку данных. Например, имя точки данных %I0001 - “DL_ON”, описание – «автоматический выключатель включен». Если при чтении программы LD вы не поймете значения точки “DL_ON” из ее названия, можете ознакомиться с описанием. Однако содержимое описания не отображается в LD.

【Used Times】: Эта функция отображает сколько раз текущая точка данных была применена с прямой адресацией в программах, данная информация доступна в **【File】** / **【Used Times】**, после сохранения нет необходимости пересчитывать в следующий раз. **Количество применения точки особенно полезно для таймера и счетчика: его можно просмотреть, чтобы избежать повторного применения.**

Особые свойства

【Module Address】: Адрес модуля действителен только для физических точек данных, таких как дискретный вход I, дискретный выход Q, аналоговый вход IW и аналоговый выход QW. Адрес модуля – это адрес модуля, в котором находится текущая точка. Он генерируется автоматически после того, как задана точка. Адрес не может быть изменен. Адрес модуля в стойке 1 - от 1 до 15, адрес модуля в стойке 2 - от 16 до 30 и т.д. Например, если модуль, на котором расположена текущая точка, установлен в третий слот второй стойки, то адрес модуля будет равен 18.

【Filter Time】: Время фильтрации применимо только к дискретному входу I. Каждому входящему цифровому сигналу может быть добавлено определенное время фильтрации для предотвращения дребезга сигналов. Единица измерения времени фильтрации - 10 мс, она может быть изменена в диапазоне 0 ~ 250. Как правило, время фильтрации по умолчанию составляет 0 мс. Если введённое значение не входит в указанный диапазон, «СКПро» отобразит это в соответствующем окне, как показано на рис.4.2:

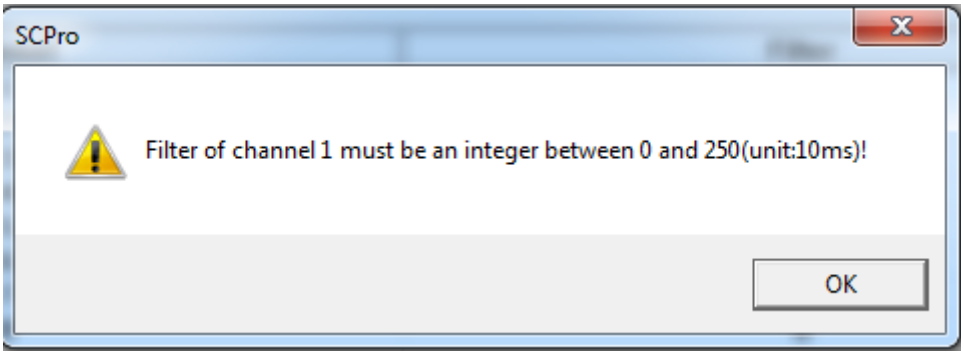


Рис.4.2 Время фильтрации

【**Force**】: Принудительный ввод (форсирование) действителен только для физических точек данных, таких как дискретный вход I, дискретный выход Q, аналоговый вход IW и аналоговый выход QW. При помощи форсирования точки, вы можете установить значение точки в соответствии с вашими требованиями, независимо от текущего значения сигнала. Принудительный ввод может быть выполнен только в режиме онлайн работы ПЛК. Дважды щелкните на панели форсирования точки, для которой необходимо установить значение. На панели появится отметка √, что говорит о том, что эта точка была форсирована. Дважды щелкните еще раз, и метка пропадет. Это означает, что эта точка не форсирована. В состоянии принудительного ввода состояние сигнала, отсканированного системой, никогда не будет отправлено в область хранения точек данных.

【**Value**】: Функция значения показывает текущие значения фактических точек данных в оперативном состоянии для дискретного входа I, дискретный выхода Q, аналогового входа IW и аналоговый выхода QW; а также показывает текущие значения виртуальных точек данных в текущем состоянии для битового регистра M, регистра слов MW, энергонезависимого битового регистра N, энергонезависимого регистра слов NW, системного битового регистра S и системного регистра слов SW. Для системного регистра S и SW оно может быть только прочитано и не может быть записано.

【**Signal Type**】: Функция "Тип сигнала" действительна только для аналогового входа IW и аналогового выхода QW. Аналоговый модуль может иметь различные типы сигналов, такие как тип "Напряжение", "Ток" или "Термосопротивление".

【**Preset Value**】: Предустановленное значение действительно только для таймера T и счетчика C (предустановленное значение в настройках аналоговых выходных каналов можно посмотреть в настройках соответствующего аналогового выходного модуля). В режиме онлайн можно просмотреть заданное время таймера или заданное значение счетчика.

【**Current Value**】: Текущее значение действительно только для таймера T и счетчика C. В режиме онлайн можно просмотреть текущее время таймера или текущее значение счетчика.

【**Dimension**】: Размерность применима только для переменной %V, доступной в Variable Table. Размерность переменной составляет не более 4096. Индекс переменной начинается с 0, например, имя %V0001 определяется как “status”, его размерность равна 100, тогда можно использовать ”status[0] ~status[99]". Если это одномерная переменная, нужно только ввести имя переменной без нижнего индекса при вводе, например, %V0001 (имя “status”), размерность одна, при использовании нужно только ввести “status” вместо “status[0]”. Имя переменной должно быть определено при использовании переменной, например, имя %V0001 - “status”, тогда вместо “%V[0]” следует использовать “status”.

【**Type**】: Тип действителен только для переменной %V. Выберите тип данных переменной с помощью выпадающего списка. Типом данных могут быть все типы в ПЛК, включая BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT и UDINT и т.д. (Для получения подробной информации о типе данных, пожалуйста, см. часть «Тип данных» этой главы), однако каждый массив переменных может быть определен только как данные одного и того же типа.

【**Zero Offset**】: Смещение нуля применимо только к аналоговому входу IW и используется для компенсации ошибок фактического сигнала.

4.2.3 Таблица переменных

В «СКПро» существует своего рода виртуальная точка данных, которая называется переменной. Переменная хранит данные в виде массива. Размерность массива и тип данных могут быть установлены в допустимом диапазоне. В режиме онлайн текущее значение данных можно просмотреть с помощью таблицы переменных, как показано на рисунке 4.3. Для получения подробной информации о просмотре и изменении переменной см. главу 11.6.

Name	Input	Data Type	Dimension	Value	Comment
HSC_CPU	HSC_CPU	HSC_DATA[]	2		
PTO_CPU	PTO_CPU	PTO_DATA[]	4		
MOT_CPU	MOT_CPU	MOT_DATA	1		
DI_1	DI_1	BOOL	1		
SetPoint	SetPoint	REAL	1		
KP	KP	REAL	1		
TI	TI	REAL	1		
TD	TD	REAL	1		
Sample_time	Sample_time	REAL	1		
Ymax	Ymax	REAL	1		
Ymin	Ymin	REAL	1		
DeadBand	DeadBand	REAL	1		
Man	Man	BOOL	1		
Yman	Yman	REAL	1		
Y	Y	REAL	1		
Status	Status	INT	1		
tEMP_INT	tEMP_INT	DINT	1		
wq	wq	REAL	1		
i	i	INT	1		
j	j	INT	1		
connect	connect	ETH_PARAM	1		
connect_status	connect_status	INT	1		

Variable Table

Рисунок 4.3 Таблица переменных

4.2.4 Таблица дополнительных точек

В режиме онлайн просмотреть / изменить значение точки можно с помощью таблицы дополнительных точек. Таблица дополнительных точек позволяет свободно определять желаемые точки, причем таким образом, чтобы при отладке программы можно было наблюдать только за желаемыми точками. В браузере проекта выберите 【**Data**】/ 【**Optional Point Table**】, откроется таблица дополнительных точек. Если она применяется в первый раз, то эта таблица точек будет пустой, как показано на рисунке 4.4:

Enter	Value	Address	Name	Force	Type	Comment	No.

Рисунок 4.4 Пустая таблица дополнительных точек

Добавление точки

Щелкните правой кнопкой мыши на пустом месте, чтобы выбрать команду **Add**, откроется диалоговое окно **Point Name**, в котором нужно вести название точки, как показано на рисунке 4.5:

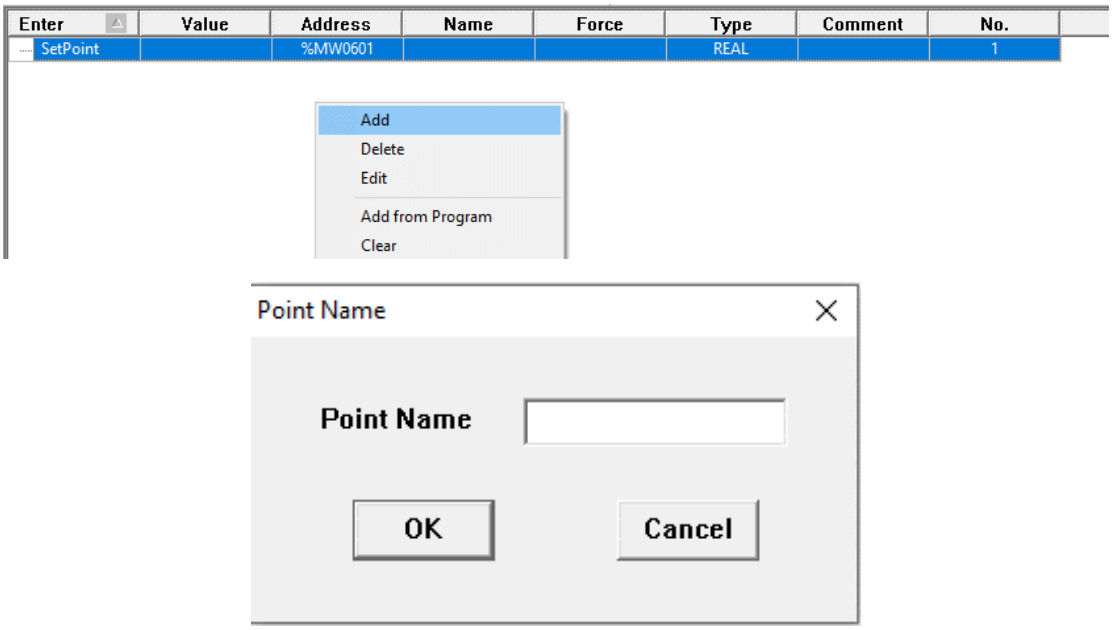


Рисунок 4.5 Добавление точки

Добавление из программы

Щелкните правой кнопкой мыши в таблице дополнительных точек, чтобы выбрать команду ***Add from Program***, откроется диалоговое окно ***Select Program***, выберите название добавляемой программы и нажмите ***OK***, затем все точки и переменные, используемые в этой программе, будут добавлены в открытую таблицу дополнительных точек, как показано на рисунке 4.6:

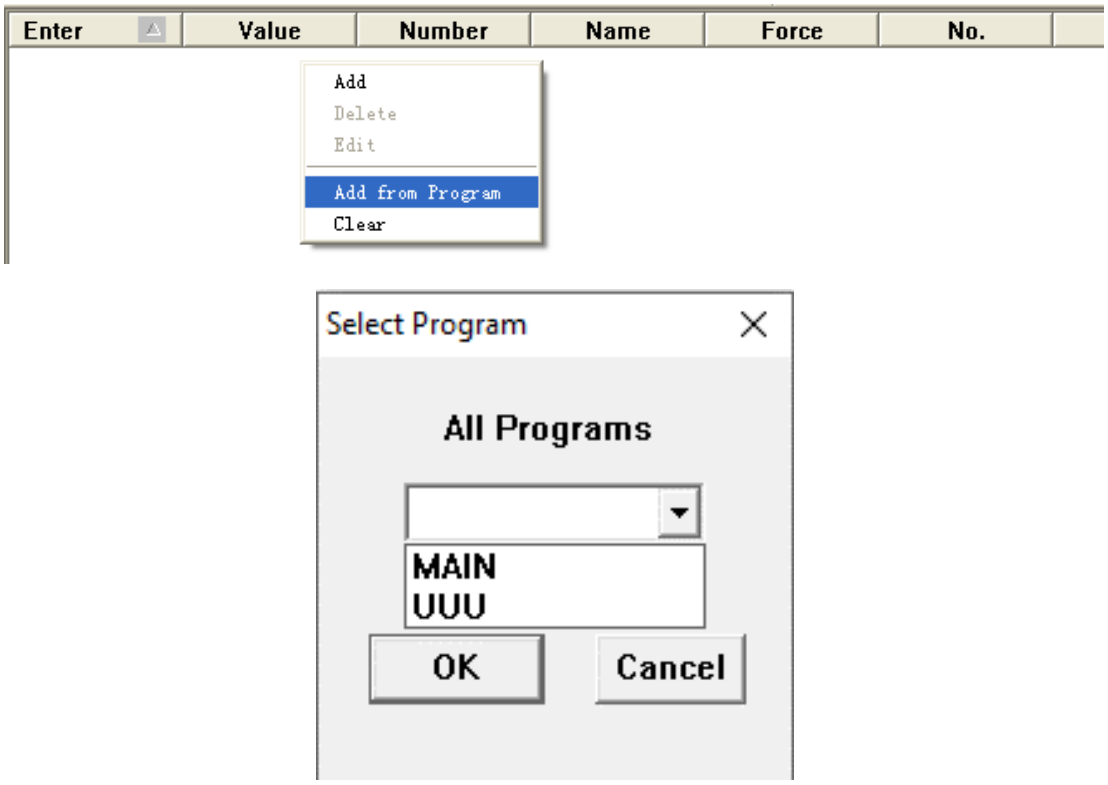


Рисунок 4.6 Добавление точки из программы

После добавления точек, информацию о значении точки, количестве, названии, принудительном вводе и т.д. можно просмотреть в таблице дополнительных точек. Как показано на рис.4.7:

Enter	Value	Address	Name	Force	Type	Comment	No.
%M3003		%M3003	Valve_1_OpOpen		BOOL	Оператор - Отк...	1
%M3004		%M3004	Valve_1_OpClose		BOOL	Оператор - зак...	2
%M3008		%M3008	Valve_1_OpReset		BOOL	Сброс ошибки	3
%MW5000		%MW5000			WORD		30
%T0001		%T0001			WORD		31
avr_work		%M6004	avr_work		BOOL		29
OK2024_1_4...		%MW20111	OK2024_1_40003...		WORD	Включение/вы...	6
valve_1_ops...		%M3005	Valve_1_OpStop		BOOL	оператор - стоп	12
Vlv[0].In.FB...		%V9110			BOOL	Закрыта	10
Vlv[0].In.FB...		%V9112			BOOL		8
Vlv[0].In.FB...		%V9109			BOOL	Открыта	9
Vlv[0].In.FB...		%V9111			BOOL		7
Vlv[0].InOut...		%V9160			BOOL	оператор - стоп	11
Vlv[0].InOut...		%V9172			BOOL		17
Vlv[0].InOut...		%V9181			BOOL		23
Vlv[0].InOut...		%V9178			BOOL		20
Vlv[0].InOut...		%V9185			BOOL		27
Vlv[0].InOut...		%V9184			BOOL		26
Vlv[0].InOut...		%V9183			BOOL		25
Vlv[0].InOut...		%V9186			BOOL		28
Vlv[0].InOut...		%V9171			BOOL	Внутренняя ко...	16
Vlv[0].InOut...		%V9180			BOOL		22
Vlv[0].InOut...		%V9177			BOOL		19

Optional Point Table

Рисунок 4.7 Таблица дополнительных точек

Значение принудительного ввода точки

Используется для изменения значения точки в таблице дополнительных точек. Для точек %M, %MW, %N, %NM, %Q, %QW и переменных, значение точки можно изменить, дважды щелкнув значение точки, которое необходимо изменить в столбце ***Value***; Для точек %I и %IW значения точек могут быть изменены в столбце ***Value*** только после того, как опция принудительного ввода будет отмечена двойным щелчком по столбцу ***Force***.

Удаление и редактирование точки

Если некоторые точки не нужны, их можно удалить или изменить, чтобы они стали другими. Операцию можно выбрать, щелкнув правой кнопкой мыши по соответствующему столбцу, как показано на рисунке 4.8:

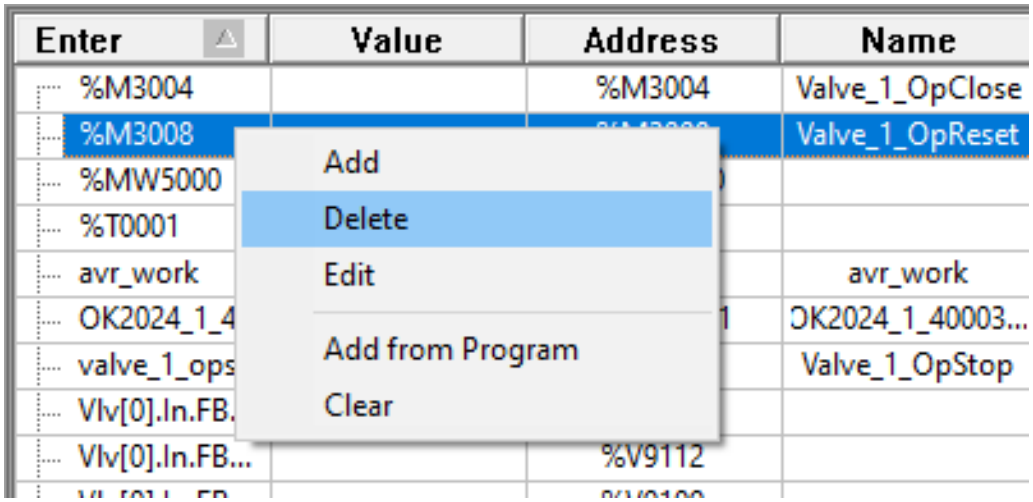


Рис.4.8 Удаление и редактирование точки

Добавление таблицы дополнительных точек

Пользователи могут создать не более 16 листов таблицы дополнительных точек. На вкладке 【**Data**】 браузера проекта щелкните правой кнопкой мыши, чтобы выбрать команду ***Add Optional Point Table***, как показано на рисунке 4.9:

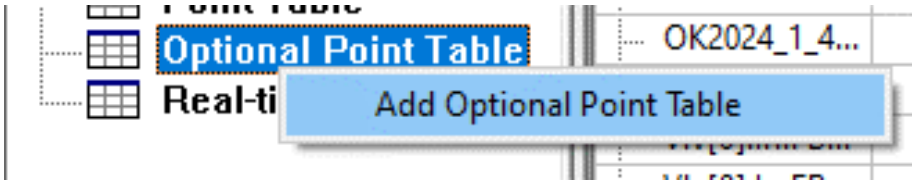


Рисунок 4.9 Добавление таблицы дополнительных точек

Удаление таблицы дополнительных точек

Если вы не хотите сохранять новую таблицу точек, а хотите удалить ее, вам нужно очистить таблицу точек, затем закрыть окно. Система автоматически удалит эту таблицу дополнительных точек, как показано на рисунке 4.10:

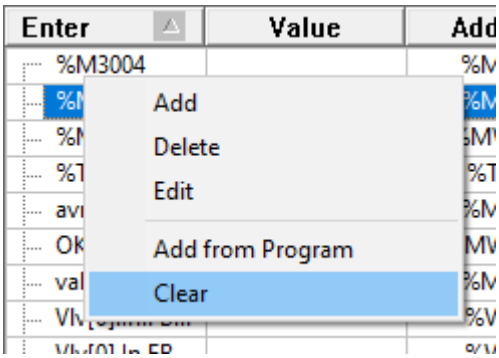


Рисунок 4.10 Очистка точки

Таблица дополнительных точек сохраняется на текущей инженерной станции для отладки, а затем при следующем подключении, таблица дополнительных точек сохранит статус, существовавший до последнего выхода. Таблица дополнительных точек никогда не загружается в ПЛК.

4.3 Режим адресации

Адресация — это метод доступа к точкам данных для языка LD и других языков программирования. Существует несколько типов режимов адресации:

Константа

Константа — это способ адресации неизменямых данных. Для следующего функционального блока функция выполнения **MOVE,** заключается в присвоении значения 100 точке %MW0001, в таком случае ввод значения 100 является режимом ввода константы.

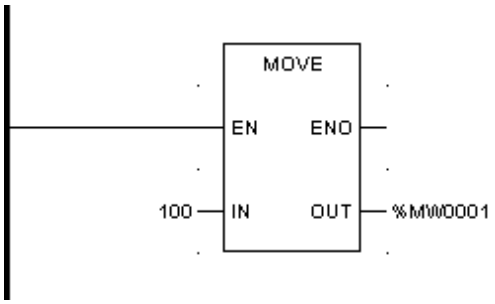


Рисунок 4.11 Константа

Указание

В режиме ввода константы, при вводе данных после шестнадцатеричных данных должна быть буква “H”, если первое число шестнадцатеричных данных является буквой, то перед буквой добавляется “0”. Например, если вы хотите ввести B021H, то записать необходимо 0B021H, в противном случае, при компиляции системы, возникает ошибка.

Прямая адресация

Тип точки, добавляющий номер точки, является объектом доступа прямой адресации. Как показано на рис.4.11, для операции **MOVE** %MW0001 является указанной точкой, это режим прямой адресации.

В качестве особого примера, использование имени точки также относится к прямой адресации, поскольку имя точки соответствует указанной точке. Например, присвойте точке %I0001 имя “GD_ON”, в таком случае имя “GD_ON” может быть использован непосредственно в программе, так как оно ссылается на %I0001.

Косвенная адресация

Номер точки объекта доступа для косвенной адресации является не константой, а другой точкой, то есть в качестве номера точки используется текущее значение этой точки. Отличие от номера точки прямой адресации в следующем: точка косвенной адресации начинается с 0, то есть, в случае косвенной адресации точки %I[%MW0001] текущее значение %MW0001 равно 1, а точка доступа - %I0002, вместо %I0001. Как показано на рис.4.12, %Q[%MW0001] — это режим косвенной адресации. Тип точки - Q, номер точки не является константой, а другой точкой - %MW0001. Операция выполнения LD такова: %MW0001 присваивается как 0, катушка выдает 1 для выходной точки %Q0001.

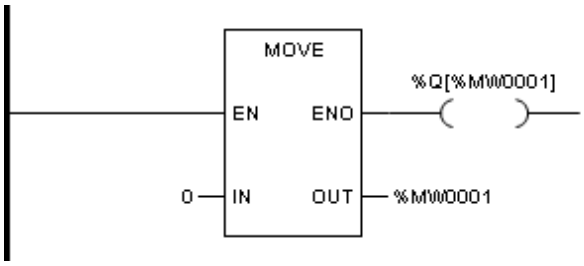


Рисунок 4.12 Косвенная адресация

5 Базовые блоки функций

В данном разделе рассматриваются базовые блоки функций, применимые для всех языков программирования.

5.1 Введение

Для графических языков (LD и FBD) блок функций (FB) обозначается как рамка с контактами ввода и вывода. Ввод всегда находится слева от рамки, а вывод всегда справа. Название блока функций (то есть тип блока функций) отображается в верхней части рамки, посередине.

В данной главе рассматриваются базовые блоки функций: список инструкций (IL), структурированный текст (ST), релейно-контактные схемы (LD), функциональные блоковые диаграммы (FBD).

5.1.1 Изменение свойств

Свойства каждого блока функций могут быть изменены. В FBD “EN/ENO” может быть отображен/скрыт. Для некоторых блоков функций количество вводов может быть увеличено.

Как показано на рис.5.1, в редакторе LD необходимо щелкнуть левой кнопкой мыши по функциональному блоку, свойство которого требуется просмотреть (блок функции **ADD** на рис.), и выбрать **Property**, щелкнув правой кнопкой мыши; откроется диалоговое окно, как показано на рис.5.2, в выпадающем списке выберите “Input Number” (количество вводов). В редакторе FBD параметр “Display EN/ENO” (Отображать EN/ENO) может быть изменен в качестве альтернативы.

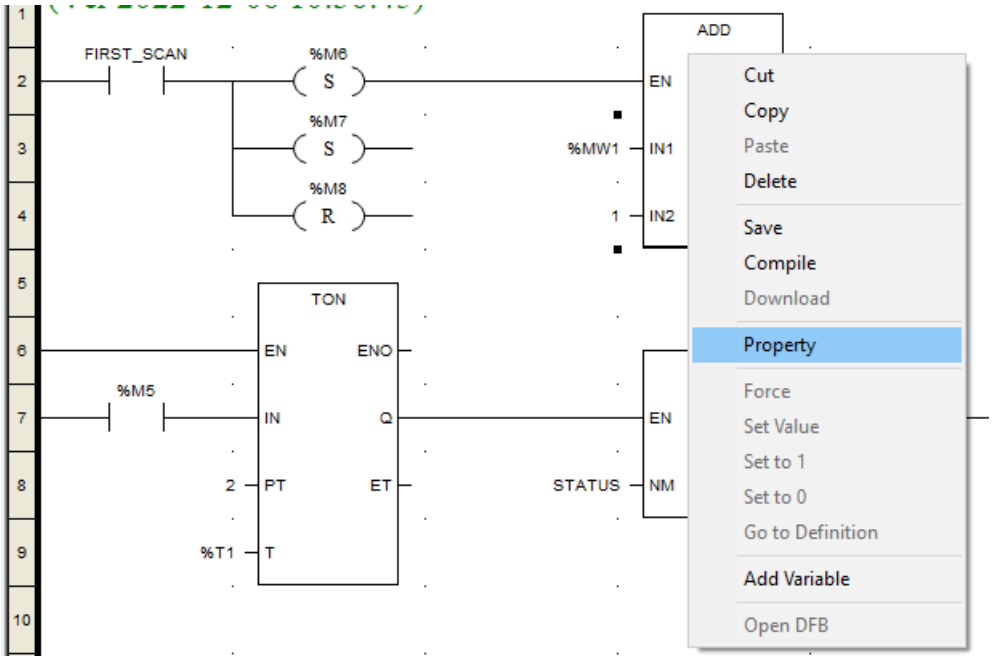


Рис.5.1 Изменение свойства FB

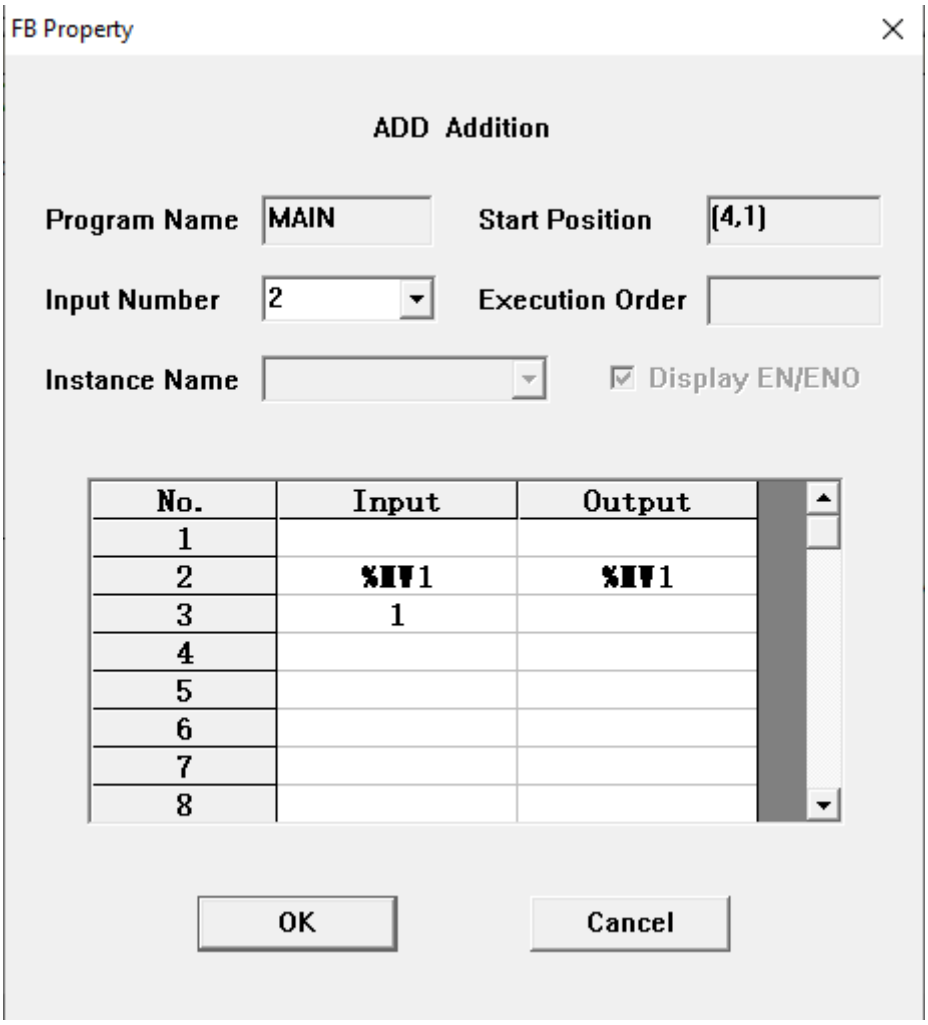


Рис.5.2 Диалоговое окно изменения свойства

5.1.2 EN/ENO

Функция настраивает ввод **EN** и вывод **ENO** для всех блоков функций. Если ввод **EN** равен 0 при вызове блока функции, то алгоритм, определенный блоком функции, не будет выполнен, вывод **ENO** в таком случае будет равен 0. Если при вызове блока функции ввод **EN** равен 1, то выполняется алгоритм, определенный блоком, и **ENO** равен 1. Если во время выполнения возникает какая-либо ошибка, **ENO** будет равен 0.

5.2 Математические блоки функций

В этом разделе описываются базовые блоки функций, связанные с выполнением базовых математических операций.

Данный раздел содержит следующие подразделы.

Тип	Описание
ADD	Прибавление: OUT = IN1 + IN2 + ... + INn
SUB	Вычитание: OUT = IN1 - IN2
MUL	Умножение: OUT = IN1 * IN2 * ... * INn
DIV	Деление: OUT = IN1 / IN2
MOD	Деление по модулю: OUT = IN1 % IN2
DIVMOD	Деление и Деление по модулю: DV = IN1 / IN2 MD = IN1 % IN2
INC	Инкремент: INOUT = INOUT + 1
DEC	Декремент: INOUT = INOUT - 1
NEG	Отрицание: OUT = 0 - IN
SIGN	Оценка знака: если IN < 0, OUT = 1 если IN ≥ 0, OUT = 0
SQRT	Квадратный корень: OUT = $\sqrt{IN}$
ABS	Абсолютное значение: OUT = | IN |
LOG	Десятичный логарифм: OUT = $\log_{10}^{IN}$
LN	Натуральный логарифм: OUT = ln ^{IN}, или OUT = $\log_{e}^{IN}$, где e = 2,718282
EXP	Натуральное возведение в степень: OUT = e ^{IN}, где e = 2,718282
EXPT	Возведение в степень: OUT = IN1 ^{IN2}
SIN	Синус: OUT = sin IN
COS	Косинус: OUT = cos IN
TAN	Тангенс: OUT = tan IN
ASIN	Синус дуги: OUT = asin IN

ACOS	Косинус дуги: OUT = acos IN
ATAN	Тангенс дуги: OUT = atan IN

При переполнении максимального/минимального значения типа данных на выходе, будет выведено значение, уменьшенное/увеличенное на максимальное/минимальное значение типа данных, будьте внимательны при выборе типов данных переменных!

При использовании математических функций для корректной работы, рекомендуем в качестве переменных использовать переменные таблицы Variable table. Например:

```
1   output1 := EXPT (input1 , input2);
```

input1	input1	REAL	1	2		%V00249
input2	input2	REAL	1	3		%V00253
output1	output1	REAL	1	8		%V00257

5.2.1 ADD: Сложение

◆ Описание функции

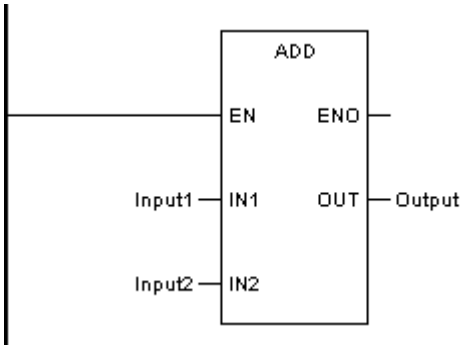
Этот блок функции складывает входные значения на вводе и присваивает результат данным на выводе.

Количество вводов может быть увеличено до 8.

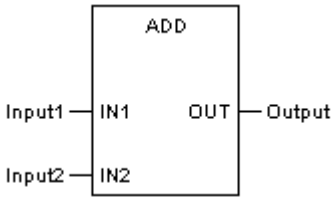
◆ Формула

OUT = IN1 + IN2 + ... + INn

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD Input1

ADD Input2

ST Output

◆ Отображение в ST

Output := ADD (Input1, Input2);

◆ Описание параметра

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

5.2.2 SUB: Вычитание

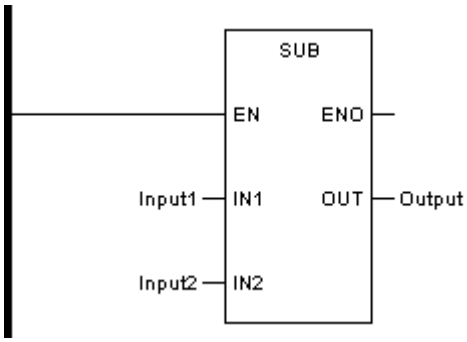
◆ Описание функции

Этот блок функции вычитает значение на вводе IN2 из значения на вводе IN1 и присваивает результат выводу

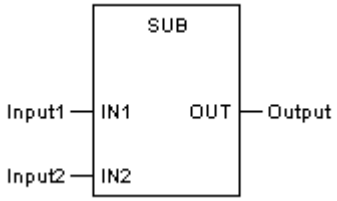
◆ Формула

OUT = IN1 - IN2

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input1
SUB	Input2
ST	Output

◆ Отображение в ST

Output := SUB (Input1, Input2);

◆ Описание параметра

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

5.2.3 MUL: Умножение

◆ Описание функции

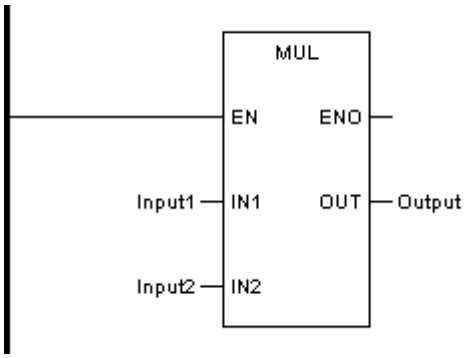
Этот блок функции перемножает значения на вводе и присваивает результат значению на выводе.

Количество вводов может быть увеличено до 8.

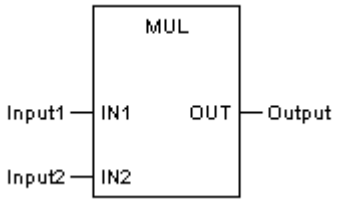
◆ Формула

OUT = IN1 * IN2 * ... * INn

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
MUL     Input2
ST      Output
```

◆ Отображение в ST

```
Output := MUL (Input1, Input2);
```

◆ Описание параметра

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

5.2.4 DIV: Деление

◆ Описание функции

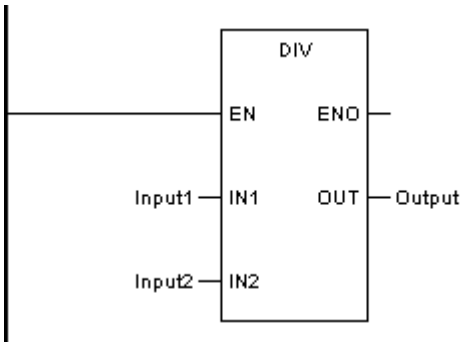
Этот блок функции делит значение на вводе IN1 на значение на вводе IN2 и присваивает результат – частное - значению вывода.

Если IN1 равна нулю, то переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing .....", с указанием FB, программы, даты и времени ошибки.

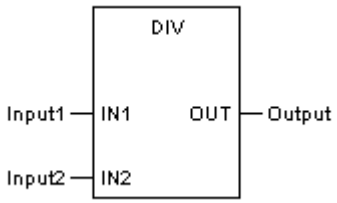
◆ Формула

OUT = IN1 / IN2

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input1
DIV	Input2
ST	Output

◆ Отображение в ST

Output := DIV (Input1, Input2);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Делимое	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Делитель	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Частное	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.5 MOD: Деление по модулю

◆ Описание функции

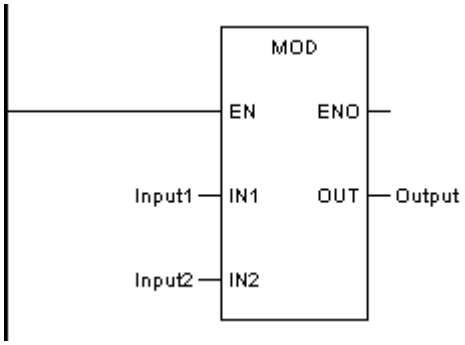
Этот блок функции делит значение на вводе IN1 на значение на вводе IN2 и присваивает значение деления по модулю – остаток – значению вывода.

Если IN1 равна нулю, то переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing .....", с указанием FB, программы, даты и времени ошибки.

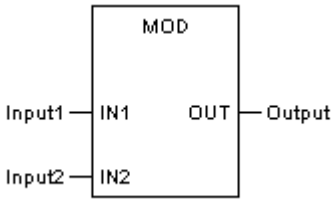
◆ Формула

OUT = IN1 % IN2

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
MOD     Input2
ST      Output
```

◆ Отображение в ST

```
Output := MOD (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Делимое	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Делитель	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Остаток	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.2.6 DIVMOD: Деление и деление по модулю

◆ Описание функции

Этот блок функции делит значение на вводе IN1 на значение на вводе IN2. Частное поступает на вывод DV. Остаток деления поступает на вывод MD.

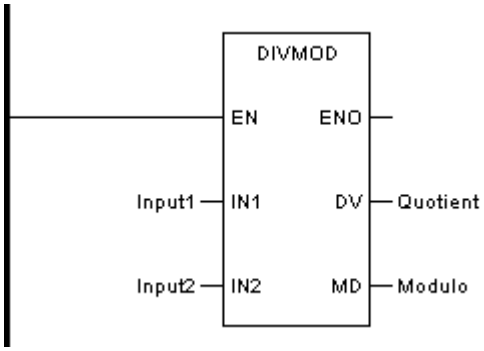
Если IN1 равна нулю, то переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing .....", с указанием FB, программы, даты и времени ошибки.

◆ Формула

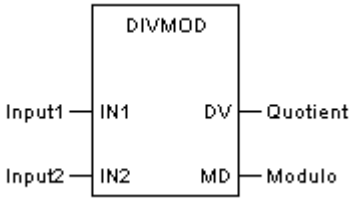
DV = IN1 / IN2

MD = IN1 % IN2

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL DIVMOD (IN1:=Input1, IN2:=Input2, DV=>Quotient, MD=>Modulo)

◆ Отображение в ST

DIVMOD (IN1:=Input1, IN2:=Input2, DV=>Quotient, MD=>Modulo);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Делимое	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Делитель	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DV	Quotient	Частное	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
MD	Modulo	Остаток	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.2.7 INC: Инкремент

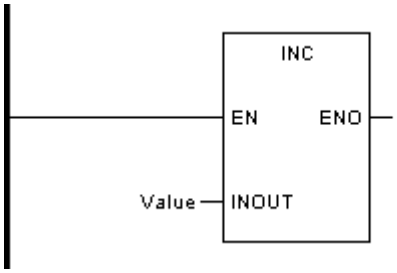
◆ Описание функции

Этот блок функции увеличивает переменную на 1.

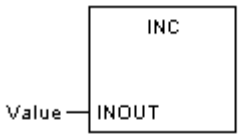
◆ Формула

INOUT = INOUT + 1

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL INC (Значение)

◆ Отображение в ST

INC (Значение);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
INOUT	Value	Каждый раз, когда программа использует этот блок функции, значение переменной увеличивается на одну единицу.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.2.8 DEC: Декремент

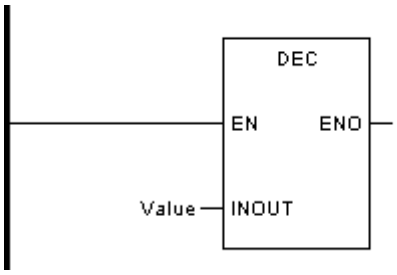
◆ Описание функции

Этот блок функции уменьшает переменную на 1.

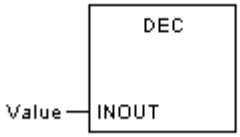
◆ Формула

INOUT = INOUT - 1

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL DEC (Значение)

◆ Отображение в ST

DEC (Значение);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
INOUT	Value	Каждый раз, когда программа использует этот блок функции, значение переменной уменьшается на одну единицу.	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.2.9 NEG: Отрицание

◆ Описание функции

Этот блок функции отменяет значение на вводе и присваивает результат значению на выводе.

Отрицание изменяет знак, например

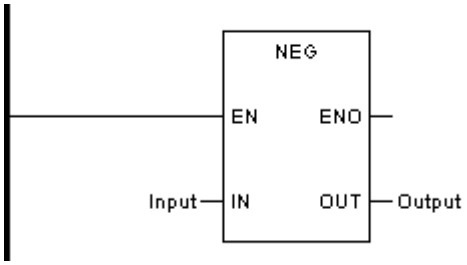
300 -> -300

-200->200

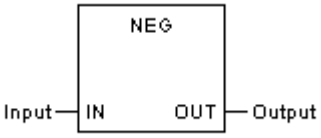
◆ Формула

OUT = 0 - IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
NEG
ST      Output
```

◆ Отображение в ST

```
Output := NEG (Input );
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Отрицательный результат	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.10 SIGN: Оценка знака

◆ Описание функции

Этот блок функции используется для обнаружения отрицательного знака.

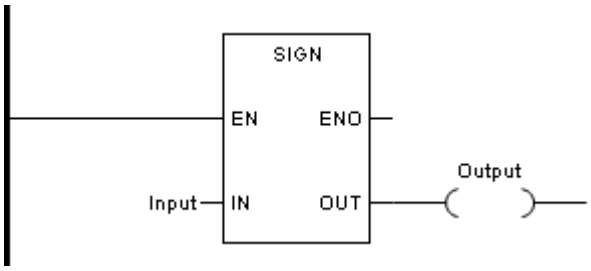
При значении ≥ 0 на вводе IN вывод OUT становится “0”. При значении <0 на вводе IN вывод OUT становится “1”.

◆ Формула

OUT = 1, if IN< 0,

OUT = 0, if IN ≥ 0.

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

```
LD      Input
SIGN
ST      Output
```

◆ Отображение в ST

Output := SIGN (Input);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод со знаком	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Оценка знака	BOOL	Q, M, N, V

5.2.11 SQRT: Квадратный корень

◆ Описание функции

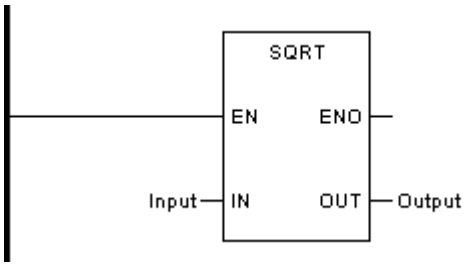
Этот блок функции извлекает квадратный корень из данных на вводе и присваивает результат данным на выводе.

Если IN отрицательное, то переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing .....", с указанием FB, программы, даты и времени ошибки.

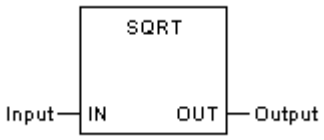
◆ Формула

OUT = $\sqrt{IN}$

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
SQRT
ST      Output
```

◆ Отображение в ST

Output := SQRT (Input);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Квадратный корень на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.12 ABS: Абсолютное значение

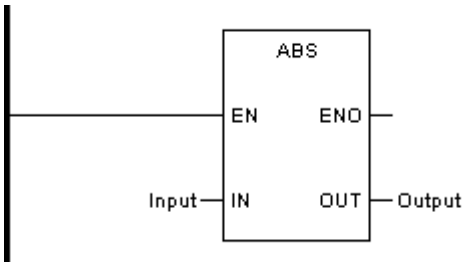
◆ Описание функции

Этот блок функции вычисляет абсолютное значение из данных на вводе и присваивает результат значению на выводе.

