Содержание

Глава I Использование Kinco Builder

1.1 Обзор

IEC61131-3 является глобальным стандартом для промышленного программирования. Его технические перспективы высоки, оставляя достаточно места для роста и дифференциации. Это упрощает, когда люди проектируют и управляют промышленными процессами по стандартному интерфейсу программирования. IEC 61131-3 имеет большое влияние в промышленной сфере управления, и принимается в качестве ориентира большинством производителей ПЛК. С его далеко идущей поддержкой он не зависит от какой-либо одной компании.

Kinco Builder это программное обеспечение для программирования серии Kinco-K5 ПЛК Micro, и это удобная и высоко-эффективная система развития с мощными функциями.

Kinco Builder разрабатывается самостоятельно и согласовывает с стандартом IEC 61131-3. Его стало легче изучать и использовать, потому что многие пользователи приобрели большинство навыков программирования через различные источники.

Kinco Builder осуществляется с учетом следующих особенностей:

★ В соответствии со стандартом IEC 61131-3
★ Поддержка двух стандартных языков программирования, то есть IL (Instruction List) и LD (Ladder Diagram)
★ Мощный набор команд, встроенных стандартных функций, функциональных блоков и других специальных инструкций
★ Поддержка структурного программирования
★ Поддержка прерываний
★ Поддержка подпрограммы
★ Поддержка прямых представленных переменных и символических переменных, легко развивать и управлять проектом пользователя.
★ Удобная и высокоэффективная среда
★ Гибкая конфигурация оборудования, вы можете определить все виды параметров оборудования

1.2 Общие обозначения в руководстве

▪ Micro PLC (программируемый логический контроллер)
В соответствии с общими правилами классификации, микро ПЛК относится к типу ПЛК с контрольными точками ниже 128. Этот тип ПЛК принимает компактную структуру, то есть определенное количество каналов ввода / вывода, выход питания, высоко-скоростной выход / вход и другие принадлежности интегрированы в модуле CPU.
▪ Корпус CPU.
А именно, модуль CPU, это основа системы управления. Программа пользователя сохраняется во внутренней памяти модуля CPU после загрузки с помощью программного обеспечения, и будет выполняться центральным процессором. Между тем, также CPU выполняет диагностику самотестирования: проверку правильной работы CPU, для областей памяти, и для статуса любых модулей расширения.
▪ Модуль расширения и расширения шины.
Модули расширения используется для расширения функций корпуса CPU и делятся на расширения ввода/вывода (для увеличения каналов ввода / вывода системы) и расширения функционального модуля (функции CPU).
Шина расширения соединяет CPU и модули расширения, а плоский 16-жильный кабель принимается как физический носитель. Шина данных, шина адреса и блок питания модуля расширения интегрированы в шине расширения.
▪ Kinco Builder
Программное обеспечение для программирования ПЛК серии Kinco-K5, согласуется с МЭК 61131-3

стандартной Kinco Builder, в настоящее время предлагает LD и IL языки, для удобства и эффективности в разработке программ управления для приложений. Kinco Builder предлагает удобную среду для разработки и отладки программ, необходимых для контроля приложений.

▪ Прошивка CPU

Это «операционная система» модуля CPU, и хранится во флешь-памяти. При включении питания, он начинает работу по управлению и планирует все задачи модуля CPU.

▪ Программа пользователя

Она также называется проект пользователь или прикладная программа, программа написана пользователем, чтобы выполнить некоторые специфические функции управления. После того как программа пользователя загружается в модуль CPU, она хранится в FRAM. При включении питания, модуль CPU должен прочитать его от FRAM в оперативной памяти, чтобы выполнить его.

▪ Основная программа и цикл сканирования

Модуль CPU выполняет ряд задач непрерывно и циклично, и мы называем это циклическим выполнением задач, как сканирование.

Основная программа является вступлением выполнения программы пользователя. В CPU, основная программа выполняется один раз за цикл сканирования. Только одна главная программа разрешается в программе пользователя.

▪ Свободный протокол связи

Корпус CPU предусматривает последовательные порты связи, которые поддерживают специальный протокол программирования Modbus RTU (в качестве ведомого устройства) и свободные протоколы. Режим свободного протокола связи позволяет вашей программе полностью контролировать коммуникационные порты CPU. Вы можете использовать режим свободного протокола связи для реализации пользовательских протоколов связи и для общения со всеми видами интеллектуальных устройств. ASCII и бинарные протоколы так же поддерживаются.

▪ Область отображения I / O

Включает области отображения ввода и области отображения выхода. В начале цикла сканирования, состояние сигнала передаётся из входных каналов в область отображения входов; в конце цикла сканирования, значения, сохраненные в области отображения выходов передаются на выходные каналы; в целях обеспечения согласованности данных и ускорения выполнения программы, модуль CPU обращается только к области отображения во время каждого цикла сканирования.

▪ Сохраняемые Диапазоны

С помощью настройки "Hardware" в Kinco Builder, вы можете определить четыре сохраняющихся диапазона для выбора областей RAM, которую вы хотите сохранить при отключении питания. В том случае, если CPU теряет питание, мгновенные данные в оперативной памяти будут поддерживаться супер конденсатором, и на сохраняемых диапазонах будут оставлены без изменений при следующем включении. Продолжительность удержания составляет 72 часа при нормальной температуре.

▪ Резервное копирование данных

Резервное копирование данных - это запись некоторых данных в E2PROM или FRAM через соответствующие инструкции для постоянного хранения. Примечание: Каждый тип постоянной памяти имеет свой срок, например, E2PROM позволяет 100 тысяч раз записи операций и FRAM позволяет 10 миллиардов раз.

Глава II Как использовать Kinco Builder (краткое руководство)

В этой главе вы узнаете, как установить Kinco Builder на компьютере и как программировать, подключать и запускать Kinco-K5 PLC. Целью этой главы является - дать вам краткое руководство, а дополнительная информация будет представлена в следующей главе.

2.1 Требования к компьютеру

2.1.1 Минимальные аппаратные требования для запуска Kinco Builder

★ Процессор: 1 ГГц или выше
★ Место на жестком диске: не менее 20M байт свободного места
★ RAM: 512M или более
★ Клавиатура, мышь, последовательный порт
★ 256 цветов VGA или выше, 1024 * 768 или выше

2.1.2 Минимальные требования к программному обеспечению для запуска Kinco Builder

★ Операционная система: Windows XP (32-разрядная), Windows Vista (32-разрядная), Windows7 (32 / 64bit), Windows8 (32 / 64bit), Windows 8.1 (32 / 64bit)
★ Пользователи могут найти ошибки при работе Kinco Builder в ОС Windows 7 или выше. Возможные решения заключаются в следующем:
★ [COM] порт в настройках связи является нулевым, Kinco Builder обнаруживает доступные COM-порты на компьютере, прочитав информацию о аппаратной части в REGEDIT.
В предыдущих версиях, Kinco Builder запрашивал команду для запуска; в противном случае он показывал нулевые списки портов.
В последней версии, Kinco Builder автоматически определит порт. Если Kinco Builder не может прочитать список портов, он будет перечислять порты от COM1 до COM9 для пользователя, чтобы выбрать вручную.
★ Kinco Builder не можете запуститься на компьютерах пользователей, но можно обратиться к режиму совместимости для запуска Kinco Builder, который устанавливается следующим образом:
★ Щелкните правой кнопкой мыши по значку «Kinco Builder V1.5.xx" и нажмите кнопку [Свойства];
★ Нажмите [Совместимость] в [Свойствах] диалогов, как показано на рисунке 2-1.

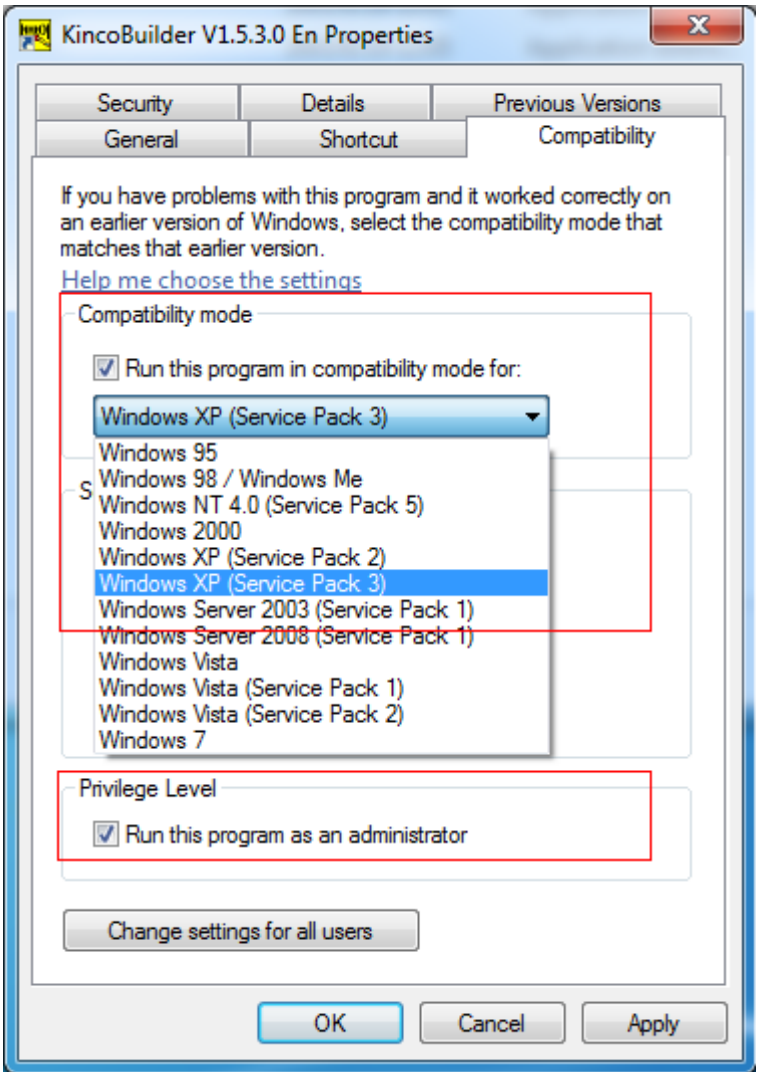


Рисунок 2-1 Установка"Режима совместимости"

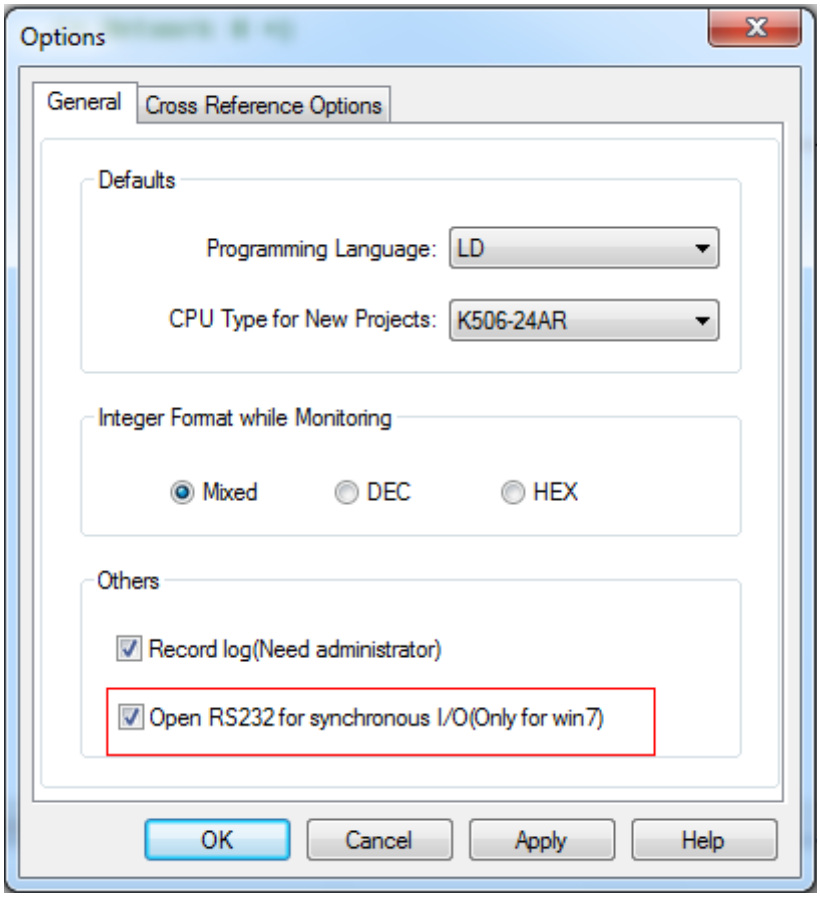


Рисунок 2-2 Открытие с синхронизацией

★ Сбой при использовании USB / RS232 преобразователя для связи с PLC.
Отказ вызван программой драйвера от преобразователя, не совместимом с компьютером. Большинство случаев вызваны с 64-битной Windows 7.
Открыть Kinco Builder и найти [Инструмент] → [Настройка Программного обеспечения] → [Открыть Параллельный порт], нажмите кнопку "Открыть порт с Синхронизацией" и нажмите "OK". Смотрите рисунок 2-2.
Впоследствии Kinco Builder откроет порты с синхронизацией и в большинстве случаев будет работать успешно.

2.2 Пользовательский интерфейс Kinco Builder

Пользовательский интерфейс использует стандартные функции интерфейса Windows, вместе с несколькими дополнительными функциями, чтобы сделать вашу среду разработки простой в использовании.

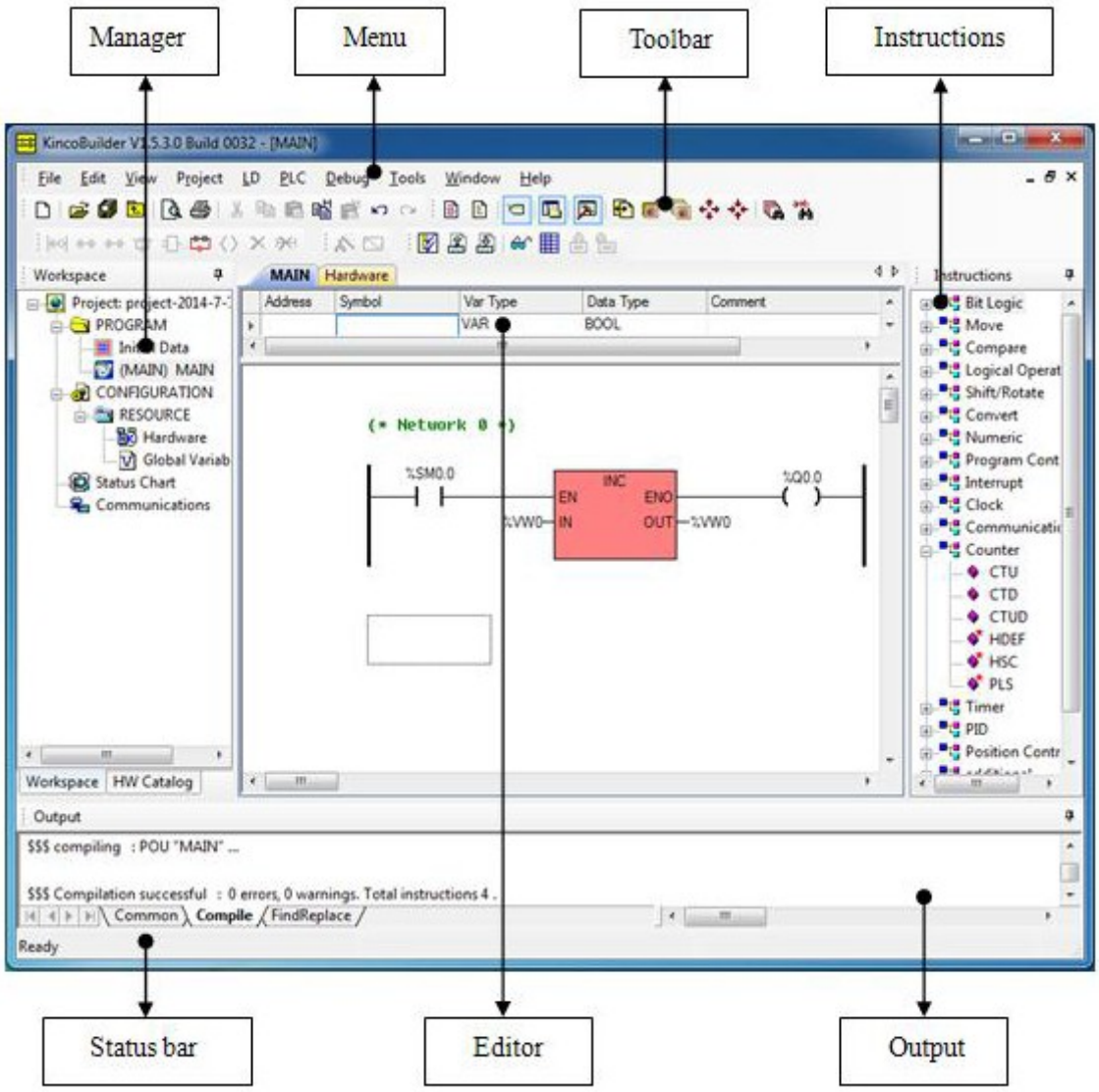


Рисунок 2-1 Интерфейс пользователя Kinco Builder

▪ Menu: содержит все команды управления в Kinco Builder.
▪ Toolbar: обеспечивает легкий доступ мыши к наиболее часто используемым командам в работе.
▪ Statusbar: предоставляет информацию о состоянии и запрашивает операции.
▪ Manager: Диспетчер окон предлагает в виде дерева все объекты проекта, в том числе PROGRAM, Hardware, Global Variable, и т.д., это может помочь вам в понимании структуры проекта. Менеджер проектов является удобным инструментом для организации управления программами и проектами. Контекстное меню появится, когда вы щелкните правой кнопкой мыши на любом узле дерева.
▪ Editor: включает в себя таблицы переменных и редактор программы (IL или LD). Вы можете программировать в редакторе программ и объявить локальные переменные и параметры ввода / вывода POU в таблице переменных.
▪ Instructions: набор команд LD и набор инструкций IL. Здесь представляется вид дерева всех имеющихся инструкций.
▪ Output: окно вывода отображает несколько типов информации. Выберите вкладку в основании окна, что бы просмотреть соответствующую информацию: "Компиляция" - окно отображает информацию о актуальной компиляции и окно "Общие" - отображает некоторую информацию о последний операции.

2.3 Используя Kinco Builder создавайте программы для ваших приложений

2.3.1 Компоненты проекта

В этом руководстве программа пользователя и проект пользователь имеет тот же смысл.
При программировании для конкретного приложения, необходимо настроить контроллеры, используемые в вашей системе управления, определить символические переменные и написать все виды POU и т.д. В KincoBuilder, все эти данные (в том числе POU, аппаратной конфигурации, глобальных переменных и т.д.) организуются в структуру проекта пользователя. Вы можете управлять информацией о проекте постояно и легко.
Компоненты проекта описаны в следующей таблице. Пункты, отмеченные как "дополнительный" не существенные компоненты в проекте, так что вы можете игнорировать их.

Таблица 2-1 Компоненты проекта

Программа	Первоначальные данные (Необязательно)	Вы можете назначить начальные числовые значения BYTE, WORD, DWORD, INT, DINT и реальные переменные в V области. Модуль CPU обрабатывает исходные данные, при включении питания, а затем начинает цикл сканирования.
	Основная программа	Это выполнение программы пользователя. Модуль CPU выполняет его один раз за цикл сканирования. Только 1 основная программа существует в проекте.
	Порядок прерываний (Необязательно)	Это прерывания, используются для обработки определенных событий прерываний. Они не вызываются основной программой. Вы осуществляете процедуру прерывания по заранее заложенным событиям прерывания, и модуль CPU выполняет эту процедуру только на каждом возникшем событии прерывания. Не более 16 прерываний разрешено в проекте.
	Подпрограммы (Необязательно)	Подпрограммы могут быть выполнены только тогда, когда они вызываются основной программой или процедурой прерывания. Подпрограммы могут быть полезными для лучшего структурирования программы пользователя. Они могут быть использованы повторно, и вы можете написать логику управления один раз в подпрограмме и вызвать его столько раз, сколько необходимо. Параметры ввода / вывода могут быть использованы в подпрограммах. Не более 16 подпрограмм допускается в проекте.
Конфигурация	Hardware	Здесь вы можете настроить модули Kinco-K5, используемые в вашей системе контроля, в том числе их адреса, параметры функции и т.д. Модуль CPU обрабатывает конфигурацию оборудования один раз при включении питания, а затем выполняет другие задачи.
	Глобальные переменные (Необязательно)	Здесь вы можете объявить глобальные переменные, необходимые в проекте.

2.3.2 Где хранить файлы проекта

При создании проекта, Kinco Builder в первую очередь попросит указать полный путь к файлу проекта, а затем пустой файл проекта (с расширением ".kpr") должен быть создан и сохранен на этом пути. Кроме того, папка с таким же именем, как проект будет создана также в этом пути; эта папка используется для хранения всех файлов программы, переменные файлы и другие временные файлы проекта.
Например, если вы создаете проект под названием "example" в "C: \ Temp", то папка и путь к файлу проекта "C: \ Temp \ example.kpr", и другие файлы хранятся в папке "C: \ Temp \ example ".

2.3.3 Импорт и Экспорт проекта

Kinco Builder предусматривает [File]>[Import…] и [File]>[Export…] в меню команд для вас сделает резервную копию и управляет проектом.
★ [Export…]
Сжимает все файлы, связанные с текущим проектом в одном файле резервной копии (с расширением ".zip").
★ Выберите [File]>[Export…] в меню команд.
Появится диалоговое окно "Экспорт проекта ...", как показано на рисунке 2-2.

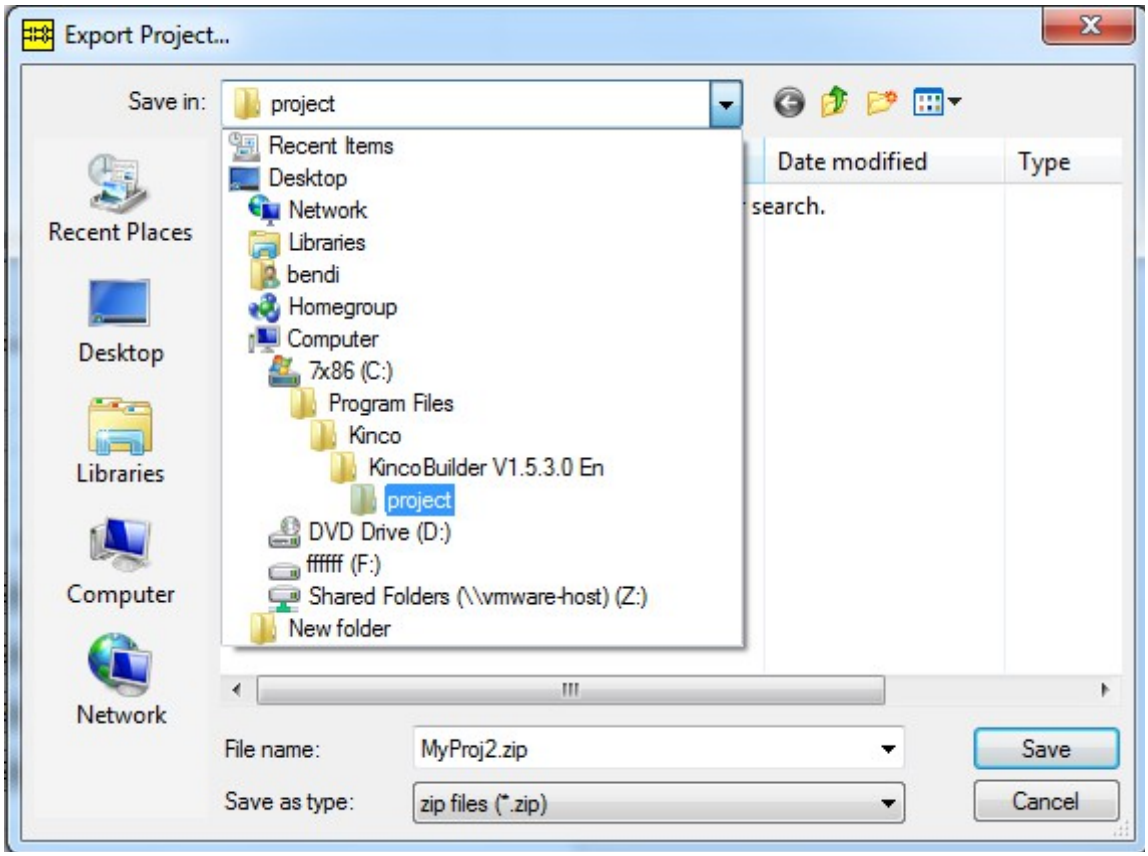


Рисунок 2-2 Экспорт проекта

★ Выберите путь и введите имя файла, а затем нажмите [Save].
Файл резервной копии для текущего проекта должен быть создан.
★ [Import ...]
Импортировать проект из существующего файла резервной копии (с расширением .zip) и открыть его.
★ Выберите [File]> [Import …] в меню команд.
Появится диалоговое окно "Импорт проекта ...", как показано на рисунке 2-3.

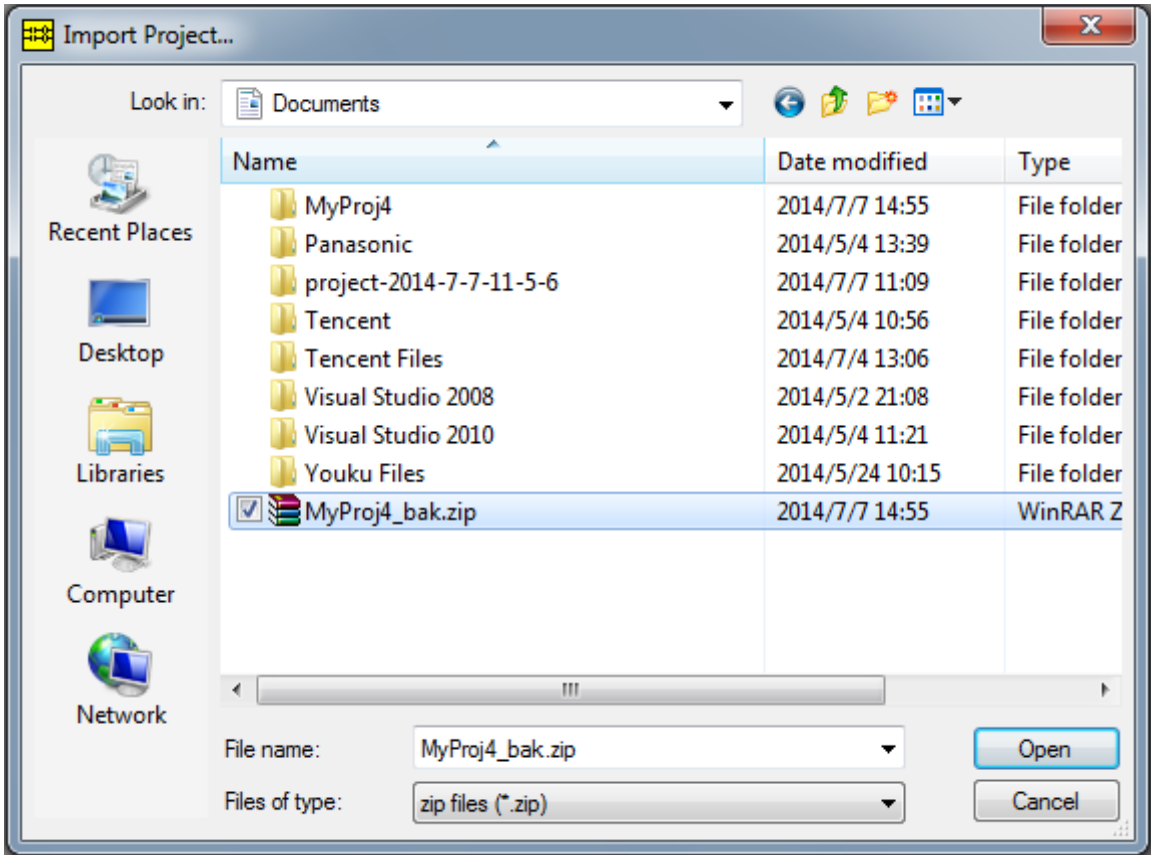


Рисунок 2-3 Импорт проекта: Выбор файла резервной копии

★ Выберите файл резервной копии, а затем нажмите [Open]. Появится следующее диалоговое окно для выбора директории для сохранения восстановленных файлов проекта.

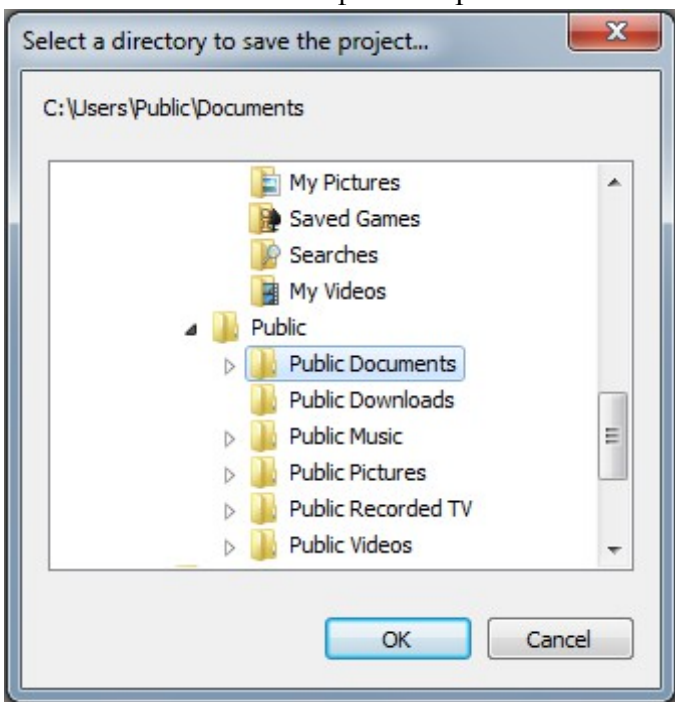


Рисунок 2-4 Импорт проекта: Выбор папки

★ Выберите директорию, затем нажмите [OK], и файлы проекта должны быть восстановлены в выбранной директории, и тот восстановленный проект будет открыт.

2.4 Как CPU Выполняет свои задачи в цикле сканирования?