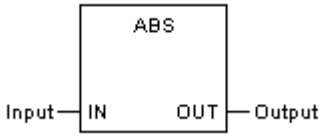
◆ Формула

OUT = |IN|

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
ABS
ST      Output
```

◆ Отображение в ST

```
Output := ABS (Input );
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Абсолютное значение на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.13 LOG: Десятичный логарифм

◆ Описание функции

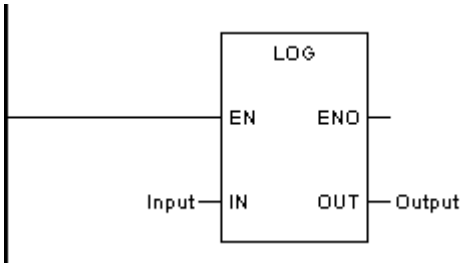
Этот блок функции вычисляет логарифм данных на вводе по основанию 10 и присваивает результат данным на выводе.

Если IN отрицательное либо равно нулю, то переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing .....", с указанием FB, программы, даты и времени ошибки.

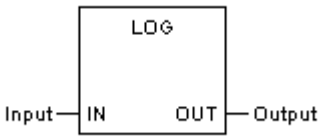
◆ Формула

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

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
LOG
ST      Output
```

◆ Отображение в ST

```
Output := LOG (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Логарифм по основанию 10 на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.14 LN: Натуральный логарифм

◆ Описание функции

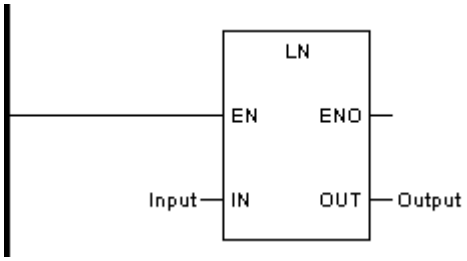
Этот блок функции вычисляет натуральный логарифм данных на вводе и присваивает результат данным на выводе.

Если IN отрицательное либо равно нулю, то переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing ……", с указанием FB, программы, даты и времени ошибки.

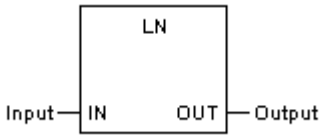
◆ Формула

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

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
LN
ST      Output
```

◆ Отображение в ST

```
Output := LN (Input );
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Натуральный логарифм на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.15 EXP: Экспонента

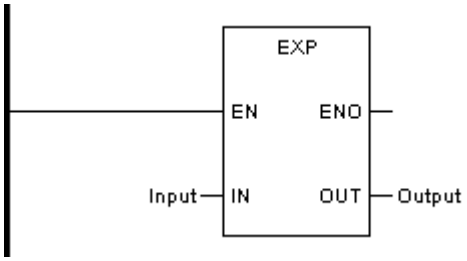
◆ Описание функции

Этот блок функции вычисляет экспоненциальную функцию для данных на вводе и присваивает результат данным на выводе.

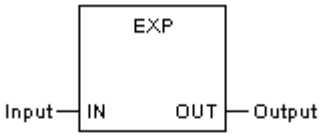
◆ Формула

OUT = e^{IN}, e=2.718282

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
EXP
ST      Output
```

◆ Отображение в ST

```
Output := EXP (Input );
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Экспоненциальная функция на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.16 EXPT: Возведение в степень

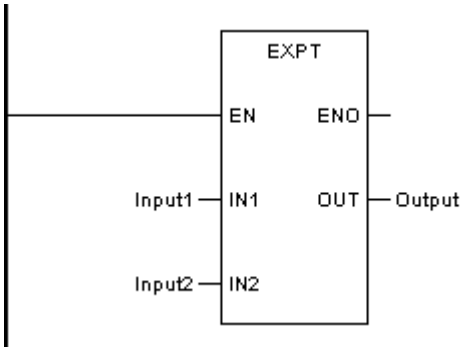
◆ Описание функции

Этот блок функции вычисляет возведение в степень одного значения на другое значение и присваивает результат данным на выводе.

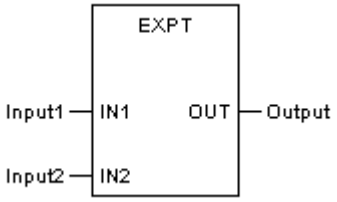
◆ Формула

$OUT = IN1^{IN2}$

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
EXPT    Input2
ST      Output
```

◆ Отображение в ST

```
Output := EXPT (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Основание	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Показатель степени	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Результат возведения в степень	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.17 SIN: Синус

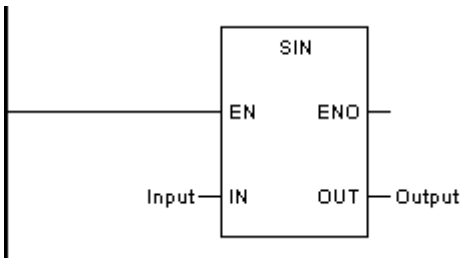
◆ Описание функции

Этот блок функции вычисляет синус угла на вводе и присваивает результат данным на выводе.

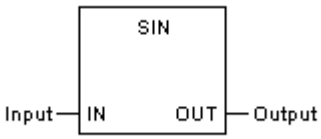
◆ Формула

OUT = sin IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
SIN
ST      Output
```

◆ Отображение в ST

```
Output := SIN (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Синус на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.18 COS: Косинус

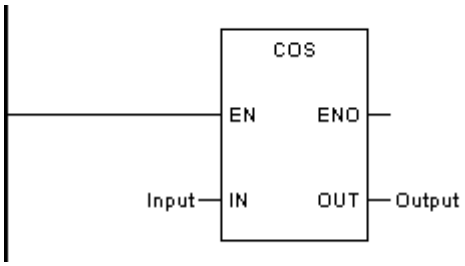
◆ Описание функции

Этот блок функции вычисляет косинус угла на вводе и присваивает результат данным на выводе.

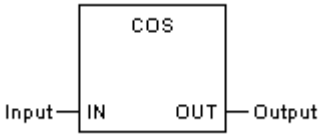
◆ Формула

OUT = cos IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
COS
ST      Output
```

◆ Отображение в ST

```
Output := COS (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Косинус на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.19 TAN: Тангенс

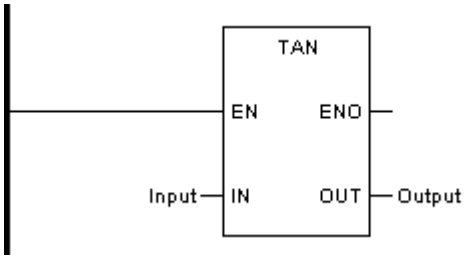
◆ Описание функции

Этот блок функции вычисляет тангенс угла на вводе и присваивает результат данным на выводе.

◆ Формула

OUT = tan IN

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

```
LD      Input
TAN
ST      Output
```

◆ Отображение в ST

```
Output := TAN (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Тангенс на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.20 ASIN: Арксинус

◆ Описание функции

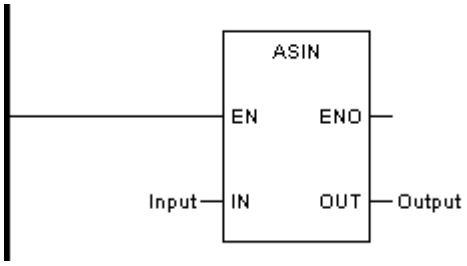
Этот блок функции вычисляет главный арксинус на вводе и присваивает результат данным на выводе в виде угла в радианах.

При значении входной переменной IN вне диапазона [-1; 1] переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing ……", с указанием FB, программы, даты и времени ошибки.

◆ Формула

OUT = asin IN

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

```
LD      Input
ASIN
ST      Output
```

◆ Отображение в ST

```
Output := ASIN (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Арксинус на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.21 ACOS: Арккосинус

◆ Описание функции

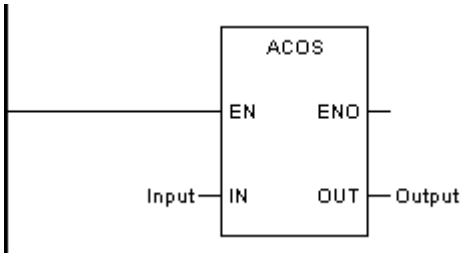
Этот блок функции вычисляет главный арккосинус на вводе и присваивает результат данным на выводе в виде угла в радианах.

При значении входной переменной IN вне диапазона [-1; 1] переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing ……", с указанием FB, программы, даты и времени ошибки.

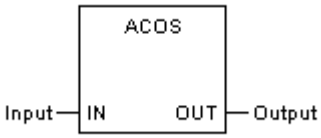
◆ Формула

OUT = acos IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
ACOS
ST      Output
```

◆ Отображение в ST

```
Output := ACOS (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Косинус дуги на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.2.22 ATAN: Арктангенс

◆ Описание функции

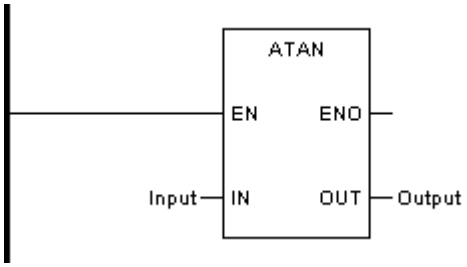
Этот блок функции вычисляет главный арктангенс на вводе и присваивает результат данным на выводе в виде угла в радианах.

При значении входной переменной IN вне диапазона [-1; 1] переменная %S0007 станет ИСТИНА, и при онлайн-подключении к ПЛК во вкладке Debug окна вывода появится сообщение "Error occurs while executing ……", с указанием FB, программы, даты и времени ошибки.

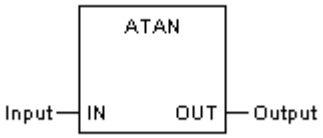
◆ Формула

OUT = atan IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
ATAN
ST      Output
```

◆ Отображение в ST

```
Output := ATAN (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Арктангенс на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.3 Статистические блоки функций

В этом разделе описываются базовые блоки функций, связанные с выполнением базовых статистических операций.

Данный раздел содержит следующие подразделы.

Тип	Описание
MIN	Минимальное значение: OUT = MIN {IN1, IN2, ... , INn} (n≤8)
MAX	Максимальное значение: OUT = MAX { IN1, IN2, ... , INn} (n≤8)
AVE	Усреднение: OUT = $\frac{IN1 + IN2 + \ldots IN\mathrm{n}}{\mathrm{n}}$ (n≤8)
LIMIT	Предел: OUT = IN, если (IN ≥ MN) & (IN ≤ MX) OUT = MN, если (IN < MN) OUT = MX, если (IN > MX)
SEL	Выбор 0/1: G = 0 -> OUT = IN0 G = 1 -> OUT = IN1
MUX	Мультиплексор: K = 0 -> OUT = IN0 K = 1 -> OUT = IN1 K = 2 -> OUT = IN2 ..... K = n -> OUT = INn (n≤6)

При переполнении максимального/минимального значения типа данных на выходе, будет выведено значение, уменьшенное/увеличенное на максимальное/минимальное значение типа данных, будьте внимательны при выборе типов данных переменных!

5.3.1 MIN: Минимальное значение

◆ Описание функции

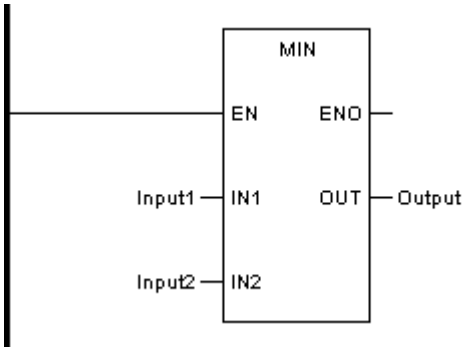
Этот блок функции присваивает значению на выводе наименьшее значение ввода.

Количество вводов может быть увеличено до 8.

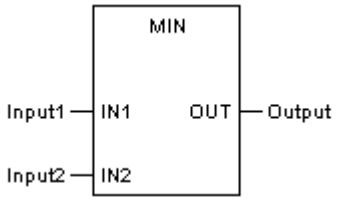
◆ Формула

OUT = MIN {IN1, IN2, ... , INn} (n≤8)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
MIN     Input2
ST      Output
```

◆ Отображение в ST

```
Output := MIN (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Минимальное значение на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.3.2 MAX: Максимальное значение

◆ Описание функции

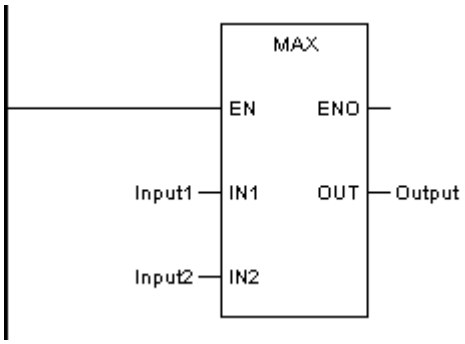
Этот блок функции присваивает значению на выводе наибольшее значение ввода.

Количество вводов может быть увеличено до 8.

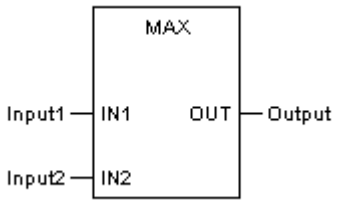
◆ Формула

OUT = MAX {IN1, IN2, ... , INn} (n≤8)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input1
MAX	Input2
ST	Output

◆ Отображение в ST

Output := MAX (Input1, Input2);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Максимальное значение на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.3.3 AVE: Среднее значение

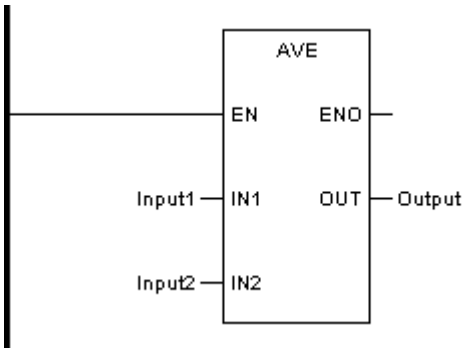
◆ Описание функции

Этот блок функции вычисляет среднее значение из значений на вводе (до 8) и присваивает результат данным на выводе.

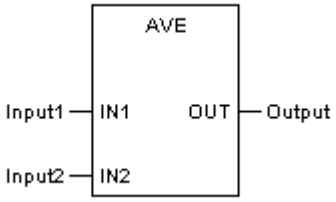
◆ Формула

$$OUT = \frac{IN1 + IN2 + \ldots IN\mathrm{n}}{\mathrm{n}} \quad (n \le 8)$$

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
AVE     Input2
ST      Output
```

◆ Отображение в ST

```
Output := AVE (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Среднее значение на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.3.4 LIMIT: Предел

◆ Описание функции

Этот блок функции передает неизмененное значение со ввода (IN) на вывод, если значение на вводе не меньше минимального значения (MN) и не больше максимального (MX).

Если значение на вводе (IN) меньше минимального значения (MN), то минимальное значение передается на вывод.

Если значение на вводе (IN) больше максимального значения (MX), то максимальное значение передается на вывод.

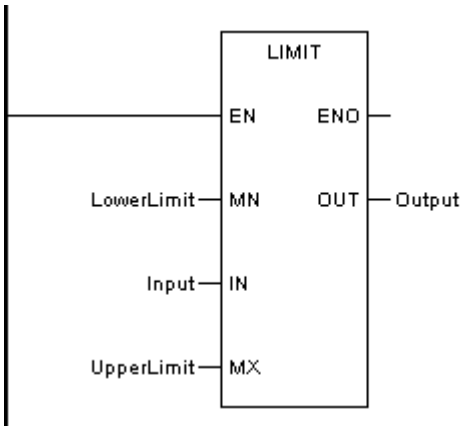
◆ Формула

OUT = IN, если (IN ≥ MN) & (IN ≤ MX)

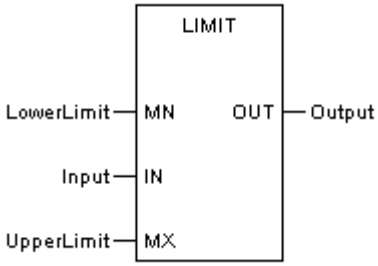
OUT = MN, если (IN < MN)

OUT = MX, если (IN > MX)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	LowerLimit
LIMIT	Input, UpperLimit
ST	Output

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
MN	LowerLimit	Минимальное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN	Input	Ввод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
MX	UpperLimit	Максимальное значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.3.5 SEL: выбор 0/1

◆ Описание функции

Этот блок функции используется для двоичного выбора между двумя значениями на вводе.

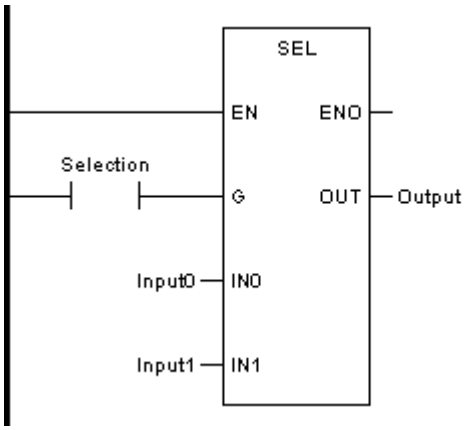
В зависимости от состояния ввода G, на вывод OUT передается либо ввод IN0, либо ввод IN1.

◆ Формула

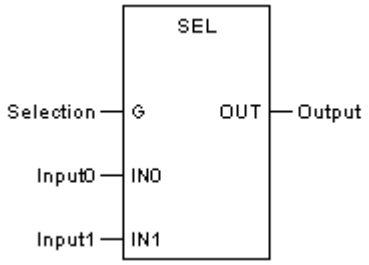
G = 0 -> OUT = IN0

G = 1 -> OUT = IN1

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Selection
SEL	Input0, Input1
ST	Output

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
G	Selection	Выбор ввода	BOOL	Константа, I, Q, M, N, S, V
IN0	Input0	Ввод 0	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.3.6 MUX: Мультиплексор

◆ Описание функции

Этот блок функции передает соответствующий ввод на вывод в зависимости от значения ввода K.

Количество вводов может быть увеличено до 7.

◆ Формула

K = 0 -> OUT = IN0

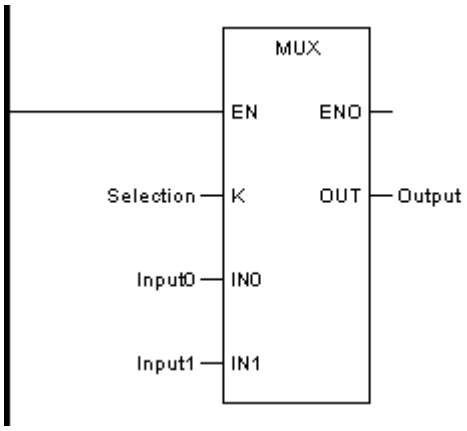
K = 1 -> OUT = IN1

K = 2 -> OUT = IN2

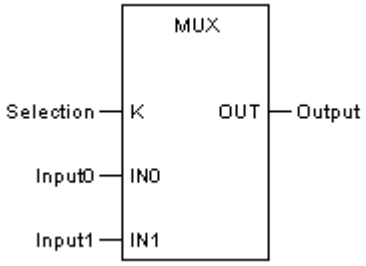
……

K = n -> OUT = INn (n≤6)

◆ Отображение в LD



◆ Отображение в FBD



- Отображение в IL

LD	Selection
MUX	Input0, Input1
ST	Output

- Отображение в ST

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

- Описание параметра

	Параметр	Описание	Тип данных	Тип точки
K	Selection	Выбор ввода	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN0	Input0	Ввод 0	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.4 Логические блоки функций

В этом разделе описываются базовые блоки функций, связанные с выполнением базовых логических операций.

Данный раздел содержит следующие подразделы.

Тип	Описание
AND	И
OR	ИЛИ
NOT	Отрицание
XOR	Исключающий ИЛИ
SHL	Сдвиг влево
SHR	Сдвиг вправо
ROL	Поворот влево
ROR	Поворот вправо
BSET	Установить бит
BCLR	Очистить бит
BTST	Протестировать бит
R_TRIG	Определение переднего фронта
F_TRIG	Определение заднего фронта
SET	Установить
RESET	Сброс
SR	Бистабильный (Доминанта включения)
RS	Бистабильный (Доминанта выключения)

При переполнении максимального/минимального значения типа данных на выходе, будет выведено значение, уменьшенное/увеличенное на максимальное/минимальное значение типа данных, будьте внимательны при выборе типов данных переменных!

5.4.1 AND: И

◆ Описание функции

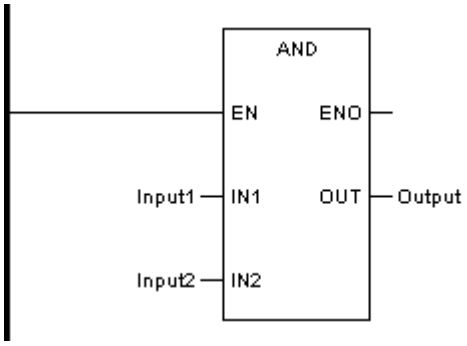
Этот блок функции анализирует каждый бит в связках AND, находящихся в последовательностях бит на вводе и присваивает результат данным на выводе.

Количество вводов может быть увеличено до 8.

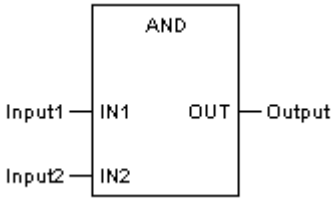
◆ Формула

OUT = IN1 AND IN2 AND … AND INn

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
AND     Input2
ST      Output
```

◆ Отображение в ST

```
Output := AND (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
IN2	Input2	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
OUT	Output	Последовательность бит на выводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Q, M, N, MW, NW, V

5.4.2 OR: ИЛИ

◆ Описание функции

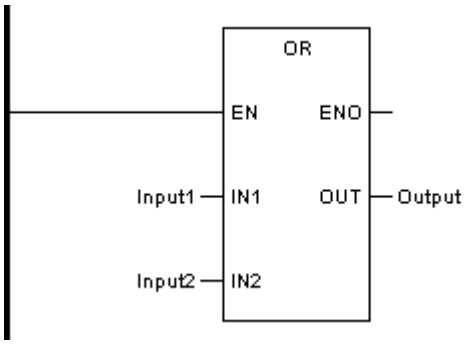
Этот блок функции анализирует каждый бит в связках OR, находящихся в последовательностях бит на вводе и присваивает результат данным на выводе.

Количество вводов может быть увеличено до 8.

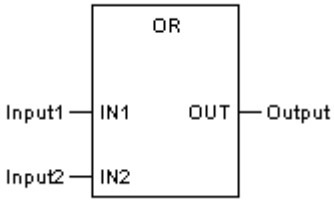
◆ Формула

OUT = IN1 OR IN2 OR ... OR INn

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
OR      Input2
ST      Output
```

◆ Отображение в ST

```
Output := OR (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
IN2	Input2	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
OUT	Output	Последовательность бит на выводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Q, M, N, MW, NW, V

5.4.3 NOT: Отрицание

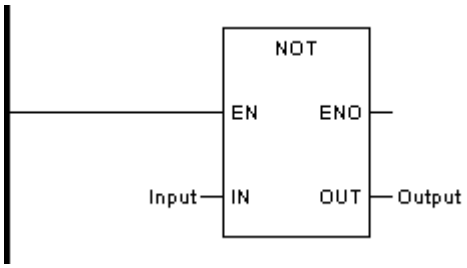
◆ Описание функции

Этот блок функции отменяет значение каждого бита, находящегося в последовательности бит на вводе, и присваивает результат значению на выводе.

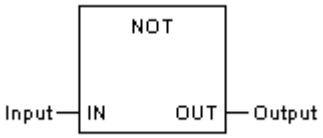
◆ Формула

OUT = NOT IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
NOT
ST      Output
```

◆ Отображение в ST

```
Output := NOT (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Последовательность бит на вводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Последовательность бит на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.4.4 XOR: Исключающее ИЛИ

◆ Описание функции

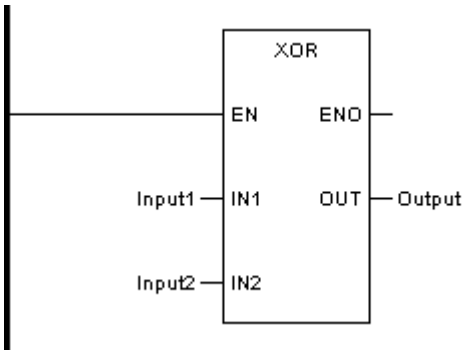
Этот блок функции анализирует каждый бит в связках XOR, находящихся в последовательностях бит на вводе и присваивает результат данным на выводе.

Количество вводов может быть увеличено до 8.

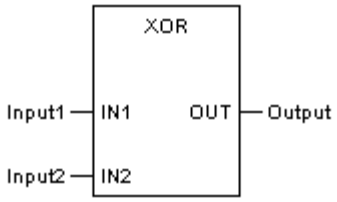
◆ Формула

OUT = IN1 XOR IN2 XOR ... XOR INn

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
XOR     Input2
ST      Output
```

◆ Отображение в ST

```
Output := XOR (Input1, Input2);
```

◆ Описание параметра

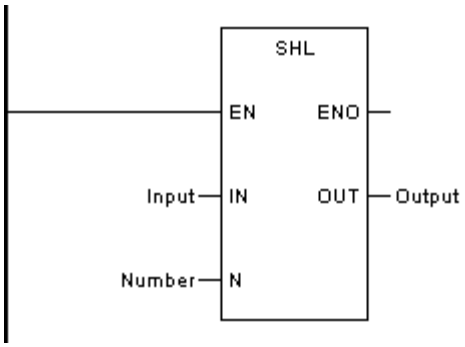
	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Последовательность бит на вводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Последовательность бит на вводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Последовательность бит на выводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.4.5 SHL: Сдвиг влево

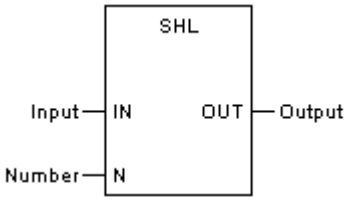
◆ Описание функции

Этот блок функции сдвигает битовый шаблон на вводе IN влево на N бит. Нули заполняются справа. Результат присваивается значению на выводе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
SHL     Number
ST      Output
```

◆ Отображение в ST

```
Output := SHL (Input, Number);
```

◆ Описание параметра

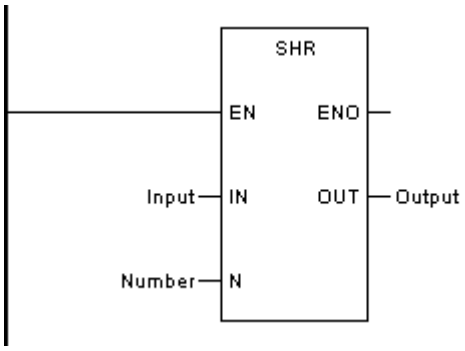
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Битовый шаблон, который будет сдвинут	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Число, которое будет сдвинуто	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Сдвинутый битовый шаблон	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.4.6 SHR: Сдвиг вправо

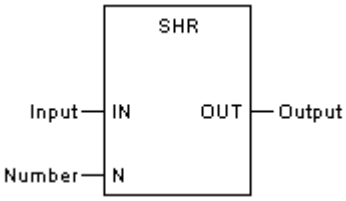
◆ Описание функции

Этот блок функции сдвигает битовый шаблон на вводе IN вправо на N бит. Нули заполняются слева. Результат присваивается значению на выводе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input
SHR	Number
ST	Output

◆ Отображение в ST

Output := SHR (Input, Number);

◆ Описание параметра

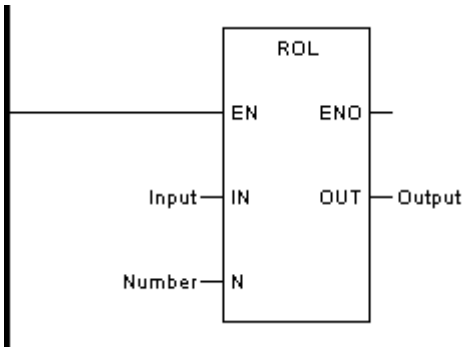
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Битовый шаблон, который будет сдвинут	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Число, которое будет сдвинуто	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Сдвинутый битовый шаблон	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.4.7 ROL: Поворот влево

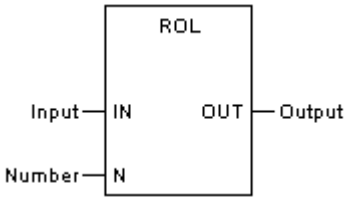
◆ Описание функции

Этот блок функции поворачивает битовый шаблон на вводе IN по кругу влево на N бит и присваивает результат данным на выводе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
ROL     Number
ST      Output
```

◆ Отображение в ST

```
Output := ROL (Input, Number);
```

◆ Описание параметра

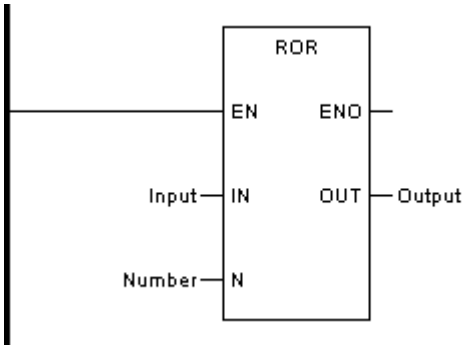
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Битовый шаблон, который будет повернут	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Число, которое будет повернуто	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Повернутый битовый шаблон	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.4.8 ROR: Поворот вправо

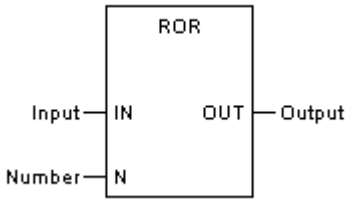
◆ Описание функции

Этот блок функции поворачивает битовый шаблон на вводе IN по кругу вправо на N бит и присваивает результат данным на выводе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
ROR     Number
ST      Output
```

◆ Отображение в ST

```
Output := ROR (Input, Number);
```

◆ Описание параметра

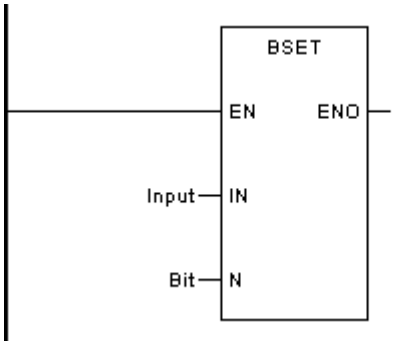
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Битовый шаблон, который будет повернут	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Число, которое будет повернуто	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Повернутый битовый шаблон	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.4.9 BSET: Установить бит

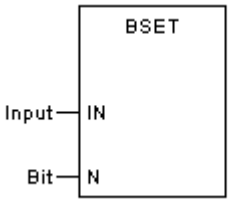
◆ Описание функции

Этот блок функции присваивает значение “1” N-ому биту последовательности бит на вводе IN.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL BSET (IN:=Input, N:=Bit)

◆ Отображение в ST

BSET (IN:=Input, N:=Bit);

◆ Описание параметра

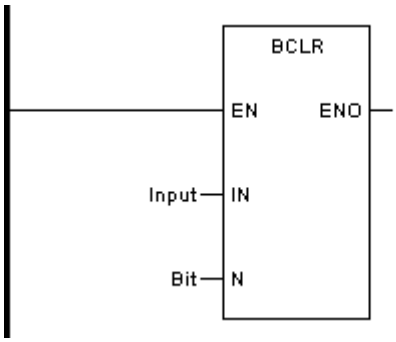
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Q, M, MW, N, NW, V
N	Bit	Номер бита	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.4.10 BCLR: Очистить бит

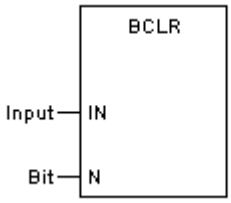
◆ Описание функции

Этот блок функции присваивает значение “0” N-ому биту последовательности бит на вводе IN.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL BCLR (IN:=Input, N:=Bit)

◆ Отображение в ST

BCLR (IN:=Input, N:=Bit);

◆ Описание параметра

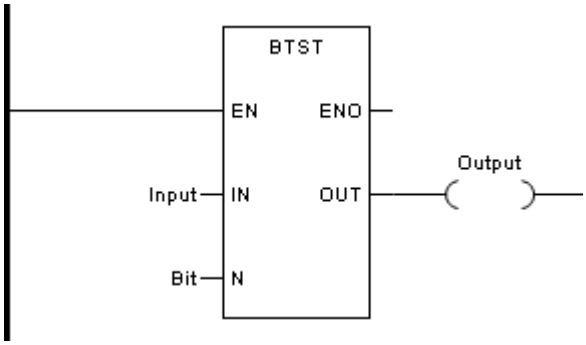
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Q, M, MW, N, NW, V
N	Bit	Номер бита	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.4.11 BTST: Протестировать бит

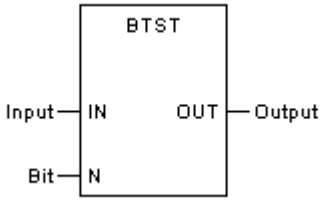
◆ Описание функции

Этот блок функции проверяет значение N-го бита (“1” или “0”) последовательности бит на вводе IN. Если результат равен “1”, то вывод становится “1”. Если результат равен “0”, то вывод становится “0”.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL BTST (IN:=Input, N:=Bit, OUT=>Output)

◆ Отображение в ST

BTST (IN:=Input, N:=Bit, OUT=>Output);

◆ Описание параметра

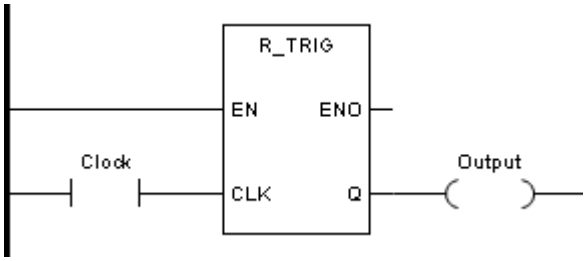
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Последовательность бит на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
N	Bit	Номер бита	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.4.12 R_TRIG: Определение переднего фронта

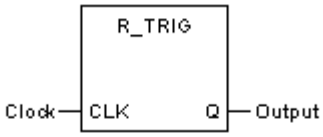
◆ Описание функции

Этот блок функции используется для определения передних фронтов 0->1. Вывод Q становится “1”, если на вводе CLK имеется переход от “0” к “1”. Вывод остается на уровне “1” от выполнения одного функционального блока до следующего (один цикл); вывод впоследствии возвращается к “0”.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

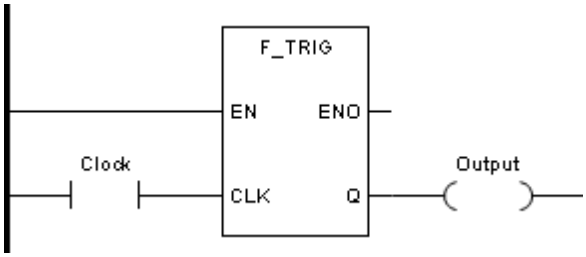
	Параметр	Описание	Тип данных	Тип точки
CLK	Clock	Вводные данные часов	BOOL	Константа, I, Q, M, N, S, V
Q	Output	Вывод	BOOL	Q, M, N, V

5.4.13 F_TRIG: Определение заднего фронта

◆ Описание функции

Этот блок функции используется для определения задних фронтов 1->0. Вывод Q становится “1”, если на вводе CLK имеется переход от “1” к “0”. Вывод остается на уровне “1” от выполнения одного функционального блока до следующего (один цикл); вывод впоследствии возвращается к “0”.

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

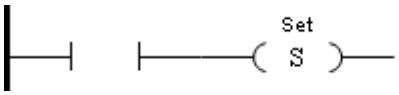
	Параметр	Описание	Тип данных	Тип точки
CLK	Clock	Вводные данные часов	BOOL	Константа, I, Q, M, N, S, V
Q	Output	Вывод	BOOL	Q, M, N, V

5.4.14 SET: Установить

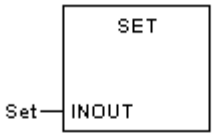
◆ Описание функции

Этот блок функции присваивает связанному с ним биту значение “1”. Функция такая же, как у "Установить катушку" -(S)-.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL SET (Set)

◆ Отображение в ST

SET (Set);

◆ Описание параметра

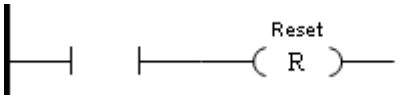
	Параметр	Описание	Тип данных	Тип точки
INOUT	Set	Бит, который нужно установить	BOOL	Q, M, N, V

5.4.15 RESET: Сброс

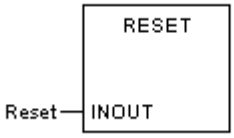
◆ Описание функции

Этот блок функции присваивает связанному с ним биту значение “0”. Функция такая же, как у "Сбросить катушку" -(R)-.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL RESET (Reset)

◆ Отображение в ST

RESET (Reset);

◆ Описание параметра

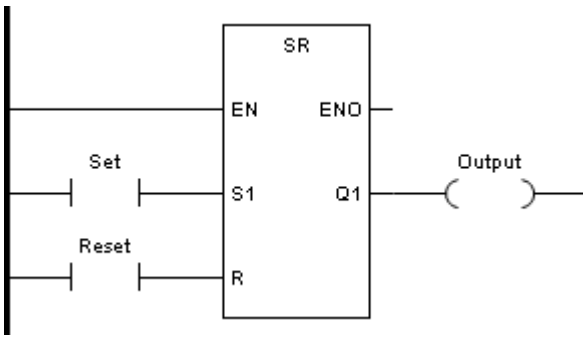
	Параметр	Описание	Тип данных	Тип точки
INOUT	Reset	Бит, который нужно сбросить	BOOL	Q, M, N, V

5.4.16 SR: SR-триггер

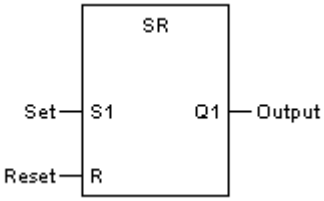
◆ Описание функции

Этот блок функции используется в качестве памяти SR со свойством “Приоритет установки”. Вывод Q1 становится “1”, когда ввод S1 становится “1”. Это состояние сохраняется, даже если ввод S1 возвращается обратно к “0”. Вывод Q1 возвращается к ”0“, когда ввод R становится "1”. Если оба ввода S1 и R имеют значение “1”, приоритетный ввод S1 установит на выводе Q1 значение “1".

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

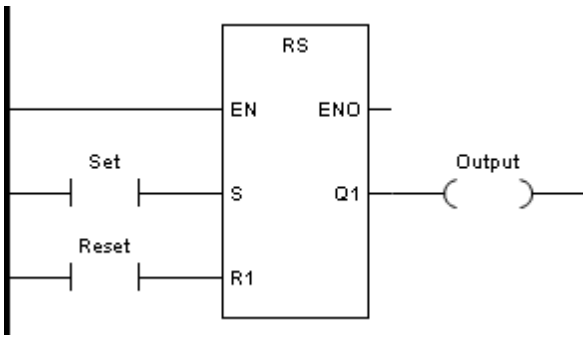
	Параметр	Описание	Тип данных	Тип точки
S1	Set	Установить (доминанта включения)	BOOL	Константа, I, Q, M, N, S, V
R	Reset	Сброс	BOOL	Константа, I, Q, M, N, S, V
Q1	Output	Вывод	BOOL	Q, M, N, V

5.4.17 RS: RS-триггер

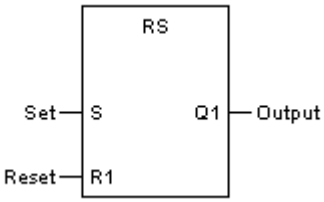
◆ Описание функции

Этот блок функции используется в качестве памяти RS со свойством “Приоритет сброса”. Вывод Q1 становится “1”, когда ввод S становится “1”. Это состояние сохраняется, даже если ввод S возвращается обратно к “0”. Вывод Q1 возвращается к ”0“, когда ввод R1 становится "1”. Если оба ввода S и R1 имеют значение “1”, приоритетный ввод R1 установит на выводе Q1 значение “0".

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
CAL    RS (S:=Set, R1:=Reset, Q1=>Output)
```

◆ Отображение в ST

```
RS (S:=Set, R1:=Reset, Q1=>Output);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
S	Set	Установить	BOOL	Константа, I, Q, M, N, S, V
R1	Reset	Сброс (доминанта выключения)	BOOL	Константа, I, Q, M, N, S, V
Q1	Output	Вывод	BOOL	Q, M, N, V

5.5 Блоки функций сравнения

В этом разделе описываются базовые блоки функций, связанные с выполнением базовых операций сравнения.

Данный раздел содержит следующие подразделы.