Модуль CPU выполняет ряд задач непрерывно и циклично, и мы называем это циклическим выполнением задач или сканирование. Только основная программа и прерывания выполняются непосредственно в модуле процессора. Основная программа выполняется один раз за цикл сканирования; программа обработки прерывания выполняется только один раз в каждом случае событии прерывания, связанного с ним.
Модуль процессора выполняет следующие задачи в цикле сканирования, как показано на рисунке 2-5:

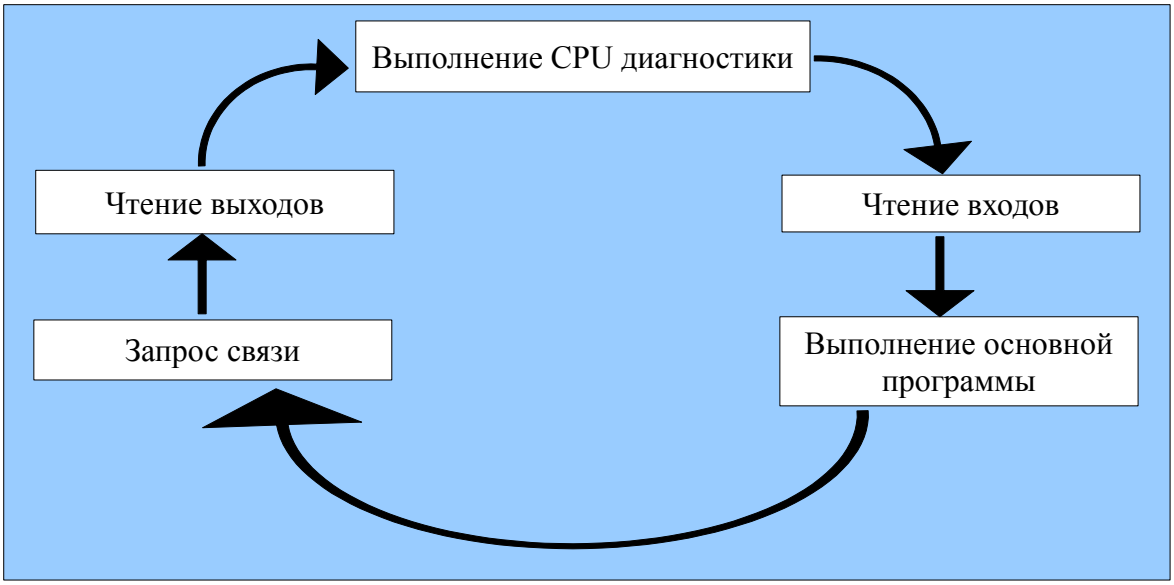


Рисунок 2-5 Цикл сканирования

★ Выполнение диагностики CPU: Модуль CPU выполняет само тестирование для проверки правильной работы CPU для областей памяти и для состояния модулей расширения.
★ Чтение входов: Kinco-K5 считывает все физические входные каналы и записывает эти значения в областях входного изображения.
★ Выполнение программы пользователя: Модуль CPU выполняет команды в основной программе непрерывно и обновляет области памяти.
★ Обработка запросов связи.
★ Запись выходов: Kinco-K5 записывает значения, хранящиеся в области вывода, на физические выходные каналы.
События прерывания могут произойти в любой момент во время цикла сканирования. Если вы используете прерывания, модуль CPU временно прерывает цикл сканирования, когда происходят события прерываний и немедленно выполнит соответствующие процедуры прерываний. После завершения выполнения подпрограммы управление возвращается в цикл сканирования в точку останова.

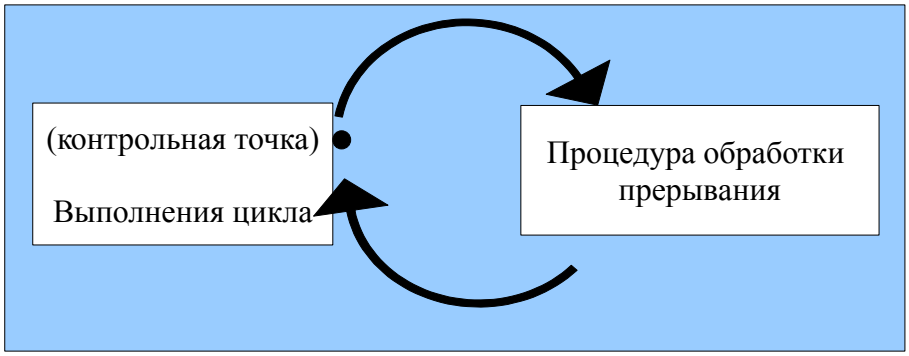


Рисунок 2-6 Выполнение прерываний

2.5 Как подключить компьютер с Kinco-K5

Модуль CPU обеспечивает интегрированный RS232 или RS485 последовательный порт для связи с другим оборудованием. Здесь мы говорим о том, как подключить модуль CPU (с портом RS232) с компьютером для начала программирования Kinco-K5 PLC используя Kinco Builder.

★ Запустите Kinco Builder, откройте существующий проект или создайте новый проект;

Подключите последовательный порт компьютера с портом модуля CPU с помощью подходящего кабеля программирования.

Примечание: соединения RS232 не имеют возможности горячей замены, поэтому вы должны выключить питание, по крайней мере с одной стороны (CPU модуля или компьютера), прежде чем подключать / отключать кабель. В противном случае, порт может быть поврежден.

★ Настройте параметры компьютера последовательного порта связи.

Примечание: Связь не может быть установлена, если параметры последовательного порта связи компьютера не идентичны с портом связи процессора.

★ Выберите [Tools]> [Communications …] в меню команд, или дважды щелкните [Communications] в окне диспетчера, или щелкните правой кнопкой мыши [Communications] и выберите команду [Open] на всплывающем меню, затем появится диалоговое окно "Связь".

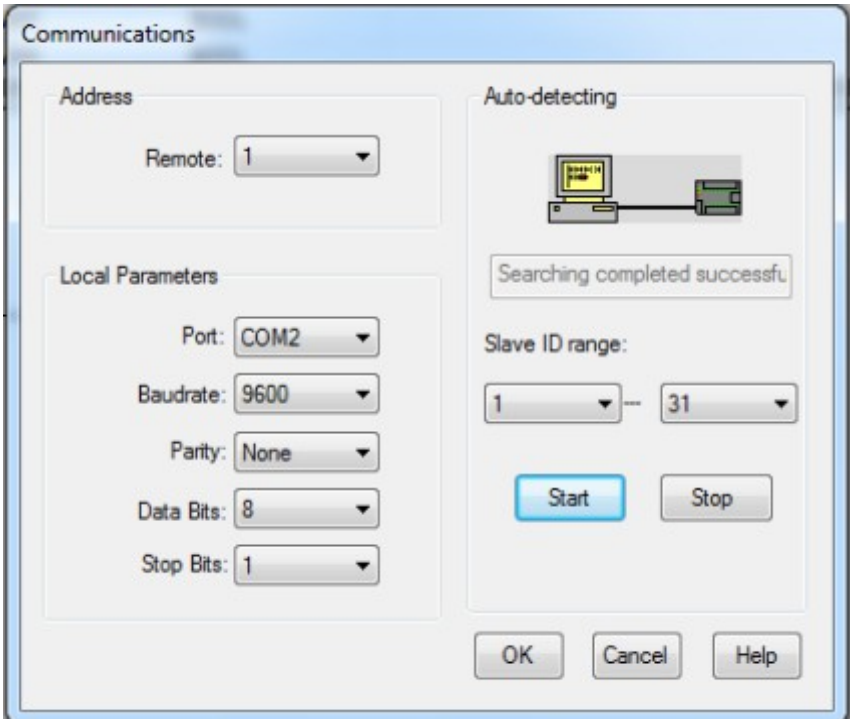


Рисунок 2-7 Диалоговое окно"Communications"

★ Выберите номер станции PLC в окне списка [Remote]; Выберите COM порт, используемый на компьютере в окне списка [Port]; Настройте параметры выбранного COM порта (в том числе [Baudrate], [Parity], [Data Bits] и [Stop Bits]) соответствующие порту CPU, а затем нажмите кнопку [OK], чтобы сохранить и применить их.

Если вы не знаете параметры связи порта CPU, как их приобрести?

Есть два способа:

★ Выберите [Port] используемый на компьютере, а затем нажмите кнопку [Search], чтобы Kinco Builder сделал поиск по параметрам онлайн модуля CPU автоматически. Это может занять от нескольких секунд до нескольких минут. Если поиск завершился успешно, Kinco Builder автоматически настроит соответствующие параметры для компьютера.

★ Выключите электропитание модуля CPU; Установите переключатель управления в положение_STOP ;

Затем включите питание, теперь CPU будет использовать параметры последовательного порта связи по умолчанию: номер станции, 1; скорость передачи данных, 19200; проверка четности, None; 8 бит данных; 1 стоп-бит. Вы можете настроить серийный COM порт компьютера в соответствии с этими параметрами.
Примечание: Не меняйте положение переключателя, пока вы не изменили параметры связи CPU.
★ После завершения настройки параметров связи COM порта компьютера, можно приступать к программированию Kinco-K5 PLC.

2.6 Как изменить параметры связи CPU

После подключения модуля CPU с компьютером, вы можете изменить его параметры связи по желанию, используя Kinco Builder.
★ Сначала откройте окно "Оборудование" с помощью одного из следующих способов:
■ Дважды щелкните [Hardware] в окне диспетчера;
■ Щелкните правой кнопкой мыши [Hardware], а затем выберите команду [open ...] на всплывающем меню.
В верхней части окна "Hardware" показан подробный список ПЛК модулей в виде таблицы, и мы называем это Таблица конфигурации .Таблица конфигурации представляет реальную конфигурацию. Вы устанавливаете свои модули в таблице конфигурации такие же, как в реальной системе управления.
В нижней части окна "Hardware" показаны все параметры выбранного модуля в таблице конфигурации, и мы называем это Окно параметров.
★ Выберите модуль CPU в таблице конфигурации, а затем выберите вкладку [Communication Ports] в окне параметров. Теперь вы можете изменить параметры связи, как показано на следующем рисунке.

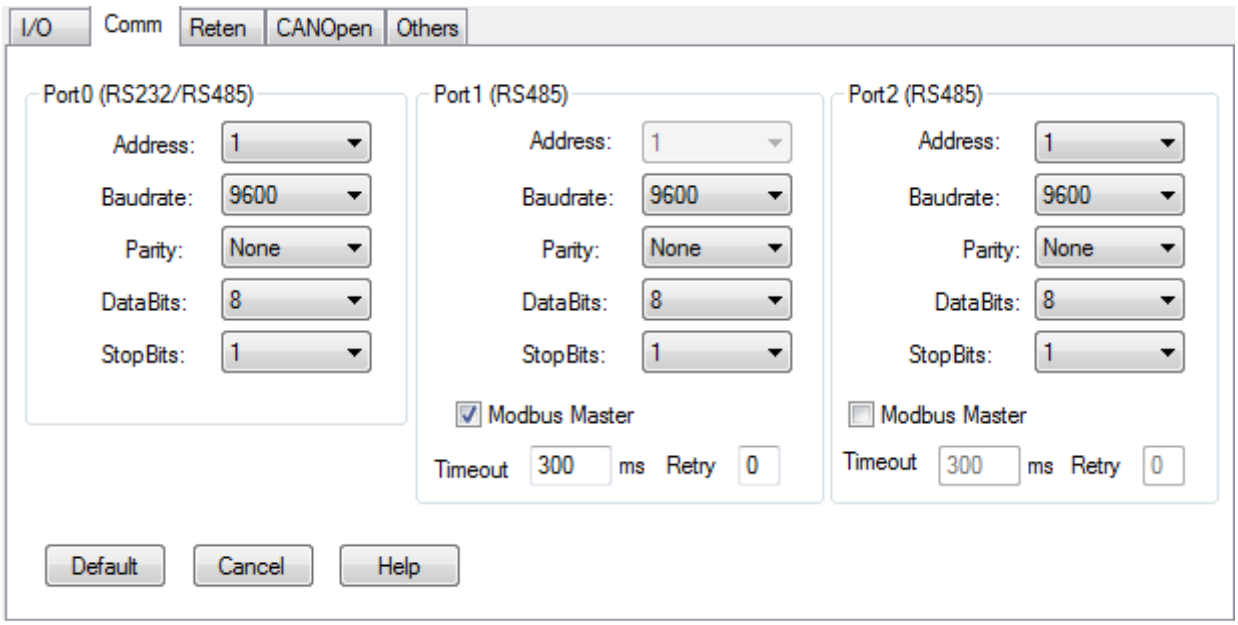


Рисунок 2-8 Параметры связи

★ После изменения параметров, необходимо загрузить их в модуль CPU.
Примечание: Параметры конфигурации не вступят в силу, если они не загружены.

2.7 Пример: Общие шаги для создания проекта

Для того, чтобы помочь начинающим быстро понять Kinco-K5, в дальнейшем мы будем использовать простой пример, чтобы представить некоторые общие шаги для создания и отладки проекта шаг за шагом. Пожалуйста, обратитесь к соответствующим разделам, чтобы узнать определенную функцию подробно в следующих главах.
Предположим, что мы должны создать следующий проект:

★ Проект: назван "Пример";
★ Оборудование: модуль CPU Kinco-K506-24AT;
★ Логика управления: переключение Q0.0 --- Q0.7, по очереди, и циклически. Для лучшей структуры, мы используем два программных модуля: подпрограмма под названием "Demo", чтобы реализовать логику управления; основная программа названа "Main", в которой будет вызываться "Demo".
★ Во-первых, запустите Kinco Builder.
★ При необходимости измените значения, используемые по умолчанию в KincoBuilder следующим образом:
★ Выберите [Tools]> [Options …] в меню команд.
Появится диалоговое окно "Options", в котором можно настроить некоторые значения по умолчанию, например, по умолчанию "Язык программирования" и т.д. Эти значения по умолчанию будут сохранены автоматически; их нужно просто настроить, до следующего изменения.
По умолчанию язык программирования [LD Ladder Diagram].
★ Создание нового проекта с помощью одного из следующих способов:
★ Выберите [File]> [New project …] в меню команд.
★ Щелкните по значку  на панели инструментов.
Появится диалоговое окно "... New project". Вам просто нужно ввести имя проекта и назначить его директорию, а затем нажать кнопку [Save], новый проект должен быть создан.
Для этого примера, давайте выберем" D: \ Temp "в качестве директории проекта, и назовём этот проект как "Example ".
★ Изменить конфигурацию оборудования. Вы можете настроить оборудование в любое время. Тем не менее, конфигурацию оборудования, необходимого для проекта, рекомендуется, завершить в первую очередь.
Когда был создан новый проект, Kinco Builder автоматически добавит CPU по умолчанию, назначенный в диалоговом окне «Options».
Пожалуйста, обратитесь к пункту 2.6 «Как изменить параметры связи CPU» для подробного описания.
Для этого примера используется модуль Kinco-K506-24AT с параметрами по умолчанию.
★ Инициализация данных
Вы можете инициализировать данные в любое время. Вы можете назначить начальные значения BYTE, WORD, DWORD, INT, DINT и реальные переменные в V области. Перед включением питания CPU и входа в основной цикл, исходные данные будут обработаны и начальные значения, назначенные пользователем, будут оцениваться соответствующими адресами.
Примечание: Любые области памяти, которые постоянно сохраняются как "initialize data" или "data maintain" будут восстановлены или оценены после входа CPU в основной цикл. Они будут следовать последовательно: восстановить память, как за это определено в "data maintain", инициализировать значение областей, как за это определено в "initialize data", восстановить постоянно сохраненные данные в соответствии с настройками пользователя.
★ Примеры создания программ.
KincoBuilder обеспечивает IL и LD языки программирования. Вы можете выбрать [Project]>[IL] или Project]>[LD] в меню команд чтобы изменить язык POU по желанию.
Для этого примера, основная программа под названием "Main" и подпрограмма под названием "Demo" должны быть написаны на языке LD.
■ Основная программа
При создании нового проекта, KincoBuilder автоматически создаст пустую основной программу под названием "MAIN".
■ Создание новой подпрограммы с помощью одного из следующих способов:
★ Выберите [New Subroutine]> [Project] в меню команд.
★ Щелкните по значку  на панели инструментов
★ Щелкните правой кнопкой мыши [PROGRAM] в окне диспетчера, а затем выберите команду [New Subroutine] на всплывающем меню.
Затем создастся новая подпрограмма, с именем по умолчанию "SBR_0". Теперь вы можете ввести следующие инструкции, как показано на рисунке 2-9.
После того как вы закончите ввод инструкций, вы можете переименовать эту подпрограмму, следующим

образом: Закройте окно подпрограммы; Щелкните правой кнопкой мыши "(SBR00) SBR_0" в окне диспетчера, а затем выберите команду [Rename] на всплывающем меню, чтобы изменить название на "Demo", или выберите [Properties ...] команды и сделайте изменение в диалоговом окне"Property ".

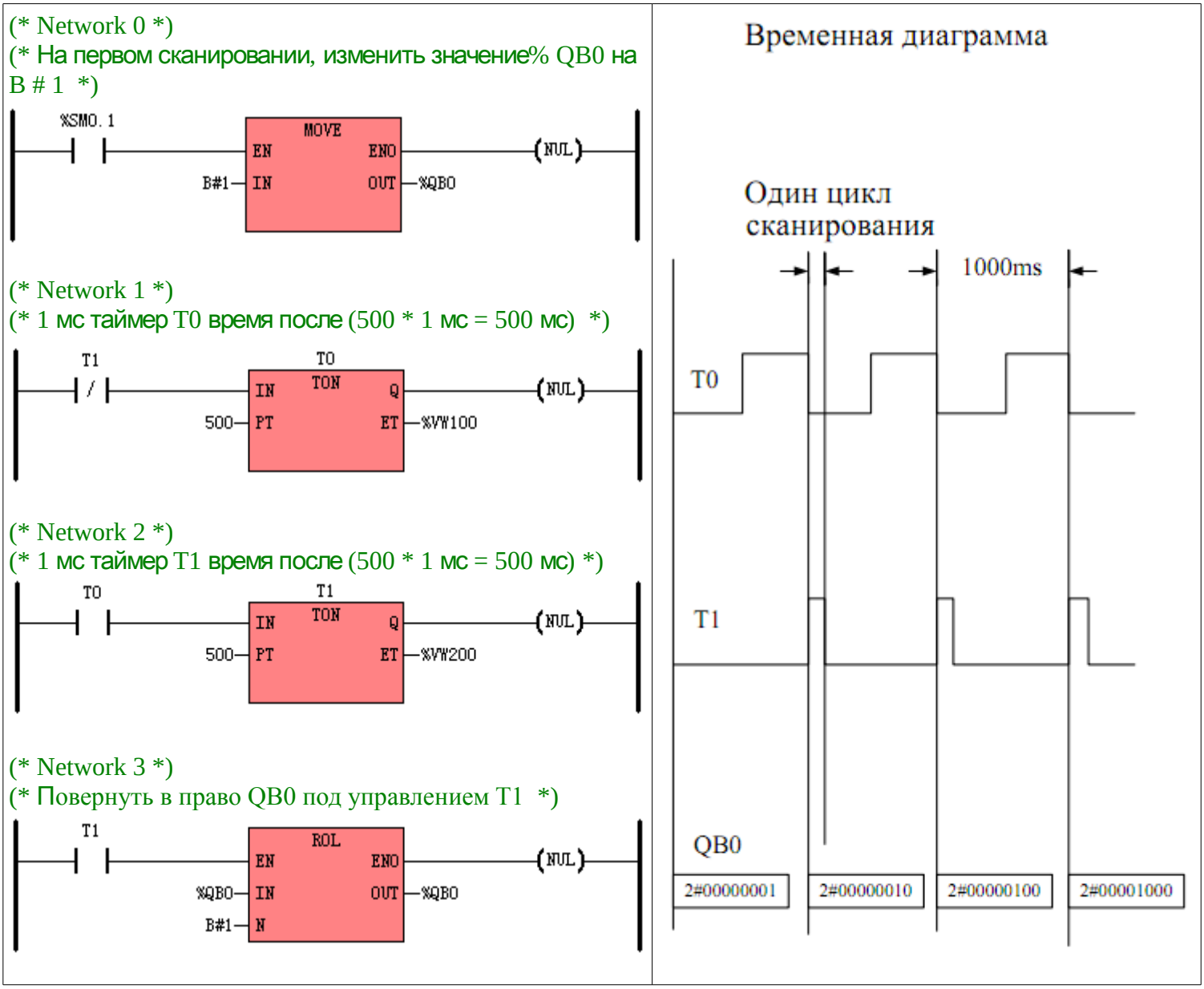


Рисунок 2-9 Подпрограмма "Demo"

■ Изменение основной программы.

Теперь мы закончили подпрограмму "Demo", и мы должны вернуться к основной программе, чтобы добавить следующие инструкции, как показано на следующем рисунке.

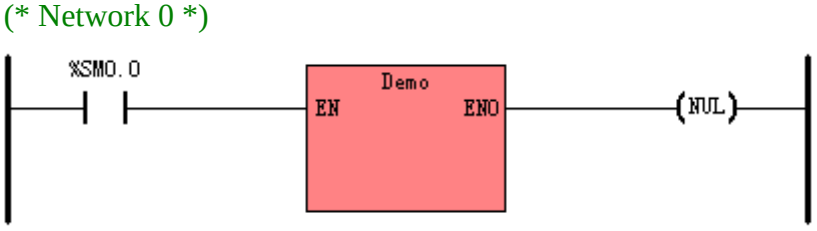


Рисунок 2-10 Основная программа

★ Компиляция проекта. После того как вы закончили весь проект, вы должны скомпилировать его. При компиляции проекта, Kinco Builder сохранит ее автоматически, но сначала убедитесь, что он является последним. Можно запустить компиляцию с помощью одного из следующих способов:
★ Выберите [PLC]> [Compile All] в меню команд
★ Щелкните по значку на панели инструментов
★ Используйте сочетание клавиш F7
Вкладка "Компиляция" в окне вывода содержит список сообщений последних компиляции. Чтобы найти исходный код, соответствующий ошибке, вы можете дважды щелкнуть на сообщении об ошибке в окне "компиляции". Вы должны сделать изменения в соответствии с сообщениями об ошибках, пока проект не будет успешно скомпилирован.
★ Теперь, пришло время, чтобы загрузить проект.
Примечание: в случае необходимости, можно изменить параметры связи последовательного порта с компьютером в диалоговом окне [Communications].
Примечание: Вы можете обратиться к пункту 2.5 «Как подключить компьютер с Kinco-K5», что бы найти параметры связи.
Вы можете скачать проект с помощью одного из следующих способов:
★ Выберите [PLC]> [Download ...] в меню команд
★ Щелкните по значку на панели инструментов
★ Используйте сочетание клавиш F8
★ Откроется диалоговое окно "Download User Project", как показано на следующем рисунке.

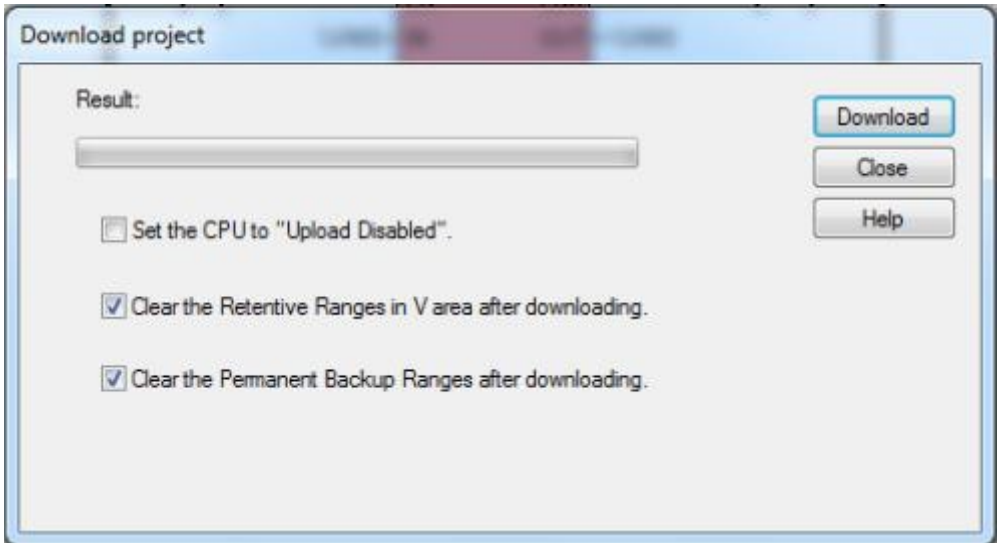


Рисунок 2-11 Диалоговое окно "Download project"

☑ [Set the CPU to “Upload disabled”]
Если этот пункт будет выбран, CPU запретит любую загрузку проекта после этой загрузки.
☑ [Clear the Retentive Ranges in V area after download]
Если этот пункт будет выбран, данные в V области и C области будут удалены;
Если не выбран, данные в V области и C области останутся без изменений.
После окончании настроек, вы можете нажать кнопку [Download] что бы загрузить проект в ПЛК и включить CPU в "RUN", чтобы проверить проект.

Если модуль CPU находится в режиме RUN, диалоговое окно предложит вам перевести его в режим STOP. Нажмите Да, чтобы перевести его в режим STOP.
После того как проект был загружен, модуль CPU перейдёт в режим RUN, а светодиоды состояния для Q0.0 --- Q0.7 должны включаться и выключаться по очереди, и циклически.
Теперь, вы завершили свой первый проект Kinco-K5.

★ Вы можете следить за программой в режиме онлайн, выбрав [Debug]> [Monitor] в меню команд или нажать значок  на панели инструментов, а затем Kinco Builder покажет значения всех переменных, используемых в программе.
Чтобы остановить модуль CPU, переведите его в режим STOP, поместив переключатель работы в позицию STOP или выбрав [Debug]> [Stop] в меню команд.
■ Debug
Вы можете использовать онлайн монитор и функцию Force для отладки
★ Онлайн монитор
Онлайн монитор содержит два режима:
■ Монитор в таблице состояния переменных. Вы можете ввести любую переменную для контроля;
■ Монитор в программе. Вы можете наблюдать, как работает программа. Программа не имеет права редактировать.
Онлайн монитор может быть эффективным только после открытия таблицы состояния переменной LD или IL. Пожалуйста, обратите внимание, что онлайн монитор является командой проверки. Каждый раз, если вы захотите закрыть онлайн статус, можно повторить команду онлайн монитора.
Вы можете использовать любую команду, чтобы войти в состояние онлайн-монитора следующим образом:
■ [Debug] → [Online monitoring];
■ Однократное нажатие значка  на панели инструментов;
■ Быстрая клавиша F6.

★ Force
Вы можете использовать функцию Force, чтобы изменить значение переменных в I области, Q области, M области, V области, AI области или AQ области, среди которых переменные в I области, Q области, M области и V области может быть отредактирован битом, байтом, словом или двойным байтом, в то время как переменные в AI области и AQ области могут быть отредактированы словом.
K5 позволяет пользователям использовать функцию Force, чтобы изменить максимум 32 переменных.
Непосредственная команда не позволяет выполнять функцию Force.
Вы можете выполнить функцию Force следующим образом:
★ Через таблицы состояния переменных. Вы можете обнаружить переменные и ввести значение для Force с помощью таблицы состояния переменных и приступить к командам меню (или вы можете щелкнуть правой кнопкой мыши и найти команду в меню [Force], [All Force] и т.д.)
★ Войдите в состоянии онлайн мониторинга и выполните функцию Force.
Вкл / Выкл переменной: Щелкните правой кнопкой мыши Contact и Coil, выполнить команду [Force 0], [Force 1] или [Force ...];
Номера вкл / выкл переменной: Щелкните правой кнопкой мыши переменную, выполните команду [Force ...].
Тем временем, переменная может иметь возможные значения: Значения, присвоенные пользователем за счёт внешнего входного сигнала (I, AI) или команды программы пользователя (Q, AQ, M, V). Поэтому следующие правила вступят в силу:
★ Что касается переменных в M области и V области, величина Force будут иметь одинаковый приоритет с командой: Force будет выполнена, но будет эффективна только один раз в одном цикле сканирования и команда будет эффективна потом;
★ Что касается переменных в I области и AI области, величина Force будет отменена от внешнего входного сигнала. Если присваивается значение Force, то он будет эффективен в первую очередь;
★ Что касается переменных в Q области и AQ области, величина внешнего входного сигнала переопределит обработку; но Force, на выходе, будет отменён с помощью цикла сканирования.
Вы можете использовать команду из меню [Cancel Force], чтобы отменить Force любой переменной, или использовать [All Cancel] для отмены Force всех переменных.
Когда CPU перезагружается, статус Force всех переменных будут отменены.

Глава III Основные понятия программирования

В этой главе будут подробно описаны основы программирования PLC Kinco-K5, используя Kinco Builder, а также некоторые основные понятия стандарта IEC61131-3, которые будут полезны для вас, чтобы использовать любой тип программного обеспечения IEC 61131-3. Целью этой главы является помощь вам на начальных этапах программирования и практики, чтобы достичь уровня "знаю, что и знаю, почему". Углублённое изучение каждого раздела рекомендуется не при первом прочтении, а на практике, так будет удобнее для облегчения понимания данного руководства.

3.1 POU (Программный Организационный Блок)

Блоки, из которых построены программы и проекты, называются «Программный Организационный Блок» (POU) в IEC61131-3. Как следует из названия, POU самые маленькие, независимые программные единицы, содержащие программный код. Следующие три типа POU определяются в IEC61131-3:

★ **Программа**

Ключевое слово: PROGRAMME

Этот тип POU представляет собой «основную программу», и может быть выполнена на контроллерах.

Programs формируют время выполнения программ, связываясь с TASK в конфигурации.

Programme может иметь входные и выходные параметры.

★ **Функция**

Ключевое слово: FUNCTION

Функции имеют и входные параметры и значения функции в качестве возвращаемого значения. Функция всегда дает тот же результат, как ее значение функции, если она вызвана с теми же входными параметрами.

★ **Функциональный блок**

Ключевое слово: FUNCTION_BLOCK

Функциональный блок - сокращённо FB в следующих разделах данного руководства.

В FB могут быть назначены параметры ввода / вывода и имеет статические переменные, а статические переменные могут запоминать предыдущее состояние. FB выдаст результаты, которые также зависят от состояния статических переменных, если он вызван с теми же входными параметрами.

Проект пользователь состоит из POU, которые, предоставлены производителем или созданы пользователем. POU могут вызывать друг друга с параметрами или без параметров, а это облегчает повторное использование программных блоков, а также модуляризацию проекта пользователя. Но рекурсивные вызовы запрещены, IEC 61131-3 четко прописывает, что POU не может вызывать само себя прямо или косвенно.

3.2 Типы данных

Типы данных определяют количество битов на один элемент данных, диапазон значений и ее исходное значение по умолчанию. Все переменные в программе пользователя должны быть идентифицированы с помощью типа данных.

Группа элементарных типов данных определяется в IEC61131-3, поэтому использование этих типов данных являются открытыми и едиными для программирования ПЛК.

Элементарные типы данных, которые поддерживает Kinco-K5 в настоящее время приведены в следующей таблице.

Таблица 3-1 Типы элементарных данных, поддерживаемые Kinco-K5

Ключевое слово	Описание	Размер в битах	Диапазон значений	Исходное значение по умолчанию
BOOL	Логический	1	истина, ложь	ЛОЖЬ
BYTE	Бит строка длиной 8	8	0 ~ 255	0
WORD	Бит строка длиной 16	16	0 ~ 65,535	0
DWORD	Бит строка длиной 32	32	0 ~ 4,294,967,295	0
INT	Целое число	16	$-2^{15} \sim (2^{15}-1)$	0
DINT	Двойное целое число	32	$-2^{31} \sim (2^{31}-1)$	0
REAL	Число с плавающей точкой, ANSI / IEEE 754--1985 стандартный формат	32	$1.18*10^{-38} \sim 3.40*10^{38}$, $-3.40*10^{38} \sim -1.18*10^{-38}$	0.0

Типы вещественных чисел в Kinco-K5 будут следовать ANSI / IEEE 754-1985, который описан как FLOAT в языке C.

★ Округление разности / погрешности данных REAL

Двойная система реального числа не является точным. Данные REAL будут занимать пространство 4 байта и представлять достоверные числа с цифрами максимум 7 знаков. Числа, которые больше, чем эта длина будут округляется.

Допустимые номера данных, начиная с первого числа, не = 0 до последнего числа.

★ Данные "0.0"

В связи с округлением разности / погрешности "0.0" не может быть точно показано на K5. Любое действительное число, находящееся в пределах [-0,000001, 0,000001] будет рассматриваться как "0,0".

★ Сравнение вещественных чисел

При использовании команд сравнения (GT, GE, EQ, NE, LT, LE), пожалуйста, следует отметить, что два действительных числа не может быть сравнены точно. Пока два реальных числа находятся в диапазоне от [-0,000001, 0,000001] K5 будет считать эти два числа в равенстве, и наоборот.

3.3 Идентификаторы

Идентификатор представляет собой строку из букв, цифр и знаков подчеркивания, которые должны начинаться с буквы или символа подчеркивания. (IEC61131-3)

3.3.1 Как определить идентификатор

Вы должны соблюдать следующие принципы при определении идентификатора:

★ Следует начать с буквы или символа подчеркивания, а далее цифры, буквы или символы подчеркивания.

★ Идентификаторы не учитывается регистром. Например, идентификаторы abc, ABC и aBC должны быть одинаковым.

★ Максимальная длина идентификатора ограничена только каждой системой программирования.

В Kinco Builder максимальная длина идентификатора составляет 16 символов.

★ Ключевые слова не могут быть использованы в качестве пользовательских идентификаторов. Ключевые слова являются стандартными идентификаторами и зарезервированы для языков программирования IEC61131-3.

3.3.2 Использование идентификаторов

Языковые элементы, которые можно использовать как идентификаторы в Kinco Builder заключаются в следующем:

★ Имя программы, имя функции и имя FB
★ Имя переменной
★ Метка и т.д.

3.4 Константа

Константа - это единица, которая представляет собой значение в программе. Используйте константы, что бы представлять числовые, символьные строки или значения времени, которые не могут быть изменены. Константы характеризуются значением и типом данных. Особенности и примеры констант, которые поддерживает Kinco-K5 в настоящее время, показаны в следующей таблице.

Таблица 3-2 Константы

Тип данных	Формат(1)	Диапазон значений	Пример
BOOL	истина, ложь	истина, ложь	ЛОЖЬ
BYTE	B # цифры	B#0 ~ B#255	B#129
	B # 2 # двоичные цифры		B#2#10010110
	B # 8 # восьмеричные цифры		B#8#173
	B # 16 # шестнадцатеричные цифры		B#16#3E
WORD	W # цифры	W#0 ~ W#65535	W#39675
	2 # двоичные цифры		2#100110011
	W# 2 # двоичные цифры		W#2#110011
	8 # восьмеричные цифры		8#7432
	W # 8 # восьмеричные цифры		8#174732
	16 # шестнадцатеричные цифры		16#6A7D
	W # 16 # шестнадцатеричные цифры		W#16#9BFE
DWORD	DW# цифры	DW#0 ~ DW#4294967295	DW#547321
	DW# 2 # двоичные цифры		DW#2#10111
	DW# 8 # восьмеричные цифры		DW#8#76543
	DW# 16 # шестнадцатеричные цифры		DW#16#FF7D
INT	Цифры	-32768 ~ 32767	12345
	I# цифры		I#-2345
	I# 2 # двоичные цифры(2)		I#2#1111110
	I# 8 # восьмеричные цифры(2)		I#8#16732
	I# 16 # шестнадцатеричные цифры(2)		I#16#7FFF
DINT	DI# цифры	DI#-2147483647 ~ DI#2147483647	DI#8976540
	DI# 2 # двоичные цифры(2)		DI#2#101111

	DI# 8 # восьмеричные цифры(2)		DI#8#126732
	DI# 16 # шестнадцатеричные цифры(2)		DI#16#2A7FF
REAL	Цифры с запятой	$1.18*10^{-38} \sim 3.40*10^{38}$, $-3.40*10^{38} \sim -1.18*10^{-38}$	1.0, -243.456
	xEy		-2.3E-23

Примечание:
★ Дескриптором не учитываются, например, константы W # 234 и w # 234, должны быть одинаковыми.
★ В двоичной, восьмеричной и шестнадцатеричной системах константы INT и DINT принимают стандартные два состояния, а MSB является битом знака: отрицательное число, если MSB = 1, и положительное число, если MSB = 0. Например, I # 16 #FFFF = -1, I # 7FFF = 32767, I # 8000 = -32768 и т.д.

3.5 Переменные

В отличие от постоянных, переменные обеспечивают средства идентификации объектов данных, содержание которых может измениться, например, данные, связанные с входами, выходами, или памятью ПЛК. В (IEC61131-3) переменные используются для инициализации и записи данных процессов объектов. Переменная должна быть объявлена фиксированным типом данных. Место хранения переменной, т.е. объект данных, связанный с переменной, может быть определен пользователем вручную, или быть выделен автоматически с помощью системы программирования.

3.5.1 Декларация

Переменная должна быть объявлена до ее использования. Переменные могут быть объявлены из POU и использоваться на глобальном уровне; Также, они могут быть объявлены как параметры интерфейса или локальных переменных в POU. Переменные делятся на разные типы переменных для целей декларирования. Стандартные типы переменных, поддерживаемые Kinco-K5 описаны в следующей таблице. В таблице, "Внутренний" указывается, может ли переменная быть считанной или записанной в рамках POU, в котором она объявлена, и "Внешний" указывает, может ли переменная быть видимой и может ли быть прочитана или записана в вызывающей POU.

Таблица 2-3 Типы переменных

Тип переменной	Внешний	Внутренний	Описание
VAR	---	Чтение / запись	Локальные переменные. Они могут быть доступны только в пределах собственной POU.
VAR_INPUT	запись	Чтение	Входные переменные вызывающего интерфейса, т.е. формальные параметры ввода. Они могут быть записаны в пределах вызывающего POU, но могут быть прочитаны только в пределах собственной POU.
VAR_OUTPUT	Чтение	Чтение / запись	Выходные переменные, которые действуют в качестве возвращаемого значения собственной POU. Они предназначены только для чтения в пределах вызывающей POU, но можно читать и записывать в пределах собственной POU.

VAR_IN_OUT	Чтение / запись	Чтение / запись	Переменные ввода / вывода вызывающего интерфейса, т.е. формальные параметры ввода / вывода. Они имеют комбинированные возможности VAR_INPUT и VAR_OUTPUT.
VAR_GLOBAL	Чтение / запись	Чтение / запись	Глобальные переменные. Могут быть считаны и записаны в в пределах всех POU.

3.5.2 Объявление переменных в Kinco Builder

Каждый тип переменных должен быть объявлен в пределах соответствующей таблицы, так будет удобнее для Вас, чтобы вводить данные. Кроме того, Kinco Builder может строго проверить входы.
Глобальные переменные объявляются в рамках таблицы Глобальных переменных, а другие переменные объявляются в таблице переменных соответствующего POU. Каждый POU имеет свою собственную отдельную таблицу переменных.
Если вы используете то же имя для переменной на локальном и на глобальном уровне, локальный имеет приоритет в рамках своей POU.

3.5.3 Проверка переменных

Kinco Builder проверяет использование каждой переменной, чтобы проверить, является ли она соответствующего типа данных и типа переменной. Например, когда значение REAL присваивается переменной WORD или переменной VAR_INPUT изменяется в POU, Kinco Builder предупредит вас и предложит для изменения.
Поскольку характеристика переменной зависит от ее типа переменной и типа данных, строгая проверка может помочь вам избежать этих ошибок, связанных с неправильным использованием переменных.

3.6 Как получить доступ к памяти PLC

Kinco-K5 хранит информацию в различных блоках памяти. Чтобы быть удобным для пользователей, Kinco-K5 обеспечивает два способа адресации для доступа единицы памяти:
★ Прямая адресация
★ Косвенная адресация, то есть указатель.

3.6.1 Типы памяти и характеристики

Память о Kinco-K5 ПЛК делится на несколько различных областей для различных целей использования, и каждая область памяти имеет свои особенности. Подробности приведены в следующей таблице.

Таблица 3-4 Типы памяти и характеристики

I	
Описание	Область DI (Digital Input). Kinco-K5 считывает все физические каналы DI в начале каждого цикла сканирования и записывает эти значения в области I
Режим доступа	Может быть доступна по битам, по байтам, слова и двойного слова
Право доступа	Только чтение
Другое	Может быть вынужденным, и не может быть сохранён

Q	
Описание	Область DO (цифровой выход). В конце каждого цикла сканирования Kinco-K5 записывает значения, хранящиеся в области Q с физическими каналами DO.
Режим доступа	Может быть доступна по битам, по байтам, слова и двойного слова
Право доступа	Чтение / запись
Другое	Может быть вынужденным, и не может быть сохранён
AI	
Описание	Область AI (аналоговый вход). Kinco-K5 замеряет все каналы AI в начале каждого цикла сканирования, и преобразует аналоговые входные значения (например, тока или напряжения) в 16-разрядные цифровые значения и записывает эти значения в AI области.
Режим доступа	Может быть доступен словом (тип данных INT)
Право доступа	Только чтение
Другое	Может быть вынужденным, и не может быть сохранён
AQ	
Описание	Область AO (аналоговый выход). В конце каждого цикла сканирования Kinco-K5 преобразует 16-битные цифровые значения, хранящиеся в области AQ в полевые значения сигналов и записывает в AO каналы.
Режим доступа	Может быть доступен словом (тип данных INT)
Право доступа	Чтение / запись
Другое	Может быть вынужденным, и не может быть сохранён
HC	
Описание	Область высокоскоростной счетчик. Используется для хранения текущего значения счета высокоскоростных счетчиков.
Режим доступа	Может быть доступны с помощью двойного слова (тип данных DINT)
Право доступа	Только чтение
Другое	Может быть вынужденным, и не может быть сохранён
V	
Описание	Область переменных. Это относительно большая область и может быть использована для хранения большого количества данных.
Режим доступа	Может быть доступна по битам, по байтам, словом и двойным словом
Право доступа	Чтение / запись
Другое	Может быть вынужденным, и может быть сохранён
M	
Описание	Область Внутренней памяти. Может быть использована для хранения внутреннего состояния или других данных. По сравнению с V областью, M область можно получить быстрее для битных операций.

Режим доступа	Может быть доступна по битам, по байтам, словом и двойным словом
Право доступа	Чтение / запись
Другое	Может быть вынужденным, и может быть сохранён
SM	
Описание	Область Системной памяти. Системные данные хранятся здесь. Вы можете прочитать некоторые SM адреса, чтобы оценить текущее состояние системы, и вы можете изменить некоторые адреса для управления некоторыми функциями системы.
Режим доступа	Может быть доступна по битам, по байтам, словом и двойным словом
Право доступа	Чтение / запись
Другое	Может быть вынужденным, и не может быть сохранён
L	
Описание	Область Локальной переменной. Kinco Builder присваивает ячейки памяти в L области для всех локальных переменных и входных / выходных переменных автоматически. Вам не рекомендуется получать доступ к L область напрямую.
Режим доступа	Может быть доступна по битам, по байтам, словом и двойным словом
Право доступа	Чтение / запись
Другое	Может быть вынужденным, и не может быть сохранён

3.6.2 Прямая адресация

Прямая адресация означает, что переменные могут быть назначены на блоках памяти для прямого доступа к ним.

★ Прямое представление переменных

В соответствии с IEC61131-3, прямой представление переменной одного элемента обеспечивается специальным символом, образованный знаком процента "%", идентификатор области памяти и размера данных назначения, одного или более целых чисел без знака, разделенных по периодам. Например, %QB7 относится к выходному байту в ячейке памяти 7.

"Непосредственно представленная переменная" соответствует "Прямому адресу" в традиционных системах PLC.

★ Символическая переменная

Вы можете назначить символическое имя "переменной прямого представления", для удобного его определения. Идентификатор должен быть использован для символического представления переменных.

В Kinco Builder, можно объявить символические переменные в таблице Глобальных переменных и таблице переменных соответствующего POU. Пожалуйста, обратитесь к соответствующим разделам для получения дополнительной информации.

3.6.2.1 Непосредственное представление переменных

Прямое представление адреса для каждой области памяти показано в следующей таблице, в которой либо x либо y представляет десятичное число.

★Область I

Бит адресация	Формат	%Ix.y
	Описание	x: байт адрес переменной y: бит номер, то есть бит байта. Его диапазон 0 ~ 7.
	Тип данных	BOOL
	Пример	%I0.0 %I0.7 %I5.6
Байт адресация	Формат	%IBx
	Описание	x: байт адрес переменной
	Тип данных	BYTE
	Пример	%IB0 %IB1 %IB5
Слово адресация	Формат	%IWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	WORD, INT
	Пример	%IW0 %IW2 %IW4
Двойное слово адресация	Формат	%IDx
	Описание	x: начальный адрес байта переменной. Поскольку размер DWORD равен 4 байта, x должно быть четное число.
	Тип данных	DWORD, DINT
	Пример	%ID0 %ID4

★Область Q

Бит адресация	Формат	%Qx.y
	Описание	x: байт адрес переменной y: бит номер, то есть бит байта. Его диапазон 0 ~ 7.
	Тип данных	BOOL
	Пример	%Q0.0 %Q0.7 %Q5.6
Байт адресация	Формат	%QBx
	Описание	x: байт адрес переменной
	Тип данных	BYTE
	Пример	%QB0 %QB1 %QB4
Слово адресация	Формат	%QWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	WORD, INT
	Пример	%QW0 %QW2 %QW4
Двойное слово	Формат	%QDx

адресация	Описание	x: начальный адрес байта переменной. Поскольку размер DWORD равен 4 байта, x должно быть четное число.
	Тип данных	DWORD, DINT
	Пример	%QD0 %QD4 %QD12

★Область AI

Слово адресация	Формат	%AIWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	INT
	Пример	%AIW0 %AIW2 %AIW12

★Область AQ

Слово адресация	Формат	%AQWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	INT
	Пример	%AQW0 %AQW2 %AQW4

★Область M

Бит адресация	Формат	%Mx.y
	Описание	x: байт адрес переменной y: бит номер, то есть бит байта. Его диапазон 0 ~ 7.
	Тип данных	BOOL
	Пример	%M0.0 %M0.7 %M5.6
Байт адресация	Формат	%MBx
	Описание	x: байт адрес переменной
	Тип данных	BYTE
	Пример	%MB0 %MB1 %MB5
Слово адресация	Формат	%MWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	WORD, INT
	Пример	%MW0 %MW2 %MW12
Двойное слово адресация	Формат	%MDx
	Описание	x: начальный адрес байта переменной. Поскольку размер DWORD равен 4 байта, x должно быть четное число.
	Тип данных	DWORD, DINT

	Пример	%MD0 %MD4 %MD12

★Область V

Бит адресация	Формат	%Vx.y
	Описание	x: байт адрес переменной y: бит номер, то есть бит байта. Его диапазон 0 ~ 7.
	Тип данных	BOOL
	Пример	%V0.0 %V0.7 %V5.6
Байт адресация	Формат	%VBx
	Описание	x: байт адрес переменной
	Тип данных	BYTE
	Пример	%VB0 %VB1 %VB10
Слово адресация	Формат	%VWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	WORD, INT
	Пример	%VW0 %VW2 %VW12
Двойное слово адресация	Формат	%VDx
	Описание	x: начальный адрес байта переменной. Поскольку размер DWORD равен 4 байта, x должно быть четное число.
	Тип данных	DWORD, DINT
	Пример	%VD0 %VD4 %VD12
REAL адресация	Формат	%VRx
	Описание	x: начальный адрес байта переменной. Поскольку размер REAL равен 4 байта, x должно быть четное число.
	Тип данных	REAL
	Пример	%VD0 %VD4 %VD1200

★Область SM

Бит адресация	Формат	%SMx.y
	Описание	x: байт адрес переменной y: бит номер, то есть бит байта. Его диапазон 0 ~ 7.
	Тип данных	BOOL
	Пример	%SM0.0 %SM0.7 %SM5.6
Байт адресация	Формат	%SMBx
	Описание	x: байт адрес переменной
	Тип данных	BYTE

	Пример	%SMB0 %SMB1 %SMB10
Слово адресация	Формат	%SMWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	WORD, INT
	Пример	%SMW0 %SMW2 %SMW12
Двойное слово адресация	Формат	%SMDx
	Описание	x: начальный адрес байта переменной. Поскольку размер DWORD равен 4 байта, x должно быть четное число.
	Тип данных	DWORD, DINT
	Пример	%SMD0 %SMD4 %SMD12

★Область L (Примечание:. Вам не рекомендуется непосредственный доступ к L области)

Бит адресация	Формат	%Lx.y
	Описание	x: байт адрес переменной y: бит номер, то есть бит байта. Его диапазон 0 ~ 7.
	Тип данных	BOOL
	Пример	%SM0.0 %SM0.7 %SM5.6
Байт адресация	Формат	%LBx
	Описание	x: байт адрес переменной
	Тип данных	BYTE
	Пример	%LB0 %LB1 %LB10
Слово адресация	Формат	%LWx
	Описание	x: начальный адрес байта переменной. Поскольку размер WORD равен 2 байта, x должно быть четное число.
	Тип данных	WORD, INT
	Пример	%LW0 %LW2 %LW12
Двойное слово адресация	Формат	%LDx
	Описание	x: начальный адрес байта переменной. Поскольку размер DWORD равен 4 байта, x должно быть четное число.
	Тип данных	DWORD, DINT, REAL
	Пример	%LD0 %LD4 %LD12

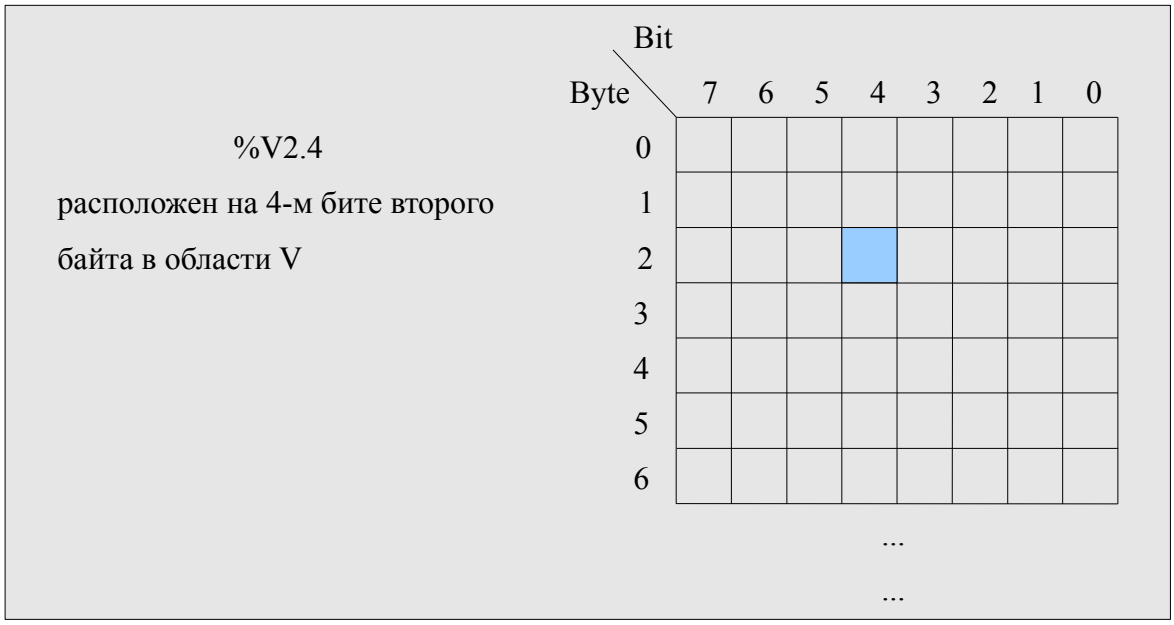
★Область HC

Двойное слово адресация	Формат	%HCx
	Описание	x: высокоскоростной счетчик
	Тип данных	DINT
	Пример	%HC0 %HC1

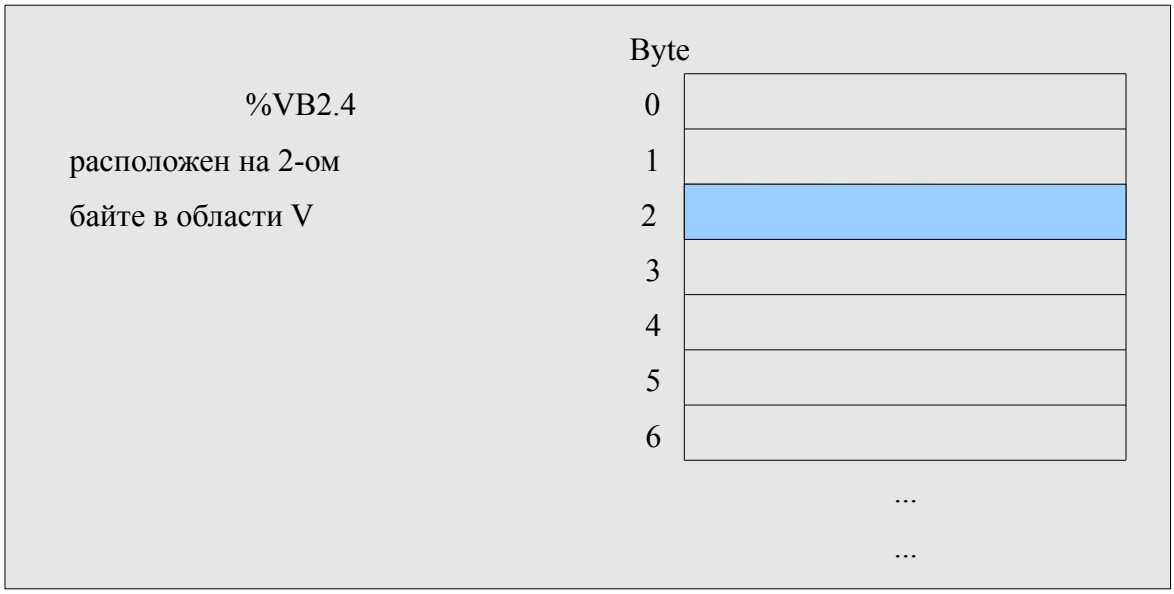
3.6.2.2 Отображение области Прямого адреса и Ячейки памяти PLC

Каждый действующий прямой адрес соответствует ячейке памяти ПЛК, отношение между ними показаны на следующей диаграмме, принимая V область в качестве примера.

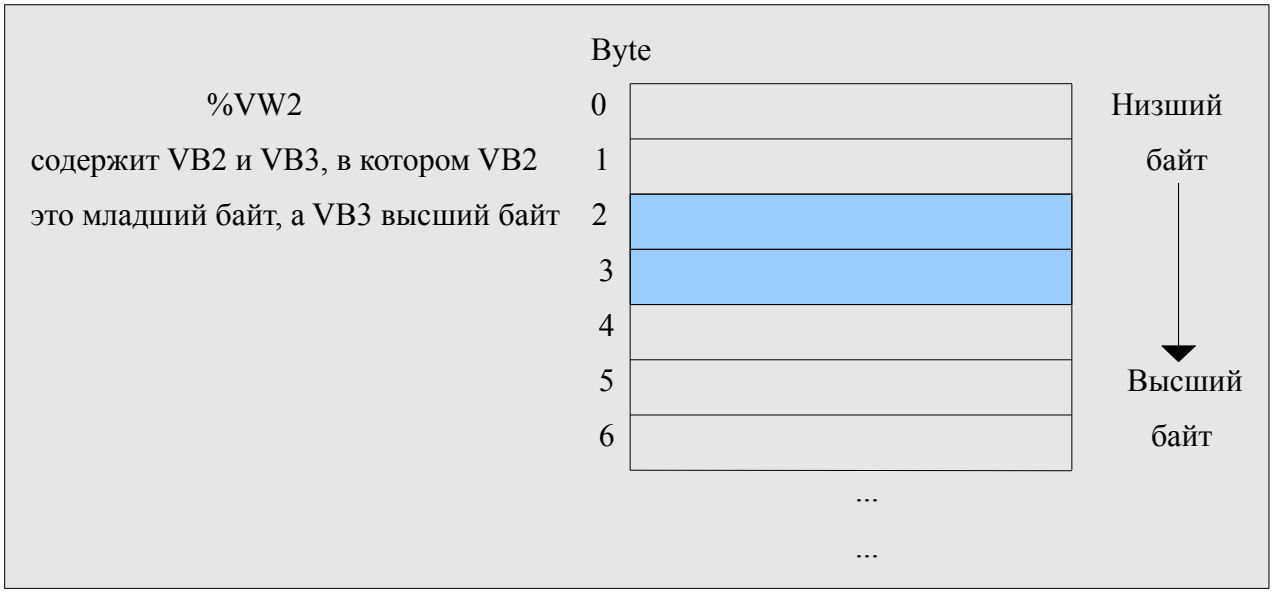
★Бит адресация



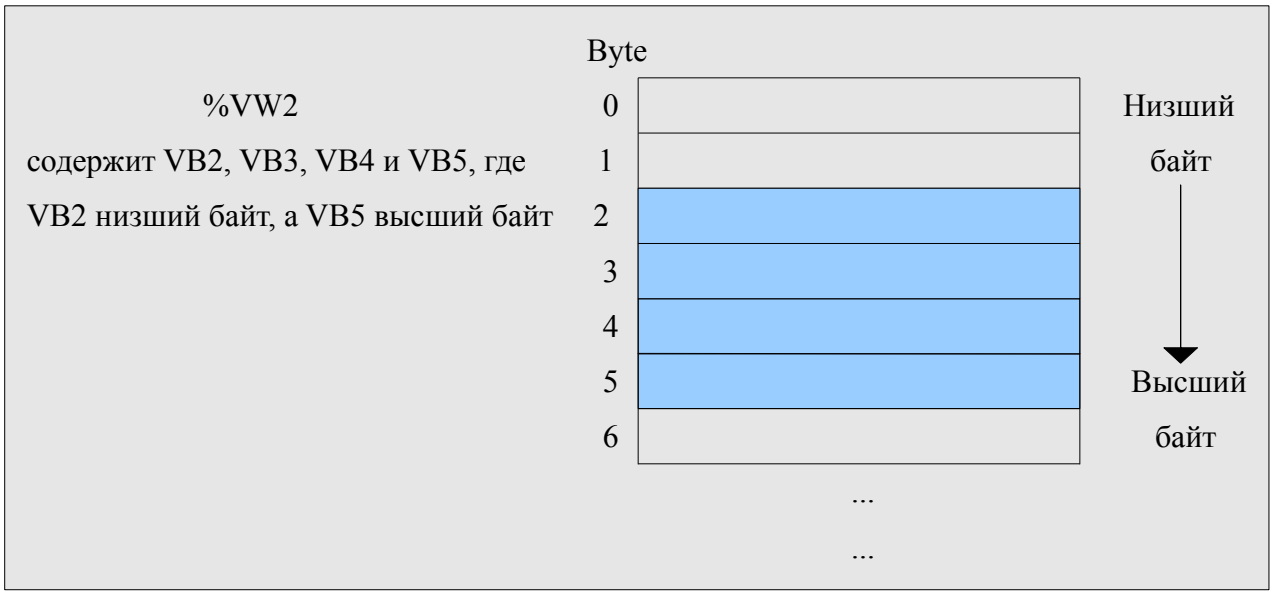
★Байт адресация



★Двойное слово адресация



★Двойное слово адресация



3.6.3 Косвенная адресация

Указатель является двойным словом переменной, которое хранит физический адрес блока памяти. Косвенная адресация использует указатель для доступа к данным в соответствующей памяти. Kinco-K5 позволяет указателям получить доступ только к области V (за исключением индивидуального бита). Кроме того, только "непосредственно представленная переменная" в V области может быть использована в качестве указателя.

3.6.3.1 Создание указателя

Для косвенного доступа к данным в запоминающем устройстве, вы должны создать указатель на этот блок. Оператор адреса "&" может быть использован, например, &VB100 означает физический адрес VB100.

Вы можете создать указатель, следующим образом: ввод оператора адреса (&) перед непосредственно представленных переменных, чтобы получить свой физический адрес, а затем написать физический адрес в других непосредственно представленных переменных как указатель, используя команду MOVE.
Например:

```
(* Создать указатель (VD204), который указывает на VW2, т.е., физический адрес VW2 хранится в VD204. *)
MOVE      &VW2, %VD204
```

3.6.3.2 Доступ к данным с помощью указателя

"*" Является оператором указателя. Ввод "*" перед указателем представляет прямой адрес переменной, на которую указывает этот указатель. При использовании указателя в качестве операнда команды, пожалуйста, обратите внимание на типы данных операндов инструкции.
Например:

```
LD%       SM0.0
MOVE      &VB0, %VD200   (* Создание указателя (VD200), который указывает на VW2. *)
MOVE      *VD200, % VB10 (* Присвоить значение VB0 в VB10. Указатель VD200 указывает на VB0 *)
                         (* поэтому * VD200 представляет собой VB0. *)
```

3.6.3.3 Изменение значения указателя

Указатель 32-битовой переменной, и поэтому значение может быть изменено с такими инструкциями, как ADD, SUB и т.д.
Всякий раз, когда значение указателя увеличивается / уменьшается на 1, прямой адрес, по которому он указывает будет увеличен / уменьшен на 1 байт соответственно. Поэтому, когда вы измените значение указателя, необходимо обратить внимание на тип данных переменной.
★ Если указатель указывает на переменную BYTE, вы можете изменить значение указателя на любое двойное целое число.
★ Если указатель указывает на WORD или переменную INT, вы можете изменить значение указателя на величину кратную 2.
★ Если указатель указывает на DWORD, DINT или переменную REAL, вы можете изменить значение указателя на величину кратную 4.

3.6.3.4 Указание по использованию указателей

★ Действия указателя являются гарантией программы пользователя. Указатели являются очень гибкими, так что вы должны быть очень осторожны при его использовании. Если указатель указывает на некорректный адрес, это может привести к непредсказуемым результатам.
★ Kinco-K5 поддерживает указатели и адреса только одного уровня, указатели и адреса нескольких уровней, являются запрещёнными. Например, следующая инструкция является запрещённой:

```
MOVE      &VB4, *VD44
```

3.6.3.5 Пример

```
(* Network 0 *)
LD        %SM0.0
MOVE      &VW0, %VD200 (* Создание указателя (VD200), который указывает на VW0. *)
MOVE      *VD200, %VW50 (* Присвоить значение VW0 к VW50. Указатель VD200 указывает на VW0 *)
                        (* поэтому VD200 представляет собой VW0. *)
ADD       DI#2, %VD200  (* Значение указателя увеличивается на 2, чтобы он указывал на _VW2 сейчас. *)
MOVE      *VD200, %VW52 (* Присвоить значение _VW2 к VW52 *)
```

3.6.4 Диапазоны адресов памяти

Kinco-K5 предоставляет несколько типов модулей CPU. Диапазоны адресов памяти различных типов CPU, могут отличаться друг от друга, а адреса за пределами соответствующего диапазона, являются незаконными. В вашей программе, вы должны убедиться, что все адреса памяти, которые вы вводите, действительны для CPU. Подробные описания приведены в следующей таблице.