Тип	Описание
EQ	Равно: OUT = (IN1 = IN2)
NE	Не равно: OUT = (IN1 <> IN2)
GT	Больше, чем: OUT = (IN1 > IN2)
GE	Больше или равно: OUT = (IN1 ≥ IN2)
LT	Меньше: OUT = (IN1 < IN2)
LE	Меньше или равно: OUT = (IN1 ≤ IN2)

5.5.1 EQ: Равно

◆ Описание функции

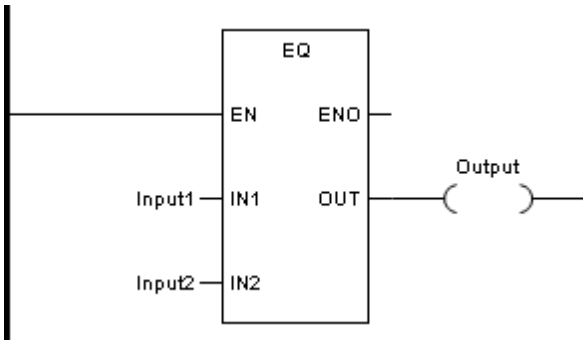
Этот блок функции проверяет данные на вводе на равенство, т.е. данные на выводе становятся “1”, если на всех вводах есть равенство; в противном случае данные на выводе остаются равными “0”.

Количество вводов может быть увеличено до 8.

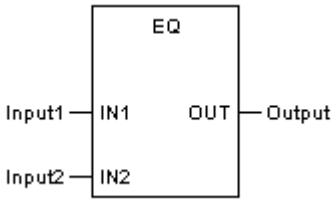
◆ Формула

OUT = 1, если (IN1 = IN2) AND (IN2 = IN3) AND ... AND (INn-1 = INn)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
EQ      Input2
ST      Output
```

◆ Отображение в ST

```
Output := EQ (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.5.2 NE: Не равно

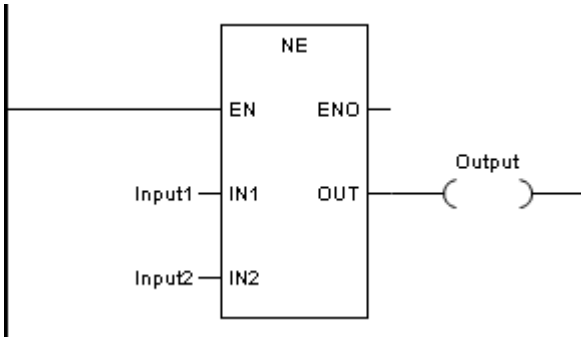
◆ Описание функции

Этот блок функции проверяет значения на вводе на наличие неравенства.

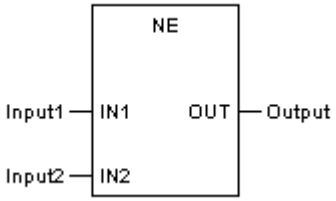
◆ Формула

OUT = 1, если IN1<>IN2

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input1
NE	Input2
ST	Output

◆ Отображение в ST

Output := NE (Input1, Input2);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.5.3 GT: Больше, чем

◆ Описание функции

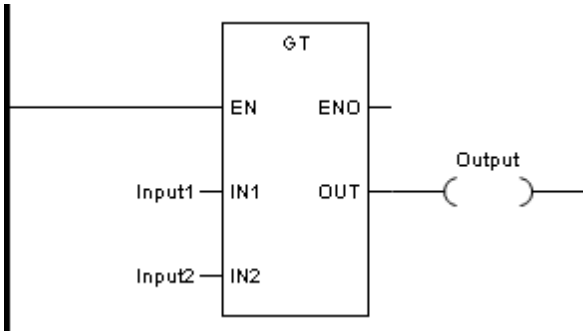
Этот блок функции проверяет значения последовательных данных на вводе на предмет убывающей последовательности.

Количество вводов может быть увеличено до 8.

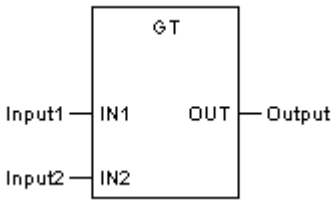
◆ Формула

OUT = 1, если (IN1>IN2) AND (IN2>IN3) AND ... AND (INn-1>INn)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input1
GT      Input2
ST      Output
```

◆ Отображение в ST

```
Output := GT (Input1, Input2);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.5.4 GE: Больше или равно

◆ Описание функции

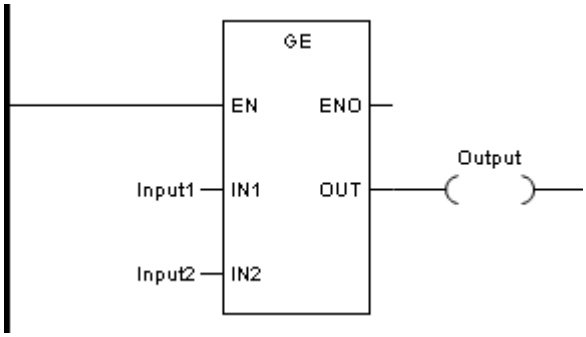
Этот блок функции проверяет значения последовательных данных на вводе на предмет убывающей последовательности или равенства.

Количество вводов может быть увеличено до 8.

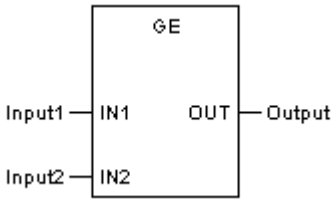
◆ Формула

OUT = 1, если (IN1≥IN2) AND (IN2≥IN3) AND ... AND (INn-1≥INn)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input1
GE	Input2
ST	Output

◆ Отображение в ST

Output := GE (Input1, Input2);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.5.5 LT: Меньше, чем

◆ Описание функции

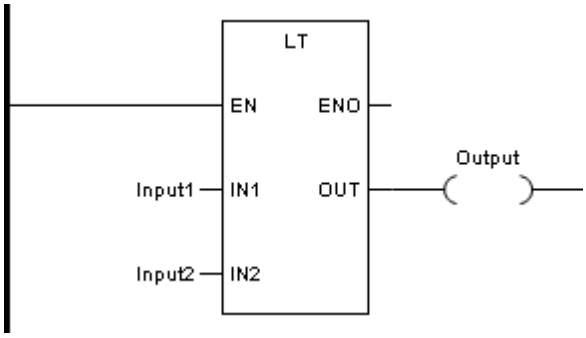
Этот блок функции проверяет значения последовательных данных на вводе на предмет возрастающей последовательности.

Количество вводов может быть увеличено до 8.

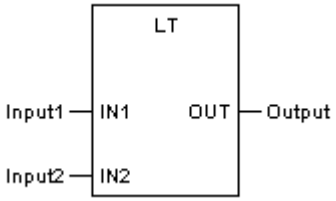
◆ Формула

OUT = 1, если (IN1<IN2) AND (IN2<IN3) AND ... AND (INn-1<INn)

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

LD	Input1
LT	Input2
ST	Output

◆ Отображение в ST

Output := LT (Input1, Input2);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.5.6 LE: Меньше или равно

◆ Описание функции

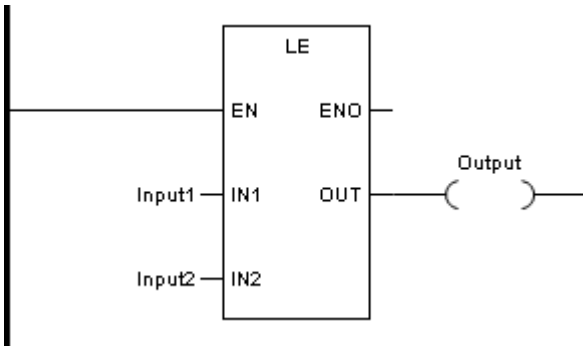
Этот блок функции проверяет значения последовательных данных на вводе на предмет возрастающей последовательности или равенства.

Количество вводов может быть увеличено до 8.

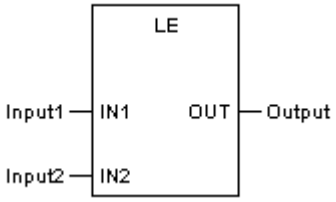
◆ Формула

OUT = 1, если (IN1≤IN2) AND (IN2≤IN3) AND ... AND (INn-1≤INn)

◆ Отображение в LD



◆ Отображение в FBD



- Отображение в IL

```
LD      Input1
LE      Input2
ST      Output
```

- Отображение в ST

```
Output := LE (Input1, Input2);
```

- Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN1	Input1	Ввод 1	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
IN2	Input2	Ввод 2	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод	BOOL	Q, M, N, V

5.6 Блоки функций преобразования

В этом разделе описываются базовые блоки функций, связанные с выполнением операций преобразования.

Данный раздел содержит следующие подразделы.

Тип	Описание
INT_TO_BCD	Преобразование целого числа в двоично-десятичный код (BCD)
BCD_TO_INT	Преобразование двоично-десятичного кода (BCD) в целое число
INT_TO_GRY	Преобразование целого числа в код Грея
GRY_TO_INT	Преобразование кода Грея в целое число
DEG_TO_RAD	Преобразование градусов в радианы
RAD_TO_DEG	Преобразование радианов в градусы

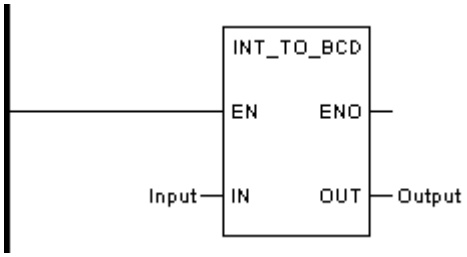
При переполнении максимального/минимального значения типа данных на выходе, будет выведено значение, уменьшенное/увеличенное на максимальное /минимальное значение типа данных, будьте внимательны при выборе типов данных переменных!

5.6.1 INT_TO_BCD: Преобразование целого числа в двоично-десятичный код (BCD)

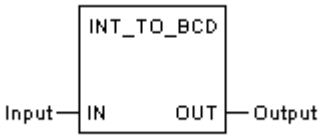
◆ Описание функции

Этот блок функции преобразует целое число в двоичной системе в целое число в двоично-десятичном коде (BCD).

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD          Input
INT_TO_BCD
ST          Output
```

◆ Отображение в ST

```
Output := INT_TO_BCD (Input);
```

◆ Описание параметра

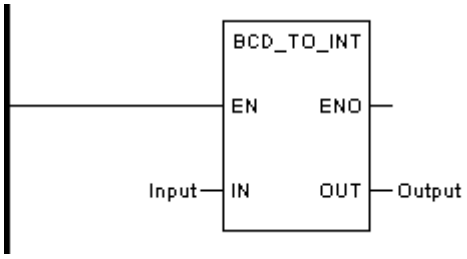
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Целое число в двоичной системе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Целое число в двоично-десятичном коде (BCD)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.6.2 BCD_TO_INT: Преобразование двоично-десятичного кода (BCD) в целое число

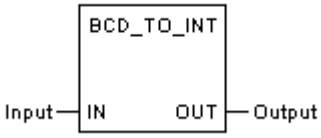
◆ Описание функции

Этот блок функции преобразует целое число в двоично-десятичном коде (BCD) в целое число в двоичной системе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD          Input
```

BCD_TO_INT
ST Output

◆ Отображение в ST

Output := BCD_TO_INT (Input);

◆ Описание параметра

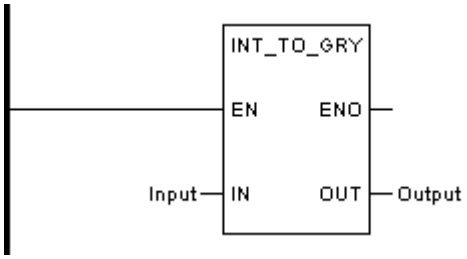
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Целое число в двоично-десятичном коде (BCD)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Целое число в двоичной системе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.6.3 INT_TO_GRY: Преобразование целого числа в код Грея

◆ Описание функции

Этот блок функции преобразует целое число в двоичной системе в целое число в код Грея.

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

LD Input
INT_TO_GRY
ST Output

◆ Отображение в ST

Output := INT_TO_GRY (Input);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки

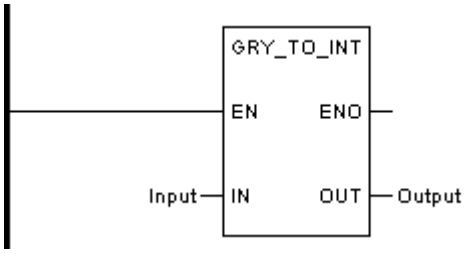
IN	Input	Целое число в двоичной системе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Целое число в коде Грея	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.6.4 GRY_TO_INT: Преобразование кода Грея в целое число

◆ Описание функции

Этот блок функции преобразует целое число в коде Грея в целое число в двоичной системе.

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

```
LD          Input
GRY_TO_INT
ST          Output
```

◆ Отображение в ST

```
Output := GRY_TO_INT (Input);
```

◆ Описание параметра

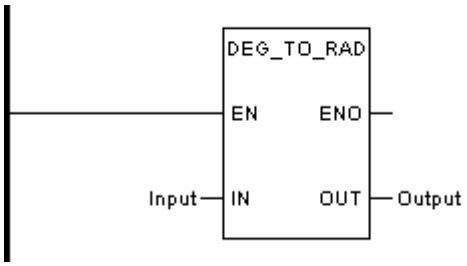
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Целое число в коде Грея	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Целое число в двоичной системе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.6.5 DEG_TO_RAD: Преобразование градусов в радианы

◆ Описание функции

Этот блок функции преобразует угол, выраженный в градусах, в радианы.

◆ Отображение в LD



◆ Отображение в FBD

◆ Отображение в IL

```
LD          Input
DEG_TO_RAD
ST          Output
```

◆ Отображение в ST

```
Output := DEG_TO_RAD (Input);
```

◆ Описание параметра

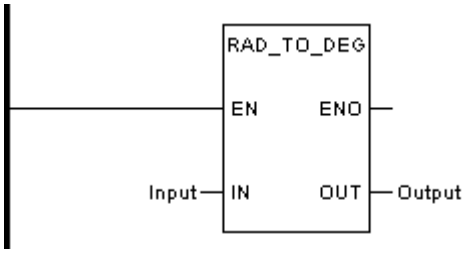
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Угол, выраженный в градусах	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Угол, выраженный в радианах	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.6.6 RAD_TO_DEG: Преобразование радианов в градусы

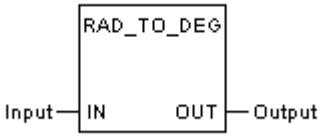
◆ Описание функции

Этот блок функции преобразует угол, выраженный в радианах, в градусы.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD          Input
RAD _TO_ DEG
ST          Output
```

◆ Отображение в ST

```
Output := RAD _TO_ DEG (Input);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Угол, выраженный в радианах	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Угол, выраженный в градусах	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.7 Блоки функций передачи данных

В этом разделе описываются базовые блоки функций, связанные с выполнением операций передачи данных.

Данный раздел содержит следующие подразделы.

Тип	Описание
MOVE	Передача значения
BLKMOV	Переместить данные блока
BLKCLR	Очистить данные блока
ETHMOV	Передача данных по Ethernet
COMMOV	Передача коммуникационных данных
READ	Чтение данных из специального модуля
WRITE	Запись данных в специальный модуль
XMT	Передача в режиме свободного порта
RCV	Прием в режиме свободного порта
LRC	Проверка LRC
CRC	Проверка CRC
MODRW	Чтение/запись данных через Modbus

5.7.1 MOVE: Передача значения

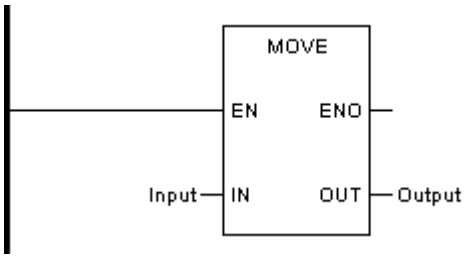
◆ Описание функции

Этот блок функции присваивает значение на вводе значению на выводе.

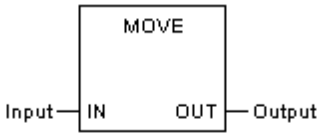
◆ Формула

OUT = IN

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
LD      Input
ST      Output
```

◆ Отображение в ST

```
Output := Input;
```

◆ Описание параметра

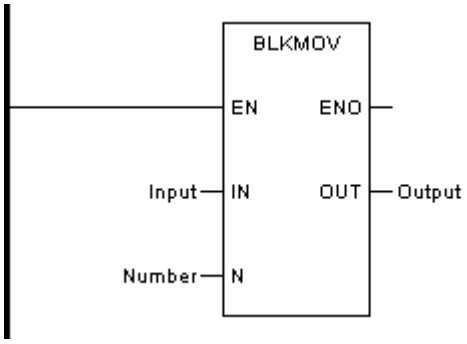
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Значение на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
OUT	Output	Значение на выводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Q, M, MW, N, NW, V

5.7.2 BLKMOV: Переместить данные блока

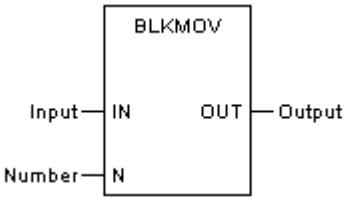
◆ Описание функции

Этот блок функции копирует блок данных на вводе и переносит их в данные на выводе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL BLKMOV (IN:=Input, N:=Number, OUT=>Output)

◆ Отображение в ST

BLKMOV (IN:=Input, N:=Number, OUT=>Output);

◆ Описание параметра

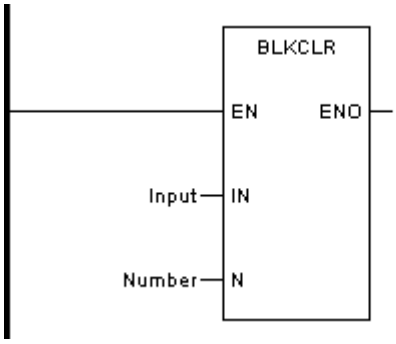
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Данные на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V
N	Number	Количество копируемых данных на вводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Данные на выводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	M, MW, N, NW, V

5.7.3 BLKCLR: Очистить данные блока

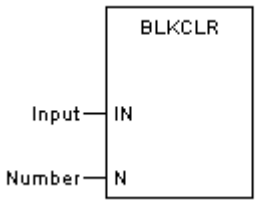
◆ Описание функции

Этот блок функции заполняет нулями указанный блок данных на вводе.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL BLKCLR (IN:=Input, N:=Number)

◆ Отображение в ST

BLKCLR (IN:=Input, N:=Number);

◆ Описание параметра

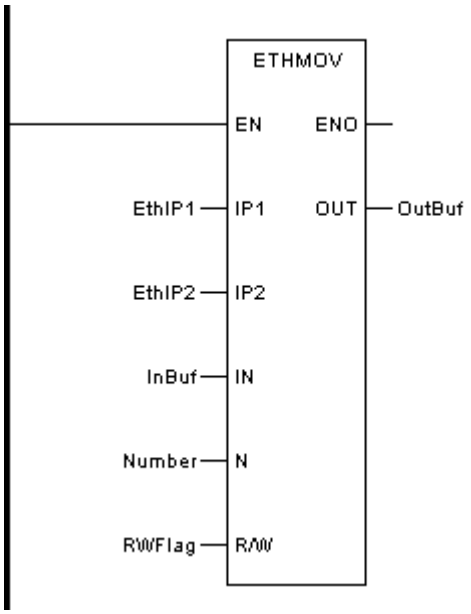
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Данные на вводе	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Q, M, MW, N, NW, V
N	Number	Количество данных на вводе, подлежащих очистке	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.7.4 ETHMOV: Передача данных по Ethernet

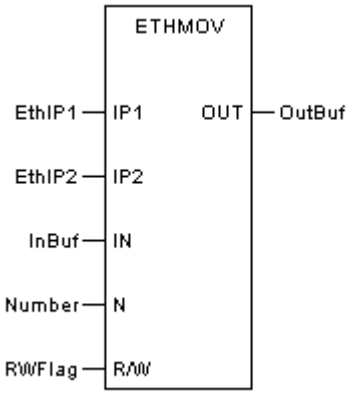
◆ Описание функции

Этот блок функции используется для связи с другими ПЛК СК4000 по протоколу MODBUS/TCP. Передаются/принимаются данные типа %MW.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL ETHMOV (IP1:=EthIP1, IP2:=EthIP2, IN:=InBuf, N:=Number, R/W:=RWFlag, OUT=>OutBuf)

◆ Отображение в ST

ETHMOV (IP1:=EthIP1, IP2:=EthIP2, IN:=InBuf, N:=Number, R/W:=RWFlag, OUT=>OutBuf);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IP1	EthIP1	Ethernet IP 1 другого ПЛК	Неприменимо, см.примеры	например, 10.40.0.22
IP2	EthIP2	Ethernet IP 2 другого ПЛК	Неприменимо, см.примеры	например, 10.40.1.22
IN	InBuf	Буфер данных этого ПЛК	WORD	MW
N	Number	Количество слов	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
R/W	RWFlag	Метка чтение/запись 0: Чтение 1: Запись	BOOL	Константа, I, Q, M, N, S, V
OUT	OutBuf	Буфер данных другого ПЛК	WORD	MW

◆ Пример в ST

```
ETHMOV(IP1:=10.40.0.22, IP2:=10.40.0.23, IN:=%MW1, N:=1, R/W:=1, OUT=>%mw1);
```

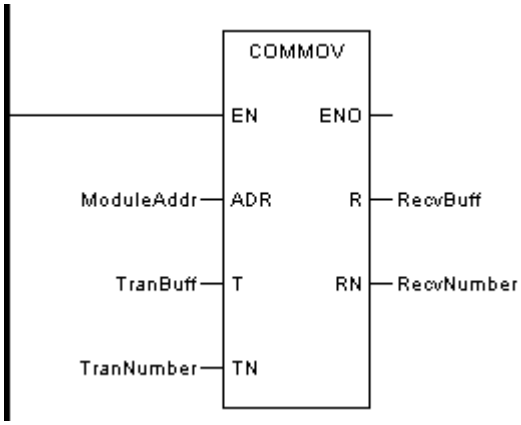
5.7.5 COMMOV: Передача коммуникационных данных

◆ Описание функции

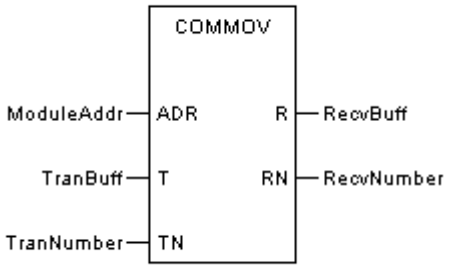
Этот блок функции используется для связи с коммуникационным модулем СК-4000, таким как модуль RS-232 или модуль PROFIBUS DP.

Более подробное описание – в документации на коммуникационные модули.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL COMMOV (ADR:=ModuleAddr, T:=TranBuff, TN:=TranNumber, R=>RecvBuff, RN=>RecvNumber)

◆ Отображение в ST

COMMOV (ADR:=ModuleAddr, T:=TranBuff, TN:=TranNumber, R=>RecvBuff, RN=>RecvNumber);

◆ Описание параметра

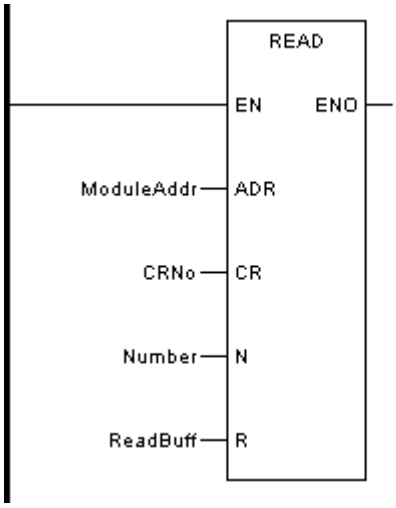
	Параметр	Описание	Тип данных	Тип точки
ADR	ModuleAddr	Адрес коммуникационного модуля	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
T	TranBuff	Буфер данных для отправки	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
TN	TranNumber	Количество байтов, подлежащих отправке	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
R	RecvBuff	Буфер данных, который должен быть получен	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
RN	RecvNumber	Количество байтов, которые должны быть получены	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.7.6 READ: Чтение данных из специального модуля

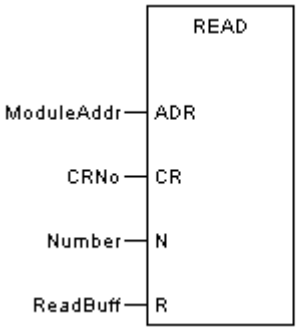
◆ Описание функции

Этот блок функции читает данные CR из специального модуля СК-4000, такого, как, например, модуль высокоскоростного счета. Более подробное описание – в документации на модули.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

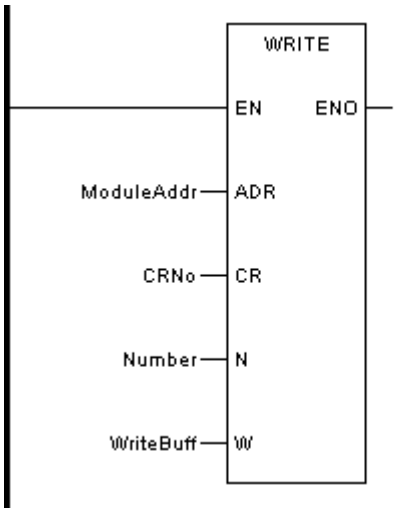
	Параметр	Описание	Тип данных	Тип точки
ADR	ModuleAddr	Адрес специального модуля	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
CR	CRNo	Начальный номер CR	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Количество CR	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
R	ReadBuff	Буфер данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.7 WRITE: Запись данных в специальный модуль

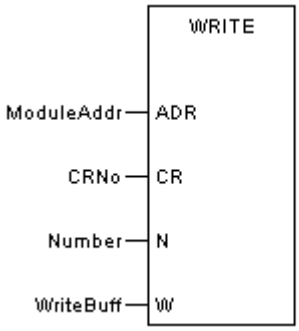
◆ Описание функции

Этот блок функции записывает данные CR в специальный модуль NA400, такой как модуль высокоскоростного счетчика. Более подробное описание – в документации на модули.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

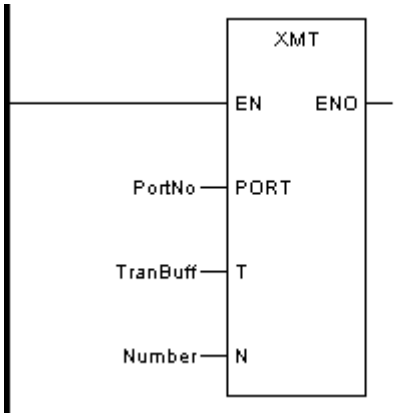
	Параметр	Описание	Тип данных	Тип точки
ADR	ModuleAddr	Адрес специального модуля	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
CR	CRNo	Начальный номер CR	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Количество CR	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
W	WriteBuff	Буфер данных	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.8 XMT: Передача в режиме свободного порта

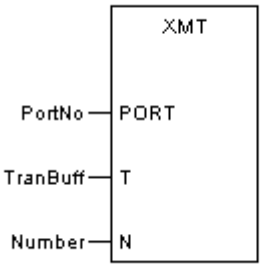
◆ Описание функции

Этот блок функции используется для передачи данных в режиме свободного порта COM1/COM2. Более подробное описание – в документации на модули.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

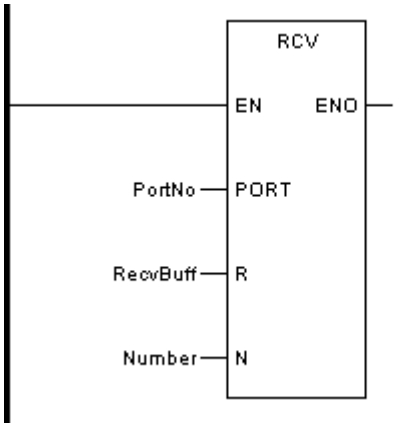
	Параметр	Описание	Тип данных	Тип точки
PORT	PortNo	Номер COM (1 или 2)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
T	TranBuff	Буфер данных для отправки	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
N	Number	Количество байтов, подлежащих отправке	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.7.9 RCV: Прием в режиме свободного порта

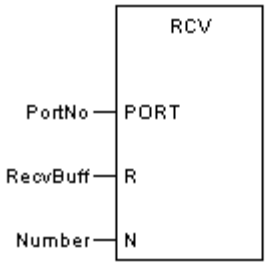
◆ Описание функции

Этот блок функции используется для приема данных в режиме свободного порта COM1/COM2. Более подробное описание – в документации на модули.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
CAL    RCV (PORT:=PortNo, R:=RecvBuff, N:=Number)
```

◆ Отображение в ST

```
RCV (PORT:=PortNo, R:=RecvBuff, N:=Number);
```

◆ Описание параметра

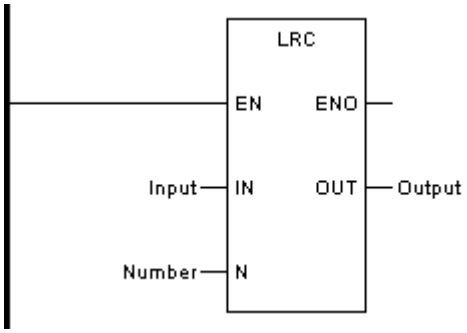
	Параметр	Описание	Тип данных	Тип точки
PORT	PortNo	Номер COM (1 или 2)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
R	RecvBuff	Буфер данных, который должен быть получен	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
N	Number	Количество байтов, которые должны быть получены	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.7.10 LRC: проверка LRC

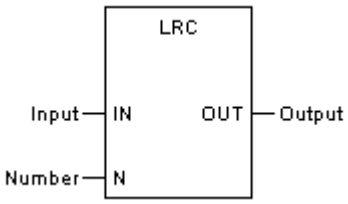
◆ Описание функции

Этот блок функции проводит расчёт контрольной суммы LRC.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

```
LRC (IN:=Input, N:=Number, OUT=>Output);
```

◆ Описание параметра

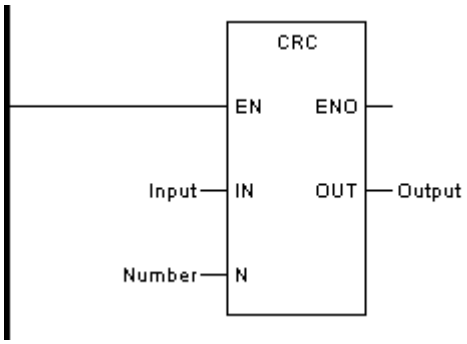
	Параметр	Описание	Тип данных	Тип точки
IN	Input	Данные на вводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
N	Number	Количество байтов	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод LRC	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.11 CRC: Проверка CRC

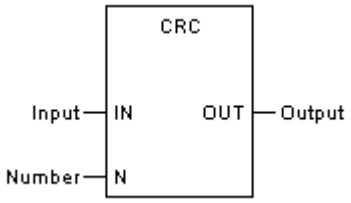
◆ Описание функции

Этот блок функции проводит расчёт контрольной суммы CRC.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

```
CRC (IN:=Input, N:=Number, OUT=>Output);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Данные на вводе	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
N	Number	Количество байтов	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
OUT	Output	Вывод CRC	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.12 MODRW: Чтение/запись данных через Modbus

◆ Описание функции

Этот блок функции используется для чтения или записи данных по мастер-протоколу MODBUS. Для этого режим порта в настройках должен быть установлен как Free port.

Состояние связи:

SW21 (состояние отправки COM1), SW23 (состояние отправки COM2)

0: в процессе отправки

1: успешная отправка

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

0: в процессе получения

1: успешное получение

2: Сбой COM 1/2

3: истечение времени приема

4: превышение максимального интервала между символами

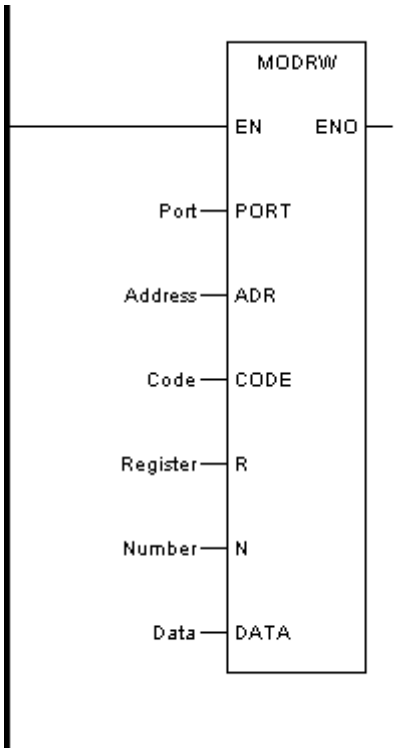
5: превышение максимального количества символов

7: ошибка ответного сообщения

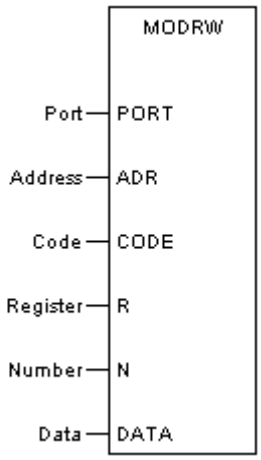
8: ошибка запроса сообщения

9: Ошибка проверки CRC16

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL MODRW (PORT:=Port, ADR:=Address, CODE:=Code, R:=Register, N:=Number, DATA:=Data)

◆ Отображение в ST

MODRW (PORT:=Port, ADR:=Address, CODE:=Code, R:=Register, N:=Number, DATA:=Data);

◆ Описание параметра

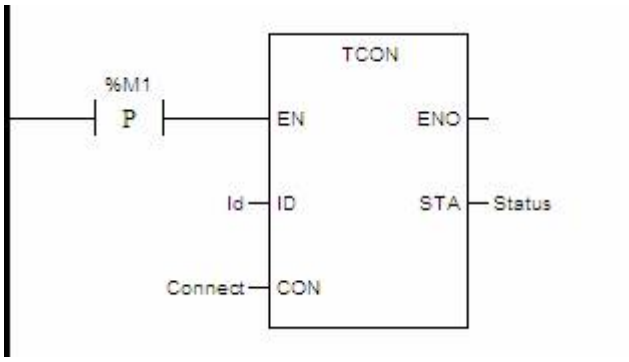
	Параметр	Описание	Тип данных	Тип точки
PORT	PortNo	Номер COM (1 или 2)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
ADR	Address	Адрес подчиненного устройства Modbus, 1~255	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
CODE	Code	Функциональный код протокола Modbus: 01, 02, 03, 04, 05, 06, 15, 16	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
R	Register	Регистр	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
N	Number	Количество регистров	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DATA	Data	Буфер данных	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	IW, QW, MW, NW, SW, I, Q, M, N, S, V

5.7.13 TCON: Активация Ethernet

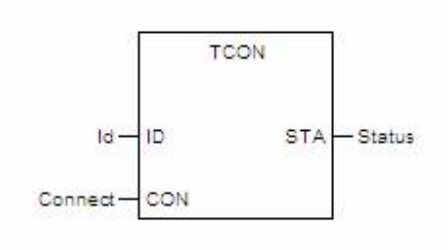
◆ Описание функции

Этот блок функции устанавливает коммуникационное соединение или регистрирует коммуникационную службу.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
CAL      TCON (ID:=Id, CON:=Connect, STA=>Status)
```

◆ Отображение в ST

```
TCON (ID:=Id, CON:=Connect, STA=>Status);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
ID	Id	Номер подключения, каждая установленная связь Ethernet с использованием уникального идентификационного номера, один и тот же набор коммуникаций в TDISCON, TUSEND, TRECV ID и номер TCON ID одинаковы	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
CON	Connect	Параметры настройки Ethernet	ETH_PARAM	V
STA	Status	Вывод состояния связи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

CON: Параметры настройки, тип данных: ETH_PARAM, конкретные настройки контактов CON следующие.

con	ETH_PARAM	1	%V00001		16	
TYPE	DWORD	1	%V00001	3	4	Protocol type (1: TCP client, 2: TCP server, 3: UDP
TIMEOUT	WORD	1	%V00005	10	2	Timeout (unit: 10ms)
PORT	WORD	1	%V00007	3000	2	Port number
IPADDR	BYTE[]	4	%V00009		4	IP address
IPADDR[0]	BYTE		%V00009	101	1	
IPADDR[1]	BYTE		%V00010	1	1	
IPADDR[2]	BYTE		%V00011	168	1	
IPADDR[3]	BYTE		%V00012	192	1	
RSVD	BYTE[]	4	%V00013		4	Reserved
RSVD[0]	BYTE		%V00013		1	
RSVD[1]	BYTE		%V00014		1	
RSVD[2]	BYTE		%V00015		1	
RSVD[3]	BYTE		%V00016		1	

TCP-Client: TYPE — 1, TIMEOUT — тайм-аут соединения/отправки/получения, единица измерения — 10 мс, IPADDR — удаленный IP-адрес (первый младший байт), PORT — номер удаленного порта;

TCP-Server: TYPE — 2, TIMEOUT — время ожидания отправки/получения, единица измерения — 10 мс, PORT — номер локального порта, другие значения можно игнорировать;

UDP: TYPE — 3, TIMEOUT — время ожидания отправки/получения, единица измерения — 10 мс, PORT — номер локального порта, другие значения можно игнорировать.

Параметр TYPE на рисунке выше имеет значение 3 – UDP коммуникация; TIMEOUT установлен равным 10, что означает 100 мс; Номер PORT установлен равным 3000 (не ставьте 502 номером порта); IPADDR указывает IP-адрес снизу вверх, а IP-адрес здесь установлен: 192.168.1.101; RSVD резервируется системой и не заполняется.

STA: Вывод состояния связи, тип данных WORD (то же самое, что и следующие функциональные блоки):

- 0x01: Операция продолжается
- 0x02: Операция прошла успешно
- 0x81: Внутренняя ошибка
- 0x82: Идентификатор подключен
- 0x83: Идентификатор не подключен
- 0x84: Ошибка подключения
- 0x85: Истекло время ожидания операции
- 0x86: Тип является незаконным
- 0x87: Достигнуто максимальное количество подключений

– 0x88: Слишком частые операции

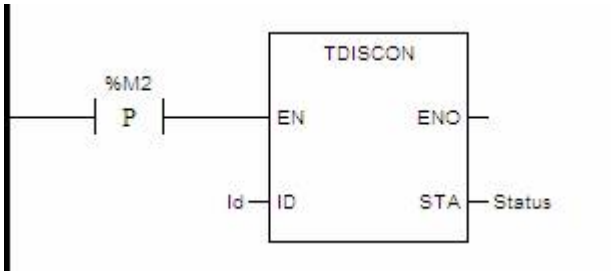
– 0x89: Ошибка отправки/получения

5.7.14 TDISCON: Деактивация Ethernet

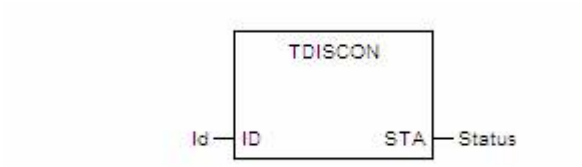
◆ Описание функции

Этот блок функции разрывает коммуникационное соединение или коммуникационную службу.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL TDISCON (ID:=Id, STA=>Status)

◆ Отображение в ST

TCON (ID:=Id, STA=>Status);

◆ Описание параметра

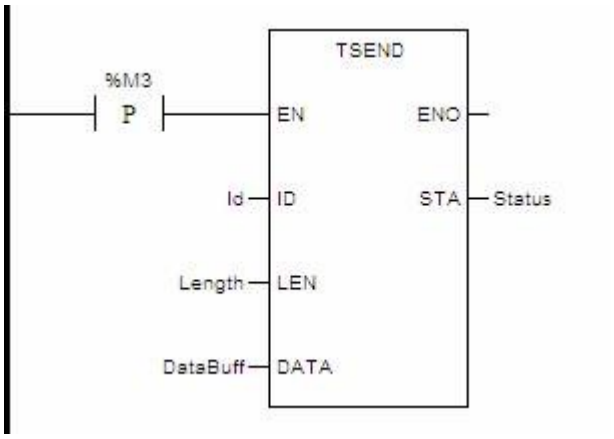
	Параметр	Описание	Тип данных	Тип точки
ID	Id	Номер подключения, каждая установленная связь Ethernet с использованием уникального идентификационного номера, один и тот же набор коммуникаций в TDISCON, TUSEND, TRECV ID и номер TCON ID одинаковы	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
STA	Status	Вывод состояния связи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.15 TSEND: Передача данных по TCP

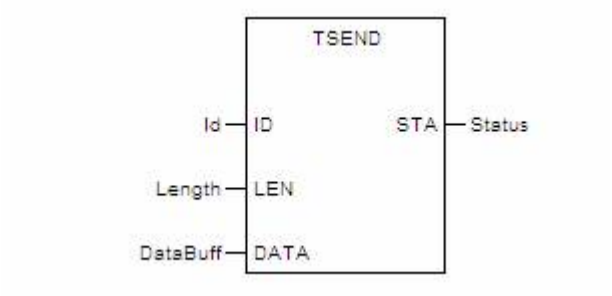
◆ Описание функции

Этот блок функции применяется для отправки данных по TCP.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

TSEND (ID:=Id, LEN:=Length, DATA:=DataBuff, STA=>Status);

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
ID	Id	Номер подключения должен совпадать с номером (ID) TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
LEN	Length	Количество отправляемых байтов (<8192 байтов)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DATA	DataBuff	Буфер данных для отправки	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

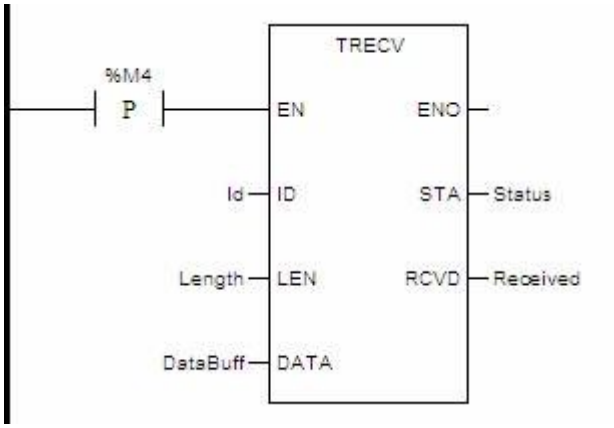
STA	Status	Вывод состояния связи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.16 TRECV: Прием данных по TCP

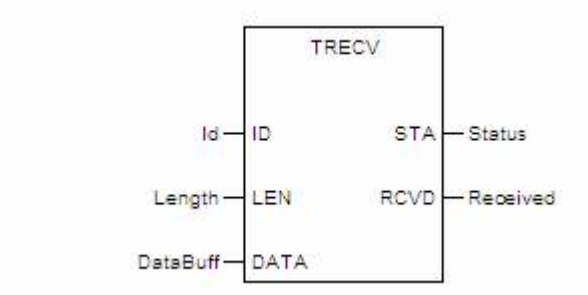
◆ Описание функции

Этот блок функции применяется для отправки данных по TCP.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
ID	Id	Номер подключения должен совпадать с номером (ID) TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

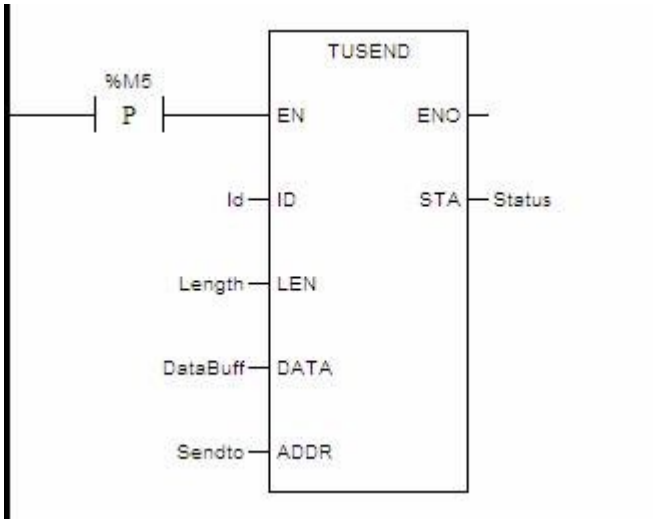
LEN	Length	Количество принимаемых байтов (<8192 байтов)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DATA	DataBuff	Буфер данных для приёма	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
STA	Status	Вывод состояния связи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
RCVD	Recieved	Количество принятых байтов	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.17 TUSEND: Отправка датаграмм UDP

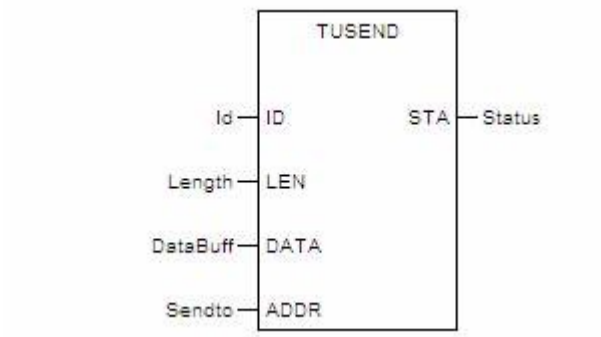
◆ Описание функции

Этот блок функции применяется для отправки датаграмм UDP.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

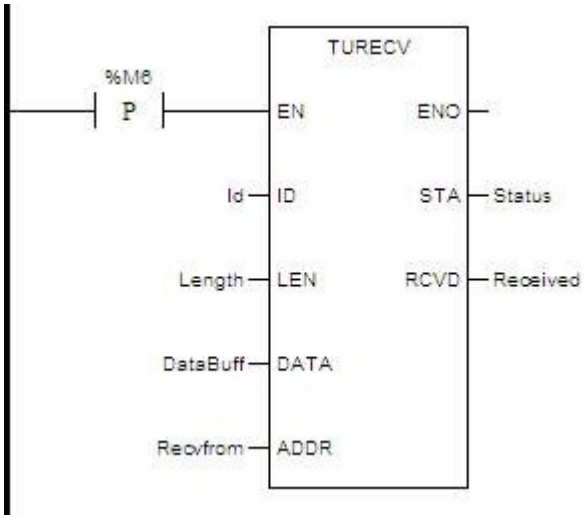
	Параметр	Описание	Тип данных	Тип точки
ID	Id	Номер подключения должен совпадать с номером (ID) TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
LEN	Length	Количество отправляемых байтов (<1472 байта)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DATA	DataBuff	Буфер данных для отправки	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
ADDR	Sendto	Параметры настройки Ethernet: IPADDR – это IP адрес, PORT – номер порта	ETH_PARAM	V
STA	Status	Вывод состояния связи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.7.18 TURECV: Прием датаграмм UDP

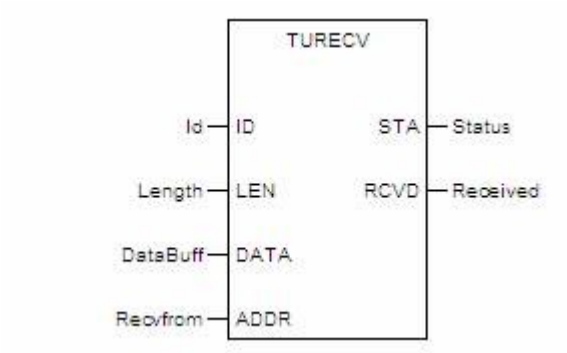
◆ Описание функции

Этот блок функции применяется для приёма датаграмм UDP.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL TURECV (ID:=Id, LEN:=Length, DATA:=DataBuff, ADDR:=Recvdfrom, STA=>Status, RCVD=>Received);

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
ID	Id	Номер подключения должен совпадать с номером (ID) TCON	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
LEN	Length	Количество принимаемых байтов (<1472 байта)	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DATA	DataBuff	Буфер данных для приема	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
ADDR	Recvfrom	Параметры настройки Ethernet: IPADDR – это IP адрес, PORT – номер порта	ETH_PARAM	V
STA	Status	Вывод состояния связи	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V
RCVD	Recieved	Количество принятых байтов	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.8 Таймеры

В этом разделе описываются базовые блоки функций, связанные с таймерами.

Данный раздел содержит следующие подразделы.

Тип	Описание
TON	Таймер задержки включения (Секунда)
TOF	Таймер задержки выключения (Секунда)
TP	Импульсный таймер (Секунда)
TON_MS	Таймер задержки включения (Миллисекунда)
TOF_MS	Таймер задержки выключения (Миллисекунда)
TP_MS	Импульсный таймер (Миллисекунда)
TON_M	Таймер задержки включения (Минута)
TOF_M	Таймер задержки выключения (Минута)
TP_M	Импульсный таймер (Минута)
TON_H	Таймер задержки включения (Час)
TOF_H	Таймер задержки выключения (Час)
TP_H	Импульсный таймер (Час)

В User Defined блоках программ переменные %T нельзя использовать как неопределённые элементы массива.

%T[NUM] вызовет ошибку при компиляции, %T[5] не вызовет ошибку при компиляции User Defined блоков.

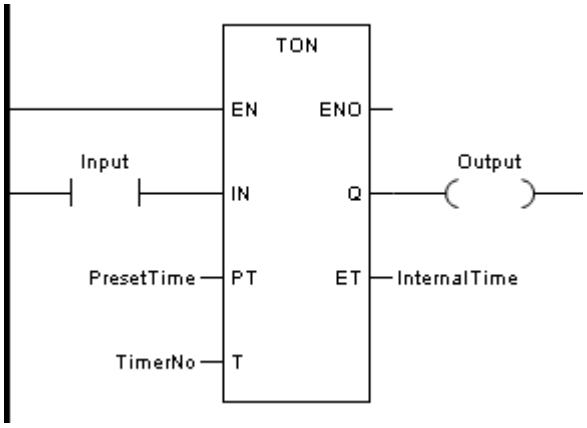
5.8.1 TON: Таймер задержки включения

◆ Описание функции

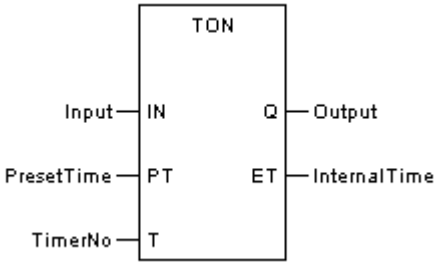
Этот блок функции используется в качестве задержки включения. Когда блок функции вызывается в первый раз, начальное состояние ET равно “0”.

TON, TON_MS, TON_M и TON_H идентичны, за исключением их единиц измерения.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL TON (IN:=Input, PT:=PresetTime, T:=TimerNo, Q=>Output, ET=>InternalTime) (ET можно не учитывать)

◆ Отображение в ST

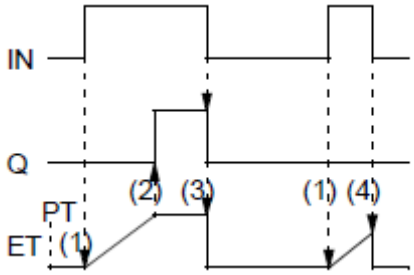
TON (IN:=Input, PT:=PresetTime, T:=TimerNo, Q=>Output, ET=>InternalTime); (ET можно не учитывать)

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Запустить задержку	BOOL	Константа, I, Q, M, N, S, V
PT	PresetTime	Заданное время задержки	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
T	TimerNo	Номер таймера		T
Q	Output	Вывод	BOOL	Q, M, N, V
ET	InternalTime	Внутреннее время	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

◆ Диаграмма тайминга

Отображение задержки включения:



(1) Если IN становится “1”, начинается отсчет внутреннего времени (ET).
(2) Если внутреннее время достигает значения PT, то Q становится “1”.
(3) Если IN становится “0”, то Q становится “0”, и внутреннее время останавливается/сбрасывается.
(4) Если IN становится “0” до того, как внутреннее время достигнет значения PT, то внутреннее время останавливается/сбрасывается без перехода Q на “1”.

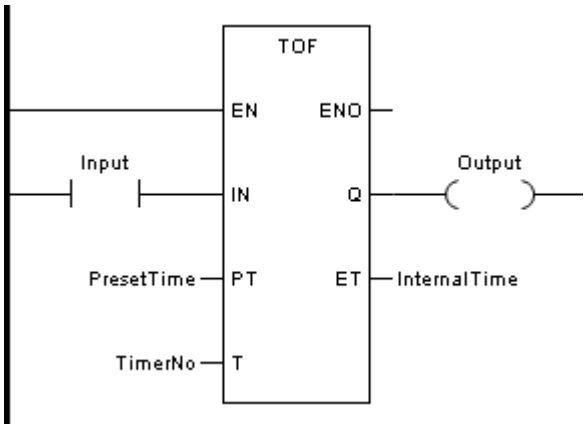
5.8.2 TOF: Таймер задержки выключения

◆ Описание функции

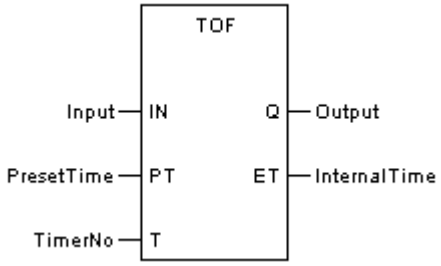
Этот блок функции используется в качестве задержки выключения. Когда блок функции вызывается в первый раз, начальное состояние ET равно “0”.

TOF, TOF_MS, TOF_M и TOF_H идентичны, за исключением их единиц измерения.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL TOF (IN:=Input, PT:=PresetTime, T:=TimerNo, Q=>Output, ET=>InternalTime) (ET можно не учитывать)

◆ Отображение в ST

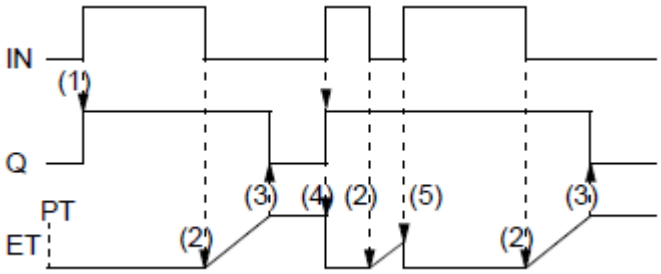
TOF (IN:=Input, PT:=PresetTime, T:=TimerNo, Q=>Output, ET=>InternalTime); (ET можно не учитывать)

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Запустить задержку	BOOL	Константа, I, Q, M, N, S, V
PT	PresetTime	Заданное время задержки	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
T	TimerNo	Номер таймера		T
Q	Output	Вывод	BOOL	Q, M, N, V
ET	InternalTime	Внутреннее время	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

◆ Диаграмма тайминга

Отображение задержки выключения TOF:



(1) Если IN становится “1”, то Q становится “1”.

(2) Если IN становится “0”, начинается отсчет внутреннего времени (ET).

(3) Если внутреннее время достигает значения PT, то Q становится “0”.

(4) Если IN становится “1”, то Q становится “1”, и внутреннее время останавливается/сбрасывается.

(5) Если IN становится “1“ до того, как внутреннее время достигнет значения PT, то внутреннее время останавливается/сбрасывается без возврата Q к значению ”0".

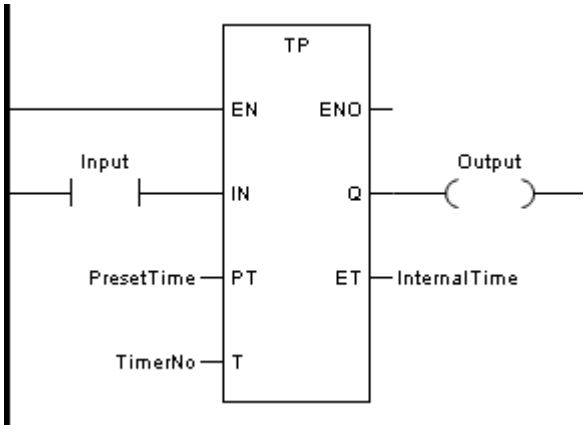
5.8.3 TP: Импульсный таймер

◆ Описание функции

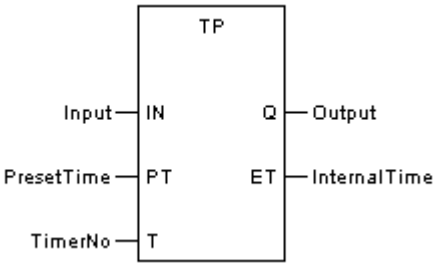
Этот блок функции используется для генерации импульса с определенной длительностью. Когда блок функции вызывается в первый раз, начальное состояние ET равно “0”.

TP, TP_MS, TP_M и TP_H идентичны, за исключением их единиц измерения.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL TP(IN:=Input, PT:=PresetTime, T:=TimerNo, Q=>Output, ET=>InternalTime) (ET можно не учитывать)

◆ Отображение в ST

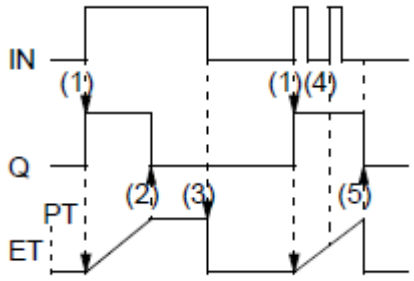
TP (IN:=Input, PT:=PresetTime, T:=TimerNo, Q=>Output, ET=>InternalTime); (ET можно не учитывать)

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
IN	Input	Пусковой импульс	BOOL	Константа, I, Q, M, N, S, V
PT	PresetTime	Предустановленная длительность импульса	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
T	TimerNo	Номер таймера		T
Q	Output	Вывод	BOOL	Q, M, N, V
ET	InternalTime	Внутреннее время	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

◆ Диаграмма тайминга

Отображение импульса TP:



(1) Если IN становится “1”, то Q становится “1” и начинается отсчет внутреннего времени (ET).

(2) Если внутреннее время достигает значения PT, то Q становится “0” (независимо от IN).

(3) Внутреннее время останавливается/сбрасывается, если IN становится “0”.

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

(5) Если внутреннее время достигло значения PT, а IN равно “0”, то внутреннее время останавливается/сбрасывается, и Q становится “0”.

5.9 Счетчики

В этом разделе описываются базовые блоки функций, связанные с счетчиками.

Данный раздел содержит следующие подразделы.

Тип	Описание
CTU	Счет на увеличение
CTD	Счет на уменьшение
CTUD	Счет на увеличение/уменьшение

В User Defined блоках программ переменные %C нельзя использовать как неопределённые элементы массива. %C[NUM] вызовет ошибку при компиляции, %C[5] не вызовет ошибку при компиляции User Defined блоков.

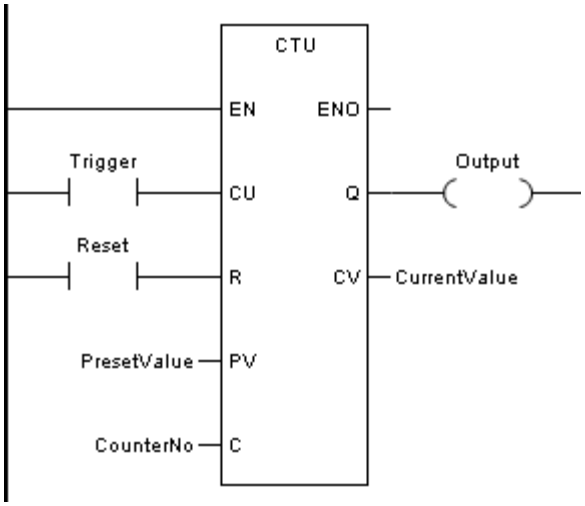
5.9.1 CTU: Счет на увеличение

◆ Описание функции

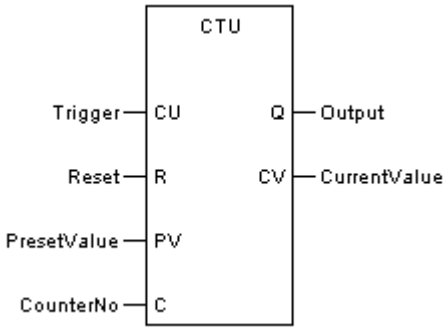
Этот блок функции используется для счета на увеличение.

Сигнал “1” на вводе R приводит к присвоению выводу CV значения “0”. При каждом переходе от “0” к “1” на вводе CU значение CV увеличивается на 1. Когда CV ≥ PV, то выводу Q присваивается значение “1” и дальнейший отсчёт прекращается.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
CU	Trigger	Триггерный ввод	BOOL	Константа, I, Q, M, N, S, V
R	Reset	Сброс	BOOL	Константа, I, Q, M, N, S, V
PV	PresetValue	Заданное значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
C	CounterNo	Номер счетчика		C
Q	Output	Вывод	BOOL	Q, M, N, V
CV	CurrentValue	Текущее значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

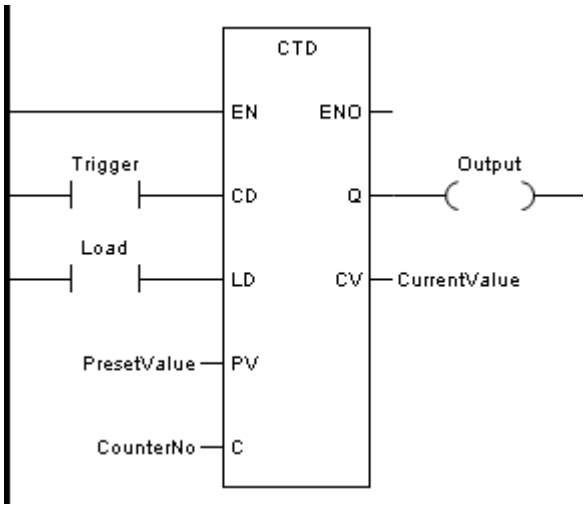
5.9.2 CTD: Счет на уменьшение

◆ Описание функции

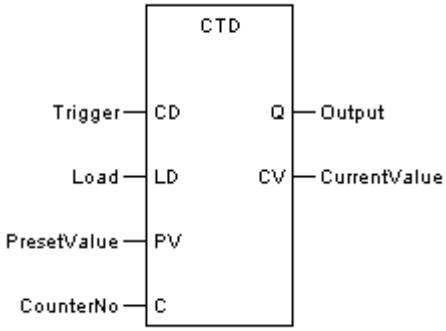
Этот блок функции используется для счета на уменьшение.

Сигнал «1» на вводе LD приводит к тому, что значение вводного сигнала PV передается на вывод CV. Отсчет начинается, если LD возвращается в состояние «0». При каждом переходе от «0» к «1» на вводе CD значение CV уменьшается на 1. Когда CV ≤ 0, то выводу Q присваивается значение «1».

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
CD	Trigger	Триггерный ввод	BOOL	Константа, I, Q, M, N, S, V
LD	Загрузка	Загрузка	BOOL	Константа, I, Q, M, N, S, V
PV	PresetValue	Заданное значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
C	CounterNo	Номер счетчика		C
Q	Output	Вывод	BOOL	Q, M, N, V
CV	CurrentValue	Текущее значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.9.3 CTUD: Счет на уменьшение/увеличение

◆ Описание функции

Этот блок функции используется для счета на уменьшение/увеличение.

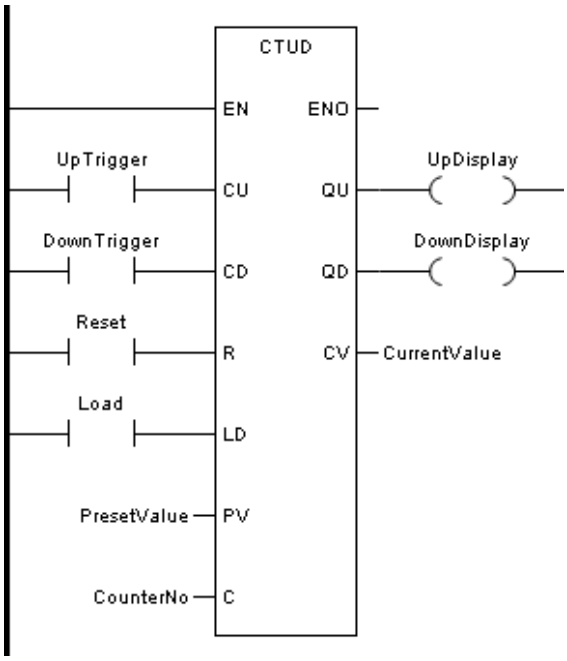
Сигнал “1” на вводе R привел к присвоению выводу CV значения “0”. Сигнал “1” на вводе LD привел к тому, что значение вводного сигнала PV было передано на вывод CV. При каждом переходе от “0” к “1” на вводе CU значение CV увеличивается на 1. При каждом переходе от “0” к “1” на вводе CD значение CV уменьшается на 1.

Если на вводы R и LD одновременно подается сигнал “1”, то ввод R имеет приоритет.

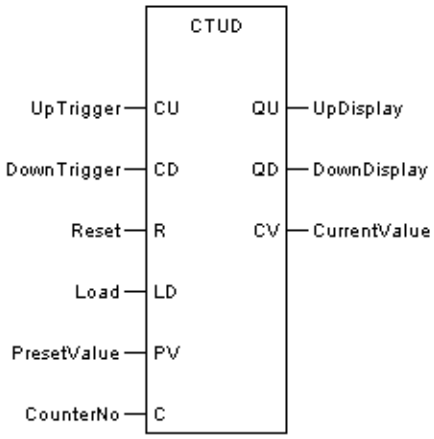
Когда CV ≥ PV, то выводу QU присваивается значение “1”.

Когда CV ≤ 0, то выводу QD присваивается значение “1”.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
CU	UpTrigger	Ввод триггера счета на увеличение	BOOL	Константа, I, Q, M, N, S, V

CD	DownTrigger	Ввод триггера счета на уменьшение	BOOL	Константа, I, Q, M, N, S, V
R	Reset	Сброс	BOOL	Константа, I, Q, M, N, S, V
LD	Загрузка	Загрузка	BOOL	Константа, I, Q, M, N, S, V
PV	PresetValue	Заданное значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
C	CounterNo	Номер счетчика		C
QU	UpDisplay	Отображение значений на увеличение	BOOL	Q, M, N, V
QD	DownDisplay	Отображение значений на уменьшение	BOOL	Q, M, N, V
CV	CurrentValue	Текущее значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	MW, NW, V

5.10 Управление программой (Control)

В этом разделе описываются базовые блоки функций, связанные с управлением программой.

Данный раздел содержит следующие подразделы

Тип	Описание
CALL	Вызвать программу
EXEC	Выполнить SCC
KILL	Остановить SCC
LOCK	Зафиксировать SCC
UNLOCK	Снять фиксацию SCC

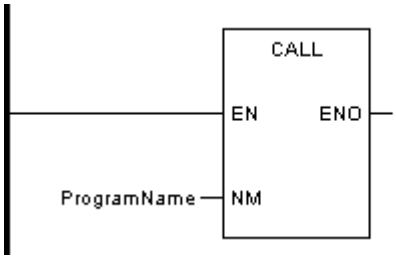
5.10.1 CALL: Вызвать программу

◆ Описание функции

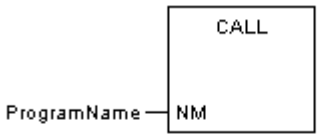
Этот блок функции вызывает соответствующую подпрограмму (другую программу LD, FBD, IL, ST).

Когда вызывается блок функции, он дает сканированию команду немедленно перейти к указанной подпрограмме и выполнить ее. После завершения выполнения подпрограммы управление возвращается на ступень логики, следующую непосредственно за функциональным блоком CALL.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL ProgramName

◆ Отображение в ST

ProgramName ();

◆ Описание параметра

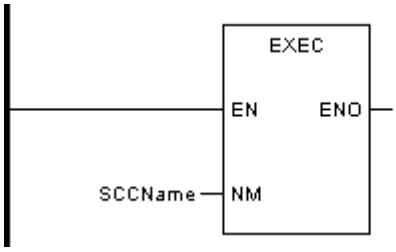
	Параметр	Описание	Тип данных	Тип точки
NM	ProgramName	Название программы	Не применимо	Не применимо

5.10.2 EXEC: Выполнить SCC

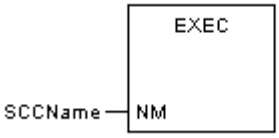
◆ Описание функции

Этот блок функции запускает соответствующую программу SCC.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL EXEC (SCCName)

◆ Отображение в ST

EXEC (SCCName);

◆ Описание параметра

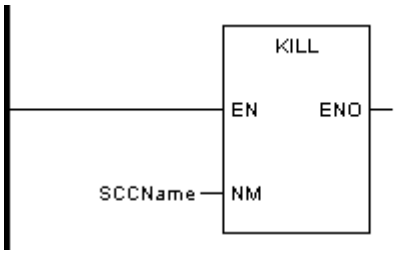
	Параметр	Описание	Тип данных	Тип точки
NM	SCCName	Название SCC	Не применимо	Не применимо

5.10.3 KILL: Остановить SCC

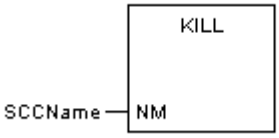
◆ Описание функции

Этот блок функции останавливает соответствующую программу SCC.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL KILL (SCCName)

◆ Отображение в ST

KILL (SCCName);

◆ Описание параметра

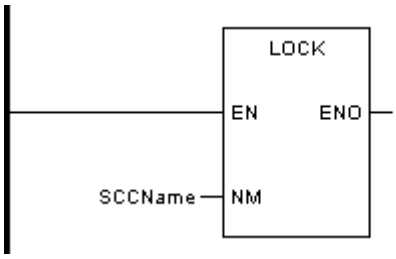
	Параметр	Описание	Тип данных	Тип точки
NM	SCCName	Название SCC	Не применимо	Не применимо

5.10.4 LOCK: Зафиксировать SCC

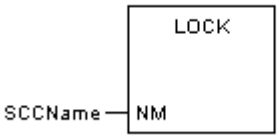
◆ Описание функции

Этот блок функции фиксирует соответствующую программу SCC.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL LOCK (SCCName)

◆ Отображение в ST

LOCK (SCCName);

◆ Описание параметра

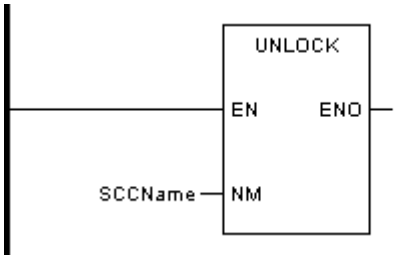
	Параметр	Описание	Тип данных	Тип точки
NM	SCCName	Название SCC	Не применимо	Не применимо

5.10.5 UNLOCK: Снять фиксацию SCC

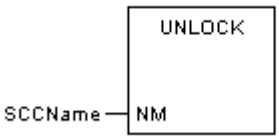
◆ Описание функции

Этот блок функции снимает фиксацию с соответствующей программы SCC.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
CAL    UNLOCK (SCCName)
```

◆ Отображение в ST

```
UNLOCK (SCCName);
```

◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
NM	SCCName	Название SCC	Не применимо	Не применимо

5.11 Блоки функций ПЛК

В этом разделе описываются блоки функций, связанные с функционалом ПЛК.

Данный раздел содержит следующие подразделы.

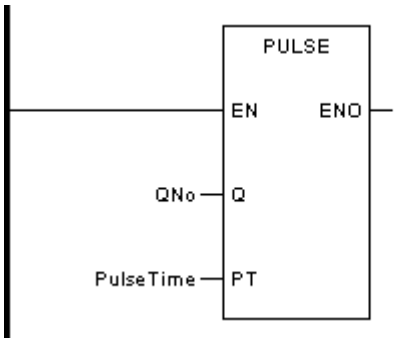
Тип	Описание
PULSE	Импульсный цифровой вывод
AOUT	Аналоговый вывод
FORCE	Принудительный ввод точки
UNFORCE	Отмена принудительного ввода
SWITCH	Переключение master/slave
ENI	Включить прерывания
DISI	Отключить прерывания

5.11.1 PULSE: Импульсный цифровой вывод

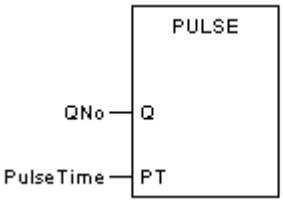
◆ Описание функции

Этот блок функции устанавливает точку цифрового вывода на “1” и удерживает в течение указанного времени, а затем автоматически переключается на “0”.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

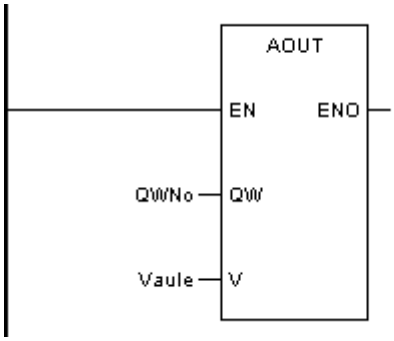
	Параметр	Описание	Тип данных	Тип точки
Q	QNo	Точка импульсного цифрового вывода		Q
PT	PulseTime	Время импульса, единица измерения: мс, 10~600000	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.11.2 AOUT: Аналоговый вывод

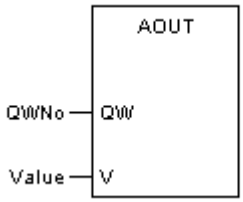
◆ Описание функции

Этот блок функции устанавливает точку аналогового вывода на заданное в коде ЦАП значение. Необходимо соблюдать пределы ЦАП канала модуля.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

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

◆ Отображение в ST

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

◆ Описание параметра

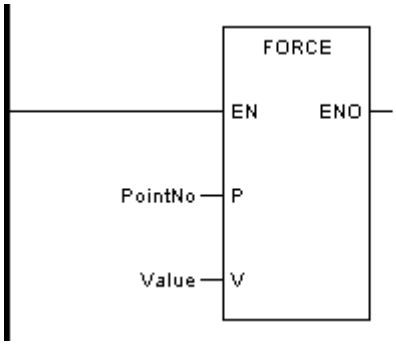
	Параметр	Описание	Тип данных	Тип точки
QW	QWNo	Точка аналогового вывода		QW
V	Значение	Указанное значение	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

5.11.3 FORCE: Принудительный ввод точки

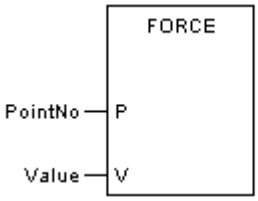
◆ Описание функции

Этот блок функции принудительно вводит заданное значение для точки (I, Q, IW, QW). Необходимо соблюдать пределы ЦАП канала модуля.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
CAL    FORCE (P:=PointNo, V:=Value)
```

◆ Отображение в ST

```
FORCE (P:=PointNo, V:=Value);
```

◆ Описание параметра

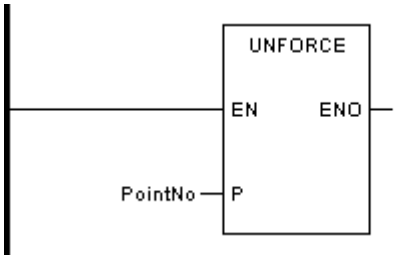
	Параметр	Описание	Тип данных	Тип точки
P	PointNo	Точка, значение которой нужно принудительно ввести		I, Q, IW, QW
V	Значение	Значение, которое нужно принудительно ввести	BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V

5.11.4 UNFORCE: Отмена принудительного ввода

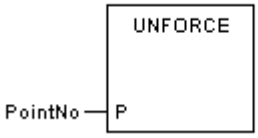
◆ Описание функции

Этот блок функции отменяет принудительный ввод для точки (I, Q, IW, QW).

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

```
CAL    UNFORCE (PointNo)
```

◆ Отображение в ST

```
UNFORCE (PointNo);
```

◆ Описание параметра

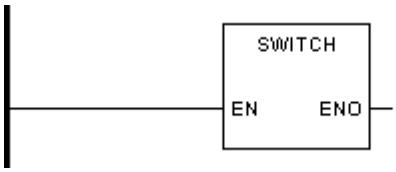
	Параметр	Описание	Тип данных	Тип точки
P	PointNo	Точка, принудительный ввод значения которой нужно отменить		I, Q, IW, QW

5.11.5 SWITCH: Переключение резервированных ЦПУ

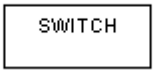
◆ Описание функции

Этот блок функции действителен только при использовании резервированной системы. Когда система оснащена двумя процессорами, один процессор используется в качестве резервируемого (master), а второй - в качестве резервного (slave). При каждом вызове функционального блока текущий резервируемый процессор станет резервным, а резервный - резервируемым. Осторожно, контролируйте переключение ПЛК с помощью системных бит "CPU1_MASTER", "CPU2_MASTER".

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL SWITCH ()

◆ Отображение в ST

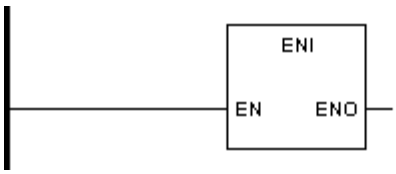
SWITCH ();

5.11.6 ENI: Включить прерывания

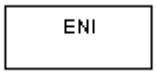
◆ Описание функции

Этот блок функции позволяет прерывать работу ПЛК. Как только блок функции вызван, при возникновении прерывания выполняется условие для вызова соответствующей программы прерывания.

◆ Отображение в LD



◆ Отображение в FBD



◆ Отображение в IL

CAL ENI ()

◆ Отображение в ST

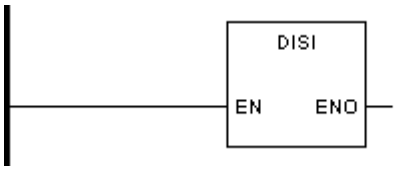
ENI ();

5.11.7 DISI: Отключить прерывания

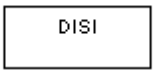
◆ Описание функции

Этот блок функции отключает прерывание ПЛК. После вызова функционального блока, когда происходит прерывание, условие для вызова соответствующей программы прерывания не выполняется.

◆ Отображение в LD



- Отображение в FBD



- Отображение в IL