Таблица 3-5 Диапазон памяти CPU

		CPU504	CPU504EX	CPU506, CPU506EA, CPU508
I	Размер	1	5	32
	Адрес бита	%I0.0 --- %I0.7	%I0.0 --- %I4.7	%I0.0 --- %I31.7
	Адрес байта	%IB0, IB1	%IB0 --- %IB4	%IB0 --- %IB31
	Адрес слова	%IW0	%IW0 --- %IW2	%IW0 ---% IW30
	Адрес двойного слова	———	%ID0	%ID0 --- %ID28
Q	Размер	1	5	32
	Адрес бита	%Q0.0 --- %Q0.7	%Q0.0 --- %Q4.7	%Q31.0 --- %Q31.7
	Адрес байта	%QB0	%QB0 --- %QB4	%QB0 --- %QB31
	Адрес слова	———	%QW0 --- %QW2	%QW0 --- %QW30
	Адрес двойного слова	———	%QD0	%QD0 --- %QD28
AI	Размер	0	16	64
	Адрес слова	———	%AIW0 --- %AIW14	%AIW0 --- %AIW62
AQ	Размер	0	16	64
	Адрес слова	———	%AQW0 --- %AQW14	%AQW0 --- %AQW62
HC	Размер	8		
	Адрес слова	%HC0, %HC1		
V	Размер	4096		
	Адрес бита	%V0.0 ---%V4095.7		
	Адрес байта	%VB0 --- %VB4095		
	Адрес слова	%VW0 --- %VW4094		
	Адрес двойного слова	%VD0 --- %VD4092		
	REAL адрес	%VR0 --- %VR4092		
M	Размер	1024		
	Адрес бита	%M0.0 --- %M1023.7		
	Адрес байта	%MB0 --- %MB1023		
	Адрес слова	%MW0 --- %MW1022		
	Адрес двойного слова	%MD0 --- %MD1020		
SM	Размер	300		
	Адрес бита	%SM0.0 --- %SM299.7		

	Адрес байта	%SMB0 --- %SMB299
	Адрес слова	%SMW0 --- %SMW298
	Адрес двойного слова	%SMD0 --- %SMD296
L	Размер	272
	Адрес бита	%L0.0 --- %L271.7
	Адрес байта	%LB0 --- %LB271
	Адрес слова	%LW0 --- %LW270
	Адрес двойного слова	%LD0 --- %LD268

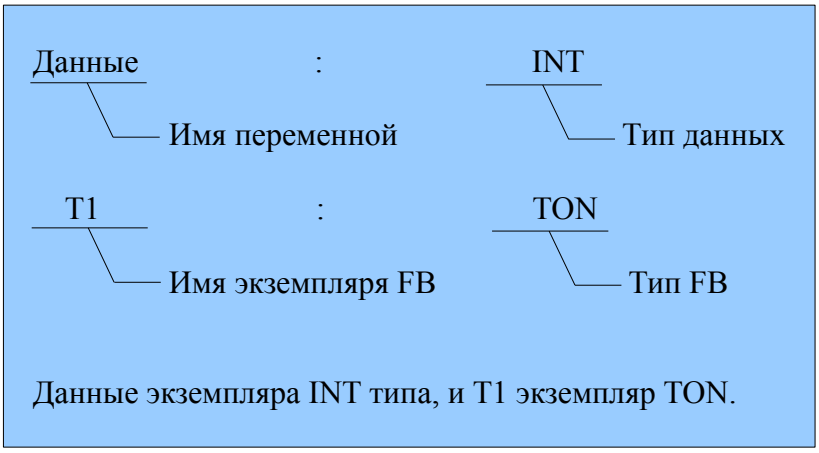
3.6.5 Функциональные блоки и их примеры

3.6.5.1 Стандартные функциональные блоки в IEC61131-3

★ Таймеры: TP --- импульсный таймер; TON --- таймер задержки включения; TOF --- таймер задержки выключения
★ Счетчики: CTU --- счетчик вверх; CTD --- счетчик вниз; CTUD --- счетчик вверх-вниз
★ Элементы двух состояний: SR --- Установить доминирующим; RS --- отменить доминирование
★ Триггеры: R_TRIG --- детектор переднего фронта; F_TRIG --- детектор заднего фронта

3.6.5.2 Примеры функциональных блоков

" Instantiation FB" является очень важным понятием в IEC61131-3.
Instantiation означает, что программист заявляет и создает переменную, указав его имя и тип данных.
После создания, переменная может быть доступна в программе.
FB также должен быть создан как переменная. После создания, FB (как примера) может быть использованы в POU, в котором она объявлена.
Как показано на следующем графике, только T1 может быть вызван и доступен.



3.6.5.3 Области памяти экземпляров FB

Область фиксированной памяти выделяется для каждого типа FB, чтобы хранить свои образцы в Kinco-K5 PLC, детали показаны в следующей таблице.

Таблица 3-6 Области памяти экземпляра FB

T	
Описание	Область памяти таймера, где экземпляры TON, TOF и TP могут быть выделены. Они используются для хранения битов состояния и текущих значений всех экземпляров таймера.
Режим доступа	Прямой доступ к биту состояния и текущему значению экземпляра таймера
Право доступа	Только чтение
Другое	Не может быть сохранён, и не может быть принужден
C	
Описание	Область памяти счетчика , где экземпляры CTU, CTD и CTUD могут быть выделены. Они используются для хранения битов состояния и текущих значений всех экземпляров счетчиков.
Режим доступа	Прямой доступ к биту состояния и текущему значению экземпляра счётчика
Право доступа	Только чтение
Другое	Может быть сохранён, и не может быть принужден
RS	
Описание	Бистабильная область RS, где экземпляры RS могут быть выделены. Они используется для хранения битов состояния для всех экземпляров RS.
Режим доступа	Прямой доступ к состоянию экземпляров RS
Право доступа	Только чтение
Другое	Не может быть сохранён, и не может быть принужден
SR	
Описание	Бистабильная область SR, где экземпляры SR могут быть выделены. Они используется для хранения состояния для всех экземпляров SR.
Режим доступа	Прямой доступ к биту состояния экземпляров SR
Право доступа	Только чтение
Другое	Не может быть сохранён, и не может быть принужден

3.6.6 Примеры использования FB

FB должен быть объявлен до его использования.
Для удобства пользователей, Kinco Builder отвечает следующим правилам: представление экземпляров FB согласуется с традиционным ПЛК, например, T0, C3; вам просто нужно назвать допустимый экземпляр FB желаемого типа в программе, и Kinco Builder будет автоматически генерировать заявления в таблице глобальных переменных.

★ T

Формат	Tx
Описание	X: десятичная цифра с указанием номера таймера
Тип данных	BOOL --- Бит состояния таймера INT --- текущее значение таймера Tx используется для доступа двух переменных. Kinco Builder будет определять доступ к любому биту состояния или текущему значению в соответствии с используемой командой: команды с BOOL операндами имеют доступ к биту состояния, а команды с INT операндами имеют доступ к текущему значению.
Пример	T0 T5 T20

★ C

Формат	Cx
Описание	X: десятичная цифра с указанием номера счётчика
Тип данных	BOOL --- Бит состояния счетчика INT --- текущее значение счета счетчика Cx используется для доступа двух переменных. Kinco Builder будет определять доступ к любому биту состояния или текущему значению в соответствии с используемой командой: команды с BOOL операндами имеют доступ к биту состояния, а команды с INT операндами доступ к текущему значению.
Пример	C0 C5 C20

★ RS

Формат	RSx
Описание	X: десятичная цифра с указанием номера RS
Тип данных	BOOL --- состояние RS
Пример	RS0, RS5, RS10

★ SR

Формат	SRx
Описание	X: десятичная цифра с указанием номера SR
Тип данных	BOOL --- состояние SR
Пример	SR0, SR5, SR10

3.6.7 Диапазоны памяти FB

Размер области памяти, который PLC может выделить типу FB ограничена ресурсами аппаратных средств; Таким образом, каждый тип Kinco-K5 CPU выделяет разные диапазоны памяти для FB. Подробное описание приведено в следующей таблице.

Таблица 3-7 Диапазоны памяти FB

T	Количество	256
	Диапазон	T0 --- T255
	Разрешение	T0 --- T3 ：1ms T4 --- T19 ：10ms T20 --- T255 ：100ms
	Макс. время	32767* Разрешение
C	Количество	256
	Диапазон	C0 --- C255
	Макс. значение счета	32767
RS	Количество	32
	Диапазон	RS0 --- RS31
SR	Количество	32
	Диапазон	SR0 --- SR31

Глава IV Как использовать Kinco Builder (основные функции)

В этой главе подробно описываются компоненты Kinco Builder, в том числе их функции и рабочие шаги. Опирается на основу базовых понятий предыдущих глав, эта глава может помочь вам получить более полное понимание Kinco Builder.
Редактор LD и редактор IL может включать в себя грамматику МЭК 61131-3, которая будет представлена в следующей главе.

4.1 Настройка основных параметров программного обеспечения

Вам нужно настроить некоторые общие параметры для Kinco Builder, например, язык программирования по умолчанию и тип процессора по умолчанию для новых проектов. Kinco Builder будет сохранять конфигурацию автоматически, поэтому вам просто нужно настроить их один раз, до следующего изменения
Выберите [Tools]> [Options …] в меню команд, а затем появиться следующее диалоговое окно:

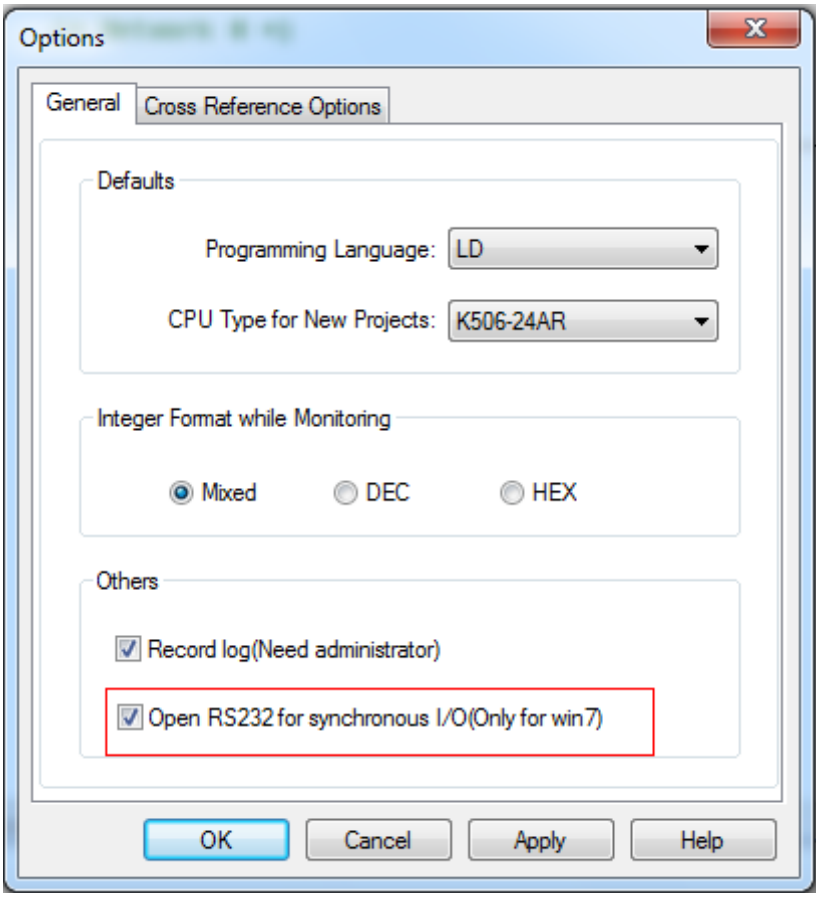


Рисунок 4-1 Диалоговое окно"Options"

■ Вкладка **General**
★ **Defaults**
• Язык программирования:
Выберите язык программирования по умолчанию для новых программ, IL или LD.
• Тип CPU для новых проектов:
Выберите тип CPU, чтобы новые проекты использовали его по умолчанию.
★ **Integer Format While Monitoring**
Выберите формат отображения для целочисленных значений в процессе мониторинга.
Mixed: Значения INT и DINT отображаются в десятичном формате;

Кроме того, значения BYTE, WORD и DWORD отображаются в шестнадцатеричном формате.
Dec: Все целые значения отображаются в десятичном формате.
HEX: Все целые значения отображаются в шестнадцатеричном формате.

★ Others

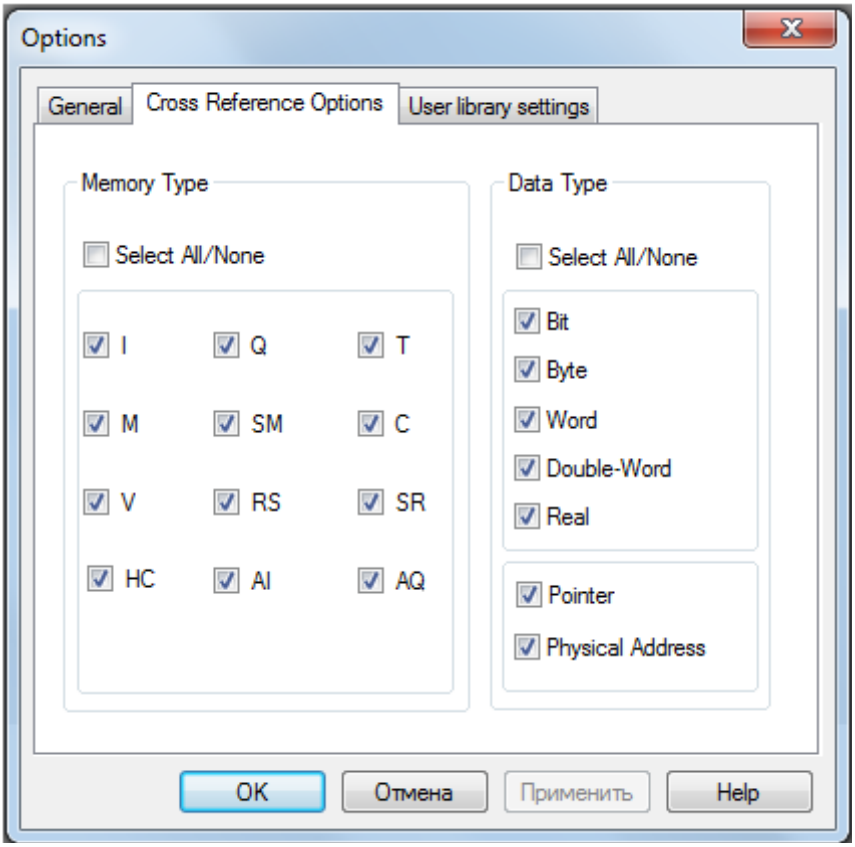
• Record log

Если этот флажок установлен, KincoBuilder создаст подкаталог с именем "KincoPlcLog" в директории установки. KincoBuilder создаст лог-файл для каждого времени, в котором сохранит файлы только одного дня. Эти файлы будут полезными для нас, чтобы отслеживать и решать баги KincoBuilder.

• Open RS232 for synchronous I/O:

Иногда KincoBuilder не может подключится к ПЛК используя некоторые преобразователи USB/RS232. Эта проблема вызвана совместимости драйвера преобразователя.
Включение этой опции может быть полезна, и в большинстве случаев это может решить проблему.

■ Вкладка **Cross Reference Options**



После компиляции, все используемые параметры и области будут отображаться в таблице Cross Reference. Все используемые параметры будут отображаться по умолчанию в Kincobuilder.
На этой странице, пользователь может выбрать [Memory Type] и [Data Type], которые будут отображаться в таблице Cross Reference.

4.2 Границы окон

В Kinco Builder Manager Window, Instructions Window, Output Window и PLC Catalog Window выполнены в виде пристыкованных окон. Эти окна имеют два режима отображения: плавающие или закрепленные. В плавающем режиме, окно может появляться в любом месте на экране. В закрепленном режиме, окно крепится вдоль любого из четырех границ главного окна Kinco Builder.

■ Для изменения закрепленного окна в плавающее окно

· Дважды щелкните на границе окна.

· Укажите на строку заголовка и перетащите окно из своей области прикрепления.

■ Чтобы закрепить плавающее окно

· Дважды щелкните заголовок окна, чтобы вернуть окно к предыдущему месту стыковки.

· Укажите на строку заголовка и перетащите окно в область прикрепления.

■ Чтобы переключить окно в режим автоматического скрытия

· Нажмите на значок, расположенный в верхнем правом углу окна.

В режиме автоматического скрытия, оно должно скрыться автоматически и остаться на границе главного окна KincoBuilder в виде иконки; Наведите курсор на эту иконку на мгновение и окно должно раскрыться.

■ Для отмены режима автоматического скрытия окна

· Нажмите на иконку, чтобы вернуться в окно к своему предыдущему месту стыковки.

4.3 Конфигурирование аппаратуры

В проекте, рекомендуется, завершить настройки оборудования в первую очередь. Когда был создан новый проект, CPU назначаются по умолчанию в диалоговом окне "Options" должны быть добавлены автоматически и вы можете изменить его по своему усмотрению.

Kinco Builder предоставляет вам полную, гибкую и удобную среду конфигурации оборудования, где можно настроить все параметры для каждого модуля PLC. Окно "Hardware" показано на рисунке 4-2. Мы видим, что это окно состоит из двух частей:

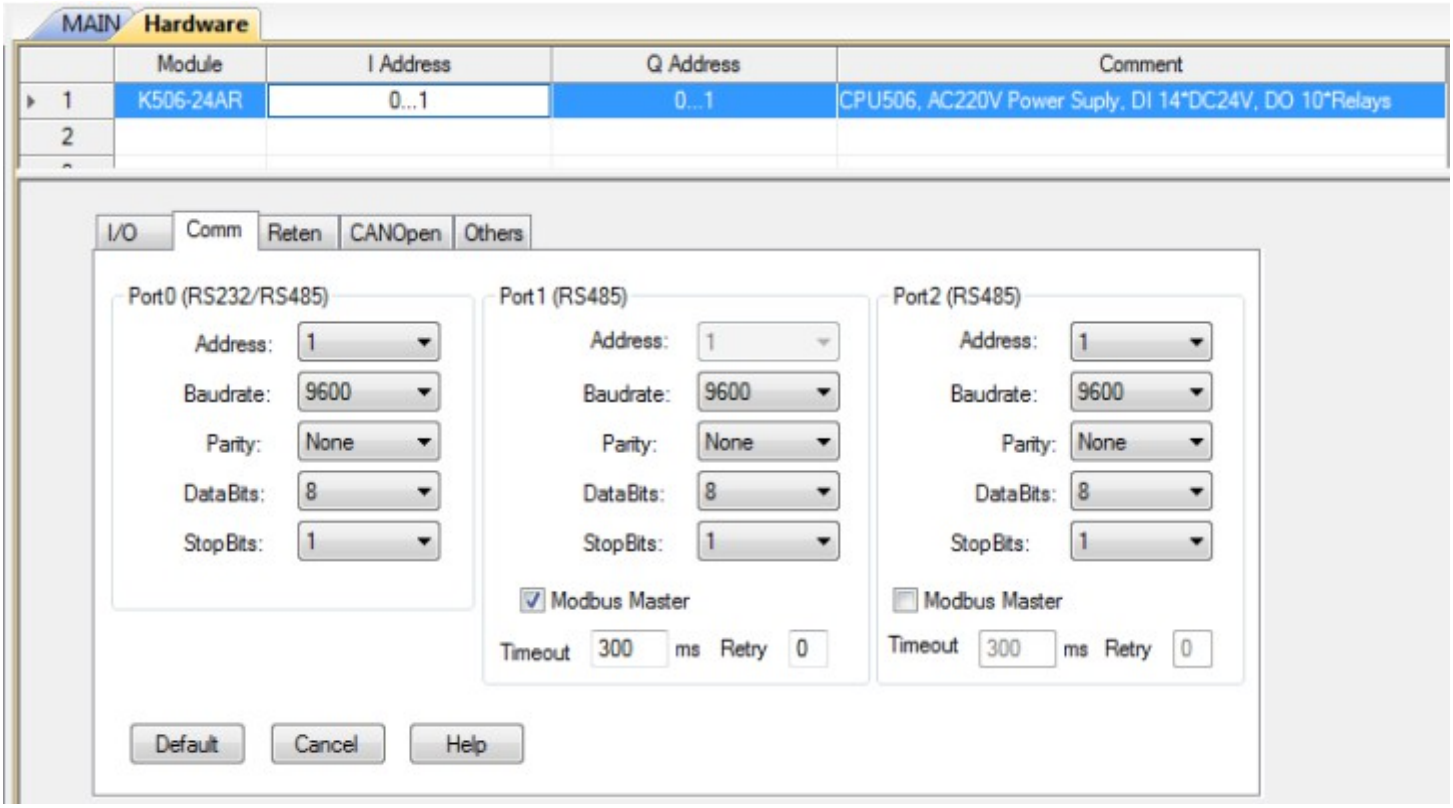


Рисунок 4-2 Окно Hardware

★ Таблица конфигурации
Верхняя часть окна оборудования показывает подробный список PLC модулей в виде таблицы, и мы называем это таблица конфигурации. Таблица конфигурации представляет реальную конфигурацию: вы располагаете ваши модули в таблице конфигурации так же, как вы сделали в реальной системе управления.
★ Окно параметров
Нижняя часть окна оборудования показывает все параметры выбранного модуля в таблице конфигурации, и мы называем это окно параметров.

4.3.1 Как открыть окно Hardware

Вы можете открыть окно "Hardware" с помощью одного из следующих способов:
★ Дважды щелкните [Hardware] в окне диспетчера.
★ Щелкните правой кнопкой мыши [Hardware], а затем выберите команду [Open] на всплывающем меню.

4.3.2 Копирование и вставка конфигурации оборудования в различных проектах

В Kinco Builder пользователи могут скопировать и вставить [Hardware Configuration] в разных проектах. [Hardware_Configuration] относится к конфигурации CANopen, порта связи и т.д., и не будет копировать информацию CPU, т.е. он может быть выполнен между CPU. Все отложенные проекты должны быть открыты в Kinco Builder. Вы можете использовать эту копию и вставить функцию в LD или IL.
Эта функция будет полезна, если вы хотите перенести конфигурацию CANopen старых проектов. Вы можете воспользоваться этой функцией следующим образом:
★ Нажмите [Copy_Hardware_Configuration], [Paste_Hardware_Configuration] в меню [Edit];
★ Щелкните правой кнопкой мыши [PLC_Hardware_Configuration] в [Project_Manager] дерева и выполнить команду [Copy_Hardware_Configuration] и [Paste_Hardware_Configuration]

4.3.3 Добавление / Удаление модулей

★ Добавить модуль
Вы можете добавить модуль, выполнив следующие действия:
★ В таблице конфигурации щелкните строку поместив внимание на нем. Если существует модуль в этом ряду, он должен быть удален перед добавлением нового модуля.
★ В окне каталога PLC дважды щелкните модуль, чтобы добавить его в текущий ряд в таблице конфигурации.
В строку 1 может быть добавлен только модуль CPU, а в другие строки могут быть добавлены только модули расширения. Там не должно быть пропущенных строк между любыми двумя модулями. Если пропущенная строка существует, Kinco Builder не позволит продолжить добавлять модули после него, и будет сообщение об ошибке во всплывающем окне, при сохранении или компиляции проекта.
★ Удалить модуль
Вы можете удалить модуль следующим способом:
· Нажмите модуль, который будет удален в таблице конфигурации, затем нажмите клавишу Del, чтобы удалить его.
· Щелкните правой кнопкой мыши модуль, который должен быть удален, а затем выберите команду [Remove] в всплывающем меню.

4.3.4 Настройка параметров модуля

После того как вы расположили модули в таблице конфигурации вы можете продолжить настраивать параметры.
Kinco Builder позволяет определить все параметры модуля.

В таблице конфигурации выберите модуль PLC, чтобы поместить курсор на нем, а затем окно параметров этого модуля появиться ниже. Вы можете назначить параметры модуля в окне параметров. Вы можете использовать клавиши со стрелкой Up и Down, чтобы переместить курсор в таблице конфигурации. В окне параметров есть две кнопки: [Default] и [Cancel].
· [Default]: Если нажать эту кнопку, Kinco Builder будет назначать параметры по умолчанию для текущего модуля.
· [Cancel]: Если нажать эту кнопку, оригинальная конфигурация текущего модуля будет восстановлена.
Примечание: Адреса модулей в одной и той же области памяти (I, Q, AI или AQ) не могут совпадать!

4.3.4.1 Параметры CPU

■ **Вкладка [I/O Configuration]**
Здесь вы можете задавать параметры ввода / вывода модуля CPU, как показано на следующем рисунке.

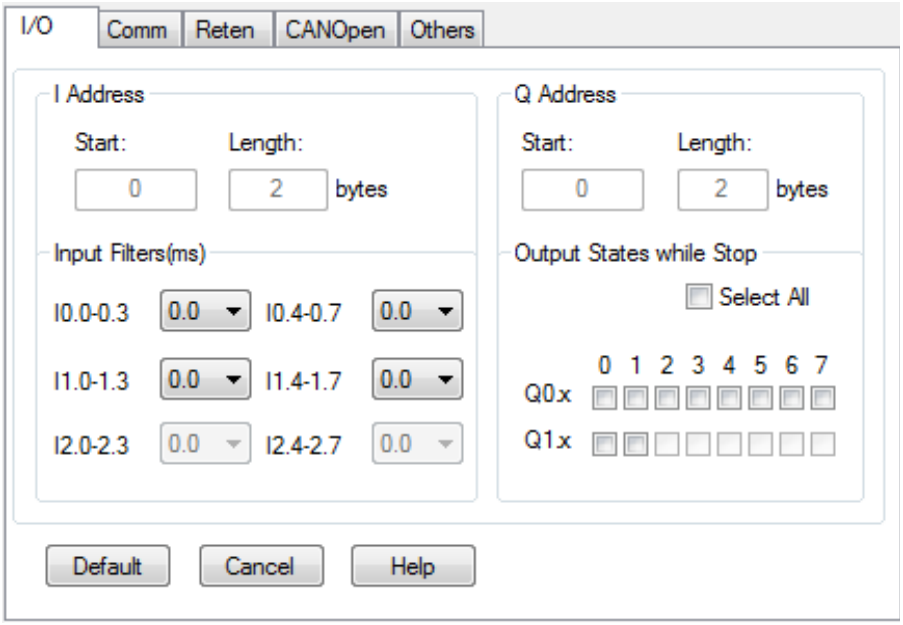


Рисунок 4-3 Параметры ввода / вывода CPU

■ **Вход:** Здесь Вы можете настроить DI каналы CPU.
★ **I Address:** начальный адрес байта каналов DI в I области. Он зафиксирован равным 0.
★ **Input Filters:** Выберите входной фильтр (мс), который определяет время задержки для DI каналов. Эта задержка полезна для фильтрации входного шума и повышения помехозащищённости системы управления. Когда состояние входа изменяется, он не будет действительным, если он не остается в течение всего времени фильтра.

■ **Выход:** Здесь Вы можете настроить DO каналы CPU.
★ **Q Address:** начальный адрес байта каналов DO в Q области. Он зафиксирован равным 0.
★ **Output States while STOP:** Задает состояние цифровых выходов во время останова CPU. Если флажок для конкретного выхода установлен, то выход должен быть установлен в положение ON (1) во время останова CPU. По умолчанию, состояние выходов во время останова CPU, является OFF (0). Эта функция важна для требований защитной блокировки после перехода RUN-to-STOP.

■ **Вкладка [Communication Ports]**
Здесь вы можете назначить параметры последовательных портов связи для Port0 и Port1 модуля CPU.

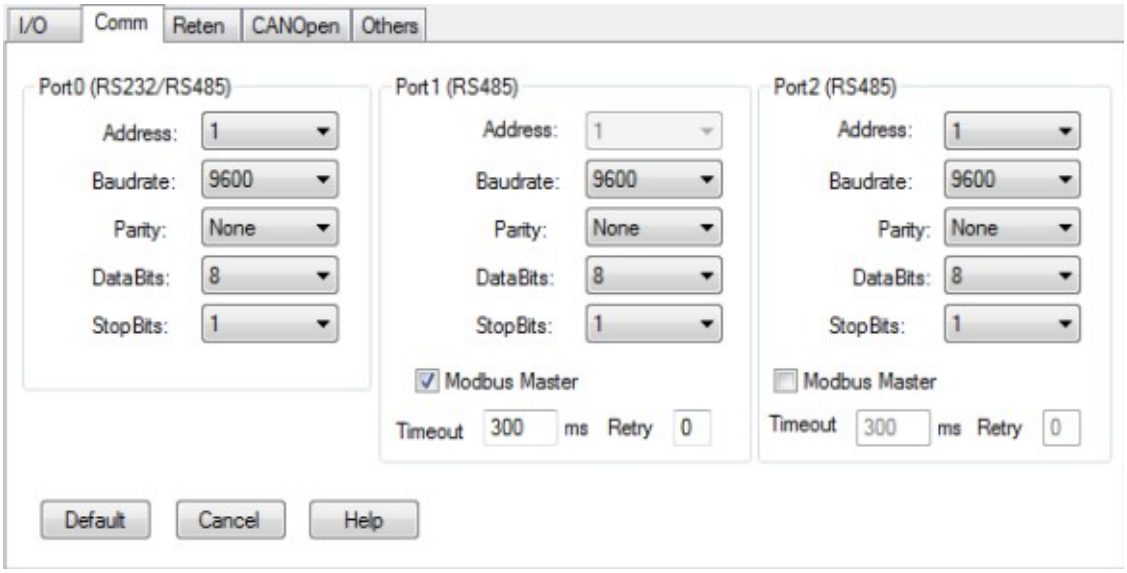


Рисунок 4-4 Параметров связи последовательных портов

■ Port0

★ Address: Выберите нужный адрес станции Port0. Этот адрес также выступает в качестве номера ведомого Modbus RTU в сети.
★ Baudrate: Выберите нужную скорость передачи. (2400, 4800, 9600, 19200, 38400, 57600or115200bps)
★ Parity: Выберите желаемую схему четности. (Без контроля по четности, Odd или Even)
★ DataBits: Выберите число бит в переданных и полученных байтах. (8)
★ StopBits: Выберите количество стоп-битов. (1)

■ Port1 и Port2

Port1 и Port2 являются RS485 портами.
★ Modbus Master: Если флажок установлен, Port1 будет работать в качестве ведущего Modbus RTU.
★ Timeout: Введите значение тайм-аута для мастера Modbus.
★ Retry: Введите значение времени повтора. Это время повторной попытки связи с ведомым устройством, когда мастер получает неправильный кадр от ведомого.
★ Baudrate: Выберите нужную частоту передачи. (2400, 4800, 9600, 19200or38400bps)
★ Parity: Выберите желаемую схему четности. (Без контроля по четности, Odd или Even)
★ DataBits: Выберите число бит в байтах переданных и полученных. (8)
★ StopBits: Выберите количество стоп-битов. (1)

■ Вкладка [Retentive Ranges]

Здесь вы можете определить четыре диапазона с возможностью сохранения, чтобы выбрать диапазоны RAM которые вы хотите сохранить при потере питания. Если CPU теряет питание, мгновенные данные в RAM будут поддерживаться супер конденсатором, и только данные в с возможностью сохранения диапазонов останутся на прежнем уровне до следующего включения.

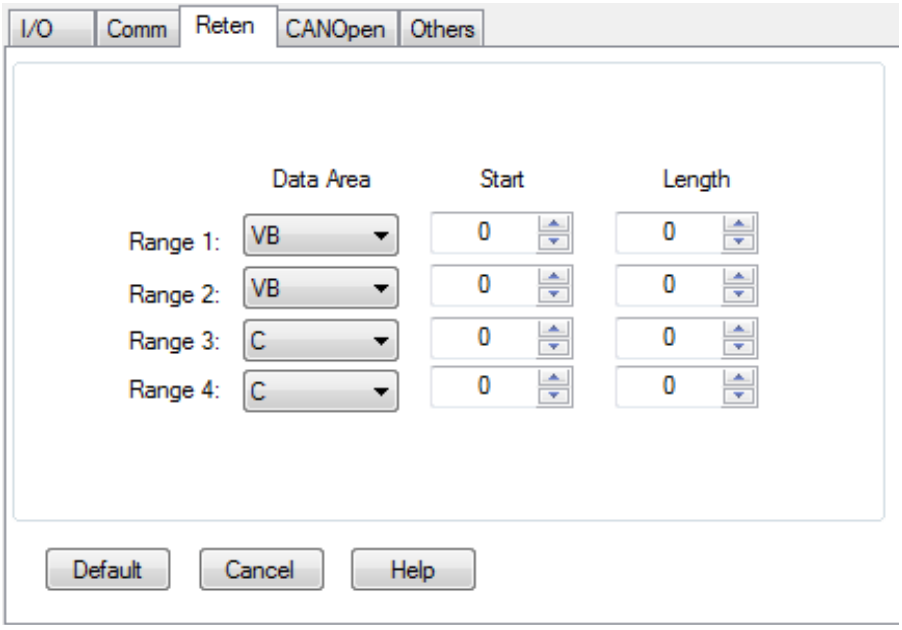
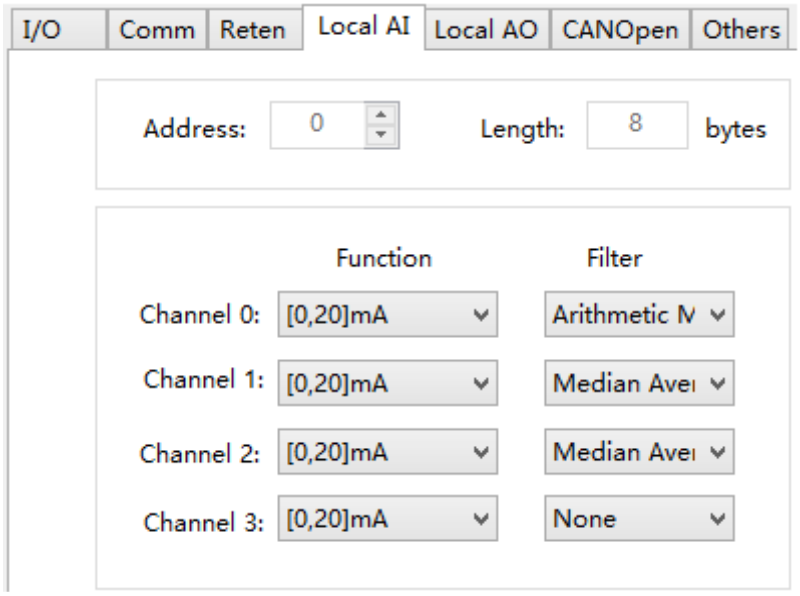
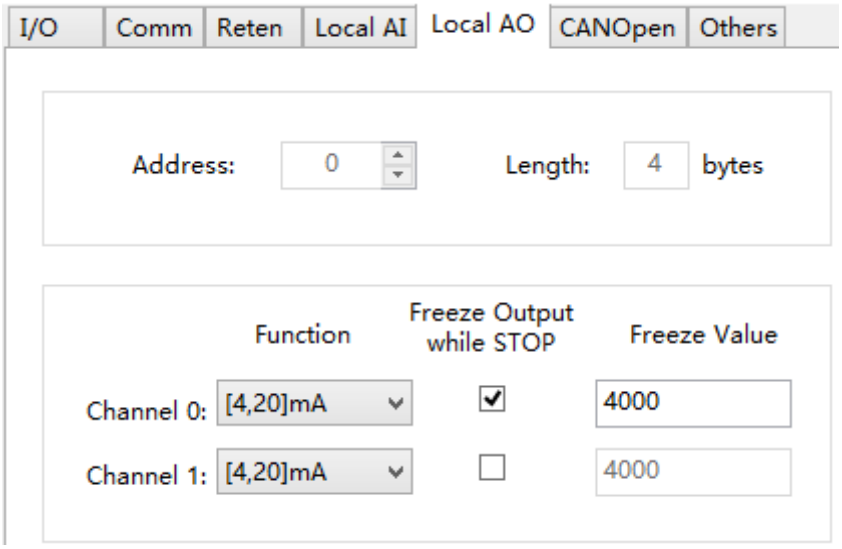


Рисунок 4-5 Сохраняемые диапазоны

■ Диапазон 1
• Область данных
Выберите область памяти для сохраняемого Диапазона 1. (V области или Counter области)
Для счетчиков, только текущие значения счета могут быть сохранятся.
• Start
Назначьте адрес начального байта Диапазона 1.
• Length
Назначьте длину Диапазона 1, блок: BYTE.
■ Диапазон 2
■ Диапазон 3
■ Диапазон 4
Пожалуйста, обратитесь к информации, указанной выше.
Как показано на рисунке 4-5, данные, хранящиеся в Диапазоне 1 (%VB0 ... %VB9), Диапазон 2 (%VB100 ... %VB199), Диапазон 3 (C0 ... C9) и Диапазон 4 (C20 ... C49) будут с возможностью сохранения при потере питания.

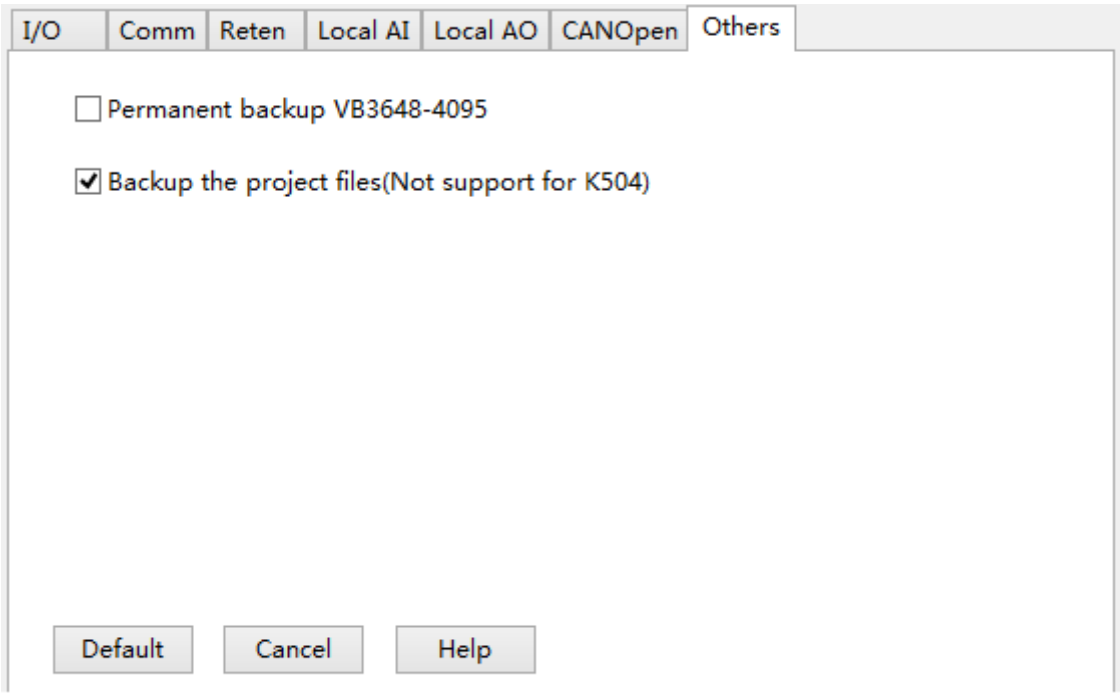
■ **Вкладка [Local AI/AO]**
K506EA-30 ат объединяет 4 каналов AI. Эти каналы отображаются в AIW0, AIW2, AIW4, AIW6 соответственно. Выборка скорость преобразования каждого канала составляет около 30 раз в секунду.
K506EA также интегрируется 2 AO каналов. Эти каналы отображаются в AQW0, AQW2 соответственно. Выборка скорость преобразования каждого канала составляет около 30 раз в секунду.
Для этих встроенным AI / AO, пользователи должны установить функции и фильтр для каждого канала в [Оборудование] - [Локальный А.И.] / вкладка [Локальный AO] следующим образом:





■ Вкладка [CANOpen]
Смотри Приложение E

■ Вкладка [Others]
1. Если нет галочки "Permanent backup VB3648-4095", то в ПЛК K5 резервное копирование будет 255 байт (VB3648-3902) по умолчанию, как и в K3 PLC. Пользователи могут использовать "Permanent backup VB3648-4095" в соответствии с программой.
2. Резервное копирование файлов проекта.
Когда пользователь создаёт новый проект, [Backup the project files(Not support for K504)] выбран по умолчанию. С помощью этой функции, проект программируемый Kincobuilder будет встроен в файл OBJ в формате ZIP. Когда пользователь загружает проект, проект будет извлечен из архива.
• По сравнению с функцией загрузки K3, отмеченные параметры идентификатора, имя программы, имя подпрограммы будут сохранены.
• Резервное копирование файлов проекта будет занимать память EEPROM. Если проект слишком большой, чтобы быть уместить в EEPROM, ПЛК проигнорирует эту опцию.
• Этот параметр увеличит время загрузки.
• Если пользователю не нужно загружать проект в PLC, а только отладить проект, то можно отключить эту опцию.



4.3.4.2 Параметры модулей дискретного входа

Вы можете установить параметры модуля DI следующим образом:

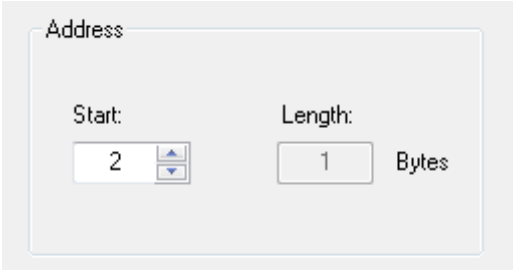


Рисунок 4-6 Параметры модуля DI

■ **Address**

• **Start**

Введите адрес начального байта адреса диапазона этого модуля в I области. Адреса каналов этого модуля на основе этого начального адреса.

• **Length**

Длина этого диапазона адресов модуля. Это значение является фиксированным и зависит от количества DI каналов модуля.

Как показано на рисунке 4-6, модуль имеет 8 каналов DI, и его начальный адрес является %IB2, поэтому адреса его каналов с %I2.0 по %I2.7.

4.3.4.3 Параметры модуля DO

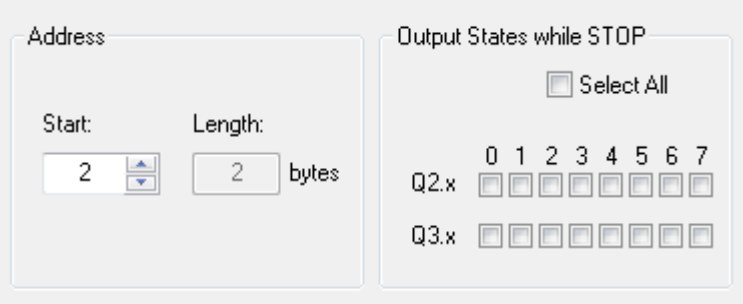


Рисунок 4-7 Параметры модуля DO

■ **Address**

• **Start**

Введите адрес начального байта адреса диапазона этого модуля в Q области. Адреса каналов этого модуля на основе этого начального адреса.

• **Length**

Длина этого диапазона адресов модуля. Это значение является фиксированным и зависит от количества DO каналов модуля.

Как показано на рисунке 4-7, модуль имеет 16 каналов DO, и его начальный адрес является %QB2, поэтому адреса его каналов с %Q2.0 по %Q3.7.

■ **Output States while STOP**

★ Здесь вы можете настроить состояние цифровых выходов во время останова CPU. Если флажок для выхода установлен, то выход должен быть установлен в положение ON (1), во время останова CPU. Состояние выходов по умолчанию во время останова CPU ВЫКЛ (0).

4.3.4.4 Параметры модуля AI

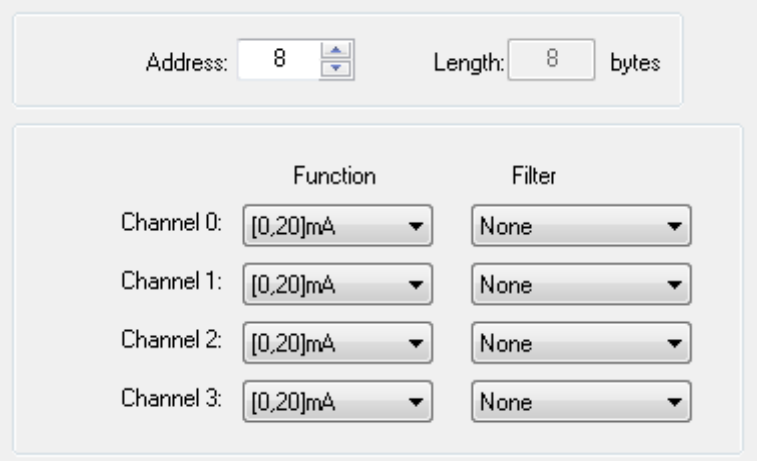


Рисунок 4-8 Параметры модуля AI

■ **Address**

• **Address**

Введите адрес начального байта (адрес первого канала) этого модуля в AI области; адреса для других каналов на основе этого начального адреса, каждый адрес занимает два байта. Это численное значение должно быть четным.

• Length
Длина этого диапазона адресов модуля. Это значение является фиксированным и зависит от количества каналов AI модуля.
Как показано на рисунке 4-8, модуль имеет 4 канала AI и его стартовый адрес %AIW0, а адреса других каналов %AIW2, %AIW4 и %AIW6.

■ Inputs
• Function
Выберите тип измерения для канала, например, 4-20mA, 1-5V, и т.д.
Пожалуйста, обратитесь к разделу **6.1.4 "Internal Presentation Format of the Measured Values of Signals"** в **"Hardware Manual"** для представления измеренного значения.
• Filter
Выберите программный фильтр для канала. Для аналогового сигнала с резким изменением, фильтр может быть полезным для стабилизации измеренного значения.
Примечание: Если система управления требует, чтобы ответ на сигнал AI был быстрым, программный фильтр соответствующего канала следует отключить.
Вы можете назначить один из следующих фильтров для канала:
No --- Программный фильтр отключена.
Arithmetic Mean --- отфильтрованное значение это среднее арифметическое значение числа сигнала на входе.
Sliding Mean --- отфильтрованное значение это скользящее среднее значение из числа выборок входного сигнала.

4.3.4.5 Параметры модуля AO

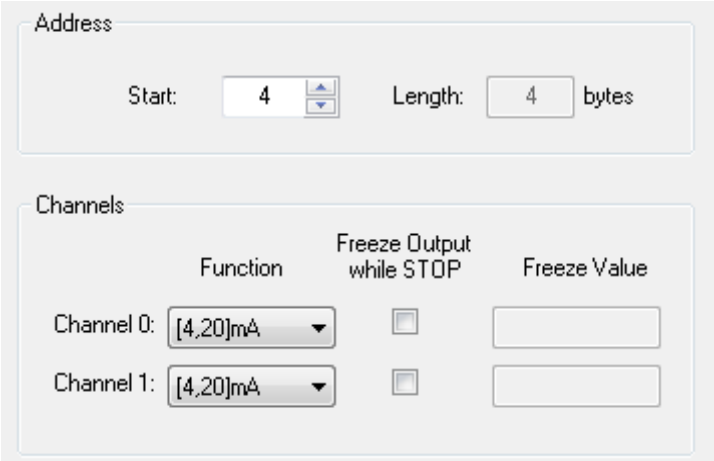


Рисунок 4-9 Параметры модуля AO

■ Address
• Start
Введите начальный адрес (адрес первого канала) этого модуля в AQ области; адреса для других каналов на основе этого начального адреса, каждый адрес занимает два байта. Это численное значение должно быть четным.
• Length
Длина этого диапазона адресов модуля. Это значение является фиксированным и зависит от количества каналов AO модуля.
Как показано на рисунке 4-9, модуль имеет два канала AQ, и его начальным адресом является %AQW0, а адрес другого канала %AQW2.

■ **Outputs**
• **Function**
Выберите тип выходного сигнала для канала, например, 4-20mA, 1-5V, и т.д.
Пожалуйста, обратитесь к разделу **"7.1.4 Internal Presentation Format of Signal Value"** в **"Hardware Manual"** для представления выходного значения.
• **Freeze Output while STOP**
Выберите, следует ли установить аналоговый выход к известному значению (Freeze Value), во время останова CPU. Если флажок для выхода установлен, выход должен сохраниться на значении во время останова CPU.
• **Freeze Value**
Здесь вы можете ввести значение, которое аналоговый выход должен поддерживать во время останова CPU.

4.4 Таблица исходных данных

В таблице исходных данных вы можете присвоить начальные численные значения BYTE, WORD, DWORD, INT, DINT и REAL переменных в V области. Модуль CPU обрабатывает исходные данные один раз при включении питания, а затем начинает цикл сканирования.
Таблица исходных данных показана на Рисунке 4-10.

Initial Data

	Address	Value	Value	Value	Value
1	%VB0	B#1	B#11	B#11	
2	%VW12	2	22	222	
3	%VD122	DI#3	DI#33	DI#3333333	
4	%VR322	4.4	4.44	4.444	

Рисунок 4-10 Таблица исходных данных

Примечание: Области памяти для "Initial Data", "Data Maintain" и "Data Backup" должны быть размещены во избежание совпадения. Данные должны быть восстановлены после включения CPU. Последовательность: восстановление данных в памяти, определённых в "Data Maintain", присвоение начального значения памяти из "Initial Data" и восстановление данных сохраненных команд.

4.4.1 Открытие таблицы исходных данных

■ Дважды щелкните [Initial Data] в окне диспетчера.
■ Щелкните правой кнопкой мыши [Initial Data], а затем выберите команду [Open] на всплывающем меню.

4.4.2 Редактирование ячейки

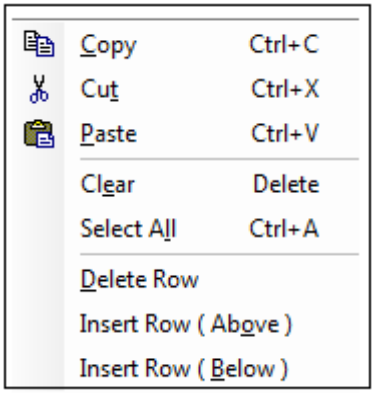
Нажмите на ячейку, чтобы перейти в режим редактирования, теперь вы можете ввести нужные данные. Кроме того, вы можете использовать для перемещения вверх, вниз, влево и правую клавиши со стрелками для перемещения курсора от одной клетки к другой, и клетки, которые получают курсор должны перейти в режим редактирования.
Когда клетка теряет курсор, её содержание сохраняется. Кроме того, вы можете использовать клавишу ENTER, чтобы подтвердить вашу работу и переместить курсор на следующую ячейку.
Не корректные данные должны выделяться красным цветом.

4.4.3 Создание данных начальных назначений

Таблица имеет 5 колонок: адрес столбца и 4 столбца значений.
★ Введите прямую переменную, то есть прямой адрес в адресную строку.
★ Введите числовые значения в столбцах Value. Вы можете ввести один или несколько значений. При вводе нескольких значений, Kinco Builder должен сделать неявное назначение адресов.
Как показано на рисунке 4-10, Строка 1 указывает, что B#1 назначается %VB0 и B#2 назначается %VB1; Строка 2 показывает, что 2, 3 и 4 назначены %VW10, %VW12 и %VW14 соответственно; Ряд 3 показывает, что DI#100, DI#200, DI#2000 и DW#2456 назначены %VD100, %VD104, %VD108 и %VD112 соответственно.

4.4.4 Редактирование таблицы исходных данных

★ Сортировка
Щелкните заголовок столбца Address, чтобы отсортировать таблицу.
★ Всплывающее меню
Щелкните правой кнопкой мыши на любой ячейке в таблице, будет показано следующее всплывающее меню:



• Delete Row: Удалить строку, в которой находится курсор.
• Insert Row (Above): Вставить новую пустую строку выше строки, в которой находится курсор.
• Insert Row (Below): Вставить новую пустую строку ниже строки, в которых находится курсор.
Пожалуйста, обратите внимание, пользуясь командой вставки: она не будет работать между различными типами таблицы и между различными рядами.

4.5 Таблица глобальных переменных

Таблица глобальных переменных состоит из двух частей: вкладка глобальных переменных и вкладка FB.
★ Вкладка глобальных переменных
Может быть использована для определения глобальной переменной, которая обращается к адресу памяти ПЛК.
Глобальная переменная в программе может заменить адрес памяти ПЛК, чтобы обеспечить читаемость программы. Каждому адресу памяти может быть назначено одно символическое имя переменной; Точно так же, одно символическое имя переменной будет иметь только один соответствующий адрес памяти.

Пожалуйста, обратитесь к пункту 3.3.1 "Как определить идентификатор", чтобы увидеть определяющие правила;
Пожалуйста, обратитесь к пункту 3.5 "Переменные", для получения подробной информации о глобальных переменных.

Вы можете объявлять здесь глобальные символические переменные, как показано на рисунке 4-11. "Таблица глобальных переменных" обычно обозначается на этой вкладке.

Как уже упоминалось в "3.6.6 Использование FB экземпляров", экземпляры FB объявляются в Kinco Builder автоматически, для облегчения пользователю. Поэтому вся информация здесь только для справки, и вы не можете её изменить.

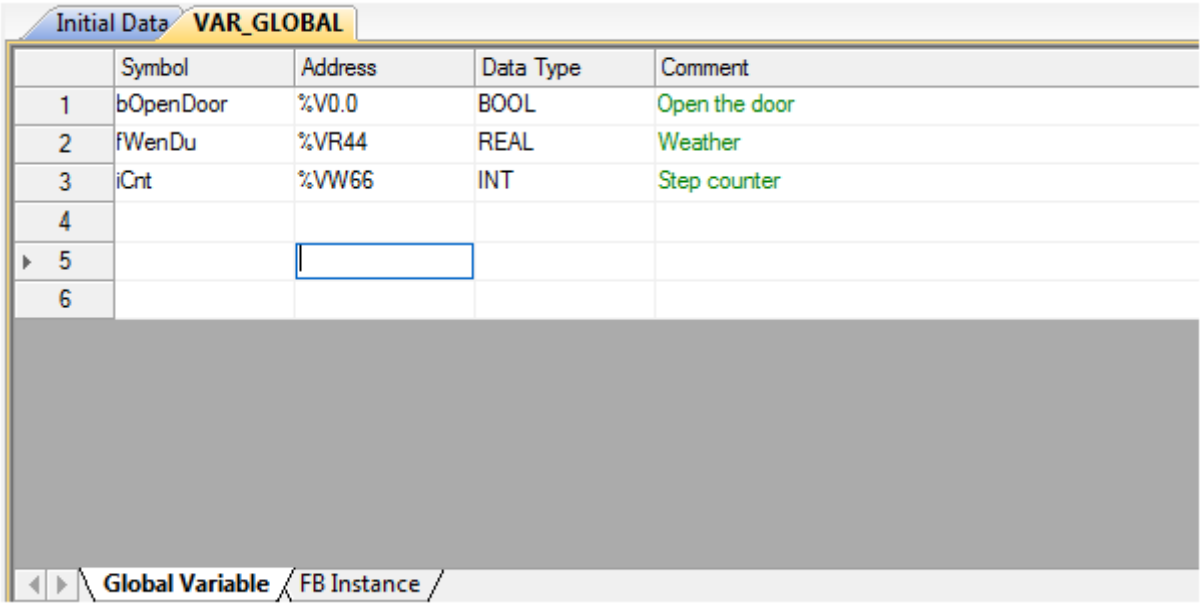


Рисунок 4-11 Вкладка глобальных переменных

★ Вкладка FB

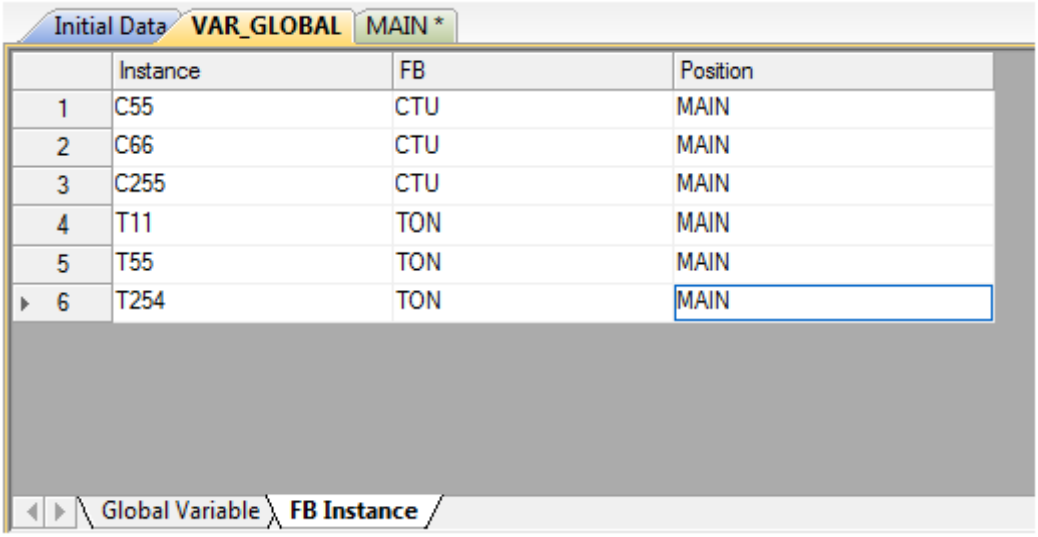


Рисунок 4-12 Вкладка FB

4.5.1 Открытие таблицы глобальных переменных

Есть три способа, чтобы открыть таблицу глобальных переменных:
★ Дважды щелкните [Global Variable] в окне диспетчера.
★ Щелкните правой кнопкой мыши [Global Variable], а затем выберите команду [Open] на всплывающем меню.
★ Выберите [Project]> [Global Variable] в меню команд.

4.5.2 Объявление глобальных переменных

Таблица имеет 4 колонки: Symbol, Address, Data Type и Comment.
★ Откройте окна таблицы глобальных переменных и выберите вкладку глобальной переменной.
★ Введите имя символа в столбце Symbol и подтвердите его.
★ Введите прямой адрес в адресную строку и подтвердите его.
★ Выберите тип данных из выпадающего списка в столбце Data Type.
★ (Необязательно) Введите комментарий .
Если вы объявляете глобальную переменную в таблице глобальной переменной, вы можете использовать её в любой POU, и прямой адрес равносилен ее символическому названию в программе пользователя. Пожалуйста, обратитесь к разделу "3.5 Переменные" для получения дополнительной информации о глобальной переменной.
Вы можете управлять таблицей глобальной переменной так же, как таблицей исходных данных. Пожалуйста, обратитесь к разделу "4.4 Таблица начальных данных" для получения дополнительной информации.

4.6 Таблица перекрёстных ссылок

Таблица перекрёстных ссылок показывает все переменные, используемые в проекте, и определяет POU, сеть или строку, и как получить доступ к операндам (чтения или записи). Таблица перекрёстных ссылок полезна, когда вы хотите знать, если символическое имя или адрес уже используется, и где он используется. Информация в Таблице перекрестных ссылок генерируется только после первой компиляции, и обновляется автоматически после каждой компиляции.

MAIN | Cross Reference

Index	Address	Symbol	POU	Position	Read/Write
0	%AIW0		MAIN	Row 5	Read
1	%VW2		MAIN	Row 5	Write

Рисунок 4-13 Таблица перекрёстных ссылок

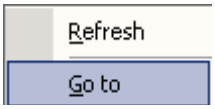
- **Address** Показывает все адреса памяти, используемые в проекте.
- **Symbol** Отображает глобальное символическое имя и адрес.
- **POU** Указывает POU в которой используется Address.
- **Position** Указывает строку или сеть, в которой используется Address.
- **Read / Write** Указывает, является ли Address чтение или запись.

Как показано на рисунке 4-13, первая строка в таблице означает, что %M1.3 используется один раз в Network0 в основной программе, и в этом случае читается.
Дважды щелкните на строке в таблице перекрестных ссылок, и вы должны перейти к соответствующей части программы.

4.6.1 Открытие таблицы перекрестных ссылок

★ Выберите [Project] > [Cross Reference] в меню команд .
★ Нажмите на иконку  на панели инструментов.
★ Используйте горячую клавишу Alt + C.

4.6.2 Всплывающее меню

Щелкните правой кнопкой мыши на любой строке в таблице, следующее меню должно появиться во всплывающем окне. 

• **Refresh:** Обновить таблицу и отображать последнюю информацию перекрестных ссылок.
• **Go to:** Перейти к соответствующей части вашей программы.

4.7 Диаграмма состояния

Вы можете использовать диаграмму состояния для мониторинга и воздействия на какую-либо переменную на прямую, используемой в проекте, после того как вы загрузили проект на PLC. Диаграмма состояния показана на рисунке 4-15.
★ Память диаграммы мониторинга
Может использоваться для обнаружения любого адреса памяти PLC;
Эта функция доступна на K3 и K5 PLC; на K5 доступна вся память, в то время как на K3 только I, Q, AI, AQ, M, и V области.

Initial Data | VAR_GLOBAL | MAIN * | **Status Chart**

	Address	Number	Format	Value 1	Value 2	Value 3
1	%V0.0	3	DEC	FALSE	FALSE	FALSE
2						
3	%MB6	7	DEC	B#0	B#0	B#0
4	%MB9			B#0	B#0	B#0
5	%MB12			B#0		
6						
7	%VR44	9	DEC	0	0	0
8	%VR56			0	0	0
9	%VR68			0	0	0
10						
11	%SMW66	5	DEC	0	0	0
12	%SMW72			0	0	
13						
14						
15						
16						
17						
18						

Рисунок 4-14 Диаграмма мониторинга

Вы можете найти в таблице:
[Memory Address]: Введите начальный адрес области памяти для обнаружения;
[Monitor Length]: Введите общее количество данных, которые должны быть обнаружены из адресов

памяти. Максимальное количество составляет 150.
Для каждой строки показано максимально 3 значения данных. Если данных больше чем 3, данные будут отдельно в разных строках.
[Display Format]: Выберите формат отображения, десятичной или шестнадцатеричной системы.
[Memory Value]: Показать память. Если бит памяти обнаружен, будут показаны только TRUE или FALSE.

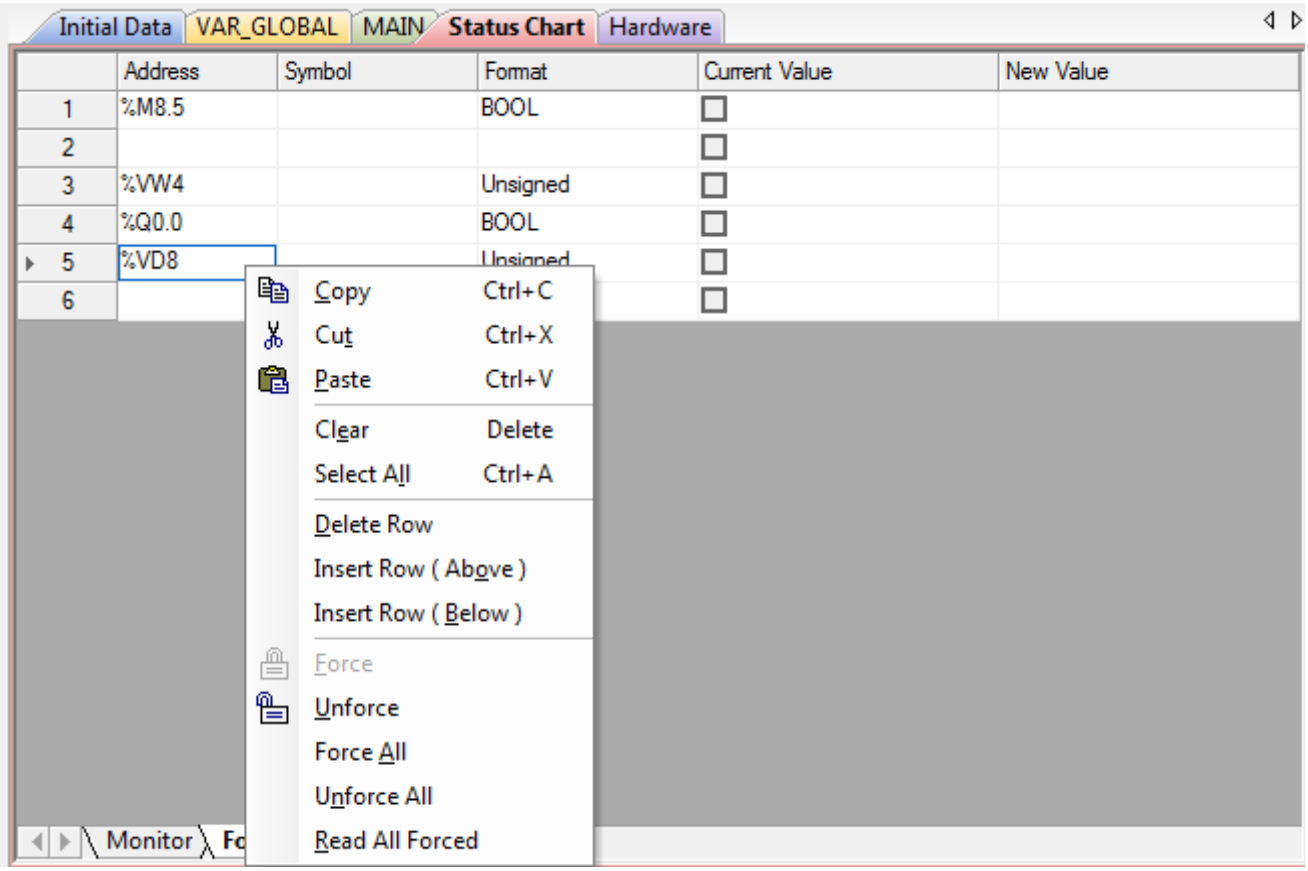


Рисунок 4-15 Диаграмма состояния

• **Address:** Введите начальный адрес, который будет контролироваться.
• **Symbol:** Отображение глобального символического имени Address.
• **Format:** Выберите формат отображения для текущего значения и новое значение. (BOOL; REAL; Signed, Unsigned, Hexadecimal или Binary)
• **Current value:** Отображение текущих значений Address от PLC.
• **New Value:** Введите значение, которое будет присвоено для Address при мониторинге.
Вы можете открыть диаграмму состояния, чтобы изменить её, но никакой информации о состоянии в столбце Current Value не отобразится, если вы не выберите команду [Monitor] в меню или на панели инструментов [Debug].
Для того, чтобы быть эффективным, Kinco Builder позволяет осуществлять мониторинг и изменять переменные, используемые в проекте. Если ввести переменные, которые не используются, Current Value и New Value не вступят в силу.