```
CAL    DISI ()
```

- Отображение в ST

```
DISI ();
```

5.12 Прочие блоки функций (Others)

В этой главе описываются элементарные функциональные блоки семейства Others (Прочее).

В этом разделе описываются прочие блоки функций

Данный раздел содержит следующие подразделы.

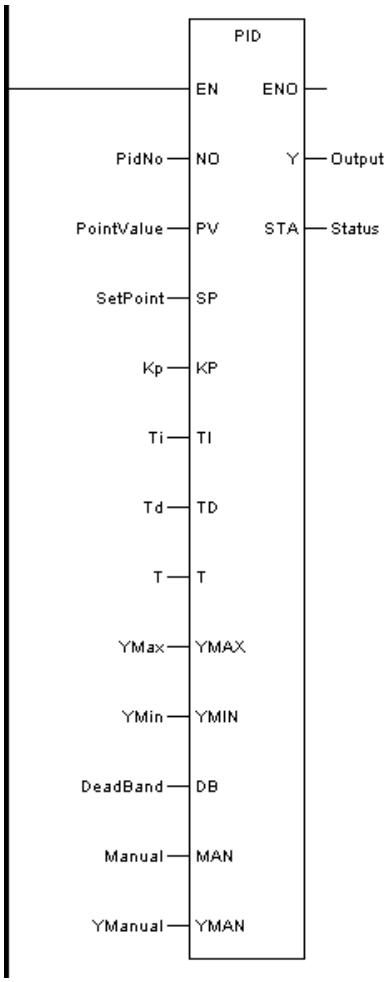
Тип	Описание
PID	ПИД-регулятор

5.12.1 PID: ПИД-регулятор

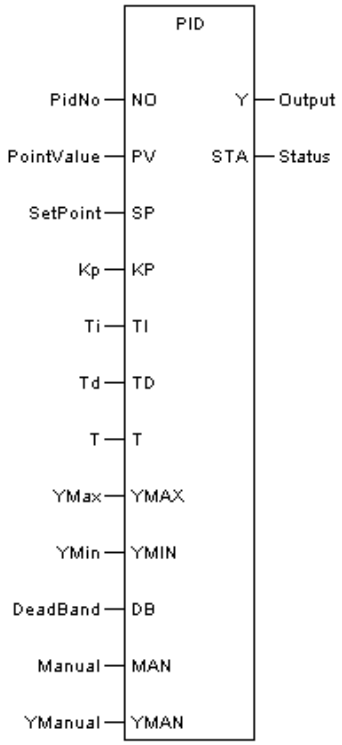
◆ Описание функции

Этот блок функции создает ПИД-регулятор.

◆ Отображение в LD



◆ Отображение в FBD



◆ Описание параметра

	Параметр	Описание	Тип данных	Тип точки
NO	PidNo	Серийный номер ПИД (см. руководство на используемый ПЛК)	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
PV	PointValue	Значение измерений контролируемой переменной	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	IW, MW, NW, V
SP	Setpoint	Задать значение	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
KP	Kp	Коэффициент пропорциональности	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
TI	Ti	Интегральное время (единица измерения: с) ≥0.01	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
TD	Td	Дифференциальное время (единица измерения: с)	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V

T	T	Время выборки (единица измерения: с) 0,01~20	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
YMAX	YMax	Верхний предел производительности	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
YMIN	YMIN	Нижний предел производительности	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
DB	DeadBand	Полоса непропускания сигнала	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
MAN	Manual	1 – Ручной режим, 0 - Автоматический режим	BOOL	Константа, I, Q, M, N, S, V
YMAN	YManual	Значение, вручную заданное на вывод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	Константа, IW, QW, MW, NW, SW, V
Y	Output	Вывод	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V
STA	Status	Состояние: 1. Ошибка инициализации 2. ПИД ВКЛЮЧЕН 3. ПИД ВЫКЛЮЧЕН 4. Ручной режим 5. Вывод > YMAX 6. Вывод < YMIN 7. YMAX < YMIN	BYTE, WORD, DWORD, SINT, INT, DINT, REAL, USINT, UINT, UDINT	MW, NW, V

5.13 Блок функции, определяемый пользователем

При написании программ пользователь может самостоятельно определить функциональные блоки (раздел User Defined FB). Данные блоки преимущественно используются для многочисленной обработки однотипных данных (входные/выходные сигналы ПЛК).

Блоки функций, определяемые пользователем можно реализовать только на языках LD, FBD ST, C/C++.

5.13.1 Новый пользовательский блок функций

Для создания FB необходимо во вкладке «Data» окна «Project Browser» открыть «DFB Type Table», и в появившемся окне «DFB Type Table» заполнить поле «Name» и выбрать тип программы функционального блока из поля «Type», как показано на рис.5.3:

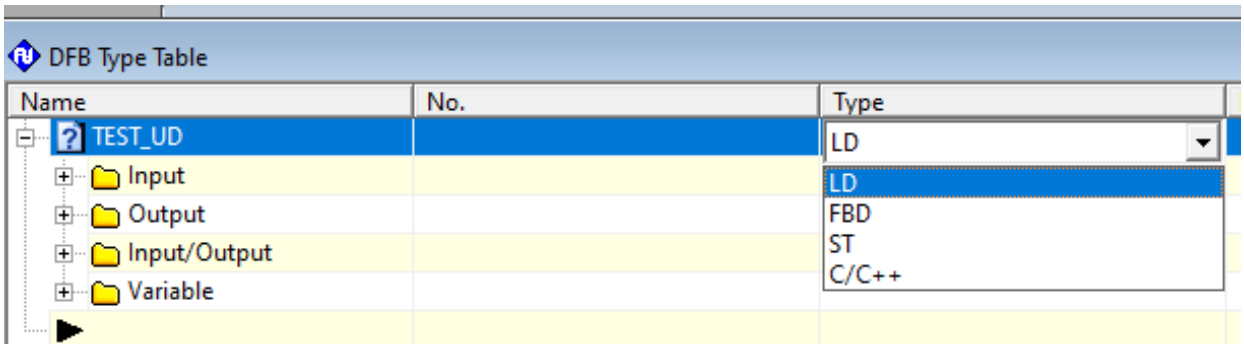


Рис.5.3 Новый пользовательский FB

Введите описание, вводы, выводы и внутренние переменные в диалоговом окне, как показано на рис.5.4:

Name	No.	Type	Dimension	Value	Comment
TEST_UD		ST			
Input					
EN	1	BOOL		1	
SAMPLETIME_IN	2	INT		0	
IN	3	Test_In			
IN1		INT	1		
In2		INT	1		
Output					
ENO	1	BOOL		1	
OUT	2	Test_Out			
Out1		INT	1		
Input/Output					
INOUT	1	Test_InOut			
Inout1		BOOL	1		
Variable					

DFB Type Table | MAIN

Рис.5.4 Определение параметров FB

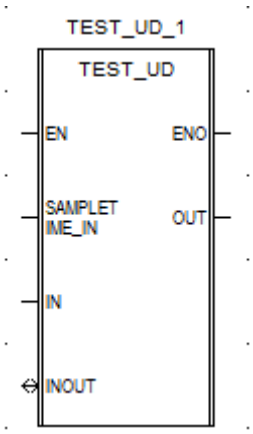


Рис.5.5 Упакованный FB

После компиляции проекта во вкладке «Program» окна «Project Browser» в группе «User Defined FB» – «Basic» появится новый FB с заданным ранее именем. После написания программы и её успешной компиляции упакованный блок функций находится в разделе **【User defined】** панели инструментов блоков функций. Метод использования такой же, как и у других базовых блоков функций.

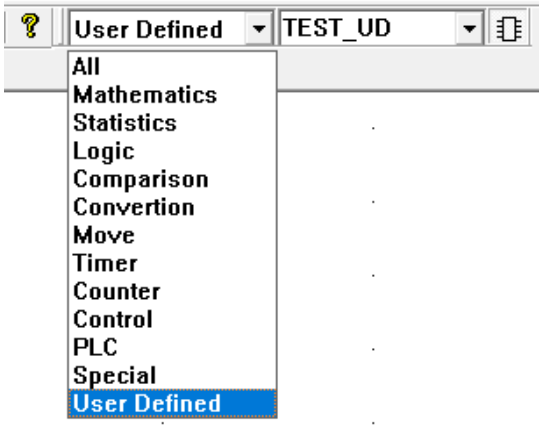


Рис.5.6 Выбор упакованного блока функций

5.13.2 Редактирование программы блока функций

Дважды щелкните на названии новой программы блока функций в браузере проекта. В открывшемся окне можно редактировать как саму программу, так и другие подпрограммы. В программе блока функций предпочтительно использование внутренних переменных, потому что они являются локальными. При использовании глобальных переменных MW, NW, N, M, V и т.д., убедитесь, что они не конфликтуют друг с другом в других программах.

Опишите необходимые входные и выходные данные и привяжите их к блоку функции, как показано на рисунке5.7:

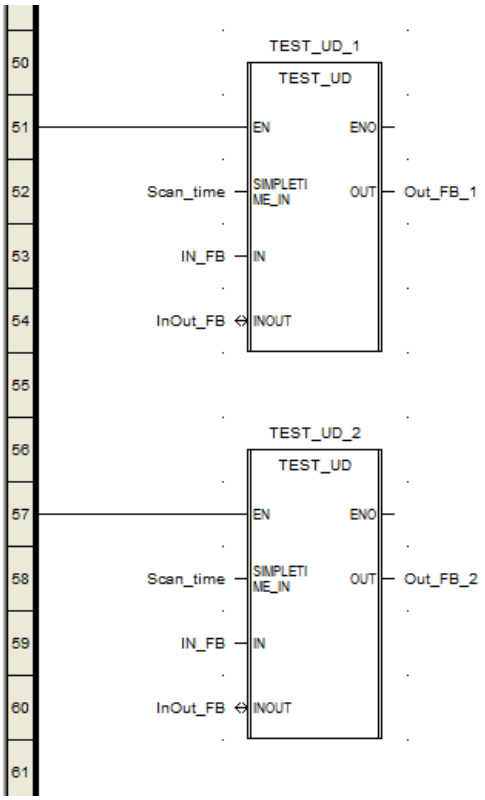


Рис.5.7 Редактирование программы блока функций

5.13.3 Исполняемая копия блока функций

Во вкладке 【**Data**】 браузера проекта (рис.5.8) дважды щелкните **DFB instance Table**. В открывшемся диалоговом окне, (рис.5.9) отредактируйте исполняемую копию блока функций.

Применение исполняемой копии блока функции: когда в программе блока функции используются таймеры и счетчики, чтобы избежать их конфликтов, необходимо определить разные исполняемые копии блока функции, то есть необходимо определить несколько исполняемых копий блока функции, когда один упакованный блок функции необходимо повторно использовать в других программах. Если в упакованном блоке функции не используются таймеры или счетчики, то вручную добавлять исполняемую копию блока функции нет необходимости, так как система добавит ее автоматически, и ее можно будет использовать повторно.

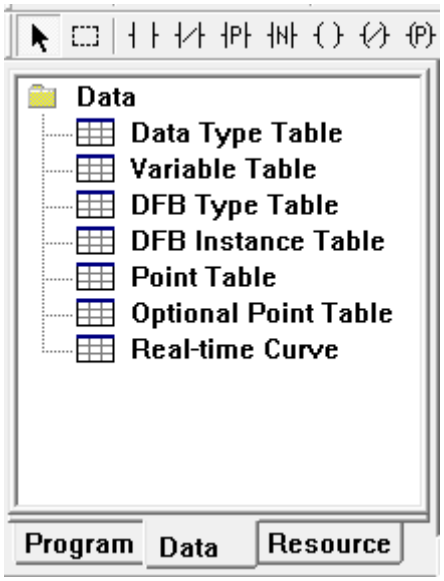
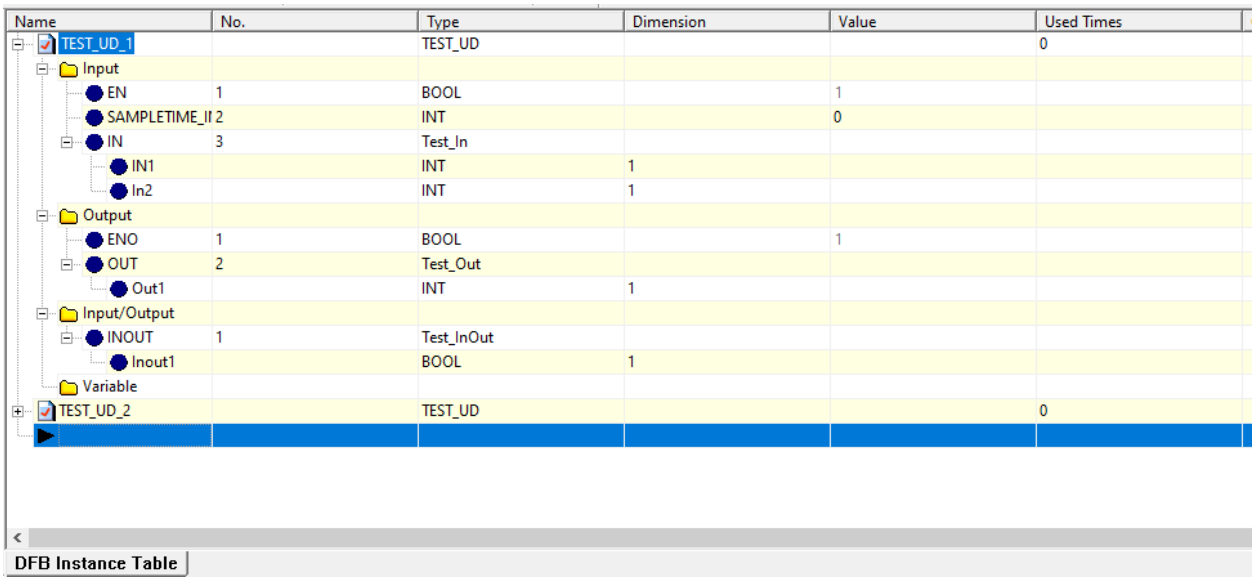


Рис.5.8 Интерфейс вкладки данных



Name	No.	Type	Dimension	Value	Used Times
TEST_UD_1		TEST_UD			0
Input					
EN	1	BOOL		1	
SAMPLETIME_II	2	INT		0	
IN	3	Test_In			
IN1		INT	1		
In2		INT	1		
Output					
ENO	1	BOOL		1	
OUT	2	Test_Out			
Out1		INT	1		
Input/Output					
INOUT	1	Test_InOut			
Inout1		BOOL	1		
Variable					
TEST_UD_2		TEST_UD			0

DFB Instance Table

Рис.5.9 Исполняющая копия блока функции

В процессе использования блока функции название добавленной исполняемой копии должно быть изменено вручную в диалоговом окне **DFB instance Table**. Название новой исполняемой копии блока функции по умолчанию - “XXX_1”. Чтобы изменить блок функции, щелкните по нему правой кнопкой мыши и выберите **Property**. Откроется диалоговое окно, как показано на рис.5.10.

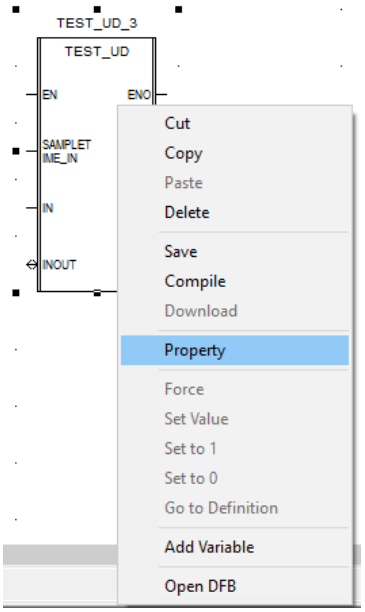


Рис.5.10 Свойство блока функции

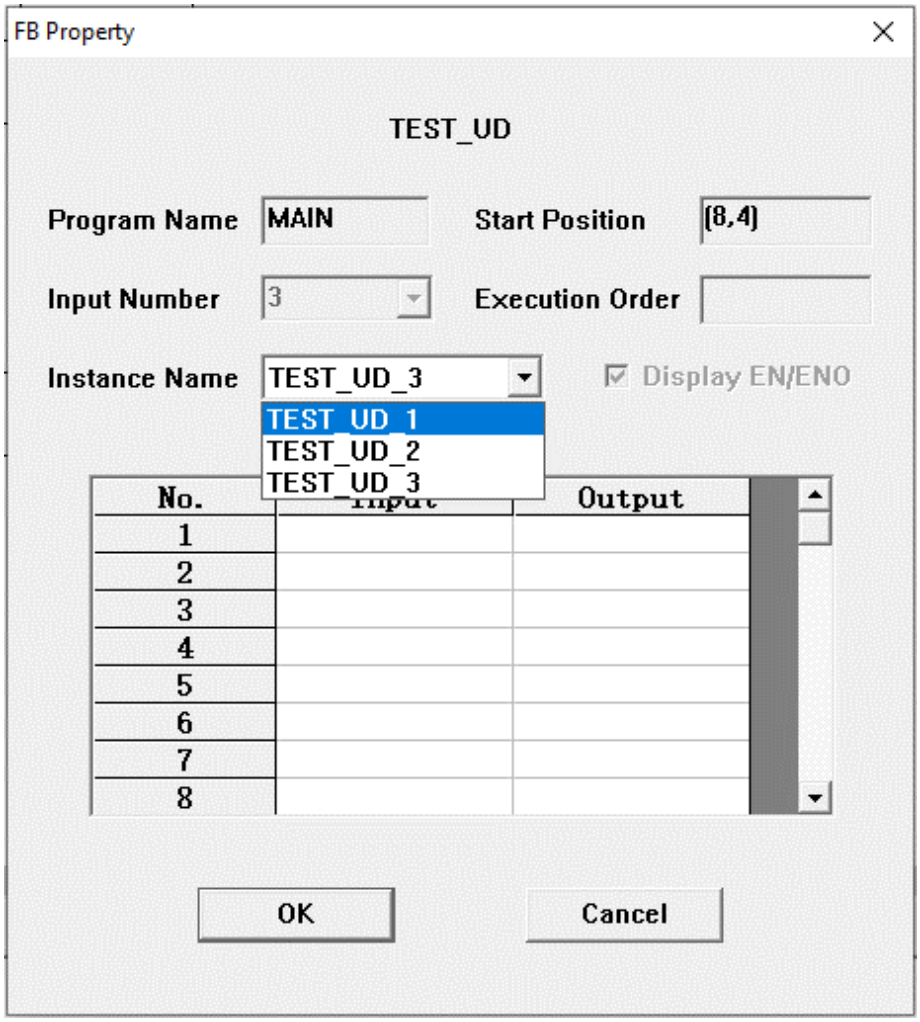


Рис.5.11 Изменение названия исполняющей копии блока функции

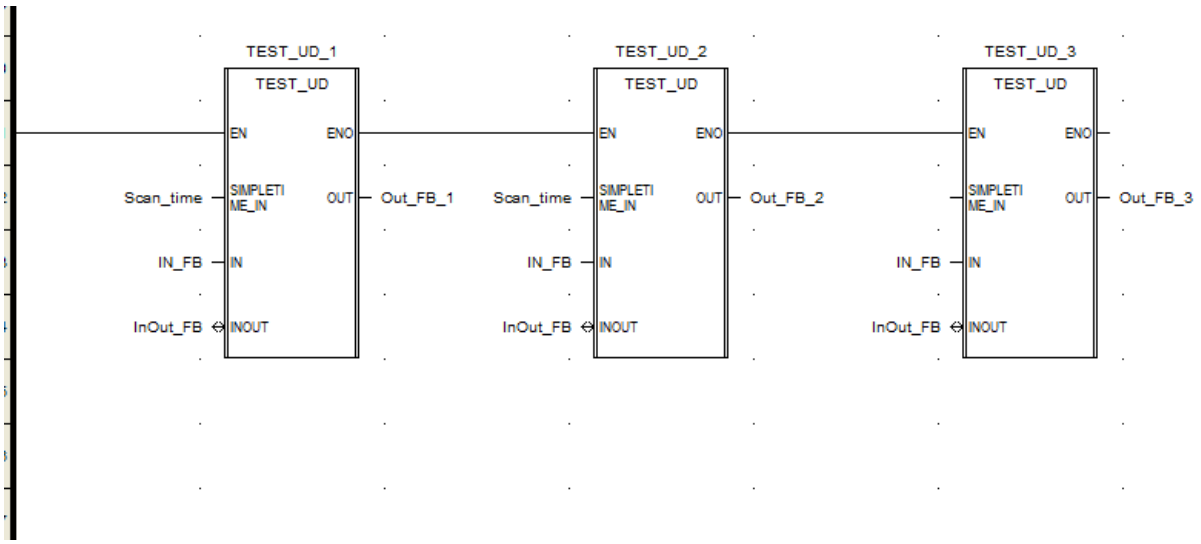


Рис.5.12 Использование исполняющей копии блока функции

5.13.4 Примечания по использованию блоков функций

1. Проблема вложенности в пользовательском блоке функции: в пользовательский блок функции может быть вложен другой упакованный блок функции, но, если пользовательский блок функции содержит таймеры или счетчики, его невозможно вложить.

2. При работе с блоком функции, определяемом пользователем, при использовании контакта/катушки положительного перехода или контакта/катушки отрицательного перехода необходимо определить несколько исполняемых копий блока функции, если упакованный блока функции необходимо повторно использовать в других программах.

6 Программирование LD

Язык релейно-контактных схем (LD) — это разновидность графического языка программирования, инструкция и синтаксис которого аналогичны принципиальной электрической схеме. С помощью LD можно отслеживать изменение данных в режиме реального времени.

Язык программирования LD был разработан на основе требований разработчиков традиционной схемотехники, поэтому существует значительное совпадение схем на языке LD и аппаратных схем, собранных из реле и электронных устройств, а написанные на языке LD управляющие программы интуитивно понятны. По прошествии времени, пользователи обнаружили, что язык программирования LD прост в освоении и удобен в использовании, поэтому он получил широкое применение. Благодаря постоянному развитию и расширению сложных вычислительных функций, - таких как таймеры, счетчики и математические вычисления, - LD стал одним из основных языков программирования для ПЛК и других устройств автоматического управления.

Многие инструкции и функциональные блоки языка программирования LD очень похожи на управляющие устройства электрической схемы. Например, нормально разомкнутый контакт (-| |-) в LD эквивалентен нормально разомкнутому контакту по месту, а нормально разомкнутая катушка (-( )-) в LD эквивалентна катушке реле.

Язык программирования LD имеет множество типов инструкций и функциональных блоков, разнообразные типы данных и удобные режимы адресации, которые требуются пользователям, поэтому знакомство с функциональными блоками и режимами программирования является предварительным условием для написания кратких и эффективных программ на LD.

6.1 Контакт, катушка и функциональный блок

6.1.1 Контакт

Контакт используется для отслеживания состояния заданной точки. Тип данных такой точки может быть только BOOL, только 0 или 1. Когда условие включения контакта выполнено, через контакт может быть «проведен ток». В таком случае, при отображении на LD в режиме реального времени, контакт станет красным, в противном случае контакт останется зеленым. Условие включения контакта зависит от состояния данной точки и типа контакта.

Тип контакта	Описание	Значок
Нормально разомкнутый контакт	Если заданная точка нормально разомкнутого контакта равна 1, то следующая за контактом инструкция будет выполнена.	⊣ ⊢
Нормально замкнутый контакт	Если заданная точка нормально замкнутого контакта равна 0, то следующая за контактом инструкция будет выполнена	⊣/⊢
Контакт с положи-тельным переходом	Если бит заданной точки контакта с положительным переходом меняется с 0 на 1, то следующая за контактом инструкция будет выполнена однократно. При сканировании контакта в следующем цикле, следующая за контактом инструкция не будет выполняться до тех пор, пока бит снова не изменится с 0 на 1.	⊣ P ⊢
Контакт с отрицательны м переходом	Если бит заданной точки контакта с отрицательным переходом меняется с 1 на 0, то следующая за контактом инструкция будет выполнена однократно. При сканировании контакта в следующем цикле, следующая за контактом инструкция не будет выполняться пока бит снова не изменится с 1 на 0.	⊣ N ⊢

6.1.2 Катушка

Катушка используется для управления заданной точкой, тип данных которой также является BOOL. Катушка управляется определенной условной логикой. Катушка начинает действовать только когда выполнено предыдущая инструкция. Существует множество типов катушек.

Катушка	Описание	Значок
Катушка	Когда катушка выполнена («проводит ток»), значение заданной точки становится равным 1. Катушка не удерживает значение, поэтому в отсутствие выполнения значение катушки становится равным 0.	—( )—
Инверсная катушка	Когда инверсная катушка не выполняется, значение заданной точки становится равным 1; когда инструкция выполняется («ток проводится»), значение заданной точки становится равным 0. Инверсная катушка также не удерживает значение.	—(/)—
Катушка положительного перехода	Если катушка положительного перехода в предыдущем цикле сканирования не выполнялась, а в текущем цикле сканирования выполняется то значение заданной точки катушки становится равным 1 и поддерживается в течение цикла сканирования. При следующем сканировании этой части LD значение заданной точки возвращается к 0.	—(P)—
Катушка отрицательного перехода	Если катушка положительного перехода в предыдущем цикле сканирования выполнялась, а в текущем цикле сканирования не выполняется, то значение заданной точки катушки становится равным 1 и поддерживается в течение цикла сканирования. При следующем сканировании этой части LD значение заданной точки возвращается к 0.	—(N)—

Катушка установки	При выполнении инструкции значение заданной точки становится равным 1. Катушка установки является удерживающей; только катушка сброса в той же точке, может изменить значение точки с 1 на 0. В противном случае данная точка поддерживает значение 1 независимо от того, проводится ток или нет.	—(S)—
Катушка сброса	При выполнении инструкции значение заданной точки становится равным 1. Катушка сброса является удерживающей; только катушка установки в той же точке, может изменить значение точки с 0 на 1. В противном случае данная точка поддерживает значение 0 независимо от того, проводится ток или нет.	—(R)—

6.1.3 Блок функции

По своим функциям, базовые блоки функций разделяются на такие группы, как математические, статистические, логические, сравнения, преобразования, перемещения данных, таймеры, счетчики, управления программой, ПЛК и другие. Сначала выберите группу на панели инструментов, как показано на рис.6.1, а затем выберите конкретный блок функции, как показано на рис.6.2. Все основные функции и параметры блоков функций описаны в главе 5 «Базовые блоки функций».

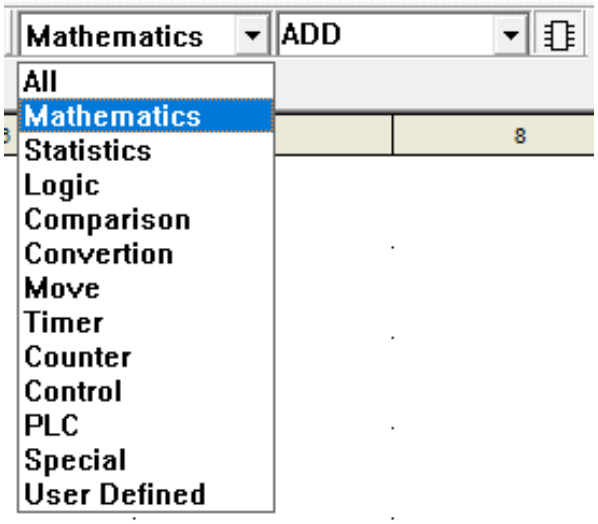


Рис.6.1 Выбор группы LD

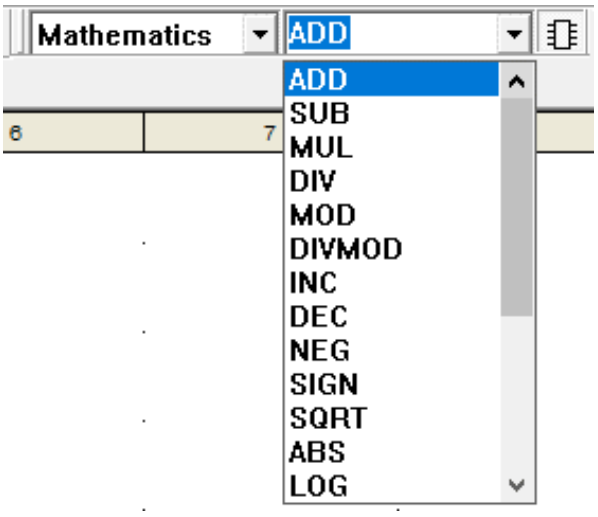


Рисунок 6.2 Выбор функционального блока LD

6.2 Работа с редактором LD

Дважды щелкните левой кнопкой мыши на названии LD в браузере проекта, чтобы содержимое программы LD отобразилось в правой боковой области редактирования и его можно было бы редактировать.

Фоном окна редактора LD является логическая сетка, нарисуйте левую шину питания слева и правую шину питания справа. Объекты LD (контакты, катушки, функциональные блоки) могут быть обработаны во время программирования LD только при подключении к левой шине питания. Выходы внутренних катушек и функциональных блоков можно не подключать, но обычно они подключаются к правой шине питания и создают «цепь тока».

Разные функциональные блоки занимают разное количество ячеек сетки. Например, самые маленькие функциональные блоки, такие как контакт и катушка, соответственно, занимают только одну ячейку сетки.

Переключаться между страницами можно, используя клавиши **PgUp** и **PgDn**, или перетаскивая полосу прокрутки в правой части области редактирования, что позволяет быстро перейти на любую нужную страницу.

Когда блок необходимо поместить в область редактирования, необходимо его сначала выбрать в меню LD или на панели инструментов LD и щелкнуть на области редактирования левой кнопкой мыши, затем блок функции можно поместить туда, куда указывает мышь. Если блок функции необходимо переместить, его необходимо выбрать и, удерживая кнопку мыши, перенести в нужное место. При работе с блоками функций, такие операции, как «вырезать», «копировать», «вставить» и «удалить» можно выполнить с помощью операций меню. Работа с несколькими блоками функций аналогична работе с одним блоком функции, но сначала нужно их выбрать с помощью операций с блоками.

Если дважды щелкнуть по контакту или катушке, то над ними появится поле ввода, в которое можно ввести требуемый параметр. Параметры и контакты специальных

блоков функций также можно ввести во всплывающем окне ввода параметров двойным щелчком мыши, как показано на рисунке 6.3.

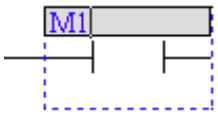


Рисунок 6.3 Ввод параметров

При вводе параметра можно использовать форму «тип + номер», такую как %I0001 (ее можно сокращать до %I1) и %MW0001 и т.д. Имя, заданное в таблице точек, также может быть использовано. При вводе переменной V, имя точки данных используется обязательно.

6.2.1 Связь

◆ **: Link**

Операция **Link** между входным и выходным выводами контакта, катушки и функционального блока эквивалентна операции меню **【LD】/【Link】.**

Функция «Магнит»

Когда два функциональных блока находятся рядом на определенном расстоянии, связь устанавливается автоматически и добавляется ссылка. Это означает, что связь установлена между обоими функциональными блоками. Когда какой-либо из двух функциональных блоков перемещается, соединение автоматически пересчитывается, и связь между ними сохраняется. Если один из них удаляется, все ссылки, связанные с этим функциональным блоком, удаляются автоматически.

Связать вручную

Выберите значок и переместите мышь в начальную точку создаваемой связи. Указатель мыши изменится на S. Затем щелкните мышью, чтобы выбрать начальную позицию создаваемой связи. Переместите мышь на конечную позицию создаваемой связи. Указатель мыши изменится на E. Затем щелкните мышью, чтобы завершить операцию создания связи.

6.2.2 Присвоить отрицательное значение

◆ **: Negate**

Операция **Negate** заключается в циклическом переключении между контактами и катушками и эквивалентна операции меню **【LD】/【Negate】.**

Переключение между контактами

Выберите контакт, который необходимо переключить, и нажмите на значок **Negate**. После каждого нажатия контакт будет циклически переключаться между следующими четырьмя типами контактов.

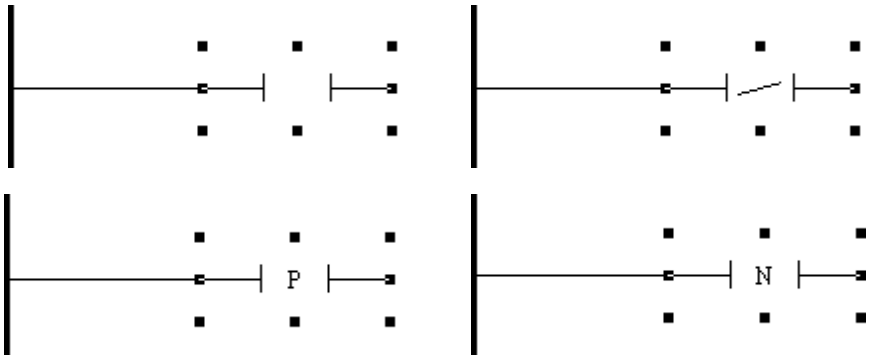


Рис.6.4 Присвоение отрицательного значения контакту

Переключение между катушками

Выберите катушку, которую необходимо переключить, и нажмите на значок **Negate**. После каждого нажатия катушка будет циклически переключаться между следующими четырьмя типами катушек.

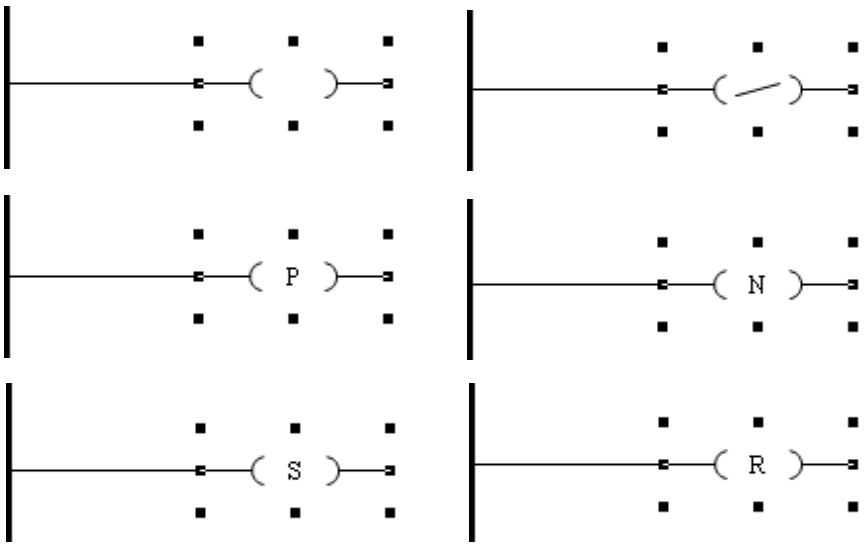


Рис.6.5 Присвоение отрицательного значения катушке

6.2.3 Метка

◆ **:Label**

Label - это метка для команды **Goto**. Операция **Label** эквивалентна операции меню **【LD】/【Label】**.

Создание метки

Выберите значок **Label** и щелкните мышью в окне редактирования LD, чтобы разместить метку. **Метка должна занимать одну строку в LD,** как показано на следующем рисунке.

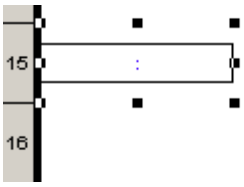


Рис.6.6 Метка в LD

Добавить название метки

Дважды щелкните по метке, как показано на следующем рисунке. Название метки может быть введено в серое поле, например «LABEL».

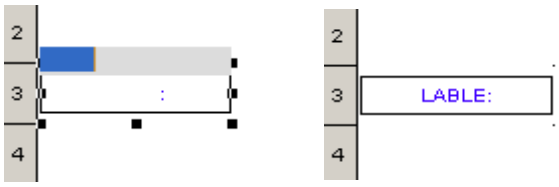
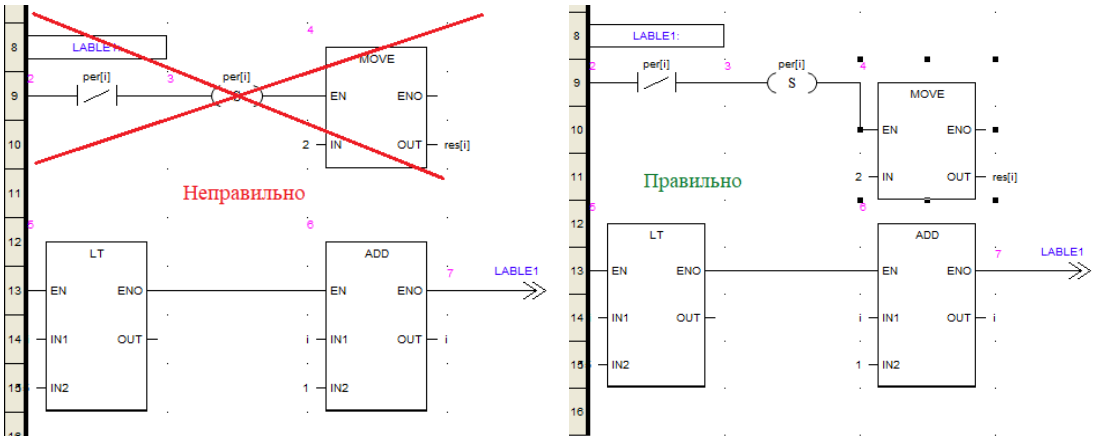


Рис.6.7 Ввод имени метки в LD

Метка должна занимать полностью 1 строку. Не должно быть пересечений с другими блоками программ. Иначе, при компиляции проекта появится ошибка “**label position invalid**”.



:Goto

Операция **Goto** эквивалентна операции меню 【**LD**】/【**Goto**】.

Размещение Goto

Выберите значок **Goto** и щелкните мышью вы окне редактирования LD, чтобы разместить значок **Goto**. Значок показан на следующем рисунке.

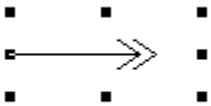


Рис.6.8 Goto в LD

Добавить имя Goto

Дважды щелкните значок **Goto**, как показано на следующем рисунке; название метки можно ввести в сером поле, например «LABEL». Это означает, что при изменении бита точки %I0001 с 0 на 1 программа перейдет в положение метки «LABEL».

Рис.6.9 Ввод имени Goto в LD

6.2.4 Возврат

◆ **:Return**

Операция **Return** эквивалентна операции меню **【LD】/【Return】.**

Когда LD выполняет операцию **Return**, программа возвращается к той её части, которая вызывает эту подпрограмму, и переходит к следующей. Все программы, следующие за операцией **Return**, не будут сканироваться и выполняться. Значок показан ниже:

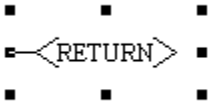


Рис.6.10 Возврат в LD

6.2.5 Комментарий

◆ **:Comment**

Операция **Comment** эквивалентна операции меню **【LD】/【Comment】**.

Текстовая аннотация может быть введена в любом месте редактора LD.

6.2.6 Масштабировать

◆ **:Zoom**

Операция **Zoom** эквивалентна операции меню **【LD】/【Zoom】**.

Эта операция изменяет масштаб редактора LD.

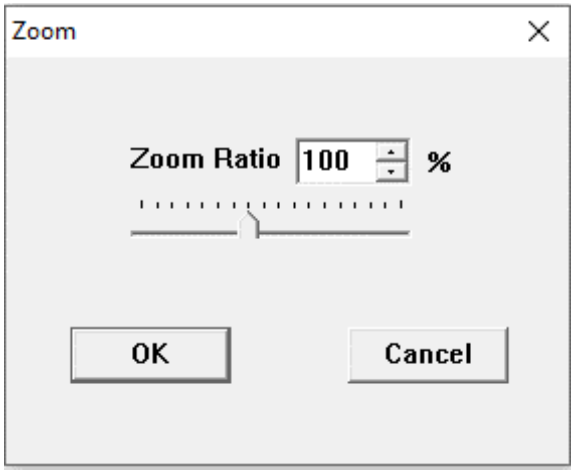


Рис.6.11 Масштабировать в LD

6.2.7 Вставка

◆ **: Insert**

Операция **Insert** эквивалентна операции меню **【LD】/【Insert】**.

Строка — это единица измерения редактора LD. Если вы хотите вставить несколько новых функциональных блоков LD между двумя строками, операция **Insert** может быть использована для увеличения пространства редактора, за счет создания дополнительного места, которое можно занять новым разделом программы.

В том месте, куда вставляется строка не должно быть других функциональных блоков.

6.2.8 Удалить

◆ **: Remove**

Операция **Remove** эквивалентна операции меню **【LD】/【Remove】**.

Если между двумя последовательными функциональными блоками в редакторе LD слишком много пустых строк, операция **Remove** может быть использована для удаления пустых строк для поддержания красивого внешнего вида.

В том месте, где будет удалена строка не должно быть других функциональных блоков.

7 Программирование FBD

7.1 Использование языка программирования FBD

7.1.1 Свойства программы FBD

◆ На фоне программы FBD расположена сетка. Ячейка сетки — это наименьшее возможное пространство между двумя объектами в редакторе FBD.

◆ Программирование в FBD не основывается на привязке блоков функция к сетке, но они все равно выравниваются по ячейкам сетки.

◆ Порядок выполнения зависит от положения блоков функция FBD в программе (выполнение ведется слева направо, сверху вниз)..

7.1.2 Новая программа FBD

Выберите **【File】/【New program】**, и откроется окно, как показано на рис.7.1. После выбора типа программы **FBD** необходимо ввести уникальное название программы в поле «Program Name», а также можно ввести комментарий в поле «Program Description». Так же необходимо выбрать задачу/прерывание, в котором будет выполняться программа **FBD** в выпадающем списке «Task/Interrupt» (более подробно про выбор задачи/прерывания см. главу 3 «Работа с проектами» Руководства).

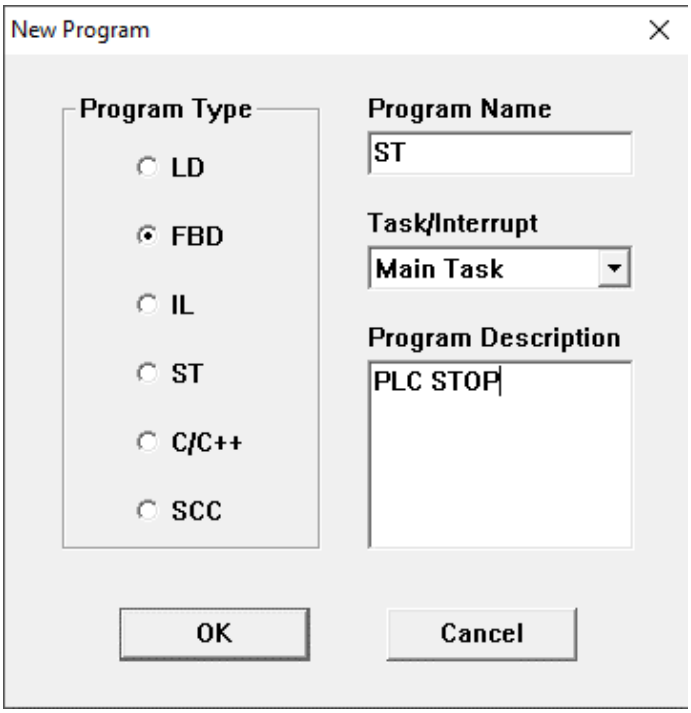


Рис.7.1 Новая программа FBD

7.2 Редактирование программы FBD

7.2.1 Блоки функций

По своему назначению базовые блоки функций разделяются на такие группы, как математические, статистические, логические, функции сравнения, функции преобразования, функции перемещения данных, таймеры, счетчики, функции управления, функции ПЛК и другие. Для выбора блока функции сначала выберите

группу на панели инструментов, как показано на рис.7.2, а затем выберите конкретный блок функции, как показано на рис.7.3. Все основные функции и параметры блоков функций описаны в главе 5 «Базовые блоки функций».

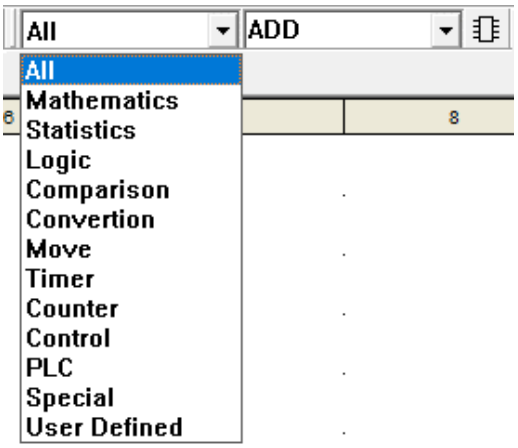


Рис.7.2 Выбор группы FBD

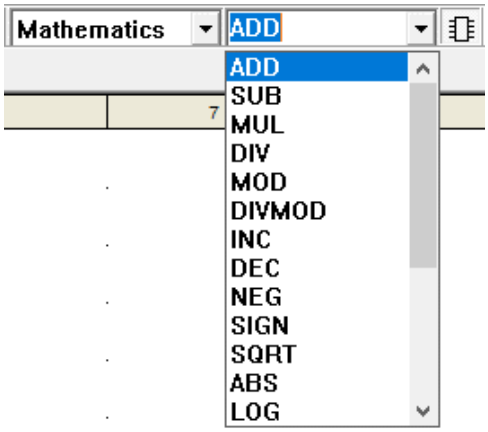


Рис.7.3 Выбор функционального блока FBD

7.2.2 Модификация свойств блока функции

Свойства каждого блока функций могут быть изменены, например, можно отображать или скрывать **EN/ENO** в FBD. Некоторые блоки функций могут увеличивать количество входных параметров. Используем блок функции **AND** в качестве примера, как показано на рис.7.4. Сначала выберите блок функции, затем щелкните правой кнопкой мыши, чтобы появилось диалоговое окно просмотра и изменения свойств, как показано на рис.7.5. Здесь можно изменить параметры блока функции.

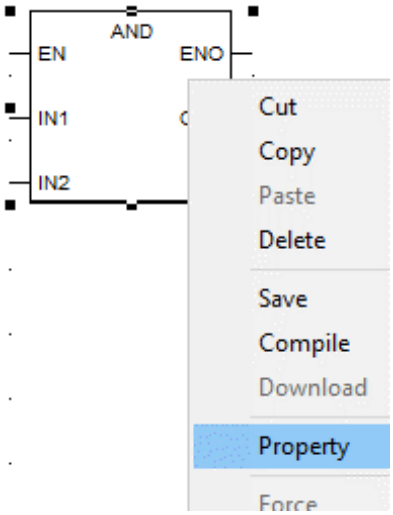


Рис.7.4 Свойство блока функции FBD

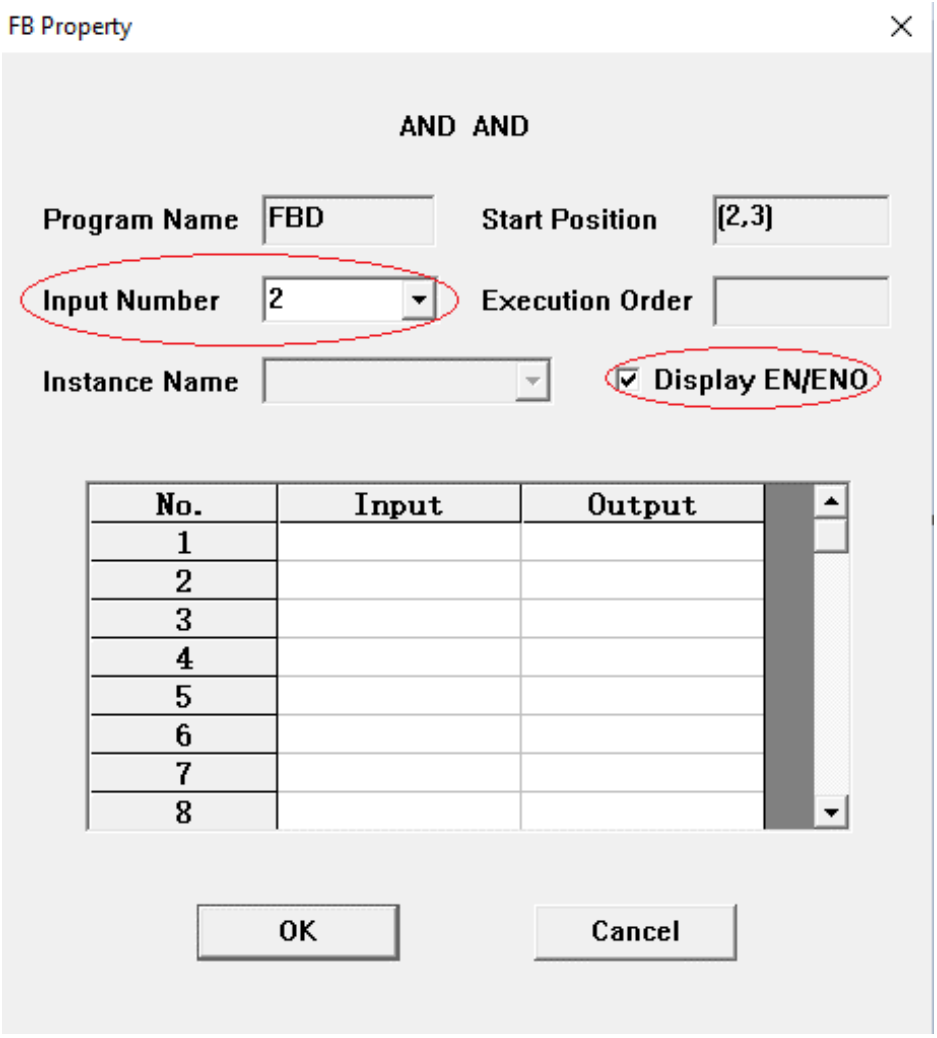


Рис.7.5 Свойства функционального блока

7.3 Операции

7.3.1 Соединение "Link"

◆ ⌐: **Link**

Операция **Link** соединяет входные и выходные параметры функциональных блоков и эквивалентна операции меню **【FBD】/【Link】.**

Соединение функциональных блоков с типами данных, созданных вручную, невозможно. Для передачи переменных подобного типа данных между функциональными блоками необходимо создать переменную данного типа и использовать её как буфер.

Возможно соединение параметров функциональных блоков совместимого типа данных. В случае, если типы данных не совместимы, СКПро при попытке создания соединения выдаст предупреждение.

Автоматическое создание соединений "Магнит"

Когда два функциональных блока находятся рядом на определенном расстоянии, связь между выходными параметрами левого блока и входными параметрами правого блока при совпадении типов данных параметров устанавливается автоматически и добавляется **Link**. Это означает, что связь установлена между обоими блоками функций. Когда какой-либо из двух блоков функций перемещается, соединение автоматически пересчитывается, и связь между ними сохраняется. Если один из функциональных блоков удаляется, все ссылки, связанные с этим блоком, удаляются автоматически.

Ручное создание соединений блоков функций

Выберите значок и переместите мышь в начальную точку создаваемой связи (входной либо выходной параметр функционального блока). Если в данной точке блока функций возможно создание связи, указатель мыши изменится на S. Затем щелкните мышью, чтобы начать создание связи. Переместите мышь на конечную позицию создаваемой связи (входной либо выходной параметр функционального блока). Если в данной точке блока функций возможно создание связи, указатель мыши изменится на E. Затем щелкните мышью, чтобы завершить операцию создания связи. Если создание связи возможно, появится **Link.**

Примечание. Если создание связи невозможно (попытка связать выходные параметры блоков, несовместимые типы данных), СКПро выдаст предупреждение "Type mismatch, please do again!". Если были связаны входные параметры блоков, СКПро выдаст ошибку при компилировании "Link invalid" с указанием расположения блока с ошибкой в сетке (например, если программа называется "Test", и выходной параметр блока ADD связан с входным, а блок располагается в ячейке [7,15], то в окне Compile при компиляции будет выведена ошибка "Test : ADD [7,15] input expected" и "Test : link invalid"). При попытка повторной связи используемого параметра блока СКПро выдаст ошибку "Link already exist, please do again**!".**

7.3.2 Инвертирование значения "Negate"

◆ **: Negate**

Операция **Negate** предназначена для инвертирования значения вводf или выводf блока и эквивалентна операции меню **【FBD】/【Negate】.**

Данная операция предназначена для использования с ограниченным набором функциональных блоков.

Если операция инвертирования значения допускается, то при выборе значка и перемещении курсора мыши на параметр блока, указатель примет вид После щелчка по параметру он изменит вид (рис.7.6, параметры IN1, IN2). Для отмены инвертирования необходимо вышеописанную процедуру повторить.

Как показано на рис.7.6, входные параметры IN1, IN2 блока функции **AND** были инвертированы, а параметры IN3 и OUT остались не инвертированы.

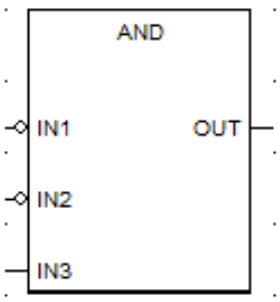


Рис.7.6 Инвертирование параметров блока функции AND

7.3.3 Метка "Label"

◆ **: Label**

Label — это метка для команды **Goto**. Операция **Label** эквивалентна операции меню **【FBD】/【Label】**.

Создание метки

Выберите значок **Label** и щелкните мышью в окне редактирования FBD, чтобы разместить метку. **Метка должна занимать одну строку в FBD,** как показано на следующем рисунке (рис.7.7).

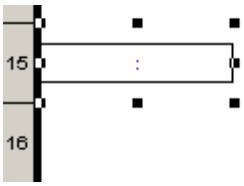


Рис.7.7 Метка в FBD

Добавить название метки

Дважды щелкните по метке, и введите название метки (например, «LABLE») в серую область, как показано на следующем рисунке.

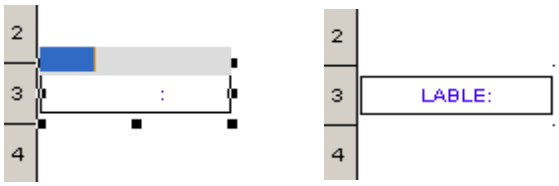
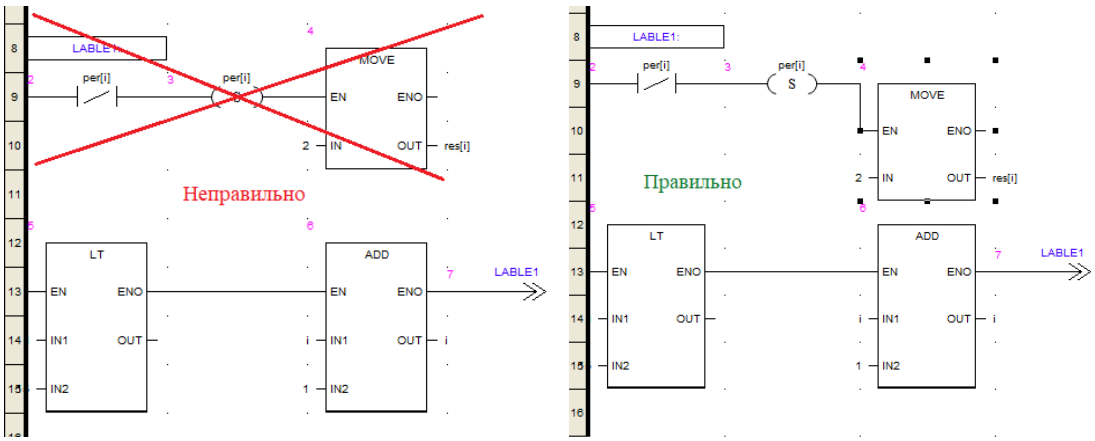


Рис.7.8 Ввод имени метки в FBD

Метка должна занимать полностью 1 строку. Не должно быть пересечений с другими блоками программ. Иначе, при компиляции проекта появится ошибка “**label position invalid**”.



7.3.4 Переход "Goto"

◆ : **Goto**

Операция **Goto** эквивалентна операции меню 【FBD】/【**Goto**】.

Размещение значка

Выберите значок **Goto** и щелкните мышью в окне редактирования FBD, чтобы разместить значок **Goto**. Значок показан на следующем рисунке.

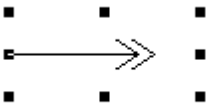


Рис.7.9 Goto в FBD

Добавить имя Goto

Дважды щелкните по значку **Goto**, и введите название метки (например, «LABLE») в серую область, как показано на следующем рисунке.

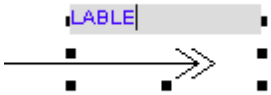


Рис.7.10 Ввод имени Goto в FBD

7.3.5 Возврат "Return"

◆ : **Return**

Операция **Return** эквивалентна операции меню 【**FBD**】/【**Return**】.

Когда FBD выполняет операцию **Return**, программа возвращается к той её части, которая вызывает эту подпрограмму, и переходит к следующей. Все программы, следующие за операцией **Return**, не будут сканироваться и выполняться. Значок показан ниже:

Рис.7.11 Return в FBD

7.3.6 Комментарий "Comment"

◆ : **Comment**

Операция **Comment** эквивалентна операции меню 【**FBD**】/【**Comment**】.

Текстовая аннотация может быть введена в любом месте редактора FBD.

7.3.7 Масштабирование "Zoom"

◆ : **Zoom**

Операция «Zoom» эквивалентна операции меню 【**FBD**】/【**Zoom**】.

Эта операция изменяет масштаб редактора FBD в пределах [30-200] %.

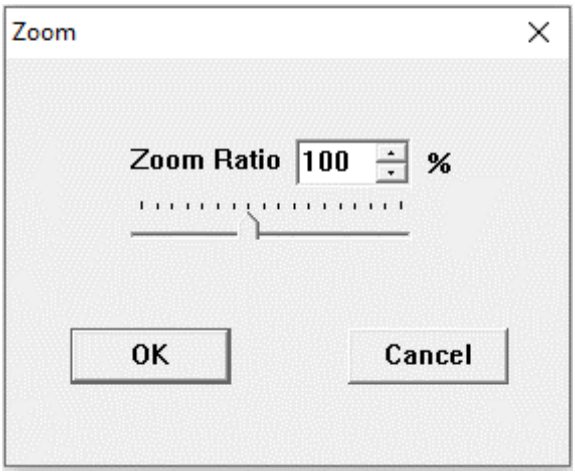


Рис.7.12 Zoom в FBD

7.3.8 Вставка "Insert"

◆ **: Insert**

Операция **Insert** эквивалентна операции меню 【**FBD**】/【**Insert**】.

Строка — это единица измерения программы редактора FBD. Если вы хотите вставить несколько новых функциональных блоков FBD между двумя строками, операция **Insert** может быть использована для увеличения пространства редактора, за счет создания дополнительного места, которое можно занять новым разделом программы.

В том месте, куда вставляется строка, не должно быть других функциональных блоков. Если в месте вставки строки имеется функциональный блок, то он будет перемещен на строку ниже, а его связи будут пересчитаны (но функционал программы не изменится).

7.3.9 Удаление "Remove"

◆ **: Remove**

Операция **Remove** эквивалентна операции меню 【**FBD**】/【**Remove**】.

Если между двумя последовательными функциональными блоками в редакторе FBD слишком много пустых строк, операция **Remove** может быть использована для удаления пустых строк для поддержания красивого внешнего вида.

В том месте, где будет удалена строка, не должно быть других функциональных блоков. При попытке удаления строки с функциональными блоками СКПро выдаст предупреждение "This line cannot be deleted!"

8 Программирование IL

С помощью языка программирования IL функциональные блоки и функции могут быть вызваны условно или безусловно; можно выполнить присвоение, а также условный или безусловный переход внутри программы IL, выполнить USER DEFINED функцию.

Программа IL состоит из ряда инструкций. Каждая инструкция состоит из следующих частей:

- Оператор
- При необходимости модификатор
- При необходимости один или несколько операндов

Если используется более одного операнда, для разделения этих операндов следует использовать запятые. Метку нужно использовать с двоеточием, за которым следует инструкция. Комментарий может быть добавлен в любом месте редактора IL.

При переполнении максимального/минимального значения типа данных операндов при выполнении операторов, будет использовано значение, уменьшенное/увеличенное на максимальное/минимальное значение типа данных, будьте внимательны при выборе типов данных операндов!

8.1 Структура языка программирования

IL — это так называемый аккумуляторно-ориентированный язык. Это означает, что каждая инструкция может использовать или изменять содержимое текущего аккумулятора (сумматора). Таким образом, IL всегда начинается с оператора «LD» (инструкция по загрузке аккумулятора).

Примеры для аккумулятора:

Инструкция	Толкование
LD 10	Загрузить значение 10 в аккумулятор.
ADD 25	Добавить 25 в аккумулятор.
ST A	Сохранить результат в «A». Теперь значение аккумулятора и «A» равно 35. Если следующая инструкция не начинается с «LD», то 35 будет распознано как значение аккумулятора.

Аналогичным образом, аккумулятор часто используется в операции сравнения. Логический результат после сравнения сохраняется в аккумуляторе как текущее значение. Биты данных аккумулятора настраиваются автоматически в соответствии с типом данных операнда.

Примеры для сравнения:

Инструкция	Толкование
LD B	Загрузить «B» в аккумулятор.
GT 10	Сравнить аккумулятор с 10.
ST A	Сохранить результат сравнения в «A». Если «B» меньше или равно 10, то значение «A» и аккумулятора равно FALSE (0). Если «B» больше 10, то значение «A» и аккумулятора равно TRUE (1).

8.2 Порядок исполнения инструкций

Инструкции выполняются построчно, сверху вниз. Этот порядок может быть изменен с помощью круглых скобок или оператора перехода.

Если соответствующие значения «A», «B», «C» и «D» равны 1, 2, 3 и 4, то инструкция работает следующим образом:

```
LD    A
ADD   B
SUB   C
MUL   D
ST    E
```

Тогда данный результат «E» равен 0.

Если инструкция работает следующим образом:

```
LD    A
ADD   B
SUB (
LD    C
MUL   D
)
ST    E
```

Тогда данный результат «E» равен -9.

8.3 Толкование инструкций

8.3.1 Операнд

Операнды могут быть константами, точками (POINT TABLE), переменными (VARIABLE TABLE) и т.д.

8.3.2 Модификатор

Модификатор «N» используется для побитового отрицания значения операнда.

Пример для модификатора «N»:

В следующем примере, если значение «A» равно TRUE (1), а значение «B» равно FALSE (0), то значение «C» равно TRUE (1).

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ANDN B	Выполнить операцию «AND» для значения аккумулятора и побитовым отрицанием значения «B».
ST C	Сохранить результат в «C».

В следующем примере, если значение «A» (INT) равно 5, то значение «C» (INT) равно -6.

Инструкция	Толкование
LDN A	Загрузить значение «A» с побитовым отрицанием в аккумулятор.
ST C	Сохранить результат в «C».

Модификатор «C» используется для выполнения соответствующих инструкций, когда значение аккумулятора равно TRUE (1).

Пример для модификатора «C»:

В следующем примере, только когда значение «A» равно TRUE (1) и значение «B» тоже равно TRUE (1), значение в аккумуляторе равно TRUE (1) и выполняется переход «JMPC» к метке «START».

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения аккумулятора и значения «B».
JMPC START	Когда значение аккумулятора равно TRUE (1), перейти к метке «START» и продолжить выполнять инструкции.

Модификатор «CN» используется для выполнения соответствующих инструкций, когда значение аккумулятора равно FASLE (0).

Пример для модификатора «CN»:

В следующем примере, только когда значение «A» равно FALSE (0) или значение «B» равно FALSE (0), выполняется переход к метке «START».

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения аккумулятора и значения «B».
JMPCN START	Когда значение аккумулятора равно FALSE (0), перейти к метке «START» и продолжить выполнять инструкции.

Модификатор «(» и «)»: Модификатор левой круглой скобки «(» используется для задержки операции до появления правой круглой скобки «)». Количество модификаторов правой круглой скобки должно быть равно количеству модификаторов левой круглой скобки. Модификатор круглой скобки может быть вложенным.

Пример для модификатора «(» и «)»:

В следующем примере, если «C» или «D» равно TRUE (1), и только когда оба «A» и «B» равны TRUE (1), то «E» равно TRUE (1).

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения аккумулятора и значения «B».
AND (	«AND» откладывается до появления правой круглой скобки.

Инструкция	Толкование
LD C	Загрузить значение «C» в аккумулятор.
OR D	Выполнить операцию «OR» для значения аккумулятора и значения «D».
)	Запускается отложенное «AND». Выполнить операцию «AND» с результатом «A AND B» и значением аккумулятора (результат «C OR D»).
ST E	Сохранить результат в «E».

8.3.3 Оператор

Опера-тор	Значение оператора	Тип модифика-тора	Тип операнда
LD	Загрузить значение операнда в аккумулятор.	N	Константа (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 BOOL значение аккумулятора равно TRUE (1), установить значение операнда равным TRUE (1).		Q, M, N, V
R	Если BOOL значение аккумулятора равно TRUE (1), установить значение операнда равным FALSE (0).		Q, M, N, V
AND	Логика AND (И)	N, N( ), ( )	Константа (constant), I, Q, IW, QW, M, MW, N, NW, S, SW, V
OR	Логика OR (ИЛИ)	N, N( ), ( )	Константа (constant), I, Q, IW, QW, M, MW, N, NW, S, SW, V
XOR	Логика исключающего OR (ИЛИ)	N, N( ), ( )	Константа (constant), I, Q, IW, QW, M, MW, N, NW, S, SW, V
ADD	Прибавление	( )	Константа (constant), IW, QW, MW, NW, SW, V
SUB	Вычитание	( )	Константа (constant), IW, QW, MW, NW, SW, V

Опера-тор	Значение оператора	Тип модифика-тора	Тип операнда
MUL	Умножение	( )	Константа (constant), IW, QW, MW, NW, SW, V
DIV	Деление	( )	Константа (constant), IW, QW, MW, NW, SW, V
MOD	Деление по модулю	( )	Константа (constant), IW, QW, MW, NW, SW, V
GT	Больше, чем	( )	Константа (constant), IW, QW, MW, NW, SW, V
GE	Больше или равно	( )	Константа (constant), IW, QW, MW, NW, SW, V
EQ	Равно	( )	Константа (constant), I, Q, IW, QW, M, MW, N, NW, S, SW, V
NE	Не равно	( )	Константа (constant), I, Q, IW, QW, M, MW, N, NW, S, SW, V
LE	Меньше или равно	( )	Константа (constant), IW, QW, MW, NW, SW, V
LT	Меньше, чем	( )	Константа (constant), IW, QW, MW, NW, SW, V
JMP	Перейти к метке	C, CN	Метка
CAL	Вызвать	C, CN	Функциональный блок, программа
RET	Возврат	C, CN	Нет

8.3.4 Метка

Метка используется для указания цели перехода и должна быть первым элементом строки. Метка должна быть уникальной для всей программы IL и не должна меняться в зависимости от условий. Максимальная длина метки может составлять 24 символа.Метка и следующая за ней инструкция должны быть разделены двоеточием «:». Метка может отображаться только в начале выражения, в противном случае в аккумуляторе появится неопределенное значение.

8.3.5 Комментарий

В редакторе IL многострочный комментарий всегда начинается с символьной строки «(*» и заканчивается символьной строкой «*)». Между этих символьных строк может быть вставлена любая другая символьная строка. Комментарий отображается другим цветом.

Символьная строка «//» используется для вставки комментария текущей строки. Пример комментариев – на рис.8.1.