4.7.1 Открытие Диаграммы состояния

★ Дважды щелкните [Status Chart] узел в окне диспетчера.
★ Щелкните правой кнопкой мыши [Status Chart], а затем выберите [Open] на всплывающем меню.
★ Выберите [Debug] > [Status Chart] в меню команд.

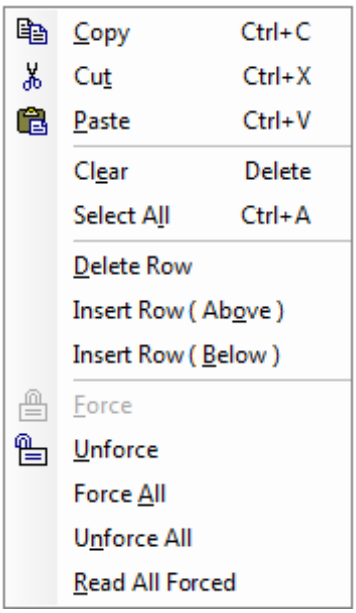
4.7.2 Мониторинг значения переменной

Вы можете следить за значение переменной по диаграмме состояния следующим образом:
★ Введите адрес памяти для наблюдения в [Address]
★ Войдите в состояние онлайн мониторинга с следующим образом:
Выполните команду [Debug] → [Online Monitor];
Нажмите на иконку на панели инструментов;
Используйте горячую клавишу F6
★ [Display Format] значения монитора может быть изменено в любое время.

4.7.3 Функция Force

Вы можете использовать функцию Force, чтобы изменить значение переменной в таких областях I, Q, M, V, AI, AQ, среди которых переменные в области I, Q , M и V могут быть изменены битом, байтом, словом или двойным байтом; переменные в AI области и AQ области могут быть изменены словом. Когда CPU перезагружается, все состояния Force будут отменены.
K5 позволяет максимально 32 переменных. Непосредственные команды не могут быть изменены принудительно.
В определенное время периода сканирования, одна переменная может иметь следующие возможности: внешний входной сигнал (I, AI) или результат программы пользователя (Q, AQ, M, V). Принципы оценки переменных заключается в следующем:
★ Переменные в M области и V области, значение Force имеет тот же приоритет выполнения программы.
★ Переменные в I и AI области, значение Force активно до появления внешнего значения входного сигнала.
★ Переменные в Q и AQ области, результат выполнения программы будет подавлять значение Force.

4.7.4 Контекстное меню



★ **Force:** Введите прямой адрес памяти в [Force Value] (Address) ПЛК.
★ **Cancel Force:** Отмена состояния Force одного элемента.
★ **Force All:** Установить всем значениям состояние Force [Force Values].
★ **All Cancel Force:** Отменить все состояния Force.
★ **Read All Forced:** Читать все переменные состояния Force и показать их в Variable Status Table.

4.7.5 Force и отмена Force

Вы можете назначить состояние Force или отменить состояние Force переменной в Variable Status Table следующим образом:
★ Введите прямой адрес памяти, что бы назначить Force, в [Address];
★ Выберите [Format] значения, и вы можете проверить его в любое время (опционально);
★ Ввод значения Force. Вы можете ввести круглое десятичное число и Kinco Builder настроит его соответствующим образом;
★ **Вы можете изменить Force состояние переменной следующим образом:**
★ Щелкните правой кнопкой мыши строку и выполните команду [Force];
★ Выберите строку и нажмите кнопку  в панели инструментов;
★ Выберите строку и выполните команду [Debug] → [Force];
★ **Вы можете отменить состояние Force непосредственно в строке [Address] следующим образом:**
★ Щелкните правой кнопкой мыши строку и выполните команду [Cancel Force];
★ Выберите строку и нажмите на иконку  в панели инструментов;
★ Выберите строку и выполнить команду [Debug] → [Cancel Force];
Variable Status Table является такой же, как Initial Data, пожалуйста, обратитесь к этой части.

4.8 Защита паролем

Kinco-K5 предусматривает защиту паролем, чтобы зашифровать CPU для ограничения доступа к определенным функциям. Если CPU в зашифрованном виде, потребуется ввести пароль, чтобы получить доступ к защищенным функциям. При этом, если введен правильный пароль, CPU разрешит выполнение соответствующих операций; если был введен неправильный пароль, CPU откажется выполнять соответствующие операции. Пароль действителен только для текущей операции. Если вы снова попытаетесь получить доступ к защищенным функциям, то вы должны ввести пароль еще раз.
Пароль сочетает буквы, цифры и знаки подчеркивания, с учетом регистра. Максимальная длина пароля составляет 8 бит.
Если вы обновили уровень защиты PLC до «3-й степени: Высокая защита" и поставил пароль или включили "Запретить Загрузить", программа пользователя в контроллере будет зашифрована.

4.8.1 Уровни защиты

Kinco-K5 предусматривает следующие 3 уровня защиты:
★ Уровень 1: Полный доступ. Нет ограничений для доступа ко всем функциям. Этот уровень по умолчанию.
★ Уровень 2: Частичный доступ. Пароль требуется при загрузке.
★ Уровень 3: Минимальный доступ. Пароль требуется во время скачивания и загрузки.

4.8.2 Как изменить пароль и уровень защиты

Выберите [PLC] > [Password...] в меню команд, чтобы открыть окно "Password". На следующем рисунке показано окно "Password" :

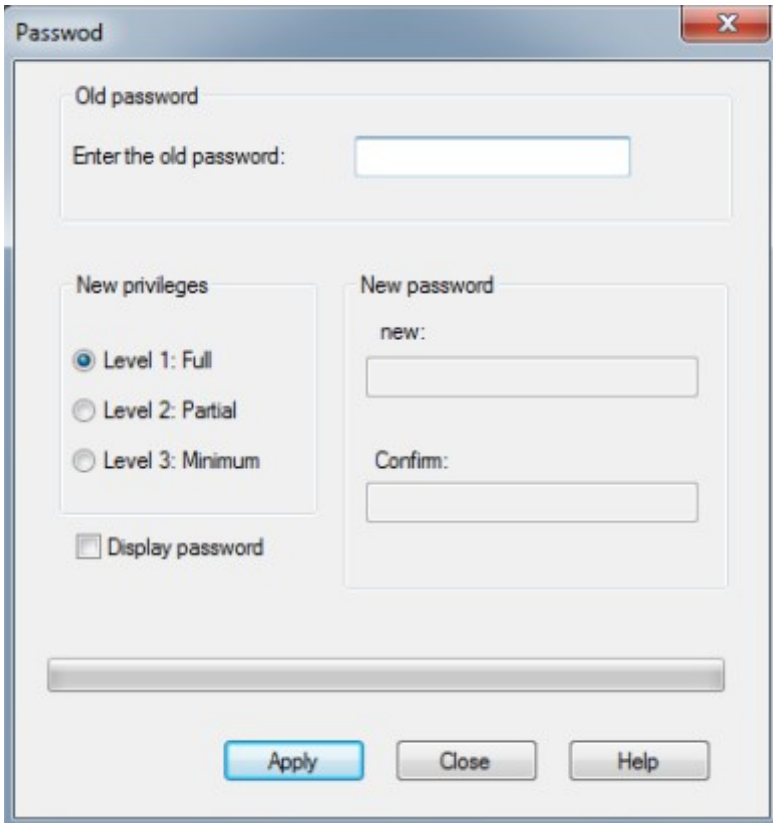


Рисунок. 4-16 Окно "Password"

■ **Old password**
Если подключенный CPU был с установленной защитой паролем, то старый пароль должен быть введён здесь для проверки. Если защита паролем не используется и никогда не была установлена, то просто оставьте поле редактирования пустым.
★ **New Privileges**
Здесь Вы можете задать новый уровень защиты и пароли для подключенного CPU.
★ **New Privileges:** Вы можете выбрать любой один из уровней: уровень 1, уровень 2 или уровень 3.
★ **New password:** Вы можете ввести новый пароль.
★ **Confirm:** Вы должны ввести новый пароль еще раз.
После окончания настройки, вы можете нажать на кнопку [Apply] для записи новых настроек в подключенном CPU, после этого новые настройки будут эффективными.

4.8.3 Как восстановить утраченный пароль

Если вы забыли пароль, вы должны очистить память CPU для дальнейшего его использования. Выберите [PLC] > [Clear …] в меню команд, чтобы очистить память CPU.
После очистки, все данные в CPU, включая программу пользователя, данные конфигурации и пароль, будут утеряны, и CPU восстановится до заводских настроек по умолчанию, за исключением RTC. При этом параметры связи будут следующие: станция № 1, скорость передачи данных составляет 9600, без контроля четности, 8 бит данных, 1 стоп-бит.

Глава V Как использовать Kinco Builder (программирование)

Kinco Builder в настоящее время поддерживает IL и LD языки программирования, а так же два редактора, для программирования: редактор IL и редактор LD. В этой главе будет подробно описано два редактора.

IEC61131-3 определяет три текстовых языка и три графических языка. Текстовые языки включают: Список команд (IL), Структурированный текст (ST) и Схемы последовательных функций (SFC, текстовый вариант); и графические языки включают: Релейная диаграмма (LD), Функциональная блок-схема (FBD) и схемы последовательных функций (SFC, графическая версия).

Kinco Builder в настоящее время предусматривает два редактора для программирования: редактор IL и редактор LD. Вы можете написать POU в языке IL или LD, то есть вы можете написать POU с помощью редактора IL или LD. С некоторыми ограничениями, POU написанная в одном редакторе можно просматривать и изменять в другом редакторе. Вы всего лишь выбираете [Project] > [IL] или [Project] > [LD] в меню команд, чтобы переключить редактор для текущего POU.

5.1 Программирование в IL

5.1.1 Обзор

IL является языком низкого уровня, который очень похож с языком ассемблера, он основан на схожих языках IL от известных производителей PLC по всему миру.
IL близок к машинному коду, поэтому является эффективным языком. IL очень подходит для опытных программистов. Иногда вы можете использовать IL, что бы решить те проблемы, которые вы не можете решить с помощью языка LD.

5.1.2 Правила

5.1.2.1 Инструкции

IL является линия-ориентированным языком. Программа IL состоит из последовательности команд. Каждая команда должна начинаться с новой строки и содержит оператор. Операнды являются необязательными, и разделяются запятыми или пробелами. Комментарий может быть введен в конце линии, используя скобки и звездочки. Пустые строки допустимы в списке команд.
На следующем рисунке показан типичный формат оператора IL:

```
Label:
Operator   Operands   (* Comment *)
```

Рисунок 5-1 Типичный формат оператора IL

★ **Label**
Необязательно. Переход используется для перехода к строке программы IL. В этом случае, метка перед линией назначения используется. Формат имени метки идентичен идентификатору.
★ **Operator**
★ **Operands**
Пожалуйста, обратитесь к инструкции настройки для подробного описания.
★ **Comment**
Необязательно. Только один комментарий допустим в линии; вложение не допускается.

Ниже пример:

```
(* Network 0 *)
begin:              (* метка, используется при переходе *)
LD      %I1.0
TP      T2, 168     (* если I1.0% истинно, то таймер T2 запускается. T2 является экземпляром TP. *)
```

5.1.2.2 Текущий результат

IL обеспечивает универсальный аккумулятор называемый "Current Result (CR)", а текущий результат логической операции сохраняется в CR.CR будет обновлен после выполнения каждого оператора, и он может действовать в качестве условия выполнения или один из операндов для следующего утверждения. Все операторы в Kinco Builder могут быть сгруппированы в соответствии с их влиянием на CR, как показано в следующей таблице. Пожалуйста, обратитесь к инструкции настройки для получения дополнительной информации.

Таблица 5-1 Группы операторов

Группа	Влияние на CR	Пример
C	Создаёт CR	LD, LDN
P	Установка CR, чтобы быть результатом операции	Бит логика, сравнение инструкции и т.д.
U	Оставляет CR без изменений	ST, R, S, JMP, и т.д.

Примечание: IEC61131-3 не определяет вышеперечисленные группы. В результате, эти группы в различных системах программирования могут быть различными.

5.1.2.3 Network

В Kinco Builder POU состоит из следующих частей:
★ Тип POU и имя POU
★ Объявление переменной части
★ Часть кода, содержащее инструкции

Network может быть принят в качестве основного сегмента кода; часть кода POU состоит из нескольких Network.
Network упрощает просмотр программы IL. Типичная Network включает в себя:
★ Метка Network
★ Комментарий Network
★ Инструкции

5.1.3 Редактор IL в Kinco Builder

Когда создается новая программа в языке IL, редактор IL будет готов к программированию; Если программа IL открывается, редактор IL также будет готов. Редактор IL показан следующим образом.

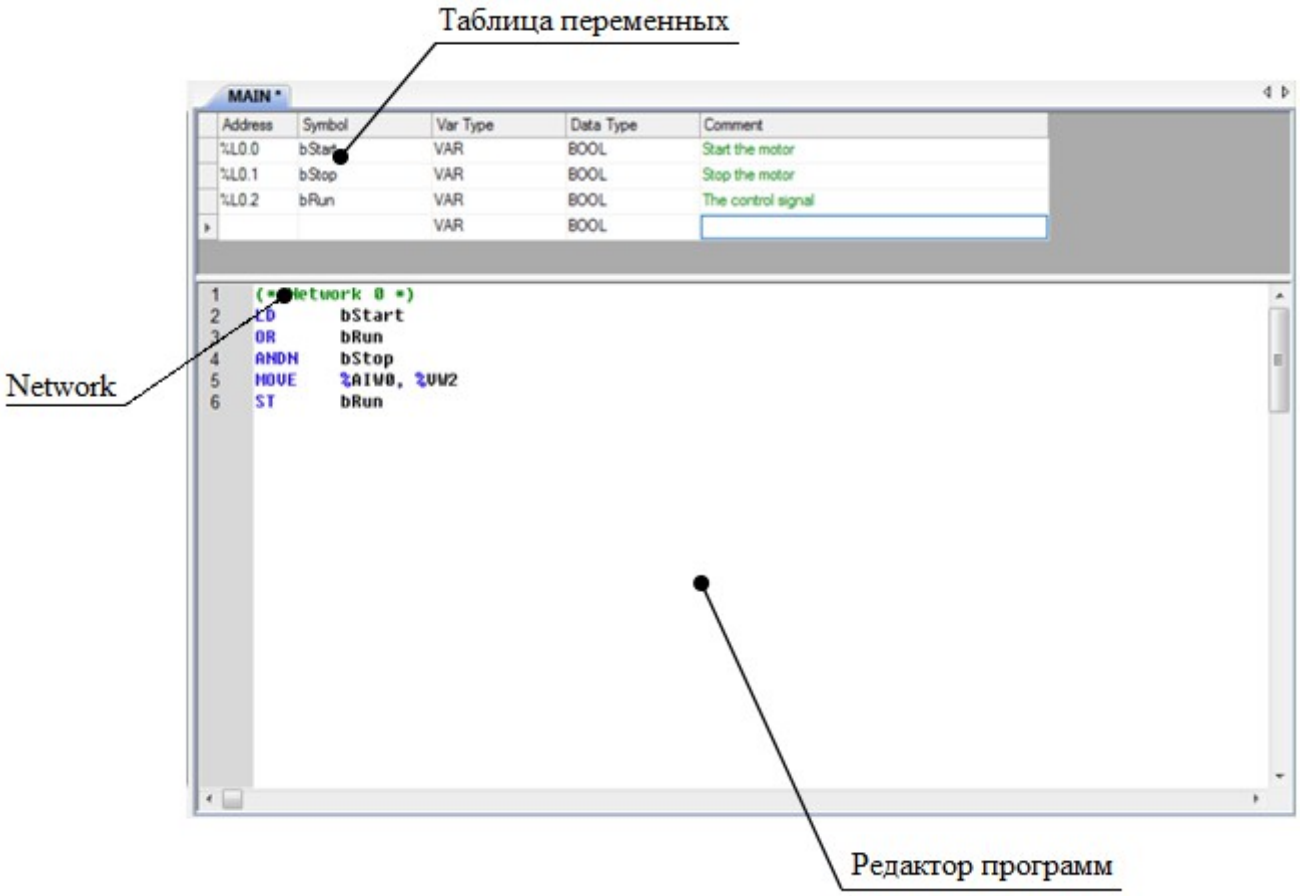


Рисунок 5-2 Редактор IL

Редактор IL состоит из двух частей:
• Таблица переменных: здесь вы можете объявить локальные переменные и параметры ввода / вывода программного модуля.
• Редактор программ: здесь можно редактировать программу управления.

5.1.3.1 Добавление сети

Используйте один из следующих способов, чтобы добавить сеть:
★ Используйте горячие клавиши Ctrl + Q
★ Щелкните правой кнопкой мыши редактор программы и выберите [Insert Network] на всплывающем меню.

5.1.3.2 Формат допустимых инструкций в Network

★ Там может быть только один оператор метка в сети. Например:

```
(* Network 0 *)
MRun:     (* Там может быть только одна метка оператора *)
```

★ Сеть может содержать некоторые заявления.
В "5.2.2.2 Current Result" мы делим все инструкции три группы ("C", "P" и "U").
Сеть должна начинаться с одной из инструкций в группе "C", и завершаться одной из инструкций в группе "P" или "U". Например:

```
(* Network 0 *)
LD   %M3.5   (* Начинается с инструкции LD *)
```

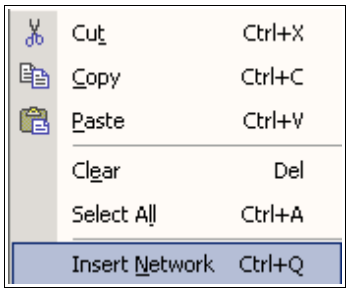
```
... ...          (* Можно ввести и другие инструкции *)
ST   %Q2.3       (* Конец с допустимой инструкцией *)
```

★ Сеть может содержать некоторые выписки меток и некоторые заявления.
Сеть должна начинаться с метки или одной из инструкций в группе "C", и завершаться одной из инструкций в группе "P" или "U". Например:

```
(* Network 0 *)
MRun:
LD   %M3.5       (* Начинается с инструкции LD *)
... ...          (* Можно ввести другие инструкции *)
ST   %Q2.3       (* Конец с допустимой инструкцией *)
```

5.1.3.3 Другие операции

Редактор IL может автоматически форматировать заявления. Он также может автоматически проверять заявления, и красный вопросительный знак (?) перед строкой будет означать, что в этой линии есть ошибка.
Редактор IL похож с текстовым редактором и поддерживает распространенные операции клавиатуры.
Все команды в меню [Edit] применимы в редакторе IL.
Щелкните правой кнопкой мыши на редакторе программ, появится следующее всплывающее меню:



★ Ввод заявления IL
Область редактирования программы IL схожа с текстовым редактором, где пользователи могут редактировать входные заявления и программу непосредственно. Он поддерживает основные операции клавиатуры, такие как Delete, Backspace и движения курсора UP/DOWN/LEFT/RIGHT.
Редактор IL может форматировать входные заявления автоматически, отображая ключевые слова синим цветом, и комментарии зеленым цветом.
Если пользователь переместит курсор к другой строке, редактор IL будет проверять синтаксис предыдущей строки. Красный (?) будет отображаться в начале строки.

★ Check/Delete/Copy/Cut/Paste
В редакторе IL, пользователи могут использовать все команды в меню [Edit], включая Check All, Copy, Cut, Paste и так далее. Также пользователи могут вызвать меню правой кнопкой мыши.
Выполнение команды **[Check All]** будет проверять весь текст в области редактирования. Переместите курсор или используйте клавиши вверх / вниз / влево / вправо нажав кнопку SHIFT, чтобы проверить пройденный текст. Проверенная область отобразится на чёрном фоне и текст будет выделен.
Удалить проверенное содержимое можно с помощью кнопки Delete или команды **[Delete]**.
Используйте горячие клавиши Ctrl + C или выполните команду **[Copy]** для копирования проверенного содержимого в буфер обмена Windows. Скопированное содержимое может быть вставлено в любом текстовом редакторе, например, Window notebook или документ Word.
Операция Cut эквивалентна Copy и Delete проверенного содержимого. Выполните команду [Cut] или используйте горячую клавишу Ctrl + X для того чтобы вырезать проверенное содержимое в буфер обмена Windows.

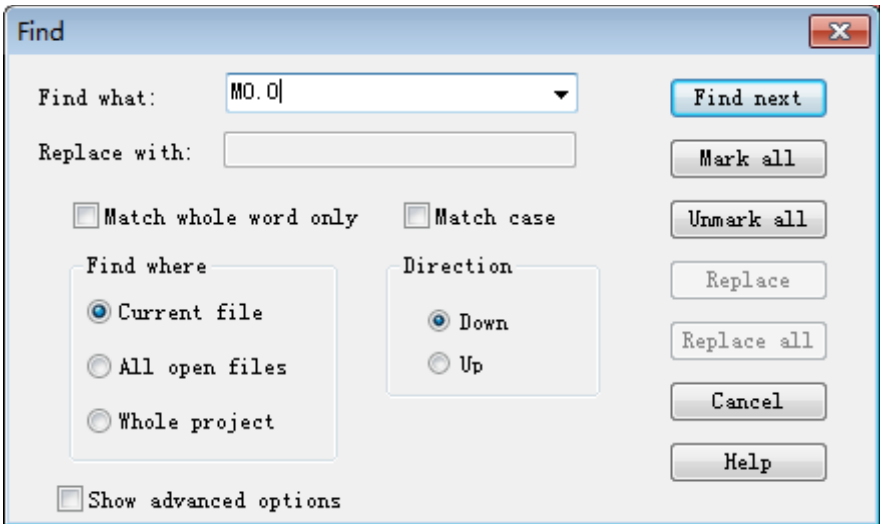
После операции копирования или вырезания, переместите курсор на нужное место вставки, а затем выполнить команду [Paste] или используйте горячую клавишу Ctrl + V, чтобы вставить содержимое буфера обмена в текущую позицию курсора.

★ Find/Replace

IL редактор поддерживает стандартные команды **Найти** и **Заменить**

• Найти

Используйте горячую клавишу Ctrl + F или кликните меню **Edit → Find**..., появится окно **Find**.

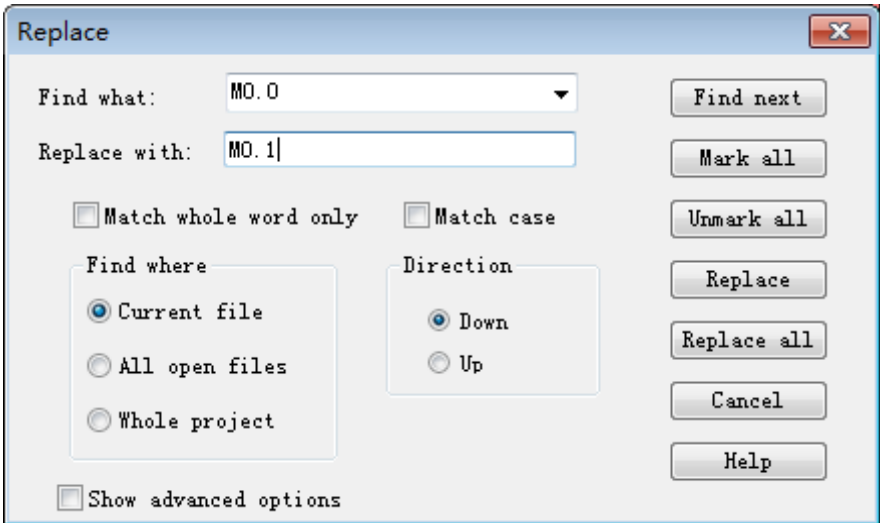


Введите символ, который вы хотите найти, в окне **Find what**, а затем нажмите кнопку **Find next**, найденное содержание будет выделено синим прямоугольником.

Другие функции в этом окне такие же, как в стандартной Windows.

• Заменить

Используйте Ctrl + R или откройте меню **Edit → Replace**, появится окно **Replace**.



Введите символ, который должен быть заменен в **Find what** и введите заданный символ в поле **Replace with**, затем нажмите кнопку **Replace**, редактор найдет символ, который должен быть заменён, и заменить его на заданный символ. Если вы нажмете кнопку **Replace all**, редактор найдет все символы, которые должны быть заменены и заменить их все на заданный символ.

Другие функции в этом окне такие же, как в стандартной Windows.

5.1.3.4 Онлайн мониторинг

После выбора [Debug] > [Monitor] в меню команд, редактор IL изменится на режим онлайн мониторинга. В этом режиме вы не можете редактировать программу.
В режиме онлайн мониторинга, исходная область редактора программ разделена на две части вертикальной линией в середине, в правом столбце отображается программа, а в левой колонке, отображаются соответствующие переменные. Чтобы изменить размеры столбцов, переместите курсор на вертикальную линию, появиться знак  , затем зажмите левую клавишу мыши и перетащите линию влево или вправо.

5.1.3.5 Пример

```
(* NETWORK 0 *)
LDN     %M0.0
TON     T0, 1000    (* Запуск T0 с выходом T1, время: 1000 * 1 мс *)
ST      %M0.1
LD      %M0.1
TON     T1, 1000    (* Запуск T1 с выходом T0, время: 1000 * 1 мс *)
ST      %M0.0
LD      %M0.1
ST      %Q0.0    (* Выход прямоугольный импульс с периодом 2с в %Q0.0 *)
```

5.1.4 Преобразование программы IL в программу LD

Вы можете выбрать [Project] > [LD] в меню команд для изменения редактора IL в редактор LD; в то же время, текущая программа IL будет преобразована в формат LD.
Не все программы IL могут быть преобразованы в формат LD; для успешного преобразования программа должна удовлетворять следующим условиям:
★ Нет ошибок в исходной программе IL.
★ Программа источник IL должна быть строго в соответствии со следующими правилами:
★ Каждая сеть должна начинаться с одной из инструкций в группе "C"; или должен быть только один оператор метки в сети.
★ Инструкция, в которой начинается сеть, следует использовать только один раз в сети.
★ Каждая сеть должна заканчиваться одной из инструкций в группе "P" или "U".

5.1.5 Отладка и мониторинг программы

5.1.5.1 Онлайн монитор IL

Вы можете войти в Online Status Monitor с редактором IL любым из способов, представленных ниже:
★ Выполните команды [Debug] → [Online Monitor];
★ Щелкните по значку  на панели инструментов;
★ Горячая клавиша F6.
В режиме Online Monitor область будет разделена на две части; правая колонка — программа, а левая колонка — соответствующие переменные. Колонки разделены линией, которую можно перемещать, для изменения пространства каждого столбца.

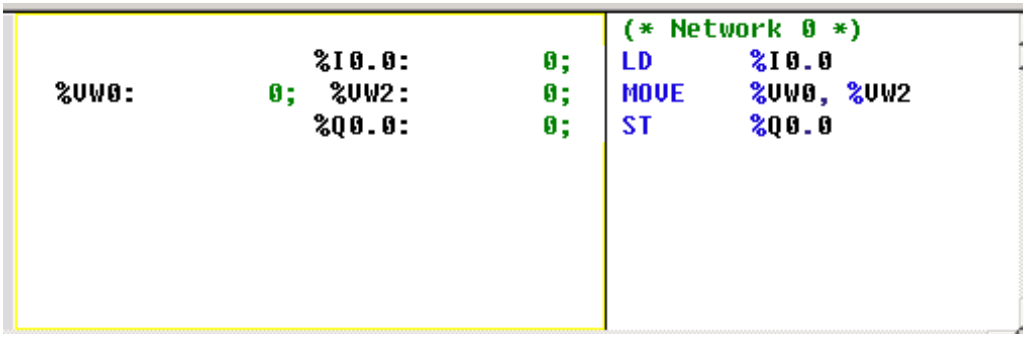
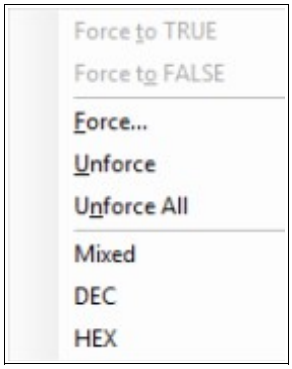


Рисунок 5-3 Вид IL в режиме Online Monitor

Примечание: Программа не может быть отредактирована в режиме Online Monitor.

5.1.5.2 Функция Force конкретных переменных

Вы можете найти подробное описание в разделе ”4.7.3 Функция Force”.
В режиме онлайн мониторинга IL, вы можете выполнить Force или отменить Force для определённых переменных в редакторе IL; щелкните правой кнопкой мыши любую переменную, появится меню (если переменная без On / Off, команды [Force to TRUE] и [Force to FALSE], будут недействительны):



★ Force to be TRUE: Принудительное назначение переменной (On / Off переменной) в 1 (TRUE)
★ Force to be FALSE: Принудительное назначение переменной (On / Off переменной), в 0 (FALSE)
★ Force to be ...: Если вы выберите эту команду, появиться диалоговое окно.

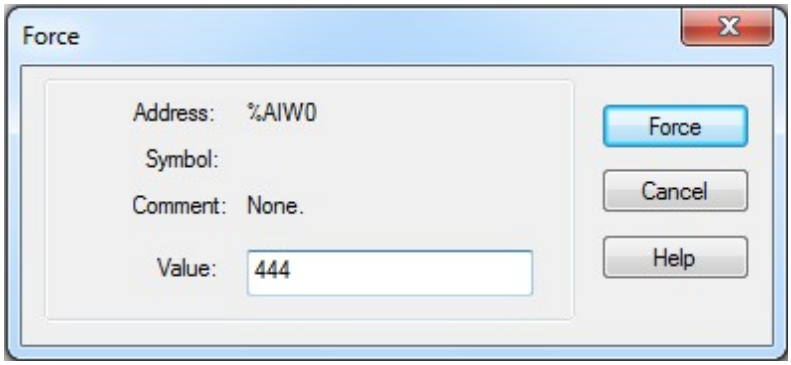


Рисунок 5-4 Диалоговое окно Force

Вы можете ввести это значение в поле [Force Value] и нажать кнопку [Force]. Вы можете обратиться к пункту «3.5 Переменные».

5.2 Программирование в LD

Некоторые определения взяты из стандарта IEC 61131-3.

5.2.1 Обзор

LD (Релейная диаграмма) является одним из наиболее часто используемых графических языков в программировании ПЛК. Язык LD основан на традиционной релейной логике. Кроме того, язык LD позволяет использовать определяемые пользователем функциональные блоки и функции, и поэтому могут быть использованы в иерархическом порядке. LD позволяет программировать с помощью стандартных графических символов, поэтому является легким в освоении и использовании. LD показывает большие преимущества в работе с булевой логикой. Ниже приведен простой программный сегмент в LD.

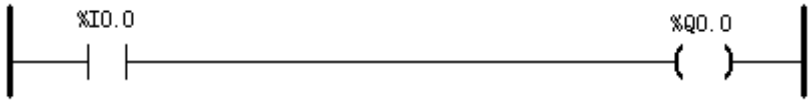


Рисунок 5-5 Образец в LD

5.2.2 Сеть

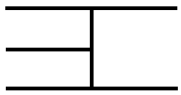
Когда вы пишете программу в LD, вы можете использовать стандартные графические символы и организовав их построить цепь логики. LD цепь должна быть ограничена слева вертикальной линией, известной как левая шина питания, и справа вертикальной линией, известной как правая шина питания. Состояние левого шины считается вкл. всё время. Для правой шины состояние не определено.

5.2.3 Стандартные графические символы

★ Линии связи
Горизонтальные связи и вертикальные связи используются в LD, соответствующие последовательному соединению и параллельному соединению, соответственно. Состояние линии связи может быть включено или выключено, соответствующее логическому значению 1 или 0, соответственно. Условие состояния связи должно быть аналогичным состоянию питания.

Таблица 5-2 Элементы линий связи

Символ	Наименование	Описание
	Горизонтальная связь	Горизонтальный элемент связи изображается как горизонтальная линия. Она передает состояние элемента находящегося слева на элемент находящийся справа.
	Вертикальная связь (С прикрепленными горизонтальными связями)	Элемент вертикальной подсистемы должен состоять из вертикальной линии, пересекающейся с одним или несколькими горизонтальными элементами связи на каждой стороне. Состояние вертикальной подсистемы должно представлять включительно OR из ON состояний горизонтальных связей с левой стороны, то есть состояние вертикальной подсистемы должно быть: - OFF, если состояние всех присоединенных горизонтальных

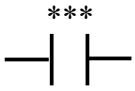
		связей слева от нее выключены; - ON, если состояние одного или более из прилагаемых горизонтальных звеньев слева от нее ON. Состояние вертикальной связи должно быть скопировано на все подключенные горизонтальные связи справа от нее.

★ Контакт

Контакт — это элемент, который придает состояние горизонтальной связи с правой стороны, которая равна логическому значению, из состояния горизонтального звена с левой стороны с соответствующей функцией ассоциированной логической переменной.

Контакт не изменяет значение связанной булевой переменной.

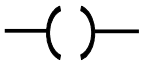
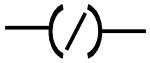
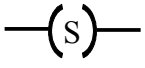
Таблица 5-3 Контакты

Символ	Наименование	Описание
*** 	Нормально открытый контакт	Состояние левой связи копируется в правую связь, если состояние ассоциированного булевой переменной (обозначено "***") является ON. В противном случае, состояние правой связи выключено.
*** 	Нормально закрытый контакт	Состояние левой связи копируется в правую связь, если состояние связанной булевой переменной OFF. В противном случае, состояние правого связи выключено.

★ Катушка

Катушка записывает состояние левой связи в соответствующую логическую переменную.

Таблица 5-4 Катушки

Символ	Наименование	Описание
*** 	Катушка	Состояние левой связи копируется в соответствующую логическую переменную и правую связь.
*** 	Инвертированная катушка	Обратное состояние левой связи копируется в соответствующей логической переменной, то есть, если состояние левой связи является OFF, то состояние соответствующего переменной ON, и наоборот.
*** 	SET (запирать) катушка	Соответствующая логическая переменная устанавливается в состояние ON, когда левая связь находится в состоянии ON, и остаётся установленной до сброса с помощью катушки RESET.
*** 	RESET (отпирать) катушка	Соответствующая логическая переменная сбрасывается в состояние OFF, когда левая связь находится в состоянии ON, и остается сброшенной, пока не установиться катушка SET.

★ Выполнение элементов управления

Передача управления программы в языке LD должна быть представлена графическими элементами показанные в таблице.

Таблица 5-5 Выполнение элементов управления

Символ	Наименование	Описание
├─(1)─ <RETURN>	Условие возврата	Выполнение программы будет возвращено обратно к началу вызывающей ссылки равной 1 (TRUE), и будет продолжаться в обычном режиме, когда логический вход = 0 (FALSE).
├─ >> Label	Безусловный переход	Выполнение программы должно быть передано в назначенную сетевую метку безусловно.
├─(1)─ >>Label	Условие перехода	Выполнение программы должно быть передано в назначенную сетевую метку, когда значение ссылки равно 1 (TRUE), и будет продолжать в обычном режиме, когда логический вход = 0 (FALSE).

Примечание: (1) графическое отображение логического результата.

★ Функции и функциональные блоки.

Функция или функциональный блок должны быть представлены прямоугольным блоком, и его фактические переменные соединения могут быть показаны путем записи соответствующей переменной вне блока, прилегающего к формальному имени переменной на внутренней стороне. По крайней мере, один Логический вход и один Логический выход должны быть показаны на каждом блоке, чтобы обеспечить поток сигнала через блок.

Функция должна иметь логическое ввод с именем EN и логический выход под именем ENO. EN используется для управления выполнением этой функции. Если EN истина, то функция будет выполнена и ENO будет установлен в качестве истины. Если EN ложно, функция не будет выполнена и ENO будет установлен как ложное.

Рисунок 5-6 Функции и функциональные блоки

5.2.4 Редактор LD в Kinco Builder

Когда создается новая программа на языке LD, редактор LD будет готов к программированию; если программа LD открывается, редактор LD также будет готов. Редактор LD показан следующим образом.

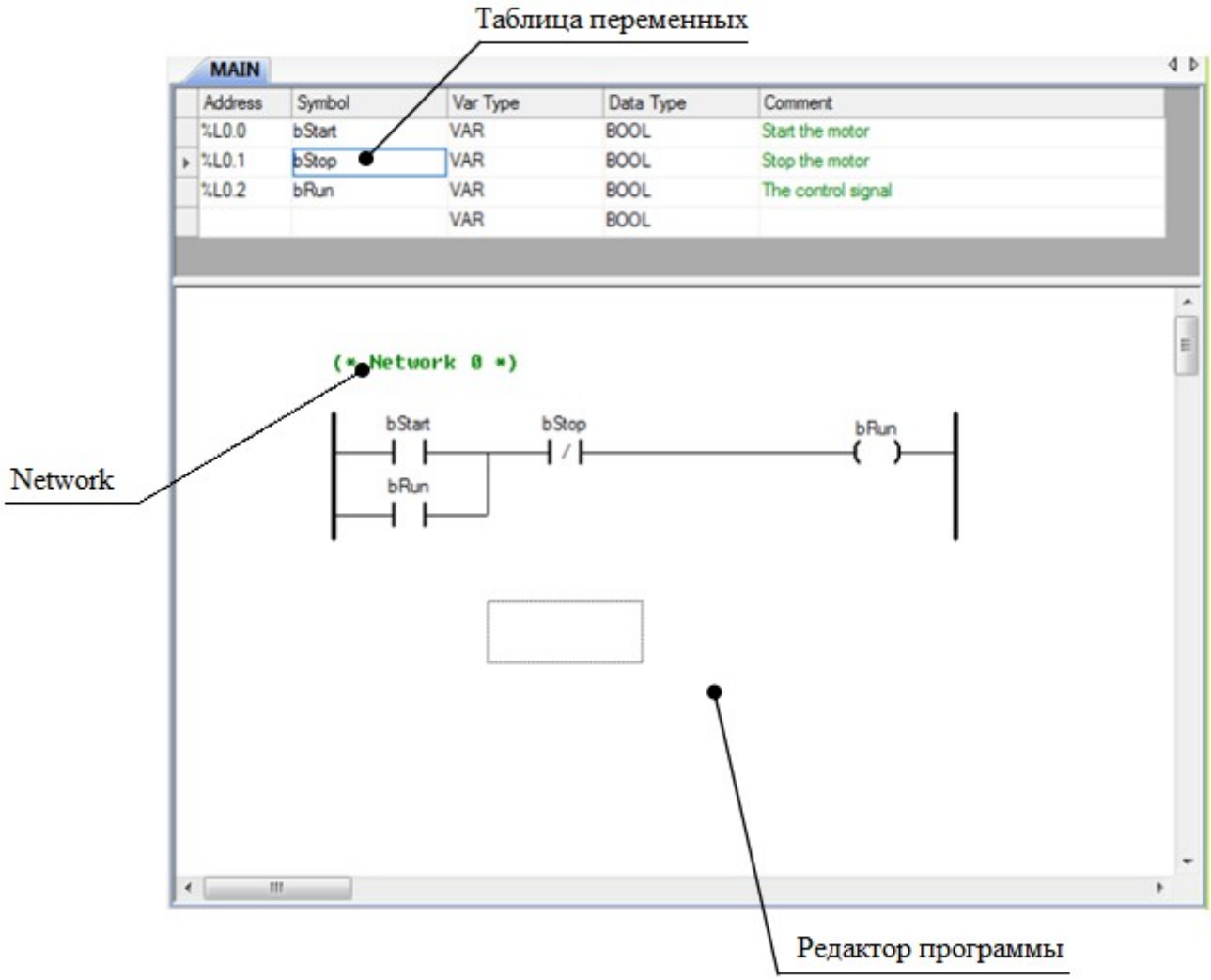


Рисунок 5-7 Редактор LD

5.2.4.1 Пределы LD программы

Максимум 200 Network допускается в программе LD.
Вы можете рассматривать окно Редактора Программы, как поле, разделенное на клетки. Внутри этого поля, Network можно продлить максимально на 32 ячейки по горизонтали и 16 ячеек по вертикали. Таким образом, максимальное количество элементов в горизонтальном положении в Network составит: если имеется только катушки и контакты, до 31 контактов и 1 катушку; если только с функциями / функциональными блоками, до 12 блоков, 1 катушки и 1 контакта. Кроме того, в сети, ветви не должны превышать 16 в параллельном соединении.
Параллельное соединение двух или более независимых функций / функциональных блоков запрещено.

5.2.4.2 Общие операции

Редактор LD поддерживает общие операции с использованием мыши:
★ Щелкните элемент, который должен быть выбран, курсором мыши (на элементе появится прямоугольная рамка);
★ Дважды щелкните элемент, появиться всплывающее диалоговое окно свойств, в котором вы сможете изменить свойства элемента;
★ Щелкните правой кнопкой мыши элемент, появиться всплывающее контекстное меню, в котором вы

сможете выбрать команду меню, чтобы выполнить соответствующую функцию.
Кроме того, редактор LD поддерживает операции клавиш:
★ Использование клавиш UP, DOWN, LEFT и RIGHT со стрелками для перемещения курсора.
★ Нажмите ENTER, чтобы выбрать параметр элемента для ввода.
★ Нажмите клавишу Del, чтобы удалить элемент, на котором находится курсор.
★ Существует комбинация клавиш, соответствующих каждой команде меню.

5.2.4.3 Шаги программирования в LD

Следующее описание будет сосредоточено на операции мыши.
★ Используйте один из следующих способов, чтобы добавить сеть:
★ Выберите [LD] > [Сеть] в меню команд
★ Нажмите на иконку на панели инструментов
★ Используйте сочетание клавиш Ctrl + W
★ Щелкните правой кнопкой мыши любой элемент, и выберите команду [Network] на всплывающем меню.
Вновь добавленная сеть выглядит следующим образом.

Рисунок 5-8 Новая сеть

Дважды щелкните сетевую метку, откроется диалоговое окно, и вы можете ввести некоторые комментарии для описания этой сети.

★ Когда вы добавляете новую инструкцию, ее переменные изначально обозначаются красными знаками вопроса (????). Эти вопросительные знаки указывают, что переменная не определена, и вы должны определить её при составлении программы.
При нажатии на переменную, появится окно для указания области переменной, и вы можете ввести переменную или константу в этом поле. Вы также можете нажать кнопку ENTER для выбора области переменной элемента, на которой находится курсор. LD редактор автоматически форматирует прямой адрес после ввода, поэтому вам не нужно вводить процентную отметку, если вы введете прямой адрес.
Кроме того, вы можете дважды щелкнуть элемент контакт или катушку, чтобы открыть диалоговое окно свойств, чтобы изменить его тип и параметры. На следующем рисунке показано диалоговое окно свойств контакта.

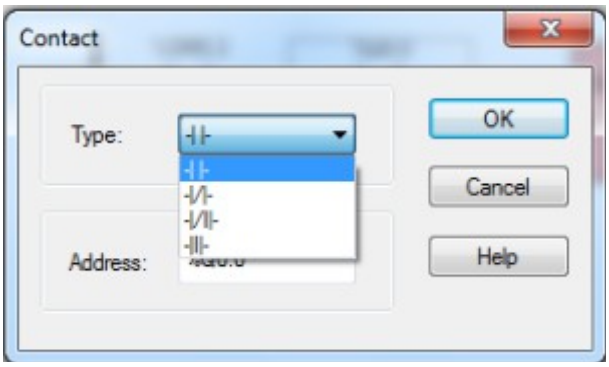
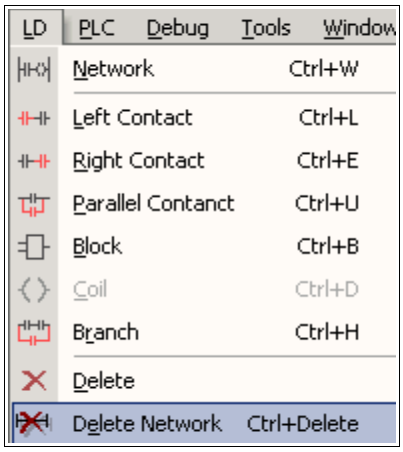


Рисунок 5-9 Диалоговое окно настройки контакта

★ Нажмите элемент и выберите его в качестве эталона, а затем продолжите, чтобы добавить другие

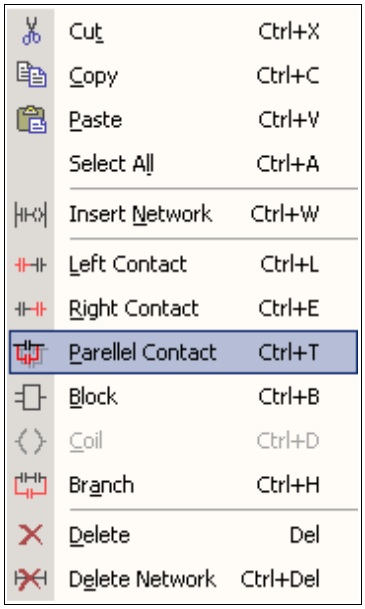
элементы, используя один из следующих способов:

★ Используйте [LD] в меню команд или горячие клавиши:



Left Contact: Добавление контакта с левой стороны опорного элемента.
Right Contact: Добавление контакта с правой стороны опорного элемента.
Parallel Contact: Добавить контакт параллельно опорному контакту.
Block: Добавить последовательный блок (функция / FB / подпрограмма).
Coil: Добавить катушку параллельно опорной катушки.
Branch: Рисование ветки параллельно с другими элементами.
Delete: Удаление выбранного элемента.
Delete Network: Удалить сеть, в которой находится выбранный элемент.

★ Используйте команды контекстного меню:
Щелкните правой кнопкой мыши по элементу, появиться следующее контекстное меню. Пожалуйста, обратитесь к вышеуказанным описанием.



★ Используйте кнопки на панели инструментов:

Что бы добавить соответствующий элемент, нажмите соответствующую кнопку на панели инструментов.

★ Двойной щелчок в дереве LD инструкции:
В дереве инструкции LD, разверните дерево, что бы найти нужную инструкцию и дважды щелкните на ней, после чего инструкция должна появиться в редакторе LD.
Предположим, что добавлен блок "MOVE". Тогда сеть выглядит следующим образом:

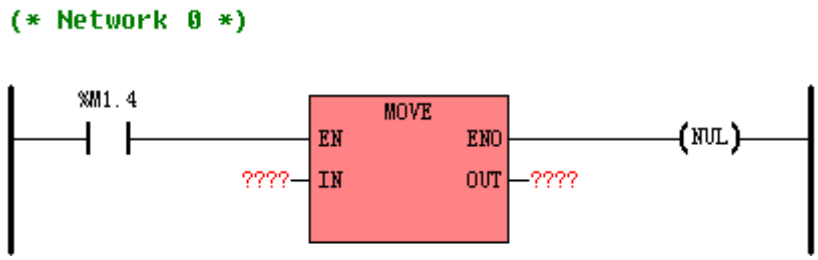


Рисунок 5-10 Добавление других элементов

★ Продолжайте использовать мышь или клавишу ENTER для выбора области переменной, чтобы изменить переменные новых элементов. Кроме того, вы можете двойным щелчком, на блочных элементах в программе, открыть диалоговое окно параметров для изменения свойств блока.

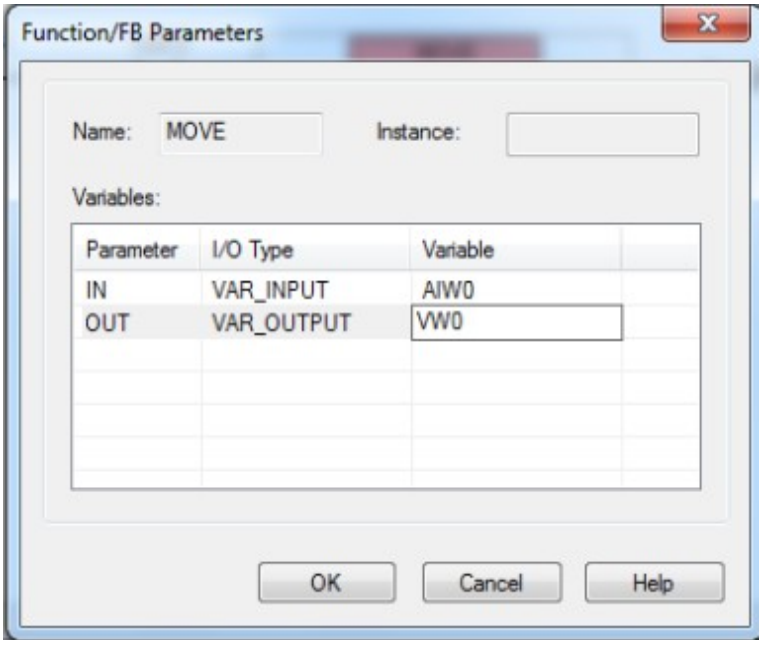


Рисунок 5-11 Диалоговое окно параметров блока

Вы можете дважды щелкнуть любую переменную в списке [Variable], чтобы изменить её, а затем нажмите клавишу Enter, чтобы подтвердить ввод. Кроме того, вы можете использовать клавиши со стрелкой вверх или вниз для выбора переменной, и нажать клавишу Enter, чтобы начать редактирование, затем нажмите кнопку ENTER, чтобы подтвердить ввод.
Kinco Builder будет строго проверять синтаксис вашей печати, неправильная переменная должна быть отказана.
Измененная сеть показана следующим образом:

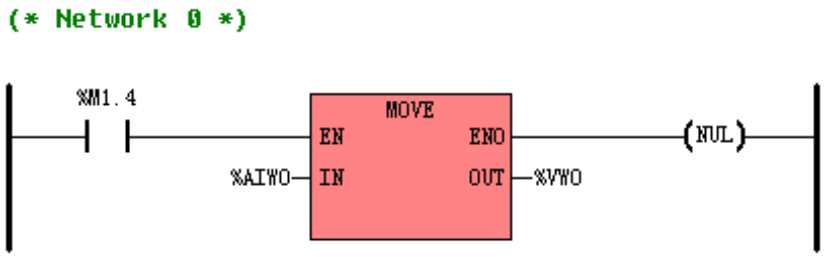


Рисунок 5-12 Измененная сеть

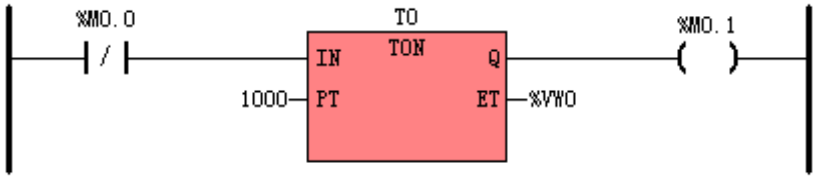
★ После этого сеть будет завершена, продолжайте добавлять и изменять новые сети, пока эта POU не будет закончена.
При добавлении новой сети, если текущая метка сети выбрана качестве опорной, то новая сеть должна быть добавлена выше текущей сети; в противном случае новая сеть должна быть добавлена ниже текущей сети. Здесь текущая сеть означает сеть, в которой находится выбранный элемент.

5.2.4.4 Онлайн мониторинг

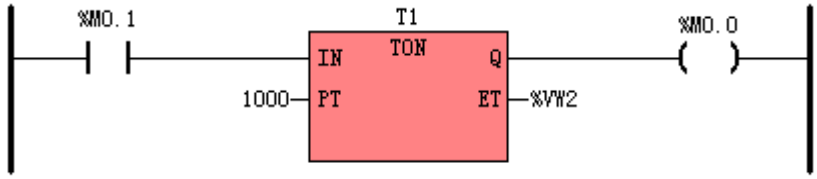
После выбора [Debug] > [Monitor] в меню команд, редактор LD перейдёт в режим онлайн мониторинга. В этом режиме все состояния данных PLC отображаются в окне редактора LD, и вы не можете редактировать программу.

5.2.4.5 Пример

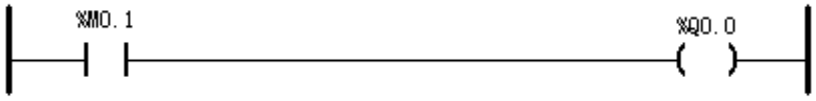
(* Network 0 *)
(* Запуск T0 с выходом T1, время: 1000 * 1 мс *)



(* Network 1 *)
(* Запуск T1 с выходом T0, время: 1000 * 1 мс *)



(* Network 2 *)
(* Выход прямоугольный импульс с периодом 2с в %Q0.0 *)



5.2.5 Мониторинг и отладка программы

5.2.5.1 Онлайн монитор LD программы

Примечание: состояние онлайн монитора не допускает редактирование программы.

Вы можете войти в состояние онлайн монитора, если редактор LD открыт.
★ [Debug] → [Online Monitor]
★ Нажмите на значок .
★ Горячая клавиша F6.

В состояние онлайн монитора значения переменных показаны в виде:

★ %I0.0 Контакт отключен, %I0.0 Контакт подключен

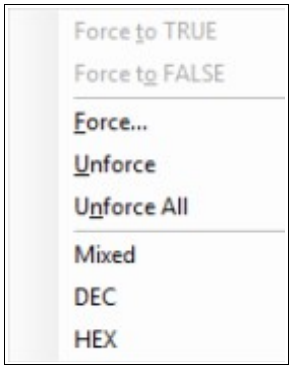
★ %M1.0 Катушка отключена, %M1.0 Катушка подключена

★ Переменные без ON / OFF будут показаны с правой стороны.
Пример:

Примечание: желтый замок означает состояние Force.

5.2.5.2 Функция Force конкретных переменных

Вы можете найти подробное описание в разделе ”4.7.3 Функция Force”.
В режиме онлайн мониторинга LD, вы можете выполнить Force или отменить Force для определённых переменных в редакторе LD; щелкните правой кнопкой мыши любую переменную, появится меню (если переменная без On / Off, команды [Force to TRUE] и [Force to FALSE], будут недействительны):



★ Force to be TRUE: Принудительное значение переменной (On / Off переменной) в 1 (TRUE)
★ Force to be FALSE: Принудительное значение переменной (On / Off переменной), в 0 (FALSE)
★ Force to be ...: Если вы выберите эту команду, появиться диалоговое окно.

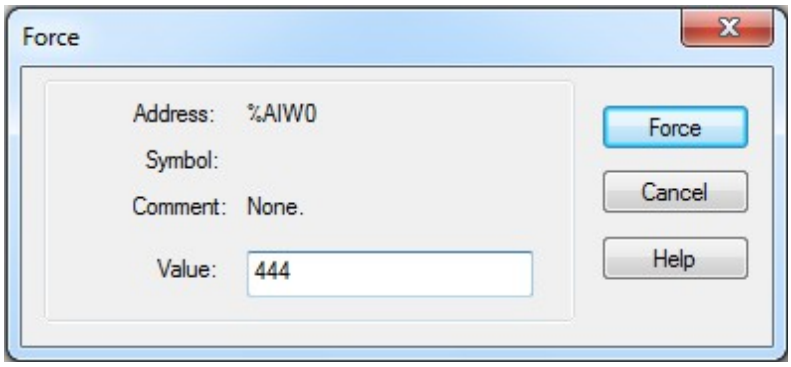


Рисунок 5-13 Диалоговое окно Force

Вы можете ввести это значение в поле [Force Value] и нажать кнопку [Force]. Вы можете обратиться к пункту «3.5 Переменные».

Глава VI Набор команд Kinco-K5

Набор инструкций Kinco-K5 согласуется с IEC 61131-3 для программирования основных инструкций и большинства стандартных функций / функциональных блоков. Кроме того, некоторые нестандартные инструкции доступны для различных пользователей и фактических потребностей приложений.

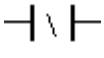
6.1 Основная информация

В этой главе представлено подробное описание и конкретные примеры применения всех инструкций. Будут описаны инструкции LD и IL.
Для значений LD, *EN* и *ENO* операнды не описаны в следующих разделах, потому что они являются одинаковыми для всех команд. *EN* и *ENO* связаны с потоком энергии. *EN* (Enable) является входом BOOL для большинства блоков, и поток энергии должен быть действителен на этом входе, что бы этот блок выполнялся. *ENO* (Enable Out) является выход BOOL для большинства блоков; если блок получает поток сигнал на входе *EN*, то блок выполняется правильно, и ENO устанавливается равным "1", и передает сигнал к следующему элементу, в противном случае сигнал здесь прекращается.
Для IL, как указано в пункте «5.1.2.2 Текущий результат» в руководстве по программному обеспечению, CR будет обновляться после каждого выполнения оператора, и он может действовать в качестве условия выполнения или один из операндов для следующего утверждения. Это описание подробное, а сокращённые группы операторов, используются в этой главе.

6.2 Битовые логические команды

6.2.1 Стандартный Контакт

★ Описание

	Название	Использование	Группа	
LD	Нормально открытый контакт	*bit* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Нормально закрытый контакт	*bit* 		
IL	LD	LD *bit*	C	
	AND	AND *bit*	P	
	OR	OR *bit*		
	LDN	LDN *bit*	C	
	ANDN	ANDN *bit*	P	
	ORN	ORN *bit*		

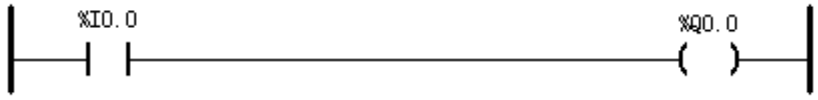
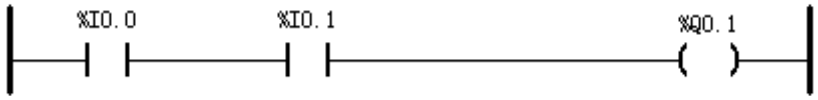
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
bit	Вход	BOOL	I, Q, V, M, SM, L, T, C, RS, SR, constant

★ LD
Когда бит равен 1, нормально открытый контакт замкнут (включен) и поток мощности передается на

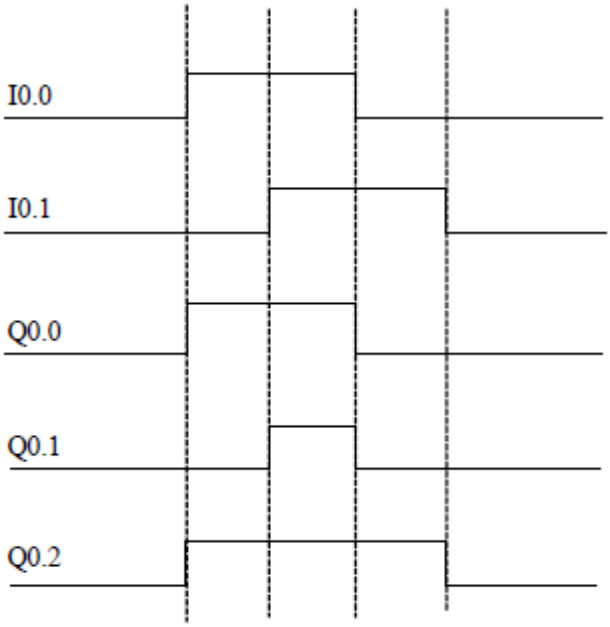
следующий элемент.
Когда бит равен 0, нормально замкнутый контакт замкнут (включен), и сигнал передается на следующий элемент.
★ IL
Нормально разомкнутые контакты представлены инструкциями LD, AND и OR.
Инструкция LD загружает бит и задает CR, равную результату.
Инструкция AND использует AND бит с CR, и устанавливает CR равную результату операции.
Инструкция OR использует OR бит с CR и устанавливает CR равную результату операции.

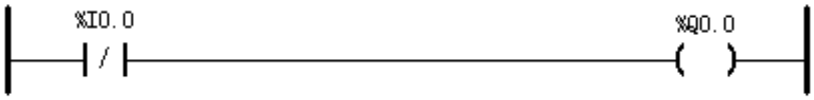
Нормально замкнутые контакты представлены в LDN, ANDN и ORN инструкции.
Инструкция LDN загружает логический NOT из битового значения, и устанавливает CR, равную результату операции.
Инструкция ANDN использует логические AND NOT битового значения с CR, и устанавливает CR, равную результату операции.
Инструкция ORN использует логические OR NOT битового значения с CR, и устанавливает CR равную результату операции.

★ Примеры

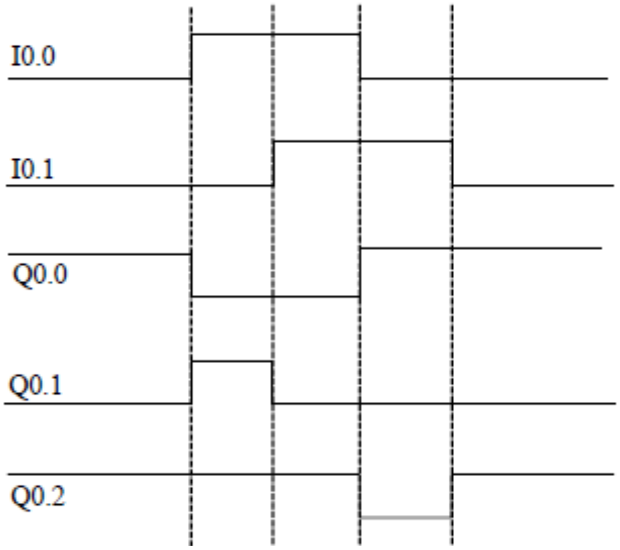
LD	IL
	LD %I0.0 ST %Q0.0
	LD %I0.0 AND %I0.1 ST %Q0.1
	LD %I0.0 OR %I0.1 ST %Q0.2

Временная диаграмма



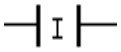
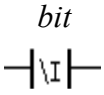
LD	IL
	LDN %I0.0 ST %Q0.0
	LD %I0.0 ANDN %I0.1 ST %Q0.1
	LD %I0.0 ORN %I0.1 ST %Q0.2

Временная диаграмма



6.2.2 Непосредственный контакт

★ Описание

	Название	Использование	Группа	
LD	Нормально открытый непосредственный контакт	*bit* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Нормально закрытый непосредственный контакт	*bit* 		
IL	LDI	LDI *bit*	C	
	ANDI	ANDI *bit*	P	
	ORI	ORI *bit*		
	LDNI	LDNI *bit*	C	
	ANDNI	ANDNI *bit*	P	
	ORNI	ORNI *bit*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
bit	Вход	BOOL	I (CPU body)

Когда непосредственная команда выполняется, он сразу же получает физическое значение входного канала, а соответствующий регистр ввода изображения не обновляется.
Непосредственные команды могут быть использованы только для каналов DI на CPU, и не зависят от времени входного фильтра настроенного в [Hardware].
В отличие от стандартного контакта, непосредственный контакт не зависит от цикла сканирования для обновления и поэтому он может реагировать на входной сигнал быстрее.

★ LD

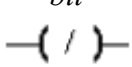
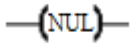
Когда физическая входная величина (бит) равен 1, нормально разомкнутый непосредственный контакт замкнут (on) и сигнал передается на следующий элемент.
Когда физическая входная величина (бит) равен 0, нормально замкнутый непосредственный контакт закрыт (on) и сигнал передается на следующий элемент.