```
1  LD 10 //загружаем в аккумулятор 10
2  ADD 5 //прибавляем к аккумулятору 5
3  (*многострочные комментарии
4  строка 1
5  строка 2
6  строка 3*)
7  st %mw1 //сохраняем значение аккумулятора в %mw1
```

Рис.8.13 Комментарии программы IL

8.4 Оператор

8.4.1 Загрузка (LD и LDN)

«LD» используется для загрузки значения операнда в аккумулятор. Биты данных аккумулятора настраиваются автоматически в соответствии с типом данных операнда. Эта операция также подходит для производного типа данных.

Пример «LD» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ADD B	Выполнить операцию «ADD» для значения аккумулятора и значения «B».
ST C	Сохранить результат в «C».

Загруженному операнду может быть присвоено отрицательное значение с помощью модификатора N.

Пример «LDN» выглядит следующим образом:

Инструкция	Толкование
LDN A	Загрузить побитовое значение NOT «A» в аккумулятор.
ADD B	Выполнить операцию «ADD» для значения аккумулятора и значения «B».
ST C	Сохранить результат в «C».

8.4.2 Сохранение (ST и STN)

«ST» используется для сохранения текущего значения аккумулятора в операнде. Тип данных операнда должен соответствовать типу данных аккумулятора. Наличие или отсутствие «LD» после «ST» определяет, будет ли использоваться исходный результат в следующей операции.

Пример «ST» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ADD B	Выполнить операцию «ADD» для значения аккумулятора и значения «B».
ST C	Сохранить результат в «C».
ADD D	Выполнить операцию «ADD» для значения «C» (то есть текущего значения аккумулятора) и значения «D».
ST E	Сохранить результат в «E».
LD F	Теперь загрузить значение «F» в аккумулятор.
SUB 2	Вычесть 2 из значения аккумулятора.
ST G	Сохранить результат в «G».

Сохраненному операнду может быть присвоено отрицательное значение с помощью модификатора N.

Пример «STN» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ADD B	Выполнить операцию «ADD» для значения аккумулятора и значения «B».
STN C	Сохранить побитовый результат NOT в «C».

8.4.3 Установить (S), Сбросить (R)

Если текущее значение аккумулятора равно логической 1, то «S» установит для операнда значение 1.

Пример «S» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
S B	Если значение аккумулятора (значение «A») равно 1, то установить значение «B» равным 1.

Оператор «R» и оператор «S» всегда используются вместе как пара (триггер).

Пример триггера «RS» (доминирующий «R») выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
S B	Если значение аккумулятора (значение «A») равно 1, то установить значение «B» равным 1.
LD C	Загрузить значение «C» в аккумулятор.
R B	Если значение аккумулятора (значение «C») равно 1, то установить значение «B» равным 0.

Если текущее значение аккумулятора равно логическому 1, то «R» установит для операнда значение 0.

Пример «R» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
R B	Если значение аккумулятора (значение «A») равно 1, то установить значение «B» равным 0.

Оператор «S» и оператор «R» используются вместе как пара (триггер).

Пример триггера «SR» (доминирующий «S») выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
R B	Если значение аккумулятора (значение «A») равно 1, то установить значение «B» равным 0.
LD C	Загрузить значение «C» в аккумулятор.
S B	Если значение аккумулятора (значение «C») равно 1, то установить значение «B» равным 1.

8.4.4 Логическая операция

И (AND, AND(), ANDN, ANDN())

Логическая операция «AND» между значением аккумулятора и операндом выполняется с помощью инструкции «AND». Для целочисленного типа данных операция выполняется побитово.

В следующем примере, если каждое из значений «A», «B» и «C» равно 1, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения аккумулятора и значения «B».
AND C	Выполнить операцию «AND» для значения аккумулятора (результат «A AND B») и значения «C».
ST D	Сохранить результат в «D».

«AND» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере, если «A» равно 1, а «B» или «C» равно 1, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND (	«AND» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
)	Запускается отложенное «AND». Выполнить операцию «AND» для значения аккумулятора (результат «B OR C») и значения «A».
ST D	Сохранить результат в «D».

«AND» может использоваться вместе с модификатором «N».

В следующем примере, если «A» равно 1, «B» и «C» равно 0, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ANDN B	Выполнить операцию «AND» для отрицательного значения «B» и значения аккумулятора.
ANDN C	Выполнить операцию «AND» для отрицательного значения «C» и значения аккумулятора (результат «A ANDN B»).

Инструкция	Толкование
ST D	Сохранить результат в «D».

«AND» может использоваться вместе с модификаторами «N» и левой круглой скобкой «(».

В следующем примере, если «A» равно 1, «B» равно 0, а «C» равно 1, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ANDN (	«AND» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
ORN C	Выполнить операцию «OR» для отрицательного значения «C» и значения аккумулятора.
)	Запускается отложенное «AND». Выполнить операцию «AND» для значения «A» и отрицательного значения аккумулятора (результат «B ORN C»).
ST D	Сохранить результат в «D».

ИЛИ (OR, OR(), ORN, ORN())

Логическая операция «OR» между значением аккумулятора и операндом выполняется с помощью инструкции «OR». Для целочисленного типа данных операция выполняется побитово.

В следующем примере, если «A» или «B» равно 1, а «C» равно 1, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
OR B	Выполнить операцию «OR» для значения аккумулятора и значения «B».
AND C	Выполнить операцию «AND» для значения аккумулятора (результат «A OR B») и значения «C».
ST D	Сохранить результат в «D».

«OR» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере, если «A» равно 1, или «B» и «C» равно 1, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
OR (	«OR» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
AND C	Выполнить операцию «AND» для значения «C» и значения аккумулятора.
)	Запускается отложенное «OR». Выполнить операцию «OR» для значения аккумулятора (результат «B AND C») и значения «A».
ST D	Сохранить результат в «D».

«OR» может использоваться вместе с модификатором «N».

В следующем примере, если «A» равно 1, или «B» равно 0, а «C» равно 1, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ORN B	Выполнить операцию «OR» для отрицательного значения «B» и значения аккумулятора.
AND C	Выполнить операцию «AND» для значения «C» и значения аккумулятора (результат «A ORN B»).
ST D	Сохранить результат в «D».

«OR» может использоваться вместе с модификаторами «N» и левой круглой скобкой «(».

В следующем примере, если «A» равно 1, или одно из значений «B» и «C» равно 0, то «D» равно 1.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ORN (	«OR» откладывается до появления правой круглой скобки.

Инструкция	Толкование
LD B	Загрузить значение «B» в аккумулятор.
AND C	Выполнить операцию «AND» для значения «C» и значения аккумулятора.
)	Запускается отложенное «OR». Выполнить операцию «OR» для отрицательного значения аккумулятора (результат «B AND C») и значения «A».
ST D	Сохранить результат в «D».

Исключающий ИЛИ (XOR, XOR(), XORN, XORN())

Логическая операция «XOR» между значением аккумулятора и операндом выполняется с помощью инструкции «XOR». Для целочисленного типа данных операция выполняется побитово.

В следующем примере, если «A» и «B» имеют разные значения (одно равно 1, а второе равно 0), то значение «D» равно 1. Если «A» и «B» имеют одинаковое значение (оба равны 0 или 1), то «D» равно 0.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
XOR B	Выполнить операцию «XOR» для аккумулятора и значения «B».
ST D	Сохранить результат в «D».

«XOR» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере, если «A» и «B AND C» имеют разные значения (одно равно 1, а второе равно 0), то значение «D» равно 1. Если «A» и «B AND C» имеют одинаковое значение (оба равны 0 или 1), то «D» равно 0.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
XOR (	«XOR» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
AND C	Выполнить операцию «AND» для значения «C» и значения аккумулятора.

Инструкция	Толкование
)	Запускается отложенное «XOR». Выполнить операцию «XOR» для значения аккумулятора (результат «B AND C») и значения «A».
ST D	Сохранить результат в «D».

«XOR» может использоваться вместе с модификатором «N».

В следующем примере, если значения «A» и «B» одинаковы (оба равны 0 или 1), то «C» равно 1. Если значения «A» и «B» не совпадают, то «C» равно 0.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
XORN B	Выполнить операцию «XOR» для значения аккумулятора и отрицательного значения «B».
ST C	Сохранить результат в «C».

«XOR» может использоваться вместе с модификаторами «N» и левой круглой скобкой «(».

В следующем примере, если значения «A» и «B AND C» одинаковы (оба равны 0 или 1), то «D» равно 1. Если значения «A» и «B AND C» не совпадают, то «D» равно 0.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
XORN (	«XOR» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
AND C	Выполнить операцию «AND» для значения «C» и значения аккумулятора.
)	Запускается отложенное «XOR». Выполнить операцию «XOR» для отрицательного значения аккумулятора (результат «B AND C») и значения «A».
ST D	Сохранить результат в «D».

8.4.5 Математическая операция

Прибавление (ADD и ADD())

С помощью инструкции «ADD» значение операнда прибавляется к значению аккумулятора.

В следующем примере используется формула: D=A+B+C.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ADD B	Прибавить значение «B» к значению аккумулятора.
ADD C	Прибавить значение «C» к значению аккумулятора («A + B»).
ST D	Сохранить результат в «D».

«ADD» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере используется формула: D=A+ (B-C).

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
ADD (	«ADD» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «ADD». Прибавить значение аккумулятора («B - C») к значению «A».
ST D	Сохранить результат в «D».

Вычитание (SUB и SUB())

С помощью инструкции «SUB» значение операнда вычитается из значения аккумулятора.

В следующем примере используется формула: D=A-B-C.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.

Инструкция	Толкование
SUB B	Вычесть значение «B» из значения аккумулятора.
SUB C	Вычесть значение «C» из значения аккумулятора («A - B»).
ST D	Сохранить результат в «D».

«SUB» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере используется формула: D=A- (B-C).

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
SUB (	«SUB» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «SUB». Вычесть значение аккумулятора («B - C») из значения «A».
ST D	Сохранить результат в «D».

Умножение (MUL и MUL())

С помощью инструкции «MUL» значение аккумулятора умножается на значение операнда.

В следующем примере используется формула: D=A*B*C.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
MUL B	Умножить значение аккумулятора на значение «B».
MUL C	Умножить значение аккумулятора («A * B») на значение «C».
ST D	Сохранить результат в «D».

«MUL» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере используется формула: D=A*(B-C).

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
MUL (	«MUL» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «MUL». Умножить значение аккумулятора («B - C») на значение «A».
ST D	Сохранить результат в «D».

Деление (DIV и DIV())

С помощью инструкции «DIV» значение аккумулятора делится на значение операнда.

В следующем примере используется формула: D=A/B/C.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
DIV B	Разделить значение аккумулятора на значение «B».
DIV C	Разделить значение аккумулятора («A / B») на значение «C».
ST D	Сохранить результат в «D».

«DIV» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере используется формула: D=A/(B-C).

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
DIV (	«DIV» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «DIV». Разделить значение «A» на значение аккумулятора («B - C»).
ST D	Сохранить результат в «D».

Деление по модулю (MOD и MOD())

С помощью инструкции «MOD» вычисляется остаток от деления значения аккумулятора на операнд.

В следующем примере используется формула: C=A-((A/B)*B), в которой (A/B) является целочисленной частью деления.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
MOD B	Результатом является остаток от деления значения аккумулятора на значение «B».
ST C	Сохранить результат в «C».

«MOD» может использоваться вместе с левой круглой скобкой - модификатором «(».

В следующем примере используется формула: D=A-(A/(B-C))*(B-C), в которой (A/(B-C)) является целочисленной частью деления.

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
MOD (	«MOD» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «MOD». Результатом является остаток от деления значения «A» на значение аккумулятора («B - C»)
ST D	Сохранить результат в «D».

8.4.6 Операция сравнения

Больше, чем (GT и GT())

С помощью инструкции «GT» сравнивается значение аккумулятора и значение операнда. Если значение аккумулятора больше значения операнда, то результатом будет логическое значение 1. Если значение аккумулятора меньше или равно значению операнда, то результатом будет логическое значение 0.

Пример «GT» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
GT 10	Сравнить значение аккумулятора с 10.
ST B	Если значение «A» больше 10, то сохранить 1 в «B». Если значение «A» меньше или равно 10, то сохранить 0 в «B».

«GT» может использоваться вместе с левой круглой скобкой - модификатором «(».

Пример «GT()» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
GT (	«GT» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.

Инструкция	Толкование
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «GT». Сравнить значение «A» со значением аккумулятора («B - C»).
ST D	Если значение «A» больше, чем («B - C»), то сохранить 1 в «D». Если значение «A» меньше или равно («B - C»), то сохранить 0 в «D».

Больше или равно (GE и GE())

С помощью инструкции «GE» сравнивается значение аккумулятора и значение операнда. Если значение аккумулятора больше или равно значению операнда, то результатом будет логическое значение 1. Если значение аккумулятора меньше значения операнда, то результатом будет логическое значение 0.

Пример «GE» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
GE 10	Сравнить значение аккумулятора с 10.
ST B	Если значение «A» больше или равно 10, то сохранить 1 в «B». Если значение «A» меньше 10, то сохранить 0 в «B».

«GE» может использоваться вместе с левой круглой скобкой - модификатором «(».

Пример «GE()» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
GE (	«GE» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «GE». Сравнить значение «A» со значением аккумулятора («B - C»).
ST D	Если значение «A» меньше, чем («B - C»), то сохранить 0 в «D». Если значение «A» больше или равно («B - C»), то сохранить 1 в «D».

Равно (EQ и EQ())

С помощью инструкции «EQ» сравнивается значение аккумулятора и значение операнда. Если значение аккумулятора равно значению операнда, то результатом будет логическое значение 1. Если значение аккумулятора не равно значению операнда, то результатом будет логическое значение 0.

Пример «EQ» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
EQ 10	Сравнить значение аккумулятора с 10.
ST B	Если значение «A» равно 10, то сохранить 1 в «B». Если значение «A» не равно 10, то сохранить 0 в «B».

«EQ» может использоваться вместе с левой круглой скобкой - модификатором «(».

Пример «EQ()» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
EQ (	«EQ» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «EQ». Сравнить значение «A» со значением аккумулятора («B - C»).
ST D	Если значение «A» не равно («B - C»), то сохранить 0 в «D». Если значение «A» равно («B - C»), то сохранить 1 в «D».

Не равно (NE и NE())

С помощью инструкции «NE» сравнивается значение аккумулятора и значение операнда. Если значение аккумулятора не равно значению операнда, то результатом будет логическое значение 1. Если значение аккумулятора равно значению операнда, то результатом будет логическое значение 0.

Пример «NE» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
NE 10	Сравнить значение аккумулятора с 10.
ST B	Если значение «A» равно 10, то сохранить 0 в «B». Если значение «A» не равно 10, то сохранить 1 в «B».

«NE» может использоваться вместе с левой круглой скобкой - модификатором «(».

Пример «NE()» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
NE (	«NE» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «NE». Сравнить значение «A» со значением аккумулятора («B - C»).
ST D	Если значение «A» не равно («B - C»), то сохранить 1 в «D». Если значение «A» равно («B - C»), то сохранить 0 в «D».

Меньше или равно (LE и LE())

С помощью инструкции «LE» сравнивается значение аккумулятора и значение операнда. Если значение аккумулятора меньше или равно значению операнда, то результатом будет логическое значение 1. Если значение аккумулятора больше значения операнда, то результатом будет логическое значение 0.

Пример «LE» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
LE 10	Сравнить значение аккумулятора с 10.
ST B	Если значение «A» больше 10, то сохранить 0 в «B». Если значение «A» меньше или равно 10, то сохранить 1 в «B».

«LE» может использоваться вместе с левой круглой скобкой - модификатором «(».

Пример «LE()» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
LE (	«LE» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «LE». Сравнить значение «A» со значением аккумулятора («B - C»).
ST D	Если значение «A» больше, чем («B - C»), то сохранить 0 в «D». Если значение «A» меньше или равно («B - C»), то сохранить 1 в «D».

Меньше, чем (LT и LT())

С помощью инструкции «LT» сравнивается значение аккумулятора и значение операнда. Если значение аккумулятора меньше значения операнда, то результатом будет логическое значение 1. Если значение аккумулятора больше или равно значению операнда, то результатом будет логическое значение 0.

Пример «LT» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
LT 10	Сравнить значение аккумулятора с 10.
ST B	Если значение «A» меньше 10, то сохранить 1 в «B». Если значение «A» больше или равно 10, то сохранить 0 в «B».

«LT» может использоваться вместе с левой круглой скобкой - модификатором «(».

Пример «LT()» выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
LT (	«LT» откладывается до появления правой круглой скобки.
LD B	Загрузить значение «B» в аккумулятор.
SUB C	Вычесть значение «C» из значения аккумулятора.
)	Запускается отложенное «LT». Сравнить значение «A» со значением аккумулятора («B - C»).
ST D	Если значение «A» меньше, чем («B - C»), то сохранить 1 в «D». Если значение «A» больше или равно («B - C»), то сохранить 0 в «D».

8.4.7 Переход (JMP, JMPC и JMPCN)

JMP

С помощью инструкции «JMP» осуществляется безусловный переход к метке.

Пример выглядит следующим образом:

Инструкция	Толкование
start: LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
ST D	Сохранить результат в «D».
JMP start	Перейти к метке «start» не принимая во внимание значение аккумулятора (значение «D»).

При реализации безусловного перехода по метке без использования прерываний и должного контроля логики программы можно перевести ПЛК в бесконечный цикл, для выхода из которого потребуется сброс ПЛК.

С помощью инструкции «JMPC» осуществляется условный переход к метке. (Условие – значение аккумулятора равно TRUE (1).)

Пример выглядит следующим образом:

Инструкция	Толкование
start: LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
JMPC START	Переход к метке «start» осуществляется только когда значение аккумулятора равно 1.

JMPCN

С помощью инструкции «JMPCN» осуществляется условный переход к метке. (Условие – значение аккумулятора равно FALSE (0).)

Пример выглядит следующим образом:

Инструкция	Толкование
start: LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
JMPCN START	Переход к метке «start» осуществляется только когда значение аккумулятора равно 0.

8.4.8 Вызов (CAL, CALC и CALCN)

CAL

С помощью инструкции «CAL» осуществляется безусловный вызов функционального блока или подпрограммы.

Пример выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
ST D	Сохранить результат в «D».
CAL IL1	Вызвать подпрограмму «IL1» не принимая во внимание значение аккумулятора (значение «D»).

CALC

С помощью инструкции «CALC» осуществляется условный вызов функционального блока и подпрограммы. (Условие – значение аккумулятора равно TRUE (1).)

Пример выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
CALC IL1	Вызвать подпрограмму «IL1» только если значение аккумулятора равно 1.

CALCN

С помощью инструкции «CALCN» осуществляется условный вызов функционального блока и подпрограммы. (Условие – значение аккумулятора равно FALSE (0).)

Пример выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
CALCN IL1	Вызвать подпрограмму «IL1» только если значение аккумулятора равно 0.

8.4.9 Возврат (RET, RETC и RETCN)

RET

С помощью инструкции «RET» осуществляется безусловный возврат из подпрограммы.

Пример выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
ST D	Сохранить результат в «D».
RET	Возврат к основной программе из подпрограммы, не принимая во внимание значение аккумулятора (значение «D»).

RETC

С помощью инструкции «RETC» осуществляется условный возврат из подпрограммы. (Условие равно 1.)

Пример выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
RETC	Возврат к основной программе из подпрограммы если значение аккумулятора равно 1.

RETCN

С помощью инструкции «RETCN» осуществляется условный возврат из подпрограммы. (Условие равно 0.)

Пример выглядит следующим образом:

Инструкция	Толкование
LD A	Загрузить значение «A» в аккумулятор.
AND B	Выполнить операцию «AND» для значения «B» и значения аккумулятора.
OR C	Выполнить операцию «OR» для значения «C» и значения аккумулятора.
RETCN	Возврат к основной программе из подпрограммы если значение аккумулятора равно 0.

8.5 Блоки функций

По своим функциям, базовые блоки функций разделяются на такие группы, как математические, статистические, логические, сравнения, преобразования, перемещения данных, таймеры, счетчики, управления программой, ПЛК и другие. Сначала выберите группу на панели инструментов, как показано на рис.8.2, затем выберите блок функции, как показано на рис.8.3, щелкните значок. Выбранный блок функции будет вставлен в редактор IL, вместе с необходимыми сопутствующими операторами. Все основные функции и параметры блоков функций описаны в главе 5 «Базовые блоки функции».

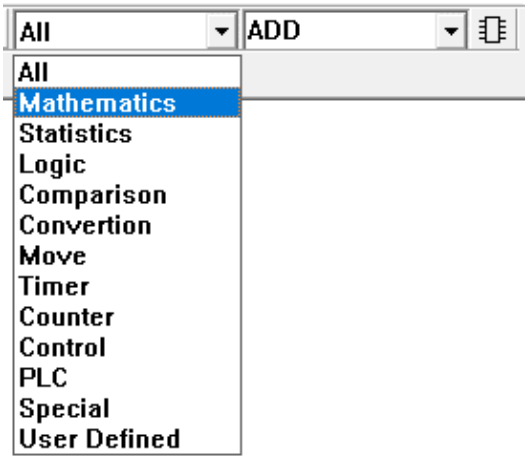


Рис.8.2 Выбор группы IL

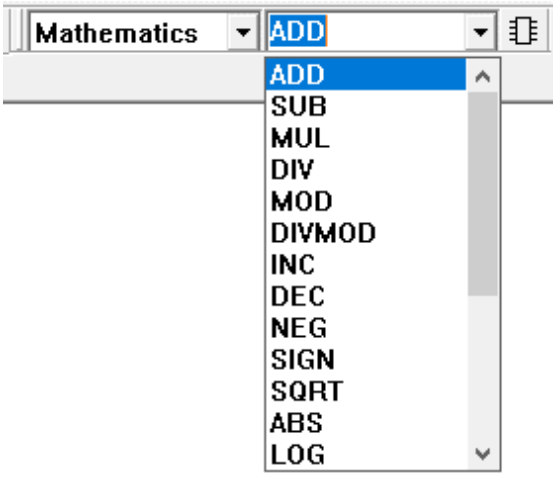


Рис.8.3 Выбор функционального блока IL

9 Программирование ST

Язык ST имеет условное сходство с классическим языком программирования Pascal. Однако ST разработан как специализированный язык программирования для задач промышленного управления. Он обладает широкими возможностями по реализации сложных программ и его используют когда необходимы, например, присвоение значений переменным и функциональным блокам, создание выражений, редактирование условных операторов, итеративных программ и т.д. ST очень удобен для приложений со сложной арифметикой, логикой, работой с массивами и т.д.

Программы ST отличаются свободным форматом, потому при редактировании вкладки, символы новой строки и комментарии могут вставляться в любые места между ключевыми словами и идентификаторами необходимо учитывать синтаксис языка, например то, что выражения должны заканчиваться точкой с запятой (;)). Для пользователей, знакомых с высокоуровневыми компьютерными языками, язык ST прост в изучении и использовании. Он также прост в чтении и понимании, особенно при использовании значимых идентификаторов и аннотаций для комментариев.

9.1 Выражение

Выражение состоит из оператора, базовой функции и операнда.

9.1.1 Операнд

Операнды могут быть константами, точками, переменными и т.д.

9.1.2 Таблица операторов

Оператор	Значение оператора	Тип операнда	Уровень
()	Последовательность выполнения	Выражение	1
NOT	Отрицание	Выражение (expression), константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V	2
*	Умножение	Выражение (expression), константа, IW, QW, MW, NW, SW, V	3
/	Деление	Выражение (expression), константа, IW, QW, MW, NW, SW, V	3
MOD	Деление по модулю	Выражение (expression), константа, IW, QW, MW, NW, SW, V	3
+	Сложение	Выражение (expression), константа, IW, QW, MW, NW, SW, V	4
-	Вычитание	Выражение (expression), константа, IW, QW, MW, NW, SW, V	4
<	Меньше, чем	Выражение (expression), константа, IW, QW, MW, NW, SW, V	5

Оператор	Значение оператора	Тип операнда	Уровень
>	Больше, чем	Выражение (expression), константа, IW, QW, MW, NW, SW, V	5
<=	Меньше или равно	Выражение (expression), константа, IW, QW, MW, NW, SW, V	5
>=	Больше или равно	Выражение (expression), константа, IW, QW, MW, NW, SW, V	5
=	Равно	Выражение (expression), константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V	6
<>	Не равно	Выражение (expression), константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V	6
AND	И	Выражение (expression), константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V	7
XOR	Исключающее ИЛИ	Выражение (expression), константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V	8
OR	Или	Выражение (expression), константа, I, Q, IW, QW, M, MW, N, NW, S, SW, V	9
:=	Присвоение	Q, QW, M, MW, N, NW, V	10

9.2 Оператор

9.2.1 Круглые скобки (( ))

Последовательность выполнения операторов может быть изменена с помощью круглых скобок.

В следующем примере, если «A» равно 1, «B» равно 2, «C» равно 3, а «D» равно –4, то:

Инструкция	Результат
E:=A+B-C*D;	«E» равно 15
E:=(A+B-C)*D;	«E» равно 0

9.2.2 Отрицание (NOT)

Операнд инвертируется по битам с помощью «NOT».

В следующем примере, если «A» равно «11011011», то:

Инструкция	Результат
B:=NOT A;	«B» равно 00100100

9.2.3 Умножение (*)

Значение первого операнда умножается на значение второго операнда с помощью «*».

В следующем примере, если «A» равно 2, а «B» равно 3, то:

Инструкция	Результат
C:=A*B;	«C» равно 6.

9.2.4 Деление (/)

Значение первого операнда делится на значение второго операнда с помощью «/».

В следующем примере, если «A» равно 6, а «B» равно 3, то:

Инструкция	Результат
C:=A/B;	«C» равно 2.

9.2.5 Деление по модулю (MOD)

Значение первого операнда делится на значение второго операнда, а остаток от деления является результатом с помощью «MOD».

Пример выглядит следующим образом:

Инструкция	Результат
C:=A MOD B;	Если «A» равно 6, а «B» равно 4, то «C» равно 2.
C:=A MOD B;	Если «A» равно 4, а «B» равно 4, то «C» равно 0.
C:=A MOD B;	Если «A» равно 7, а «B» равно 3, то «C» равно 1.

9.2.6 Сложение (+)

Значение первого операнда добавляется к значению второго операнда с помощью «+».

В следующем примере, если «A» равно 6, а «B» равно 4, то:

Инструкция	Результат
C:=A+B;	«C» равно 10.

9.2.7 Вычитание (-)

Значение первого операнда вычитается из значения второго операнда с помощью «-».

В следующем примере, если «A» равно 6, а «B» равно 4, то:

Инструкция	Результат
C:=A-B;	«C» равно 2.

9.2.8 Больше, чем (>)

С помощью операции «>» значение первого операнда сравнивается со значением второго операнда. Если значение первого операнда больше значения второго операнда, то результатом будет логическое значение 1. Если значение первого операнда меньше или равно значению второго операнда, то результатом будет логическое значение 0.

Пример выглядит следующим образом:

Инструкция	Результат
C:=A>B;	Если «A» равно 6, а «B» равно 4, то «C» равно 1.
C:=A>B;	Если «A» равно 2, а «B» равно 4, то «C» равно 0.
C:=A>B;	Если «A» равно 2, а «B» равно 2, то «C» равно 0.

9.2.9 Больше или равно (>=)

С помощью операции «>=« значение первого операнда сравнивается со значением второго операнда. Если значение первого операнда больше или равно значению второго операнда, то результатом будет логическое значение 1. Если значение первого операнда меньше значения второго операнда, то результатом будет логическое значение 0.

Пример выглядит следующим образом:

Инструкция	Результат
C:=A>=B;	Если «A» равно 6, а «B» равно 4, то «C» равно 1.
C:=A>=B;	Если «A» равно 4, а «B» равно 4, то «C» равно 1.

Инструкция	Результат
C:=A>=B;	Если «A» равно 2, а «B» равно 4, то «C» равно 0.

9.2.10 Равно (=)

С помощью операции «=» значение первого операнда сравнивается со значением второго операнда. Если значение первого операнда равно значению второго операнда, то результатом будет логическое значение 1. Если значение первого операнда не равно значению второго операнда, то результатом будет логическое значение 0.

Пример выглядит следующим образом:

Инструкция	Результат
C:=A=B;	Если «A» равно 4, а «B» равно 4, то «C» равно 1.
C:=A=B;	Если «A» равно 4, а «B» равно 6, то «C» равно 0.
C:=A=B;	Если «A» равно 6, а «B» равно 4, то «C» равно 0.

9.2.11 Не равно (<>)

С помощью операции «<>» значение первого операнда сравнивается со значением второго операнда. Если значение первого операнда не равно значению второго операнда, то результатом будет логическое значение 1. Если значение первого операнда равно значению второго операнда, то результатом будет логическое значение 0.

Пример выглядит следующим образом:

Инструкция	Результат
C:=A<>B;	Если «A» равно 4, а «B» равно 4, то «C» равно 0.
C:=A<>B;	Если «A» равно 4, а «B» равно 6, то «C» равно 1.
C:=A<>B;	Если «A» равно 6, а «B» равно 4, то «C» равно 1.

9.2.12 Меньше, чем (<)

С помощью операции «<» значение первого операнда сравнивается со значением второго операнда. Если значение первого операнда меньше значения второго операнда, то результатом будет логическое значение 1. Если значение первого операнда больше или равно значению второго операнда, то результатом будет логическое значение 0.

Пример выглядит следующим образом:

Инструкция	Результат
C:=A<B;	Если «A» равно 4, а «B» равно 6, то «C» равно 1.
C:=A<B;	Если «A» равно 6, а «B» равно 4, то «C» равно 0.
C:=A<B;	Если «A» равно 6, а «B» равно 6, то «C» равно 0.

9.2.13 Меньше или равно (<=)

С помощью операции «<=» значение первого операнда сравнивается со значением второго операнда. Если значение первого операнда меньше или равно значению второго операнда, то результатом будет логическое значение 1. Если значение первого операнда больше значения второго операнда, то результатом будет логическое значение 0.

Пример выглядит следующим образом:

Инструкция	Результат
C:=A<=B;	Если «A» равно 4, а «B» равно 6, то «C» равно 1.
C:=A<=B;	Если «A» равно 6, а «B» равно 6, то «C» равно 1.
C:=A<=B;	Если «A» равно 6, а «B» равно 4, то «C» равно 0.

9.2.14 И (AND)

С помощью операции «AND» производится логическое умножение операндов. Для целочисленного типа данных операция выполняется побитово.

Пример выглядит следующим образом:

Инструкция	Результат
D:=A AND B AND C;	Если любое из «A», «B» и «C» равно 0, то «D» равно 0, в противном случае «D» равно 1.

9.2.15 Или (OR)

С помощью операции «OR» производится логическое сложение операндов. Для целочисленного типа данных операция выполняется побитово.

Пример выглядит следующим образом:

Инструкция	Результат
D:=A OR B OR C;	Если любое из «A», «B» и «C» равно 1, то «D» равно 1, в противном случае «D» равно 0.

9.2.16 Исключающее ИЛИ (XOR)

С помощью операции «XOR» производится логическое вычитание операндов. Для целочисленного типа данных операция выполняется побитово.

В следующем примере, если «A» отличается от «B», то «C» равно 1. Если «A» совпадает с «B» (например, оба они равны 0 или 1), то «C» равно 0.

Инструкция	Результат
C:=A XOR B;	Если «A» равно 1, а «B» равно 0, то «C» равно 1.
C:=A XOR B;	Если «A» равно 0, а «B» равно 1, то «C» равно 1.
C:=A XOR B;	Если «A» равно 1, а «B» равно 1, то «C» равно 0.
C:=A XOR B;	Если «A» равно 0, а «B» равно 0, то «C» равно 0.

9.2.17 Присвоение (:=)

С помощью операции «:=» текущее значение переменной заменяется на результат присвоения выражения. Присвоение включает в себя спецификацию левой переменной, второй следующий оператор присвоения «:=» и третье следующее выражение для присвоения.

В следующем примере данная операция используется для копирования значения одной переменной и присвоения этого значения другой переменной.

Инструкция	Толкование
A:=B;	Заменить значение «A» на текущее значение «B».

В следующем примере данная операция используется для непосредственного присвоения значения переменной.

Инструкция	Толкование
A:=10;	Присвоить 10 значению «A».

В следующем примере данная операция используется для присвоения переменной результата, возвращаемого из функционального блока.

Инструкция	Толкование
A:=ABS(B);	Присвоить результат функционального блока «ABS» значению «A».

В следующем примере данная операция используется для присвоения переменной результата операции.

Инструкция	Толкование
A:=(B+C-D)*E;	Присвоить результат операции «(B+C-D)*E» значению «A».

9.3 Оператор

9.3.1 Инструкция

Инструкция — это «команда» структурированного языка программирования, которая должна заканчиваться точкой с запятой. Несколько команд могут быть введены в одну строку (каждая команда разделяется точкой с запятой).

9.3.2 IF...THEN...ELSE...END_IF

Когда логическое значение инструкции, стоящей за «IF», равно 1 (TRUE), то будет выполнена инструкция или набор инструкций (разделенных «;»), стоящий за «THEN». Когда логическое значение инструкции, стоящей за «IF», равно 0 (FALSE), то будет выполнена инструкция или набор инструкций (разделенных «;»), стоящий за «ELSE». Инструкция «END_IF» используется для обозначения конца инструкции.

Пример выглядит следующим образом:

Инструкция	Толкование
IF %M1 = 1	Оценить, соответствует ли «%M1» условию?
THEN %MW1 := 1;	Если соответствует, то выполнить эту инструкцию.
ELSE %MW1 := 2;	Если не соответствует, то выполнить эту инструкцию.
END_IF;	Отметка об окончании инструкции.

9.3.3 CASE...OF... ELSE... END_CASE

Инструкция «CASE» состоит из выражения целочисленного типа данных и набора инструкций. Каждый набор имеет метку, и эта метка состоит из одного или нескольких целых чисел. Если метка содержит вычисленное значение выражения инструкции, то команда будет выполнена; если метка не содержит этого значения, то команда не будет выполнена. Инструкция «OF» указывает на начало метки. Команда «ELSE» может быть выполнена в инструкции «CASE»; однако она может быть выполнена только тогда, когда метка не содержит никакого значения селектора. «END_CASE» используется для обозначения конца инструкции.

Пример выглядит следующим образом:

Инструкция	Толкование
CASE A+B OF	Каким условиям соответствует значение «A+B» в операторе с несколькими ответвлениями?
1,5: C:=SIN(A) * COS(B);	Если значение равно 1 или 5, то выполняется эта инструкция.
2: B:=C-A;	Если значение равно 2, то выполняется эта инструкция.
3,4,6: C:=C*A;	Если значение равно 3, 4 или 6, то выполняется эта инструкция.
ELSE B:=C*A; C:=A / B;	Если какое-либо из вышеуказанных условий не выполняется, то выполняется эта инструкция.
END_CASE;	Отметка об окончании инструкции.

9.3.4 FOR...TO...BY...DO...END_FOR

Инструкция «FOR» используется при условии, что можно определить точное количество текущих совпадающих элементов. Если количество текущих совпадающих элементов не может быть определено, можно использовать только инструкции «WHILE» или «REPEAT». Инструкция «FOR» инициирует повторение последовательности инструкций до тех пор, пока не будет инициирована инструкция «END_FOR». Количество совпадающих элементов зависит от начального значения, конечного значения и управляющей переменной, которые должны иметь один и тот же тип данных и не могут быть изменены одной из повторяющихся инструкций. Инструкция «FOR» используется для добавления значения управляющей переменной от начального до конечного значения. Значение инкрементного значения по умолчанию равно 1. Если используется другое значение, можно указать явное инкрементное значение (константу). Перед каждым запуском цикла обновления необходимо проверять значение переменной. Если значение переменной находится вне диапазона начального и конечного значений, цикл бесполезен. Перед запуском первого цикла необходимо убедиться, что инкрементное значение управляющей переменной задано верно, (что оно начинается с начального значения и переходит к конечному). Если перемещения нет, то цикл не будет выполнен, а без использования прерываний и должного контроля логики программы ПЛК может перейти в бесконечный цикл, для выхода из которого потребуется сброс ПЛК. При соблюдении данного принципа, цикл будет выполняться в нормальном порядке. Команда «DO» используется для обнаружения окончания повторяющихся определений и начала команд. Повторение можно завершить раньше при помощи инструкции «EXIT». «END_FOR» используется для обозначения конца инструкции.

Пример «FOR» с инкрементным значением 1: в следующем примере «I» — это управляющая переменная, 1 - начальное значение, а 5 - конечное значение.

Инструкция	Толкование
FOR I:= 1 TO 5 DO	«I» увеличивается на значение по умолчанию 1, выполняется циклическое выполнение следующей инструкции, когда «I» больше 5, циклическая инструкция «FOR» завершается.
C:= C+4;	Набор инструкций для выполнения цикла.
END_FOR;	Отметка об окончании инструкции.

Если инкрементное значение не равно 1, то инкрементное значение можно задать при помощи «BY». Направление обработки (вперед или назад) зависит от символа константы «BY». Если символ положительный, то цикл выполняется вперед, если отрицательный - назад.

Пример «FOR» с инкрементным значением, не равным 1: в следующем примере «I» — это управляющая переменная, 1 - начальное значение, а 5 - конечное значение.

Инструкция	Толкование
FOR I:= 1 TO 5 BY 2 DO	«I» увеличивается на инкрементное значение 2, выполняется циклическое выполнение следующей инструкции, когда «I» больше 5, циклическая инструкция «FOR» завершается.
C:= C+4;	Набор инструкций для выполнения цикла.
END_FOR;	Отметка об окончании инструкции.

9.3.5 WHILE...DO...END_WHILE

Результат выполнения команды «WHILE» таков: последовательность команд будет выполняться повторно до тех пор, пока связанное с ней логическое выражение не станет равным 0 (FALSE). Если начало выражения равно 0, то набор команд не будет выполнен. Команда «DO» используется для обнаружения окончания повторяющихся определений и начала команд. Процесс можно завершить раньше при помощи инструкции «EXIT». «END_WHILE» используется для обозначения конца инструкции. В случае, если логическое выражение никогда не станет равным 1 (TRUE), без использования прерываний и должного контроля логики программы ПЛК может перейти в бесконечный цикл, для выхода из которого потребуется сброс ПЛК.

Пример выглядит следующим образом:

Инструкция	Толкование
A := 1;	Присвоить константу 1 к «A».
WHILE A<= 100 DO A := A + 4;	Если «A» меньше или равно 100, то повторно выполнить инструкцию цикла «A := A + 4».;

Инструкция	Толкование
	Если «A» больше 100, то больше не выполнять инструкцию цикла «A := A + 4».
END_WHILE;	Отметка об окончании инструкции.

9.3.6 REPEAT…UNTIL…END_REPEAT

Результат выполнения команды «REPEAT» таков: последовательность команд будет выполняться повторно (по крайней мере, один раз) до тех пор, пока связанное с ней логическое выражение не будет равно 1 (TRUE). Инструкция «UNTIL» используется для обозначения конца условия. Процесс можно завершить раньше при помощи инструкции «EXIT». «END_REPEAT» используется для обозначения конца инструкции. В случае, если логическое выражение никогда не станет равным 0 (FALSE), то без использования прерываний и должного контроля логики программы ПЛК может перейти в бесконечный цикл, для выхода из которого потребуется сброс ПЛК.

Пример выглядит следующим образом:

Инструкция	Толкование
A := 1;	Присвоить константу 1 к «A».
REPEAT	
A := A + 2	Выполнить инструкцию цикла «A := A + 2».
UNTIL A >= 100	Если «A» меньше 100, то выполнить инструкцию цикла «A := A + 2».; Если «A» больше или равно 100, то завершить инструкцию цикла.
END_REPEAT;	Отметка об окончании инструкции.

9.3.7 EXIT

Инструкция «EXIT» используется чтобы завершить инструкцию цикла (такую как, «FOR», «WHILE», «REPEAT») до выполнения условия завершения. Если инструкция «EXIT» находится во встроенном соответствующем элементе, то останется только самый глубокий внутренний цикл (то есть позиция «EXIT»). Следующая операция заключается

в выполнении первой инструкции, следующей за завершением цикла («END_FOR», «END_WHILE» или «END_REPEAT»).

9.3.8 RETURN

Инструкция «RETURN» используется для прерывания сканирования подпрограммы и возврата к основной программе без каких-либо условий.

9.3.9 Комментарий

В редакторе ST комментарий всегда начинается с символьной строки «(*» и заканчивается символьной строкой «*)». Между этих символьных строк может быть вставлен любой комментарий. Также комментарий может быть введен в любое место редактора ST. Комментарий отображается другим цветом.

Символьная строка «//» используется для вставки комментария текущей строки.

9.3.10 GOTO

В редакторе ST метка заканчивается на «:», например «AA:». Оператор «GOTO» осуществляет переход к указанной метке, например «GOTO AA;».

9.4 Функциональный блок

По своим функциям, основные функциональные модули разделяются на такие группы, как математика, статистика, логика, сравнение, преобразование, перемещение, таймер, счетчик, управление, ПЛК и другие. Выберите группу на панели инструментов, как показано на рис.9.1, затем выберите функциональный блок, как показано на рис.9.2, щелкните значок. Выбранный функциональный блок вставляется в редактор ST. Все основные функции и параметры функциональных блоков описаны в главе 5 «Базовый функциональный блок».

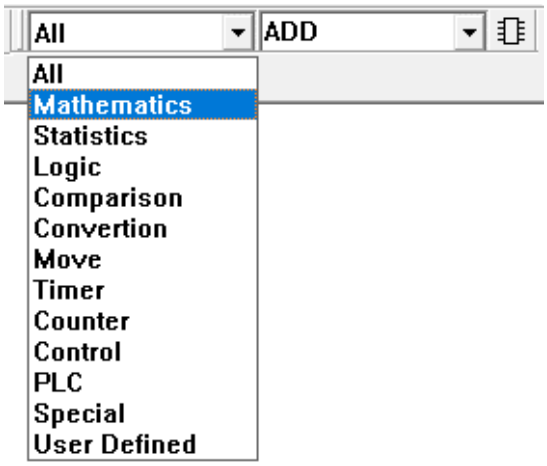


Рис.9.1 Выбор группы ST

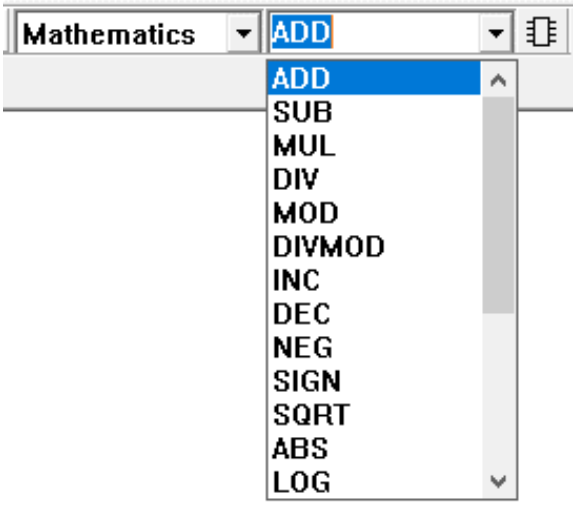


Рис.9.2 Выбор функционального блока ST

10 Программирование SCC

Современные задачи управления включают в себя последовательное управление, непрерывное управление, управление перемещением и т.д. Традиционный подход к процессу последовательного управления заключается в том, чтобы сначала сконструировать основную последовательную схему управления, а затем реализовать процесс последовательного управления при помощи таких языков программирования, как LD и ST.

Такой традиционный подход не совсем неудобен, так как требует большой рабочей нагрузки и не позволяет быстро выявлять проблемы в процессе последовательного управления. Язык SCC (язык последовательных управляющих схем) — это новый язык программирования, разработанный для устранения вышеуказанных трудностей в процессе последовательного управления. В нем используются специальные диаграммы для описания последовательного процесса управления. Такой подход интуитивно понятен, удобен и лёгок для восприятия пользователем. SCC предоставляет простое графическое описание процесса, и это один из наиболее естественных языков для описания задач последовательного управления.

При работе с SCC, после двойного щелчка левой кнопкой мыши по названию программы SCC в браузере проекта содержимое этой программы SCC отображается в правой части области редактирования, где его можно отредактировать.

Область редактирования SCC управляется с помощью многостраничного режима. Каждая программа SCC поддерживает до 16 страниц. Размещение функционального элемента блок-схемы SCC не ограничено страницей. Он может быть размещен на разных страницах, последовательность которых может не соблюдаться. Однако последовательность с первой страницы по шестнадцатую соблюдается при печати, поэтому программу SCC нужно прописывать как можно более удобной для чтения.

Для того, чтобы поместить функциональный элемент блок-схемы в область редактирования, необходимо выбрать функциональный элемент блок-схемы в меню SCC или на панели инструментов и щелкнуть на области редактирования левой кнопкой мыши. Функциональный элемент блок-схемы будет помещен в месте нажатия мыши. Когда функциональный элемент блок-схемы необходимо переместить, его нужно выбрать, и, удерживая левую кнопку мыши после нажатия на нем переместить в необходимое место. Такие операции, как «вырезать», «копировать», «вставить» и «удалить» могут быть выполнены с помощью операций меню. Работа с несколькими функциональными элементами блок-схемы ничем не отличается, но сначала с помощью операций с блоками необходимо выбрать несколько функциональных элементов блок-схемы.

Дважды щелкните на функциональном элементе блок-схемы, чтобы отредактировать его свойства.

Когда точка используется функциональным элементом блок-схемы, можно использовать форму «тип + номер», такую как %I0001 (при вводе она может быть сокращена до %I1), %MW0001 и т.д. (Тип I, MW и т.д. следует писать с заглавной буквы), и также можно использовать имя, заданное в таблице точек. Однако это имя обязательно должно использоваться, когда вводится переменная V.

Когда два функциональных элемента блок-схемы находятся достаточно близко (выше и ниже), связь между вводами/выводами соответствующего типа будет установлена автоматически. Она также может быть установлена вручную с помощью меню **【Link】**. Если изменится расположение двух функциональных элементов блок-схемы относительного друг друга, направление связи между ними также автоматически изменится. Когда функциональный элемент блок-схемы удаляется, все ссылки с этим функциональным элементом блок-схемы удаляются автоматически.

10.1 Структура данных

SCC — это графическое описание процесса управления. При фактическом использовании, для формирования языка последовательного управления должна использоваться комбинация данных.

10.1.1 Оператор

Операторы, используемые в SCC, приведены ниже. Сначала выполняются операции с более высоким приоритетом, после них - операции с более низким приоритетом. Операции с одинаковым приоритетом будут выполняться слева направо. Если сначала необходимо выполнить операцию с более низким приоритетом, следует добавить круглые скобки.

Оператор	Описание	Приоритет
( )	Круглые скобки	1
!	Логическое NOT	2
~	Побитовое NOT	3
*	Умножение	4
/	Деление	4
+	Прибавление	5
-	Вычитание	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
&	Логическое AND	9
^	Логическое XOR	10
|	Логическое OR	11
&&	Сравнение AND	12
||	Сравнение OR	13
=	Присвоение	14

10.1.2 Базовая функция

«x» и «y» — это константы, переменные или выражения.

Имя функции	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)	Натуральное возведение в степень
lg(x)	Логарифм
expt(x, y)	Возведение в степень
abs(x)	Абсолютное значение
sqrt(x)	Квадратный корень

10.1.3 Variable (Переменная)

Если данные, используемые в SCC, не являются постоянными (константами), то их можно назвать переменными. Переменные разделяются на локальные и глобальные.

Локальная переменная подразделяется на два типа: целочисленный тип (int) и тип с плавающей запятой (float). В начале работы с любой новой программой, система определяет пять целочисленных переменных (int): m_i1, m_i2, m_i3, m_i4, m_i5 и пять переменных с плавающей запятой (float): m_f1, m_f2, m_f3, m_f4, m_f5. Локальные переменные действуют только в текущей программе, поэтому в разных программах могут использоваться локальные переменные с одинаковым именем без взаимодействия между собой.

Глобальные переменные — это точки, созданные в файле проекта, такие как: %I0001, %MW0001 и т.д. Глобальные переменные также могут использовать имя точки, заданное в таблице точек, например, имя %I0001 может быть определено как «DL_ON» и SCC может обращаться непосредственно к нему. Глобальные переменные действуют во всех программах LD и других программах. Если значение точки изменено в любом SCC, то это значение изменится во всех программах.

10.1.4 Выражение

Выражение может быть одной константой, переменной или комбинацией оператора с операндом. Например, все следующие выражения являются допустимыми:

m_i1 = %MW0001

m_i2 = 5 + var (Примечание: «var» — это имя точки %IW0001, заданное в таблице точек)

m_i3 = m_i4 * 3

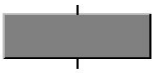
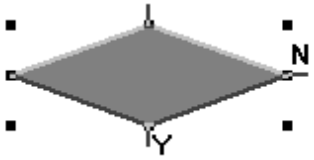
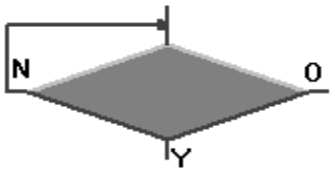
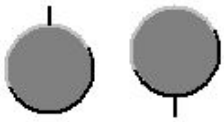
%Q0001 = 1

10.2 Функциональный элемент блок-схемы

Прежде чем использовать SCC для программирования, ознакомьтесь с различными функциональными элементами блок-схемы. Для установки функционального элемента блок-схемы, выберите его в панели инструментов SCC и поместите его в области редактирования. Когда параметры функции необходимо отредактировать, дважды щелкните по этому функциональному элементу блок-схемы.

Функциональные элементы блок-схемы, используемые в SCC, приведены ниже:

Функциональный элемент блок-схемы	Описание	Значок
Начало	Каждый SCC должен иметь уникальный начальный элемент.	

Завершение	SCC должен иметь конечный элемент. Их может быть несколько.	
Элемент исполнения	Когда необходимо выполнить операцию, можно использовать элемент выполнения.	
Условие	Элемент «условие» используется для определения выполняется ли условие.	
Условие, ограниченное по времени	Элемент «условие, ограниченное по времени» используется для определения того, выполняется ли условие в течение определенного времени.	
Соединитель	Соединитель используется для соединения нескольких частей программы или для перехода между различными частями программы.	

10.2.1 Начало

Каждый SCC должен иметь уникальный начальный элемент. Это начало программы SCC. Если его нет, то «СКПро» выдаст сообщение об ошибке при компиляции программ. Начальному элементу не требуется установка параметров, как показано на рис.10.1:

Рис.10.1 Начало

10.2.2 Завершение

SCC должен иметь конечный элемент. Их может быть несколько, что позволяет SCC выходить из разных ветвей. Конечному элементу также не требуется установка параметров. Есть SCC, которые выполняются циклически. Они могут не иметь конечного элемента. Как показано на рис.10.2:

Рис.10.2 Завершение

10.2.3 Элемент исполнения

Когда необходимо выполнить операцию, можно использовать элемент исполнения. Как показано на рис.10.3:

Рис.10.3 Элемент исполнения

Функция исполнения содержит 13 основных операций и может быть подобрана в соответствии с требованиями. При открытии элемента исполнения, можно выбрать тип операции из списка основных операций, а параметры операции могут быть установлены после нажатия кнопки «Property». Как показано на рис.10.4:

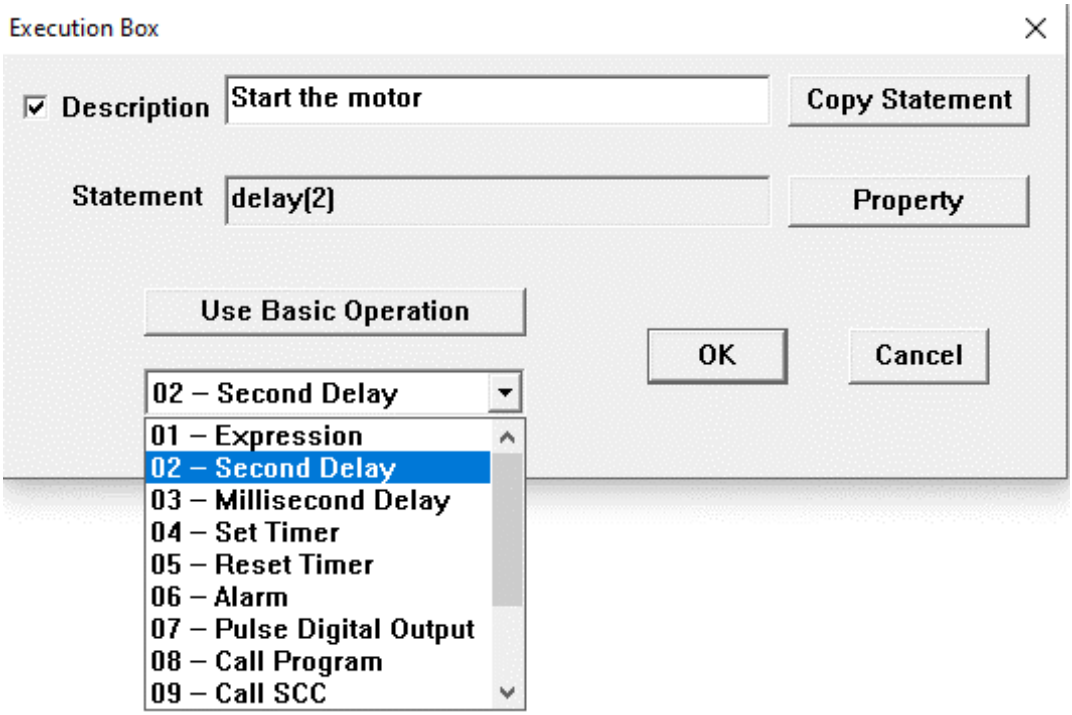


Рис.10.4 Операции элемента исполнения

✧ **Выражение**

Выражения могут выполнять такие операции, как присвоение, чтение данных, вычисления и так далее. Пользователь может использовать операторы, указанные слева, и определенные переменные справа, или точки и константы напрямую. Как показано на рис.10.5:

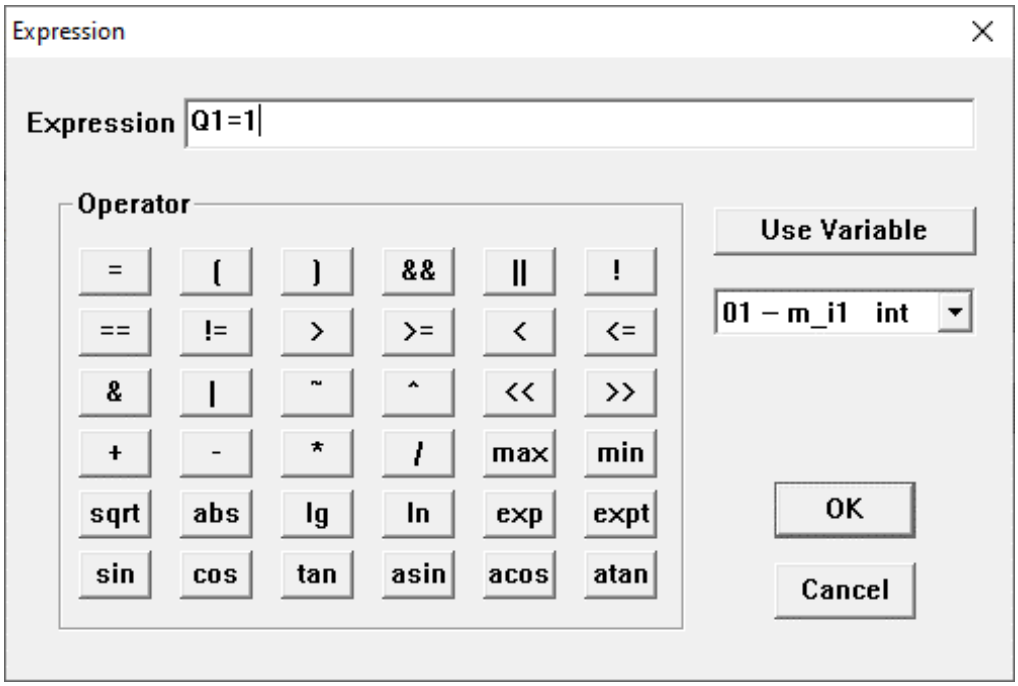


Рис.10.5 Выражение

Общими операциями выражений являются:

【Assignment】: Выражение может задавать параметры точки или переменной. Например, %Q0001 = 1, %MW0001 = 100 и так далее.

【Read Value】: Зачитать значение точки и сохранить его в переменной. Например, v_1 = %IW0001.

【Computing】: Вычислить данные точки. Например, v_1 = sin %MW0001.

✧ **Задержка**

После выполнения определенных операций управления (таких как цифровой вывод), операции, следующие за ними, должны выполняться спустя определенный промежуток времени, или после того, как сигнал поступит обратно. Для этого существует операция «Delay». Введите секунды для определения длины задержки (положительное целое число), и нажмите кнопку «OK», как показано на рис.10.6:

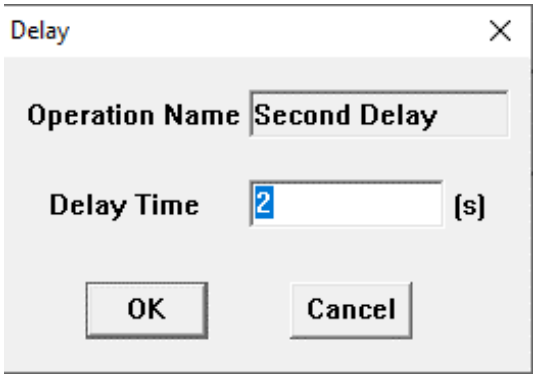


Рис.10.6 Задержка

✧ **Задержка на миллисекунды**

Операция «Millisecond delay» аналогична операции «Delay», но время задержки короче, а единица измерения равна миллисекунде. Минимальное время задержки — это время цикла сканирования системы, и, если установленное время меньше времени цикла сканирования, система автоматически установит время задержки равным циклу сканирования.

✧ **Установить таймер**

При последовательном управлении может возникнуть необходимость завершения процесса или операции в течение определенного времени, тогда для ограничения времени используется таймер. Когда время истечет, программа перейдет к исполнению метки выхода. Вручную таймер сбрасывать не нужно, потому что система сделает это автоматически. Если операция завершится до истечения времени таймера, следует добавить операцию «Reset timer» (сброс таймера). Использование этой операции показано на рис.10.7:

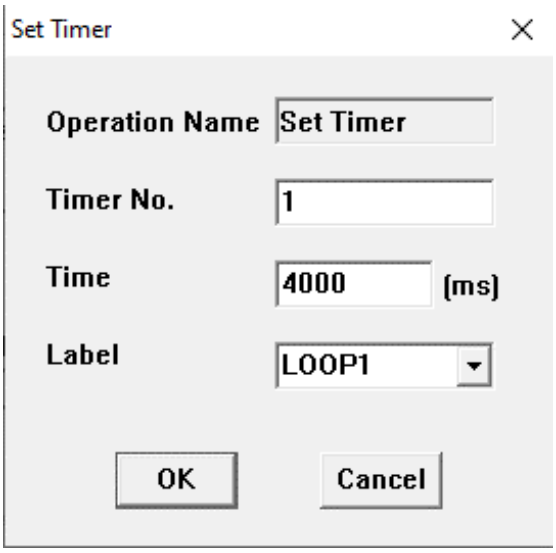


Рис.10.7 Установить таймер

Введите номер таймера (1-9), значение времени в миллисекундах и метку выхода, к которой осуществляется переход по истечении времени. Место ввода метки — это поле списка, в котором перечислены все метки, используемые в процедуре. Если необходимая метка уже существует, вы можете выбрать ее без ручного ввода.

✧ **Сбросить таймер**

Когда операция завершена, но время таймера не истекло, таймер необходимо сбросить. Когда установленное время таймера истекает, таймер сбрасывается автоматически, без участия пользователя. Введите номер таймера, который необходимо сбросить (положительное целое значение от 1 до 9), и нажмите кнопку «OK», как показано на рис.10.8:

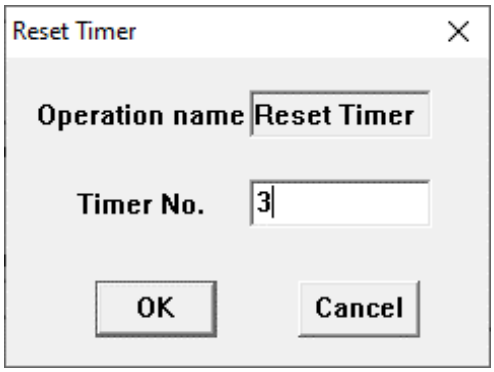


Рис.10.8 Сбросить таймер

✧ **Предупреждение**

На определенном этапе пользователь должен своевременно узнавать о состоянии выполнения последовательного управления, а некоторые неудачные операции необходимо выделять особенно, поэтому система поддерживает функцию оповещения. Когда последовательный элемент управления выполняет операцию «Предупреждение», содержимое оператора предупреждения появляется во вкладке Alarm окна вывода «СКПро». Пользователю просто нужно ввести оператор предупреждения, а затем нажать кнопку «OK», как показано на рис.10.9:

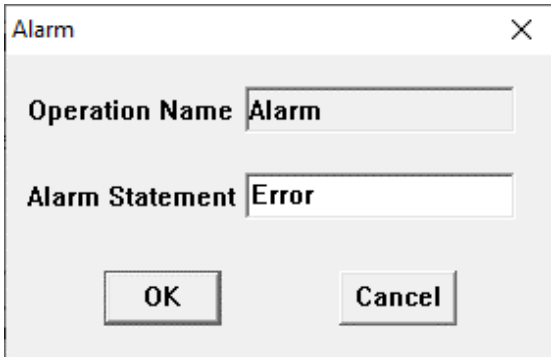


Рис.10.9 Предупреждение

✧ **Импульсный цифровой вывод**

Значение точки цифрового вывода всегда должно быть установлено в режиме последовательного управления. Если требуется длительное время вывода или время вывода является неопределенным, можно использовать выражение: сначала установить точку вывода равной 1, а затем сбросить ее на 0. Однако обычно делается так, что точка цифрового вывода выдает фиксированный импульс в течение нескольких секунд. В это время импульс может быть использован, как показано на рис.10.10:

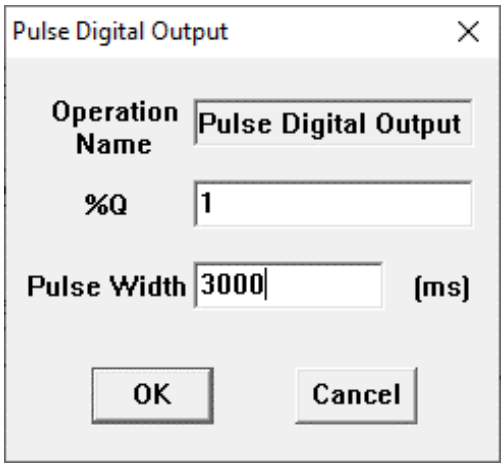


Рис.10.10 Импульсный цифровой вывод

Введите номер точки цифрового вывода и ширину выходного импульса. Помните, что единицей измерения длительности импульса является миллисекунда. После выполнения вышеуказанного функционального элемента будет выведено значение 1 для % Q0001, а затем, после удержания в течение 3 секунд, оно автоматически переключится на 0.

✧ **Вызвать программу**

SCC может напрямую вызывать программы LD, FBD, IL и ST. Когда SCC выполняет операцию «Call Program», система переходит к вызываемой программе и возвращается к исходному SCC после завершения одного сканирования. В диалоговом окне все программы текущего проекта перечислены в поле списка. Выберите и подтвердите программу, которую необходимо вызвать, как показано на рис.10.11:

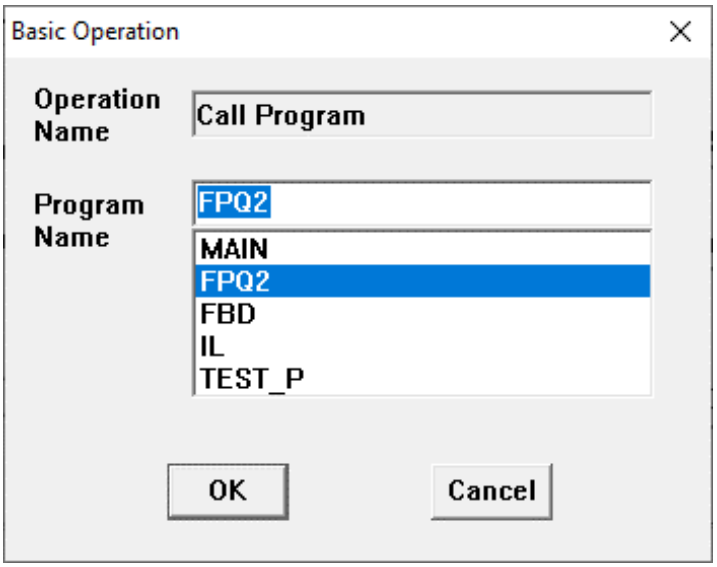


Рис.10.11 Вызвать программу

✧ **Вызов SCC**

Вызов SCC означает переход к выполнению другого SCC во время процесса выполнения программы изначально запущенного SCC. После завершения выполнения программы вызываемого SCC, система вернется к исходному SCC и продолжит, как показано на рис.10.12:

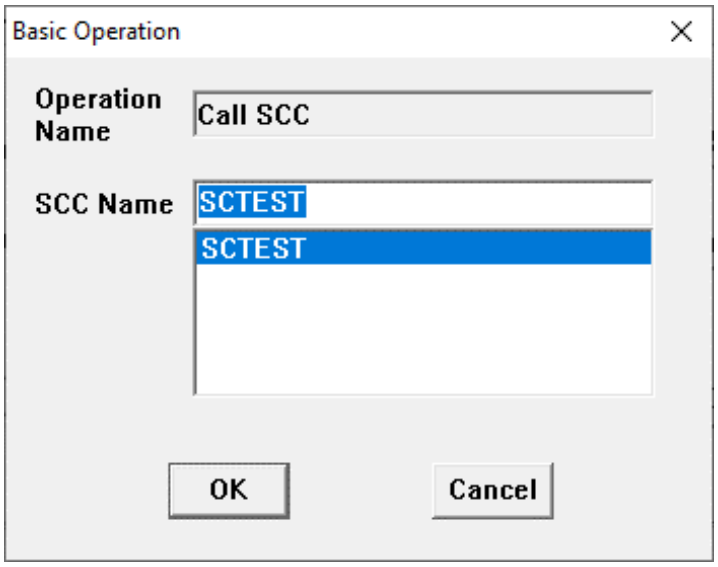


Рис.10.12 Вызов SCC

✧ **Выполнить SCC**

Иногда, при выполнении одного SCC, может потребоваться одновременное выполнение другого SCC, в этот момент может быть использована операция «Execute SCC». Исходный SCC продолжает выполняться параллельно с новым. Друг на друга они не влияют. В диалоговом окне все программы SCC текущего проекта перечислены в поле списка. Выберите и подтвердите программу, которую необходимо выполнить, как показано на рис.10.13:

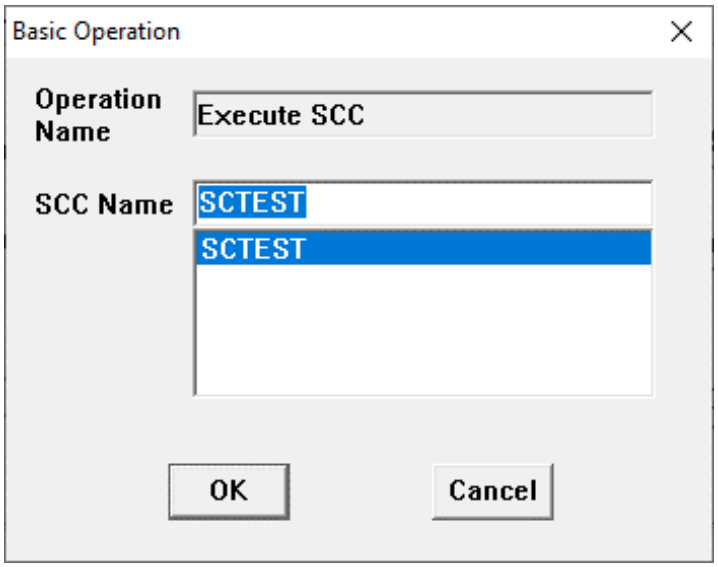


Рис.10.13 Выполнить SCC

✧ **Остановить SCC**

Иногда, когда выполняется одна программа SCC, необходимо остановить другую программу SCC, система поддерживает и такую операцию. Операция «Stop SCC» заключается в принудительной остановке другого SCC, но если в программе SCC были установлены определенные регистры, то система не сможет очистить их. Поэтому, при использовании операции «Stop SCC» обращайте внимания на необходимость очистки этих регистров. Выберите название программы SCC, которую требуется остановить, и нажмите кнопку «OK», как показано на рис.10.14:

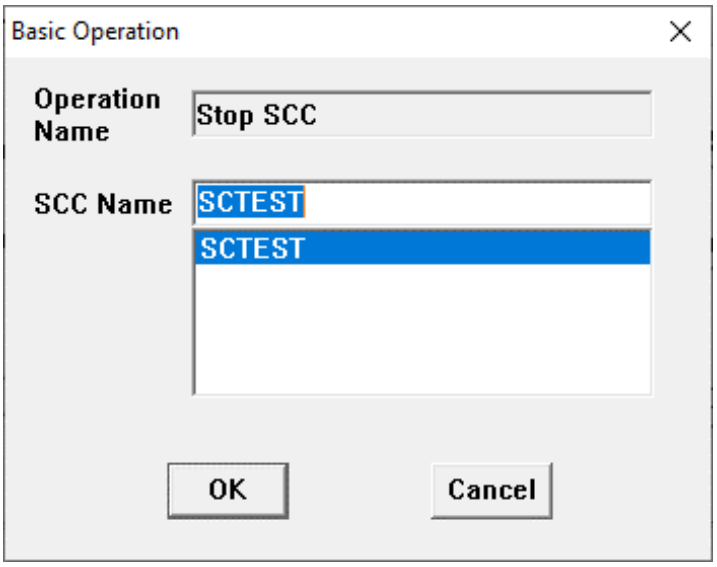


Рис.10.14 Остановить SCC

✧ **Заблокировать SCC и разблокировать SCC**

Когда выполняется один SCC, выполнение другого SCC должно быть запрещено (то есть, заблокировано); и когда выполнение этого SCC завершено - возобновлено (то есть, разблокировано) выполнение SCC, который был заблокирован. Эти две операции также существуют в системе. Выберите название SCC и нажмите кнопку «OK», как показано на рис.10.15 и рис.10.16:

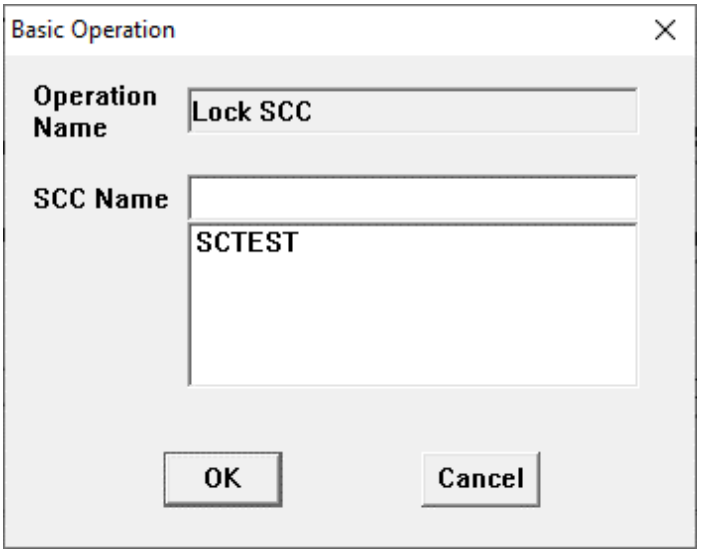


Рис.10.15 Заблокировать SCC

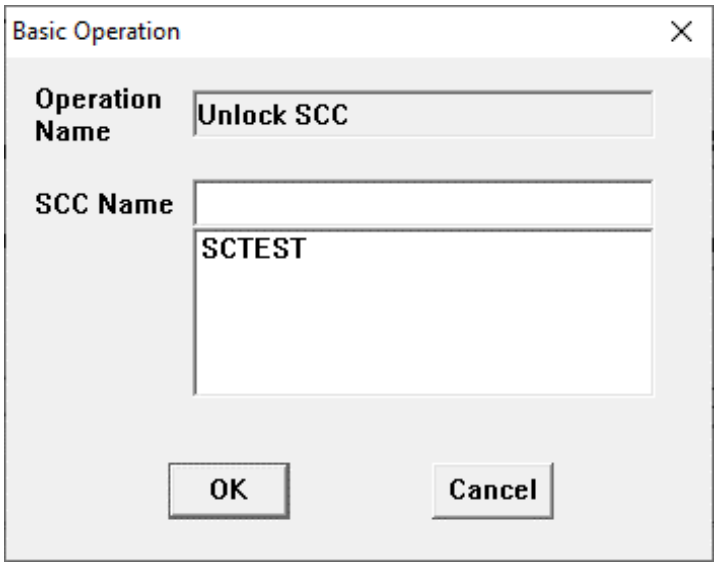


Рис.10.16 Разблокировать SCC

10.2.4 Условие

Элемент «условие» используется для определения выполняется ли условие. Если условие выполняется, перейдите в ветвь «Y», если условие не выполняется, перейдите в ветвь «N». Дважды щелкните на элементе, чтобы отредактировать условие оценки. Выражение условия поддерживает все вышеупомянутые операторы. Переменные также могут быть использованы, как показано на рис.10.17:

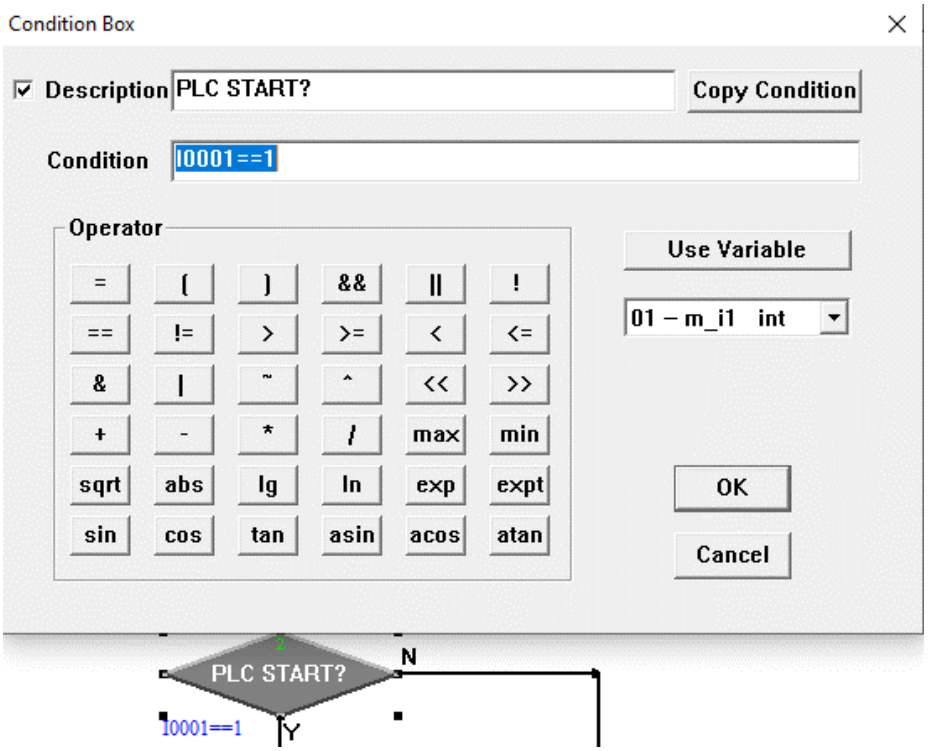


Рис.10.17 Условие

10.2.5 Условие, ограниченное по времени

Некоторые условия требуют, чтобы их оценивали в течение определенного времени. Если условие выполняется в течение ограниченного времени, перейдите в ветвь «Y»; если условие не выполняется в течение ограниченного времени, система продолжит

оценивать его до тех пор, пока не будет установлено, что по истечении ограниченного времени это условие не выполнено. В таком случае, перейдите в ветвь «O». Таймер используется для ограничения времени, и операции установки и сброса таймера будут выполняться системой автоматически, без прямого вмешательства пользователем, как показано на рис.10.18:

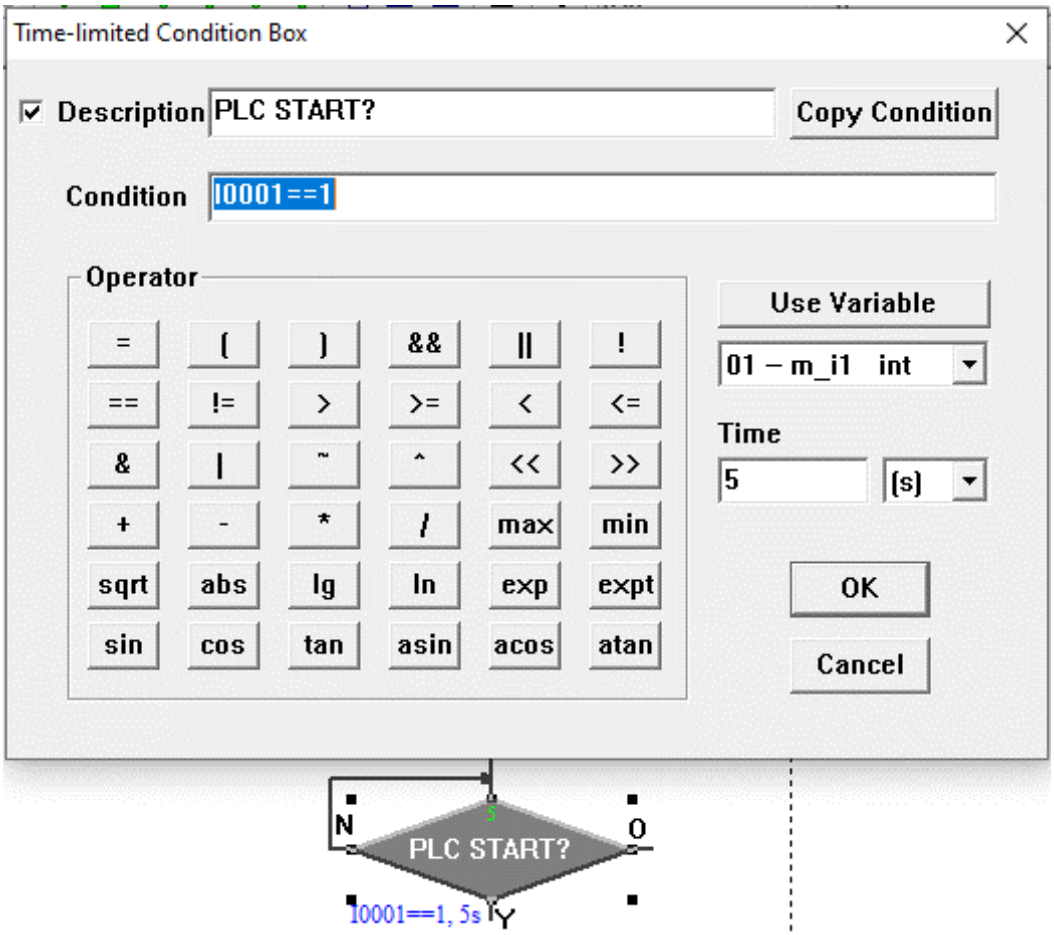


Рис.10.18 Условие, ограниченное по времени

10.2.6 Соединитель

Когда SCC является сложным, программа может быть разделена на несколько частей. В таком случае соединители используются для соединения каждой части программы или для выполнения перехода между различными частями, а также для того, чтобы слишком длинные связи не нарушали красивый внешний вид. Введите название соединителя и нажмите кнопку «OK», чтобы завершить соединение, как показано на рис.10.19:

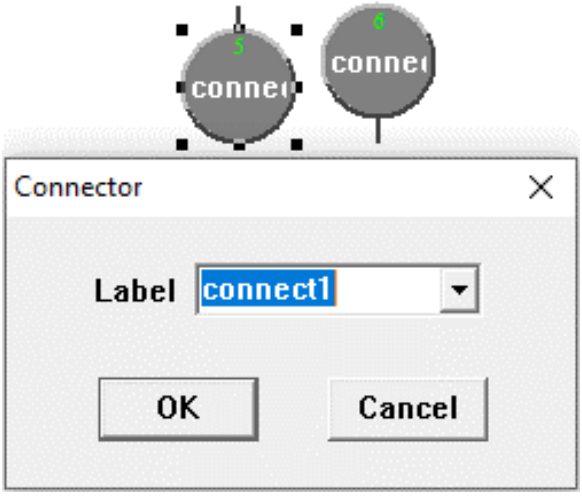


Рис.10.19 Соединитель

10.2.7 Комментарий

При написании программы, как правило, необходимо оставлять комментарии, чтобы облегчить понимание программы, в это время можно выбрать поле функции комментариев. Нажмите левую кнопку мыши там, где необходимо добавить комментарий, после чего появится диалоговое окно следующего вида (ниже). Введите нужные строки (например, «WELCOME»), нажмите кнопку «OK», и затем введенные строки появятся в программе. При необходимости перетащите этот комментарий, чтобы изменить его положение. Если хотите изменить содержание комментария, дважды щелкните на нем, как показано на рис.10.20:

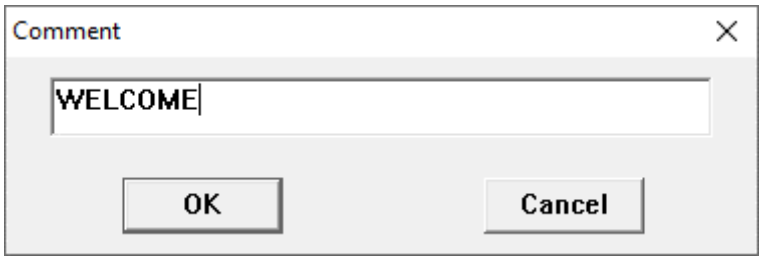


Рис.10.20 Комментарий

10.3 Связь с направлением потока

Связи каждого функционального элемента блок-схемы в SCC указывают последовательность выполнения каждого шага программы SCC. Ввод и вывод каждого шага должны быть связаны с вышестоящими и нижестоящими элементами, в противном случае возникнет ошибка. Метод создания связи приведен ниже.

10.3.1 Функция «Магнит»

Когда два функциональных элемента находятся на определенном расстоянии (снизу вверх), связь устанавливается автоматически и добавляется связь со стрелкой. Это указывает на то, что связь установлена между обоими функциональными элементами. Когда какой-либо функциональный элемент перемещается, соединение автоматически пересчитывается, и связь между обоими функциональными элементами сохраняется.

10.3.2 Создание связи с помощью мыши

Выберите кнопку **Link** на панели инструментов SCC, чтобы использовать функцию создания связи. Существует принцип связи, согласно которому ввод одного функционального элемента может быть связан только с выводом другого, в противном случае возникнет ошибка. Вывод одного функционального элемента не может быть одновременно связан со вводами двух функциональных элементов. Но ввод одного функционального элемента может быть связан с выводами нескольких функциональных элементов. Направление связи автоматически пересчитывается в соответствии с относительным положением двух функциональных элементов.

10.3.3 Удалить связь

Наведите указатель мыши на любой сегмент связи, нажмите левую кнопку мыши, ссылка станет синей, затем нажмите правую кнопку мыши, и появится всплывающее меню. Выберите пункт **Delete** в меню и нажмите левую кнопку, после чего выбранная связь будет удалена. Либо удалите связь с помощью клавиши **Delete** на клавиатуре.

10.3.4 Переместить связь

Иногда направление потока связи может быть нелогичным, проходить через другие функциональные элементы и портить внешний вид SCC. В этом случае направление потока связи можно отрегулировать по месту.

Существует принцип, согласно которому сегмент связи, непосредственно связанный с функциональным элементом, не может быть перемещен. Другие сегменты связи могут быть перемещены. Способ перемещения показан следующим образом: выберите **Move** на панели инструментов SCC, выберите сегмент связи, который необходимо переместить, нажмите левую кнопку, и тогда вся связь станет синей. Удерживайте левую кнопку рядом со связью, которую хотите переместить в соответствии с направлением данной связи. Перемещайте мышь в нужном направлении, и связь будет перемещена так, как вам требуется.

11 Отладка программы

ПО «СКПро» поддерживает множество языков программирования, таких как LD, FBD, IL, ST и SCC. В соответствии с характеристиками каждого языка для «СКПро» разработаны различные методы отладки, чтобы помочь специалистам модифицировать и дорабатывать программы в ходе проекта. Существует две среды отладки: отладка в режиме работающего ПЛК и отладка в симуляторе. Для отладки с ПЛК в онлайн-режиме, «СКПро» напрямую подключается к ПЛК, и изменения загружаются в него напрямую. Симулятор имитирует среду, в которой «СКПро» и ПЛК связаны виртуально. В нем выполняются предварительные тесты корректности выполнения программы посредством создания переменных, форсирования точек ввода-вывода и т.д., что обеспечивает успех дальнейшей отладки и безопасность эксплуатации оборудования. Ниже приведены иллюстрации шагов отладки для различных языков программирования.

11.1 Отладка LD / FBD

11.1.1 Онлайн-модификация ПЛК

Отладка LD/FBD интуитивно понятна. После включения все проводящие контуры становятся красными (%M0008), непроводящие - зелеными (%M0007), как показано на рисунке 11.1. Программа LD обычно используется для оценки состояния, поэтому в ней отображается непосредственно состояние точки ввода-вывода.

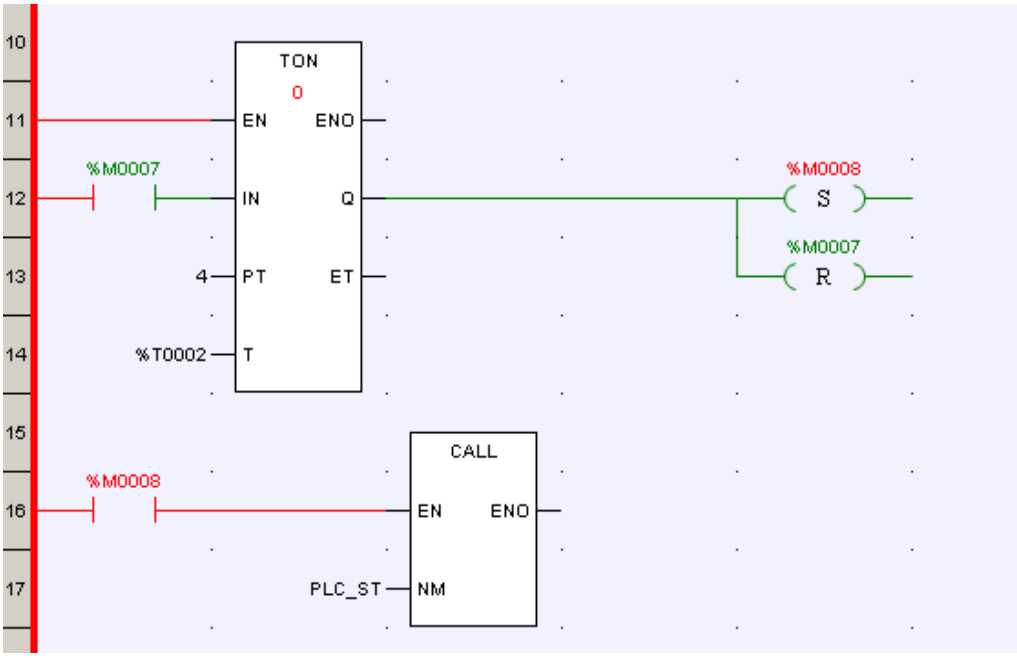


Рисунок 11.1 LD в режиме онлайн

После подключения, модификация LD/FBD может быть так же выполнена, как показано на рисунке ниже.

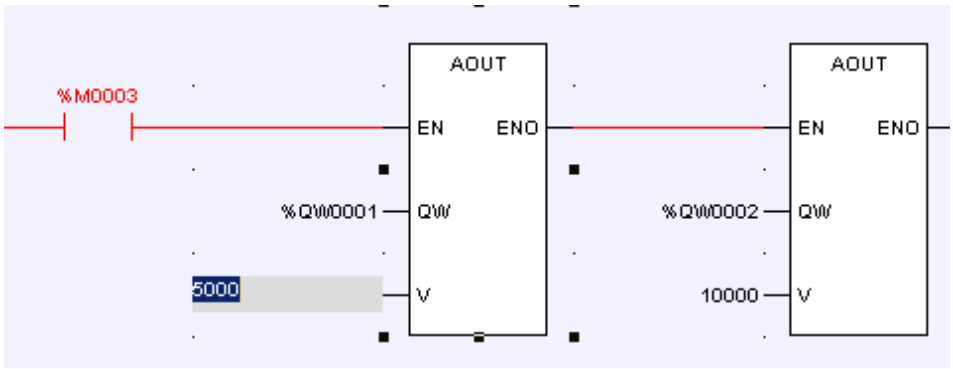


Рисунок 11.2 Модификация параметров

После подключения и внесения изменений в программу к названию программы в браузере проекта будет добавлен знак «*», например «MAIN*», как показано на рисунке 11.3. Это указывает на то, что текущая программа была изменена, но еще не была загружена.

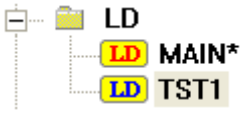


Рисунок 11.3 Программа изменена, но не загружена в ПЛК

После завершения модификации выберите значок в системной панели инструментов, чтобы сохранить измененный файл, затем выберите **【Online】/【Refresh Program】**, как показано на рисунке ниже, выгрузите измененную программу в ПЛК.

Рисунок 11.4 Обновить программу

11.1.2 Отладка с включенным ПЛК

Функция онлайн-отладки обычно используется при одноэтапной отладке программы. Точка прерывания может быть установлена в любом функциональном блоке LD/FBD во время отладки. Когда программа доходит до точки прерывания, она останавливается и ожидает выполнения **шага (Step)** инструкции. Если отладка

программы завершена, очистите все точки прерывания и нажмите кнопку ***Continue***, и программа продолжит обычное сканирование.

• Шаг (Step)

При отладке программы каждый блок функции представляет собой один шаг, поэтому программа останавливается в исходном месте после завершения каждого блока функции и ожидает инструкции **Step**. Инструкции **Step** отправляются следующим образом: наведите указатель мыши на кнопку ***Step*** на панели инструментов LD/FBD, а затем нажмите левую кнопку, чтобы отправить эту инструкцию.

• Продолжить (Continue)

Когда выполнение программы остановится, нажмите кнопку ***Continue*** на панели инструментов LD/FBD, после чего программа продолжит выполнение. Если в программе есть точки останова, то программа остановится, дойдя до первой точки останова и будет ждать инструкции **Step**. Если в программе нет точек останова, то программа будет выполняться непрерывно.

• Вставка/Удаление точки останова

Точка останова может быть установлена в любом функциональном блоке программ. Устанавливается точка останова следующим образом: выберите функциональный блок, в котором необходимо установить точку останова (наведите мышь на функциональный блок и нажмите левую кнопку мыши, чтобы выбрать его), затем наведите курсор мыши на кнопку ***Insert/Remove Breakpoint*** на панели инструментов LD/FBD, нажмите левую кнопку мыши, чтобы в правом верхнем углу выбранного блока функции появилась зеленая точка, это указывает на то, что точка останова успешно установлена. Если снова нажать кнопку ***Insert/Remove Breakpoint*** на панели инструментов LD/FBD, то точка останова будет удалена. Если нажать кнопку в третий раз, то точка останова будет установлена снова. Программа может одновременно установить до 10 точек останова в разных местах, как показано на рисунке 11.5:

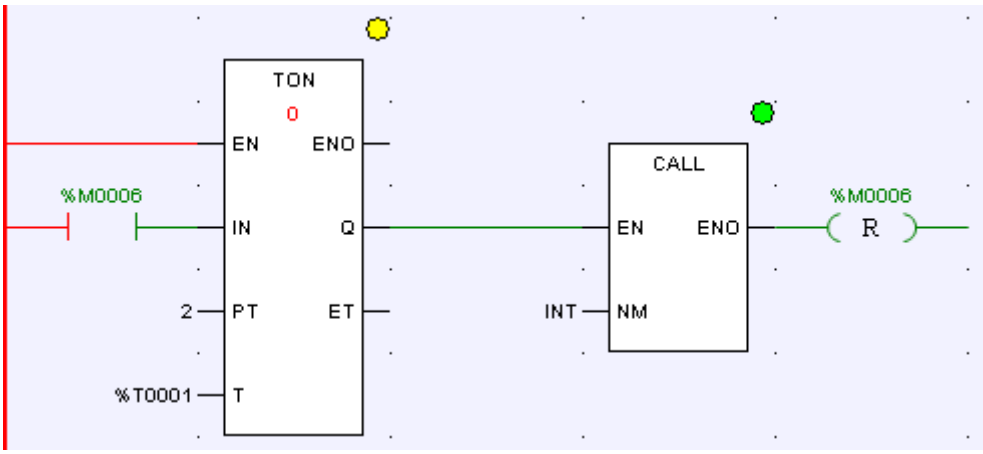


Рисунок 11.5 Установка точки останова в LD

● **Очистить все точки останова**

Все точки останова в программе могут быть очищены одновременно. Чтобы запустить очистку, наведите курсор мыши на кнопку ***Clear All Breakpoints*** на панели инструментов LD/FBD и нажмите левую кнопку, после чего все точки останова будут удалены.

Указание
После выполнения все точки останова будут автоматически удалены.

11.2 Отладка SCC

После того, как программа SCC скомпилирована и загружена, ее необходимо отладить. Перед отладкой систему необходимо подключить к ПЛК или симулятору. В этом случае можно начать использовать панель инструментов SCC, и могут быть отправлены инструкции по отладке.

11.2.1 Автоматическое выполнение

Во время работы в режиме автоматического выполнения ПЛК не будет выгружать информацию в процессе выполнения программы, и процесс выполнения программы в данный момент не может быть отражен в режиме реального времени на компьютере для отладки. Чтобы запустить автоматический режим, нажмите кнопку **Automatic** на панели инструментов SCC.

11.2.2 Наблюдение за выполнением

В режиме наблюдения за выполнением ПЛК загружает результат выполнения программы в режиме реального времени и процесс выполнения программы можно наблюдать на компьютере для отладки (только на компьютере, который отправил инструкции по отладке). В этом режиме выполнения отладочный персонал может отслеживать и управлять процессом выполнения программы в режиме реального времени и анализировать условия работы различных устройств (это может быть отражено в программе). В процессе выполнения программы, запущенный функциональный элемент отображается красным цветом, выполненные функциональные элементы отображаются синим цветом, а невыполненные - серым цветом. Чтобы запустить режим наблюдения, наведите курсор мыши на кнопку ***Watching*** на панели инструментов SCC и нажмите левую кнопку. В случае вызова подпрограммы система автоматически переключится на окно подпрограммы и будет следить за выполнением подпрограммы. После завершения подпрограммы система автоматически вернется к окну текущей программы, как показано на рисунке ниже:

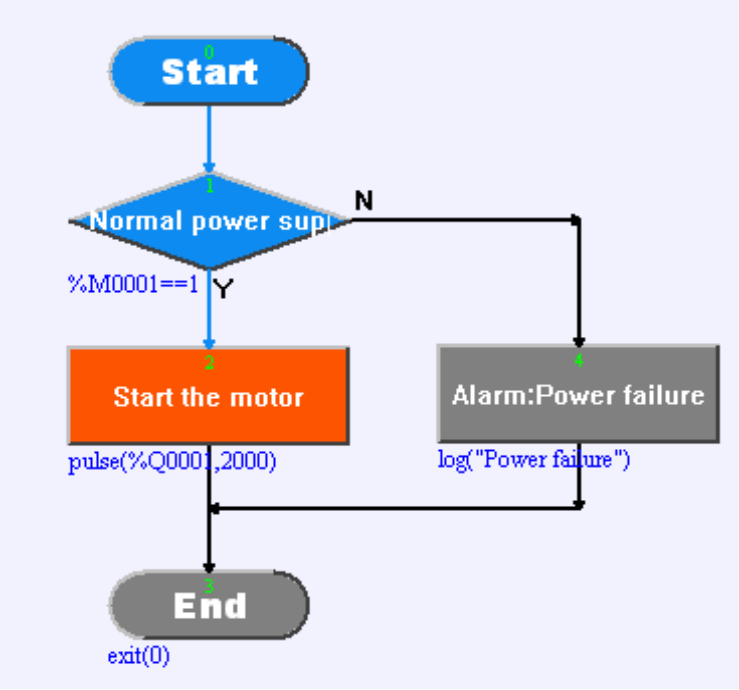


Рисунок 11.6 Наблюдение за выполнением SCC

11.2.3 Остановить выполнение

Когда наблюдение за выполнением программы находится в определенной точке, вы можете отправить инструкцию **Stop**, если хотите остановить выполнение программы. Чтобы остановить наблюдение, наведите курсор мыши на кнопку **Stop** на панели инструментов SCC и нажмите левую кнопку, после чего будет отправлена инструкция **Stop**. ПЛК остановит выполнение программы после получения инструкции.

11.2.4 Выполнение отладки

В отличие от просмотра выполнения, режим отладки выполнения больше подходит для отладки программ и поддерживает множество методов. Чтобы запустить отладку, выберите кнопку ***Debugging*** на панели инструментов SCC и нажмите левую кнопку мыши, затем отправляется инструкция **Debugging**, и программа выполняется. Однако, выполнение программы пошаговое, и после завершения первого шага программа остановится и будет ждать инструкции **Step**.

- **Шаг (Step)**

Когда программа находится в режиме выполнения одноступенчатой отладки, каждый функциональный элемент блок-схемы представляет собой один шаг. Программа останавливается в исходном положении и ждет инструкцию **Step** после завершения одного шага. Чтобы отправить инструкцию, наведите курсор мыши на кнопку ***Step*** и нажмите левую кнопку мыши, после чего инструкция будет отправлена.

•Вставка/Удаление точки останова (Insert/Remove Breakpoint)

Точка останова может быть установлена в любом функциональном блоке программ. Устанавливается точка останова следующим образом: выберите функциональный блок, в котором необходимо установить точку останова (наведите мышь на функциональный блок и нажмите левую кнопку мыши, чтобы выбрать его), затем наведите курсор мыши на кнопку ***Insert/Remove Breakpoint*** на панели инструментов LD/FBD, нажмите левую кнопку мыши, чтобы на выбранном функциональном блоке появилась зеленая точка, это указывает на то, что точка останова успешно установлена. Если снова нажать кнопку ***Insert/Remove Breakpoint*** на панели инструментов SCC, то точка останова будет удалена. Если нажать кнопку в третий раз, то точка останова будет установлена снова. Программа может одновременно установить до 10 точек останова в разных местах, — см. Рисунок 11.7:

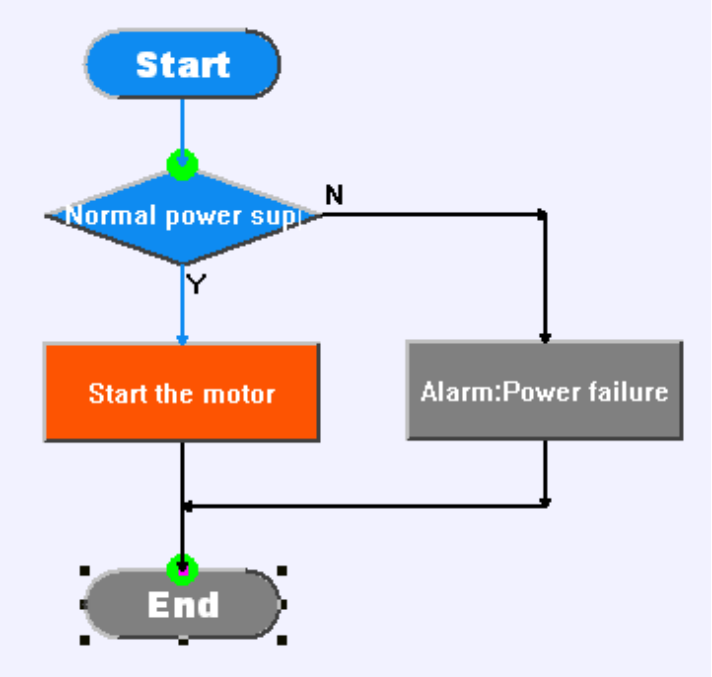


Рисунок 11.7 Установка точки останова в SCC

• Продолжить (Continue)

Находясь в режиме пошагового выполнения, нажмите кнопку ***Continue*** на панели инструментов SCC, после чего программа продолжит выполнение. Если в программе есть точки останова, то программа остановится, дойдя до первой точки останова и будет ждать инструкции **Step**. Если в программе нет точек останова, то программа будет выполняться до завершения.

• Очистить все точки останова

Все точки останова в программе могут быть очищены одновременно в режиме пошагового выполнения. Чтобы запустить очистку, наведите курсор мыши на кнопку ***Clear All Breakpoints*** на панели инструментов SCC и нажмите левую кнопку, после чего все точки останова будут удалены.

Указание

После выполнения все точки останова будут автоматически удалены.

• Перезапуск

Когда в режиме пошагового выполнения программа выполнена наполовину и ее требуется перезапустить, есть два способа сделать это:

① Нажмите кнопку ***Restart*** на панели инструментов SCC, и программа будет запущена заново.

② Сначала нажмите кнопку ***Stop Debugging*** на панели инструментов SCC, чтобы выполнить операцию **Stop debugging**. Затем, чтобы запустить программу и перезапустить выполнение с самого начала, выберите метод отладки.

• Прекратить отладку

В режиме выполнения отладки нажмите кнопку ***Stop Debugging***, и выполнение отладки программы будет остановлено.

11.2.5 Блокировка и разблокировка

С помощью кнопок ***Lock*** и ***Unlock*** можно вручную выполнить блокировку и разблокировку программы, как показано на рисунке 7.8.

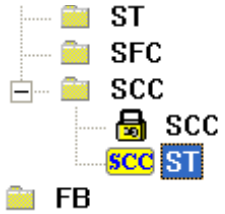


Рисунок 7.1 Блокировка SCC

11.3 Отладка IL

Функция отладки в реальном времени в основном используется при пошаговой отладке программы. В процессе отладки точка останова может быть установлена в любой допустимой строке IL. Программа остановится при достижении точки останова и будет ждать инструкции **Step**. Если отладка завершена, все точки останова будут очищены. Нажмите на кнопку ***Continue***, чтобы программа продолжила обычное сканирование.

• Шаг (Step)

При отладке программы каждая строка кода представляет собой один шаг. Программа останавливается в исходном положении и ждет инструкцию **Step** после завершения одного шага. Чтобы отправить инструкцию **Step**, наведите курсор мыши на кнопку ***Step*** на панели инструментов IL и нажмите левую кнопку мыши, после чего инструкция **Step** будет отправлена.

• Продолжить (Continue)

Находясь в режиме пошагового выполнения, нажмите кнопку ***Continue*** на панели инструментов IL, чтобы продолжить выполнение программы. Если в программе есть точки останова, то программа остановится, дойдя до первой точки останова и будет ждать инструкции **Step**. Если в программе нет точек останова, то программа будет выполняться постоянно.

• Вставка/Удаление точки останова

Точку останова можно установить в любой допустимой строке программы. Способ установки точки останова: наведите курсор на строку, где нужно установить точку останова, затем нажмите левой кнопкой мыши на кнопку ***Insert/Remove Breakpoint*** на панели инструментов IL. Строка загорится зеленым, что означает, что точка останова успешно установлена. Если снова нажать кнопку ***Insert/Remove Breakpoint***, то точка останова будет удалена. Если нажать кнопку в третий раз, то точка останова будет установлена снова. Программа может одновременно установить до 10 точек останова в разных местах, как показано на рисунке 7.9:

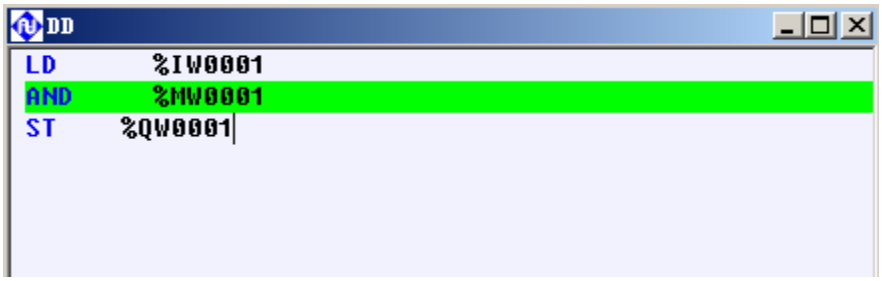


Рисунок 7.2 Установка точки останова в IL

• Очистить все точки останова

Все точки останова в программе могут быть очищены одновременно. Чтобы запустить очистку, нажмите левой кнопкой мыши на кнопку ***Clear All Breakpoints*** на панели инструментов IL после чего все точки останова будут удалены.

Указание
После отключения все точки останова удаляются автоматически.

11.4 Отладка ST

Функция отладки в реальном времени в основном используется при пошаговой отладке программы. В процессе отладки точка останова может быть установлена в любой допустимой строке ST. Программа остановится при достижении точки останова и будет ждать инструкции **Step**. Если отладка завершена, все точки останова будут очищены. Нажмите на кнопку ***Continue***, чтобы программа продолжила обычное сканирование.

• Шаг (Step)

При отладке программы каждая строка кода представляет собой один шаг. Программа останавливается в исходном положении и ждет инструкцию **Step** после завершения одного шага. Чтобы отправить инструкцию **Step**, наведите курсор мыши на кнопку ***Step*** на панели инструментов ST и нажмите левую кнопку мыши, после чего инструкция **Step** будет отправлена.

• Продолжить (Continue)

Когда выполнение программы остановится, нажмите кнопку ***Continue*** на панели инструментов ST, после чего программа продолжит выполнение. Если в программе есть точки останова, то программа остановится, дойдя до первой точки останова и будет ждать инструкции **Step**. Если в программе нет точек останова, то программа будет выполняться постоянно.

• Вставка/Удаление точки останова

Точку останова можно установить в любой допустимой строке программы. Способ установки точки останова: наведите курсор на строку, где нужно установить точку останова, затем нажмите левой кнопкой мыши на кнопку ***Insert/Remove Breakpoint*** на панели инструментов ST. Строка загорится зеленым, что означает, что точка останова успешно установлена. Как показано на рисунке 10.10. Если снова нажать кнопку ***Insert/Remove Breakpoint***, то точка останова будет удалена. Если нажать кнопку в третий раз, то точка останова будет установлена снова. Программа может одновременно установить до 10 точек останова в разных местах. Как показано на рисунке 7.10:

```
R0010 :=0;
FOR i := 1 TO II_PCNT BY 1 DO
 IF II_BUF[i]=1     THEN
R0010 := R0010+1;
```

Рисунок 7.3 Установка точки останова в ST

• Очистить все точки прерывания

Все точки останова в программе могут быть очищены одновременно. Чтобы запустить очистку, нажмите левой кнопкой мыши на кнопку **Clear All Breakpoints** на панели инструментов ST после чего все точки останова будут удалены.

Указание

После отключения все точки останова удаляются автоматически.

11.5 Симулятор

ПО «СКПро» предоставляет два режима отладки для тестирования готовых программ: в режиме подключенного онлайн ПЛК и в симуляторе в режиме реального времени.

При использовании симулятора система работает в виртуальном режиме, то есть «СКПро» не подключается к ПЛК и выполняет виртуальную отладку готовой программы ПЛК в режиме реального времени или выполняет симуляционный тест без реального ПЛК, а также моделирует производственный процесс на реальном испытательном участке и тестирует готовые программы, чтобы предоставить возможность просмотра результата выполнения текущих программ и их корректности.

При применении режима подключенного онлайн ПЛК «СКПро» подключается к самому ПЛК. Используя коммуникации и встроенные возможности мониторинга в режиме реального времени, на основе фактических или аналоговых сигналов, а также процессах управления, система может осуществлять мониторинг и отладку отредактированной программы в режиме реального времени, что позволяет просматривать результаты выполнения текущих программ и их корректность.

Различия между режимом подключенного онлайн ПЛК и симулятором заключаются в следующем:

В симуляторе в режиме реального времени «СКПро» не связывается с ПЛК, принудительный ввод и модификация всех сигналов происходят только на компьютере, где находится «СКПро». Результаты операции или данные выводятся только на компьютере, однако он не осуществляет реального управления процессом.

В режиме подключенного онлайн ПЛК, «СКПро» связывается с ПЛК, форсирование и модификация всех сигналов происходят в реальном ПЛК. Результаты работы или данные выводятся с ПЛК и могут управлять реальными процессами.

Симулятор можно запустить с помощью меню **【Online】/【Connect】/ 【Simulator】**, а режим подключенного онлайн ПЛК можно запустить с помощью меню **【Online】/【Connect】/【PLC】**. Выйти из обоих режимов можно с помощью меню **【Online】/【Disconnect】**.

Оба режима позволяют просматривать результаты выполнения текущих программ и их корректность. В целом, симулятор подходит для этапа предварительного проектирования, в то время как режим подключенного онлайн ПЛК подходит для этапа отладки на заводе или в полевых условиях после продажи.

Основные функции симулятора приведены ниже:

Он может принудительно вводить все сигналы ввода и вывода и изменять их значение.

Он может полностью симулировать фактические результаты выполнения логических связей, инструкций и программ.

12 Приложение

12.1 Сервис и поддержка

12.1.1 Контакты службы технической поддержки

Запрос по телефону

Обращение по телефону службы технической поддержки: +7 (347) 223-99-22

Запрос по электронной почте

Обращение на адрес электронной почты службы технической поддержки: support@sybcom.ru

Запрос через интернет

Обращение через заполнение формы на сайте технической поддержки. www.sybcom.ru/support

12.1.2 Порядок оказания технической поддержки по изделию

Пользователь обращается в службу технической поддержки путем формирования обращения с указанием идентифицирующей пользователя информации и описанием возникшей проблемы.

При обращении пользователя в службу технической поддержки фиксируются его контактные данные (Фамилия, Имя, Отчество (при наличии), место работы и должность, адрес электронной почты в корпоративном домене, телефон), описываются причины обращения. В случае отказа Пользователя сообщить идентифицирующую его информацию, Сотрудник Службы технический поддержки имеет право не оказывать такому Пользователю услуг по технической поддержке.

Обращения в службу технической поддержки регистрируются в виде заявки с присвоением уникального номера. Подтверждением регистрации обращения для его инициатора служит номер заявки, передаваемый техническими средствами, входящими в инструментарий службы технической поддержки. Указание пользователем номера исходной заявки при повторных обращениях позволяет сотрудникам службы технической поддержки оперативно коммуницировать с пользователем.

Пользователь принимает на себя обязанность своевременного и квалифицированного взаимодействия со службой технической поддержки в соответствии с настоящим регламентом. При необходимости он самостоятельно информирует других пользователей, действующих в интересах того же юридического лица, о статусе обращения или делегирует им работу с обращением, уведомив об этом службу технической поддержки.

12.2 Лист изменений

Версия	Дата	Изменение
V1.0	01/2023	Первое издание для версии «СКПро» 6.2.11
V1.0.1		Добавлена информация о настройке цикла опроса для модулей AI-4000-0801, AI-4000-0802, AI-4000-1601, AI-4000-0804
V1.0.2		Уточнено количество запросов состояния основной связи MODBUS/TCP
V1.1	03/2023	Исправлена информация о приоритете задач. Добавлена информация по инструкциям TCON, TDISCON, TSEND, TRECV, TUSEND, TURECV
V1.1.1		Добавлены методические указания по использованию меток в языках LD и FBD. Добавлены методические указания по применению математических функций.
V1.1.2		Добавлена информация о настройке MODRW