★ IL

Нормально открытый непосредственный контакт представлен инструкциями LDI, ANDI и ORI.
Инструкция LDI загружает значение физического входа (бит) и устанавливает CR, равную результату.
Инструкция ANDI используется для физического значения AND входного сигнала (бит) с CR и устанавливает CR равным результату работы.
Инструкция ORI используется для физическая величина OR входного сигнала (бит) с CR и устанавливает CR равным результату работы.
Нормально замкнутый непосредственный контакт представлен в LDNI, ANDNI и ORNI инструкции.
Инструкция LDNI загружает логическое значение NOT физического входного сигнала (бит) и устанавливает CR, равную результату операции.
Инструкция ANDNI используется для AND логическое NOT физической величины входного сигнала (бит) с CR и устанавливает CR равным результату работы.
Инструкция ORNI используется для OR логическое NOT физического значения входного сигнала (бит) с CR и устанавливает CR равным результату работы.

6.2.3 Катушка

★ Описание

	Название	Использование	Группа	
LD	Катушка установки	*bit* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Катушка сброса	*bit* 		
	Нулевая катушка			
IL	ST	ST *bit*	U	
	STN	STN *bit*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
bit	Выход	BOOL	Q, V, M, SM, L

■ LD

Инструкция Катушка записывает сигнал в регистр выходного изображения для бита.

Инструкция Инвертированная Катушка записывает инверсию сигнал в регистр выходного изображения для бита.

Функция сброса катушки: если сигнал равен 1, регистр изображения для выхода бита установлен равным 0, в противном случае регистр остается неизменным.

Функция набора катушки: если поток сигнал 1, регистр изображения выхода для бита устанавливается равным 1, в противном случае регистр остается неизменным.

Функция Нулевая катушка для обозначения конца сети, эта инструкция только для облегчения в программировании, и не выполняет какую-либо конкретную операцию.

■ IL

Катушки представлены с ST, STN, R и S инструкциями.

Инструкция ST пишет CR в регистр выходного изображения для бита.

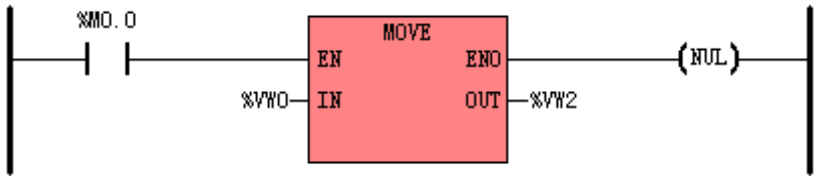
Инструкция STN пишет обратное CR в регистр выходного изображения для бита.

Функция инструкции R: если CR равен 1, регистр выходного изображения для бита устанавливается равным 0, в противном случае регистр остается неизменным.

Функция инструкции S: если CR равен 1, регистр выходного изображения для бита устанавливается равным 1, в противном случае регистр остается неизменным.

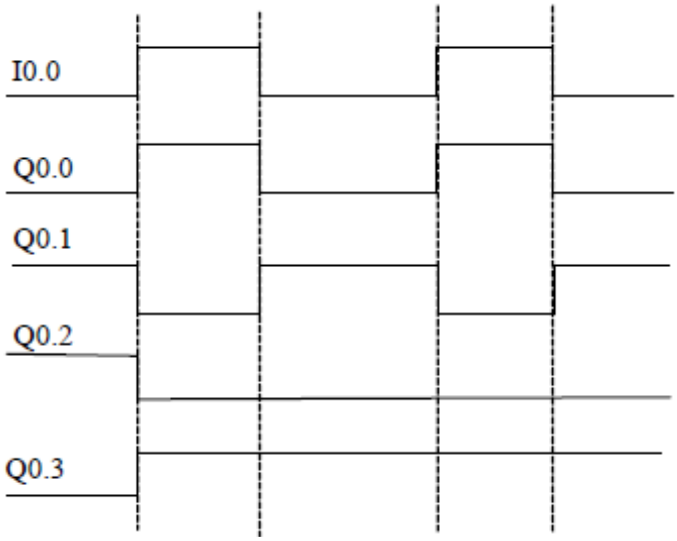
ST, STN, R и S инструкции не влияют на CR.

★ Пример

LD	IL
	LD %I0.0 ST %Q0.0 STN %Q0.1 R %Q0.2 S %Q0.3
	LD %M0.0 MOVE %VW0, %VW2

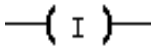
Временная диаграмма

Предположим, что значения Q0.1 и Q0.2 являются 1 и 0 соответственно, прежде чем эти инструкции выполнятся.



6.2.4 Непосредственная Катушка

★ Описание

	Название	Использование	Группа	
LD	Установка непосредственной катушки	*bit* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	STI	STI *bit*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
bit	Выход	BOOL	Q (CPU body)

Эти непосредственные инструкции могут быть использованы только для каналов DO на CPU.

■ LD

Когда инструкция Непосредственной Катушки выполняется, она сразу же передаёт сигнал для физического выхода (бит) и соответствующего регистра выходного изображения.

Когда инструкция Сброса Непосредственной Катушки выполняется, если сигнал равен 1, и физический выход (бит) и соответствующий регистр выходного изображения устанавливаются равными 0 сразу же, в противном случае они остаются неизменными.

Когда инструкция Установки Непосредственной Катушки выполняется, если сигнал равен 1, то физический выход (бит) и соответствующий регистр вывода изображения устанавливаются равными 1 немедленно, в противном случае они остаются неизменными.

■ IL

Непосредственные катушки представлены инструкциями STI, RI и SI.

Когда инструкция STI выполняется, она сразу пишет CR как к физическому объему (бит) и соответствующему регистру выходного изображения.

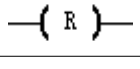
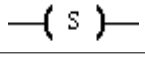
Когда инструкция RI выполнена, если CR равен 1, и физический выход (бит) и соответствующий регистр выходного изображения устанавливаются равными 0 сразу же, в противном случае они остаются неизменными.

Когда инструкция SI выполняется, если CR равен 1, и физический выход (бит) и соответствующий регистр выходного изображения устанавливаются равными 1 сразу же, в противном случае они остаются неизменными.

STI, RI и SI инструкции не влияют на CR.

6.2.5 Катушки Set и Reset

★ Описание

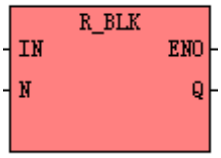
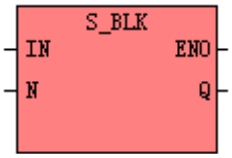
	Название	Использование	Группа	
LD	Reset	*bit* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Set	*bit* 		
IL	Reset	R *bit*	U	
	Set	S *bit*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
bit	Выход	BOOL	Q (CPU body)

■ LD
Функция Reset катушки: если сигнал равен 1, регистр изображения выхода для бита установлено равным 0, в противном случае регистр остается неизменным.
Функция Set катушки: если сигнал равен 1, регистр изображения выхода для бита установлено равным 1, в противном случае регистр остается неизменным.
■ IL
Функция инструкции R: если CR равен 1, регистр выходного изображения для бита устанавливают равным 0, в противном случае регистр остается неизменным.
Функция инструкции S: если CR равен 1, регистр выходного изображения для бита устанавливают равным 1, в противном случае регистр остается неизменным.
R и S инструкции не влияют на CR.

6.2.6 Блоки Set и Reset Катушки

★ Описание

	Название	Использование	Группа	
LD	Блок катушки Reset			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Блок катушки Set			
IL	R_BLK	R_BLK *N,Q*	U	
	S_BLK	S_BLK *N,Q*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BOOL	Поток мощности
N	Вход	INT	L, M, V, constant
Q	Выход	BOOL	Q, V, M, SM, L

■ LD
Действие R_BLK: если значение IN равно 1, он установит последовательные биты из адреса Q в 0, в противном случае остаётся неизменным.
Действие S_BLK: если значение IN равно 1, он установит последовательные биты из адреса Q в 1, в противном случае остаётся неизменным.
■ IL
Действие R_BLK: если значение CR = 1, он установит последовательные биты из адреса Q в 0, в противном случае остаётся неизменным.
Действие S_BLK: если значение CR = 1, он установит последовательные биты из адреса Q в 1, в противном

случае остаётся неизменным.
Выполнение R_BLK и S_BLK не влияет на CR.

Макс количество параметров N составляет 1024.
Параметр Q является начальным адресом блока памяти с переменной длиной. Пожалуйста, обратите внимание, что весь блок памяти запрещено вводить в нелегальные зоны памяти, в противном случае последствия могут быть ужасны.

6.2.7 Непосредственные Катушки Set и Reset

★ Описание

	Название	Использование	Группа	
LD	Reset непосредственной катушки	*bit* —(RI)—		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Set непосредственной катушки	*bit* —(SI)—		
IL	RI	RI *bit*	U	
	SI	SI *bit*		

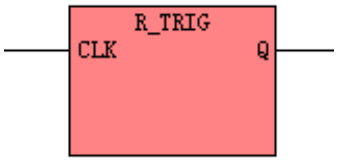
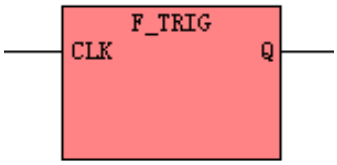
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
bit	Выход	BOOL	Q (CPU body)

Эти непосредственные команды могут быть использованы только для каналов DO на теле CPU.

■ LD
Инструкция Reset Immediate Coil выполняется, если сигнал равен 1, и физический выход (бит) и соответствующий регистр вывода изображения устанавливаются равными 0 немедленно, в противном случае они остаются неизменными.
Инструкция Set Immediate Coil выполняется, если сигнал равен 1, и физический выход (бит) и соответствующий регистр вывода изображения устанавливаются равными 1 немедленно, в противном случае они остаются неизменными.
■ IL
Инструкция RI выполняется, если CR равен 1, и физический выход (бит) и соответствующий регистр выходного изображения устанавливаются равными 0 сразу же, в противном случае они остаются неизменными.
Инструкция SI выполняются, если CR равен 1, и физический выход (бит) и соответствующий регистр выходного изображения устанавливаются равными 1 сразу же, в противном случае они остаются неизменными.
RI и SI инструкции не влияют на CR.

6.2.8 Детектор фронта

★ Описание

	Название	Использование	Группа	
LD	Детектор нарастающего фронта			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	Детектор падающего края			
IL	R_TRIG	R_TRIG	P	
	F_TRIG	F_TRIG		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
CLK (LD)	Вход	BOOL	Поток мощности
Q (LD)	Выход	BOOL	Поток мощности

■ LD

Функция команды R_TRIG заключается в обнаружении нарастающего фронта на входе CLK: после перехода из состояния 0 в состояние 1 на входе CLK, выход Q устанавливается в 1 на один цикл сканирования, а затем возвращается к 0.

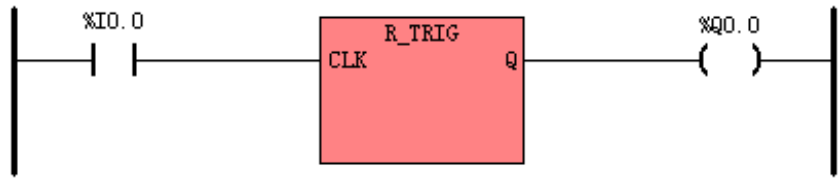
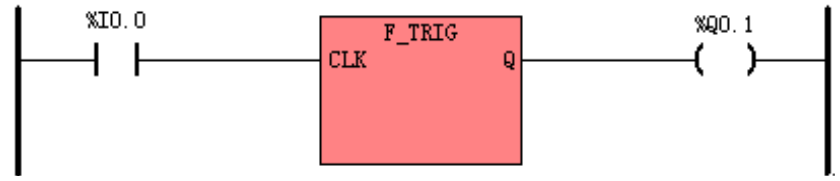
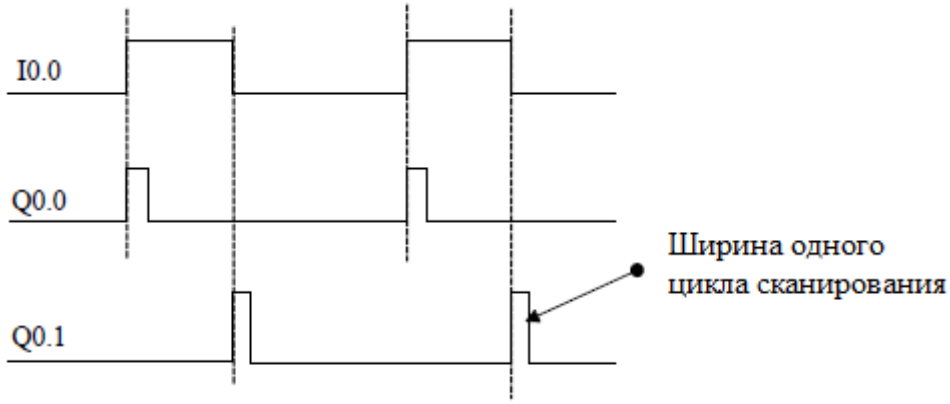
Функция команды F_TRIG заключается в обнаружении заднего фронта сигнала на входе CLK: после перехода из состояния 1 в состояние 0 на входе CLK, выход Q устанавливается в 1 на один цикл сканирования, а затем возвращается к 0.

■ IL

Функция команды R_TRIG заключается в обнаружении нарастающий фронт CR: после перехода из состояния 0 в состояние 1 в CR, выход Q устанавливается в 1 на один цикл сканирования, а затем возвращается к 0.

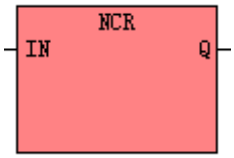
Функция команды F_TRIG заключается в обнаружении заднему фронту CR: после перехода из состояния 1 в состояние 0 на CR, выход Q устанавливается в 1 на один цикл сканирования, а затем возвращается к 0.

★ Пример

LD	IL
	LD %I0.0 R_TRIG ST %Q0.0
	LD %I0.0 F_TRIG ST %Q0.1
Временная диаграмма	
	

6.2.9 NCR (инвертирование)

★ Описание

	Название	Использование	Группа	
LD	NCR			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	NCR	NCR	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BOOL	Поток мощности
Q	Выход	BOOL	Поток мощности

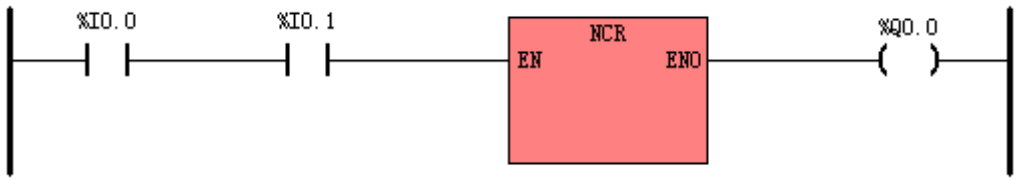
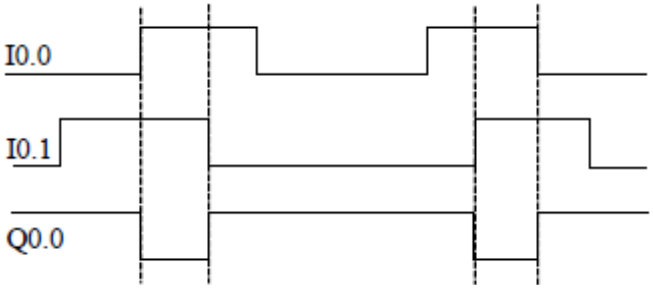
■ LD

Инструкция NCR изменяет состояние сигнала от 1 до 0, или от 0 до 1.

■ IL

Инструкция NCR меняет CR от 1 до 0, или от 0 до 1.

★ Примеры

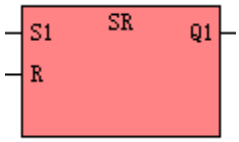
LD	IL
	LD %I0.0 AND %I0.1 NCR ST %Q0.0
Временная диаграмма	
	

6.2.10 Бистабильные элементы

Бистабильный элемент является одним из функциональных блоков, определенных в стандарте IEC61131-3, в двух видах, то есть Set Dominant Bistable (SR) и Reset Dominant Bistable (RS).
Пожалуйста, обратитесь к пункту 3.6.5 «Функциональные блоки и их примеры», для получения более подробной информации.

6.2.10.1 SR (Set Dominant Bistable)

★ Описание

	Название	Использование	Группа	
LD	SR	*SRx* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SR	LD *S1* SR *SRx*, *R*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
SRx	-	пример SR	SR
S1	Вход	BOOL	Поток мощности
R	Вход	BOOL	I, Q, V, M, SM, L, T, C, RS, SR
Q1	Выход	BOOL	Поток мощности

Set Dominant Bistable (SR) является бистабильным элементом, где входные команды являются управляющими. Если set (S1) и reset (R) оба входа 1, то и выход Q1 и значение состояния SRX будет 1. Ниже приводится таблица истинности для команды SR:

S1	R	Q1, SRx
0	0	Предыдущее значение
0	1	0
1	0	1
1	1	1

6.2.10.2 RS (Reset Dominant Bistable)

★ Описание

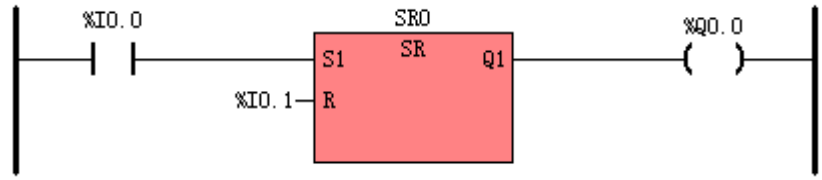
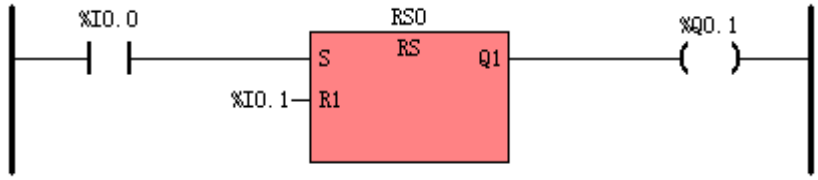
	Название	Использование	Группа	
LD	RS	*RSx* 		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	RS	LD *S* RS *RSx, R1*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
RSx	-	SR instance	RS
S	Вход	BOOL	Поток мощности
R1	Вход	BOOL	I, Q, V, M, SM, L, T, C, RS, SR
Q1	Выход	BOOL	Поток мощности

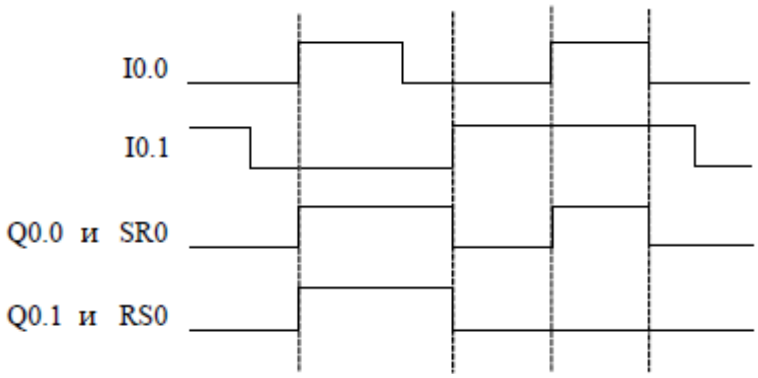
Reset Dominant Bistable (PC) является бистабильным элементом, где вход reset доминирует. Если set (S) и reset (R1) входы установлены как 1, то выход Q1 и значение состояния RSX будет 0.
Ниже приводится Таблица истинности для Инструкцией RS:

R1	S	Q1, RSx
0	0	Предыдущее значение
0	1	1
1	0	0
1	1	0

6.2.10.3 Пример

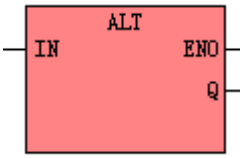
LD	IL
(* Network 0 *)  (* Network 0 *)  	(* Network 0 *) LD %I0.0 SR SR0, %I0.1 ST %Q0.0 (* Network 1 *) LD %I0.0 RS RS0, %I0.1 ST %Q0.1

Временная диаграмма



6.2.11 ALT (Замещение)

★ Описание

	Название	Использование	Группа	
LD	ALT			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ALT	ALT *Q*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN (LD)	Вход	BOOL	Сигнал
Q	Выход	BOOL	Q, V, M, SM, L

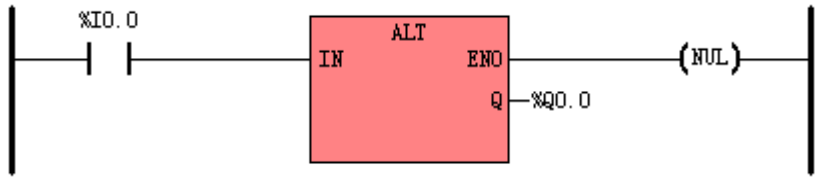
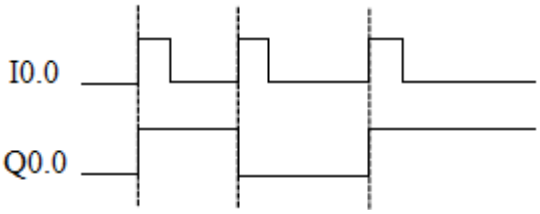
■ LD

Инструкция ALT изменяет значение Q из 1 в 0 или из 0 в 1 по нарастающему фронту на входе IN.

■ IL

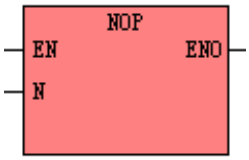
Инструкция ALT изменяет значение Q от 1 до 0 или от 0 до 1 на нарастающем фронте CR. Эта инструкция не влияет на CR.

★ Примеры

LD	IL
	LD %I0.0 ALT %Q0.0
Временная диаграмма	
	

6.2.12 NOP (без операции)

★ Описание

	Название	Использование	Группа	
LD	Dummy			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	Dummy	NOP *N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
N	Вход	INT	Постоянная (положительная)

Инструкция NOP ничего не делает и не имеет никакого влияния на исполнение программы пользователя. Выполнение программы продолжается со следующей команды.
Инструкция NOP, как правило, используется для генерации задержки в выполнении программы. Операнд N является положительным целым числом, константой.

Проверьте следующие данные тестирования для K5:
Создайте новый пустой проект, вызовите в этом проекте 12 инструкции NOP. В каждой инструкции NOP, установите входной параметр N = 32767, а затем загрузите проект в ПЛК, время сканирования ПЛК 37ms. То есть, выполнение времени = 393204 инструкции NOP по 37ms, что составит 37000μs или 37000000ηs. Тогда время выполнения = 10000 инструкциями NOP по 37ms, составит 941μs или 940987ηs. Таким образом, среднее время выполнения одной инструкции NOP составит 94ηs, но фактическое время выполнения одной инструкции NOP больше, чем 94ηs, потому что вызов команды и одной инструкции NOP не очень большое самом деле.

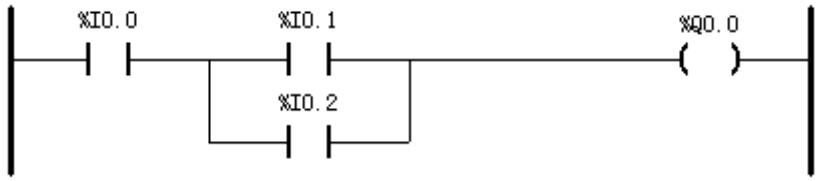
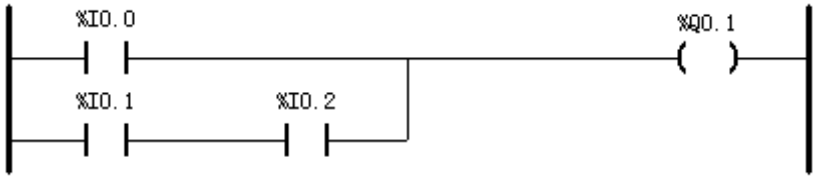
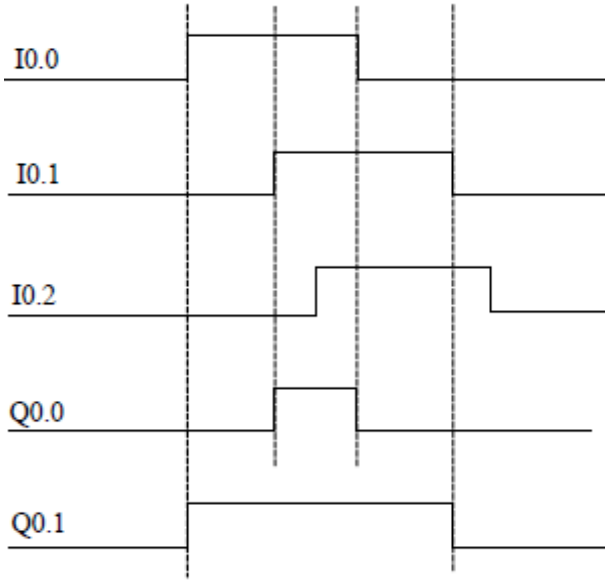
6.2.13 Скобки Модификатора

★ Описание

	Название	Использование	Группа	
IL	AND(	AND(	U	☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	OR(	*OR(*		
	)	)	P	

Скобки модификаторов представлены только в IL. В LD, ST и др. могут принимать сложные выражения в качестве операндов, но только IL обеспечивает простые выражения. Таким образом, стандарт IEC61131-3 определяет скобки модификаторов в IL для решения сложных выражений. Любые «AND(» или «OR(» в паре с «)». В программе IL, перед выполнением операторов между «AND(» и «)», CR сначала временно сохраняется; затем выполняются утверждения в скобках, и результат выполнения операции AND с сохранённой CR, и окончательно CR устанавливается равным результату работы.
Кроме того, перед выполнением операторов между «OR(» и «)», CR сначала временно сохраняется; потом выполняются утверждения в скобках, и результат выполнения оператора OR с сохранённой CR, и окончательно CR устанавливается равным результату работы.

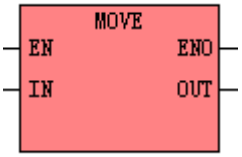
★ Примеры

LD	IL
	LD %I0.0 AND (LD %I0.1 OR %I0.2) ST %Q0.0
	LD %I0.0 OR (LD %I0.1 AND %I0.2) ST %Q0.1
Временная диаграмма	
	

6.3 Инструкции Move (перемещение)

6.3.1 MOVE

★ Описание

	Название	Использование	Группа	
LD	MOVE			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	MOVE	MOVE IN, OUT	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT	Выход	BYTE, WORD, DWORD, INT, DINT, REAL	Q, M, V, L, SM, AQ, курсор

Инструкция MOVE перемещает значение IN на адрес OUT. Эта инструкция выполняет операцию присваивания, и вход и выход должны быть того же типа данных.

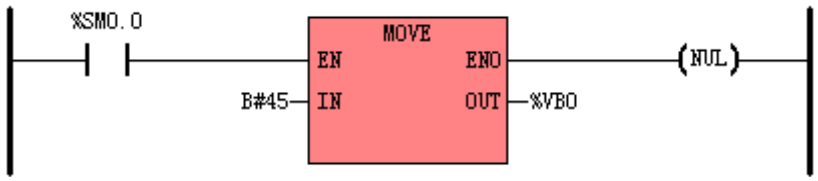
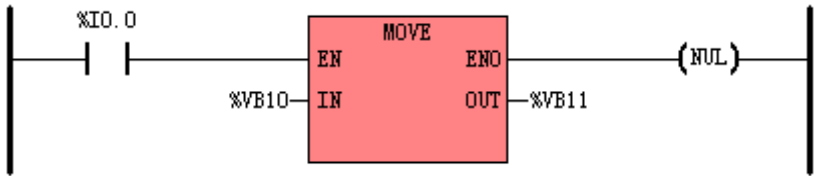
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

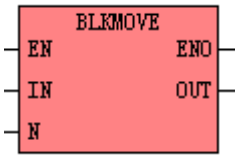
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		%SM0.0 всегда включен, поэтому MOVE всегда выполняется: B # 45 назначен %VB0.
		Если %I0.0 равно 0, MOVE не выполняется. Если %I0.0 равно 1, значение %VB10 присваивается %VB11.
IL	LD %SM0.0 (* CR создается с %SM0.0 *) MOVE B#45, %VB0 (* B#45 назначается %VB0 *)	
	LD %I0.0 (* CR создается с %I0.0 *) MOVE %VB10, %VB11 (* Если CR равно 1, значение %VB10 назначается %VB11 *) (* В противном случае, это утверждение не выполняется,% VB11 остается неизменной *)	

6.3.2 BLKMOVE (Блок Move)

★ Описание

	Название	Использование	Группа	
LD	BLKMOVE			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	BLKMOVE	BLKMOVE IN, OUT, N	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC
N	Вход	BYTE	I, Q, M, V, L, SM, константа
OUT	Выход	BYTE, WORD, DWORD, INT, DINT, REAL	Q, M, V, L, SM, AQ

Вход и выход должны быть одинакового типа данных.
Инструкция BLKMOVE перемещает количество N переменных из последовательного диапазона, который начинается с адреса IN в последующий диапазон, который начинается с адреса OUT.

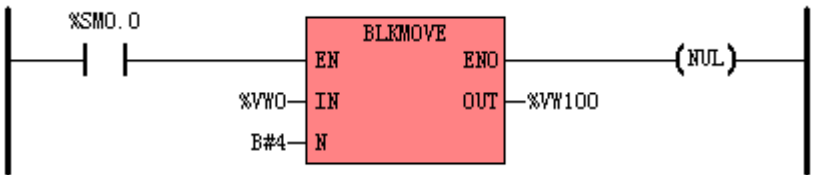
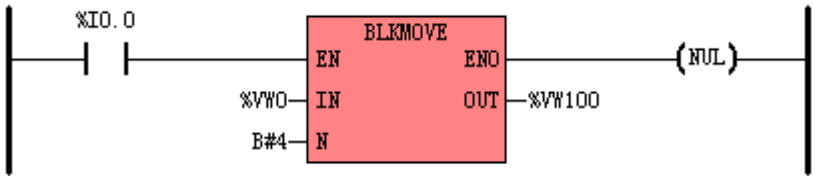
■ LD

Если EN равен 1, эта инструкция выполнена.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

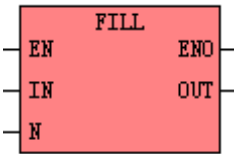
★ Примеры

	Использование	Описание
LD		% SM0.0 всегда включен, поэтому BLKMOVE всегда выполняется: данные в %VW0 - %VW6 перемещаются в %VW100 - %VW106.
		Если %I0.0 равен 1, данные в %VW0 - %VW6 перемещаются в %VW100 - %VW106. В противном случае, BLKMOVE не выполняется.
IL	LD %SM0.0 BLKMOVE %VW0, %VW100, B#4	(* CR создается с %SM0.0 *) (* Данные в VW0-VW6 перемещаются в %VW100-%VW106 *)
	LD %I0.0 BLKMOVE %VW0, %VW100, B#4	(* CR создается с %I0.0 *) (* Если CR равно 0, это утверждение не выполняется *) (* Если CR равно 1, то данные в %VW0 - %VW6 перемещаются в %VW100 - %VW106 *)

Результат	VW0	VW2	VW4	VW6
	0	10	20	30
	VW100	VW102	VW104	VW106
	0	10	20	30

6.3.3 FILL (Заполнение памяти)

★ Описание

	Название	Использование	Группа	
LD	FILL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	FILL	FILL IN, OUT, N	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	константа
N	Вход	BYTE	константа
OUT	Выход	BYTE	M, V, L

Инструкция FILL устанавливает число N последовательных переменных, начиная с адреса OUT, в указанную постоянную IN.

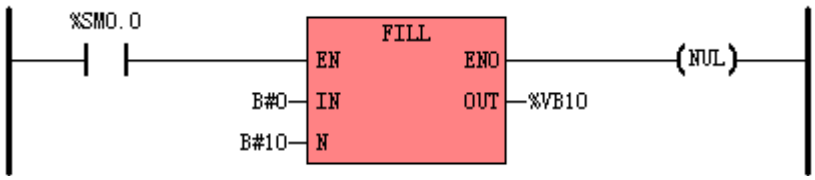
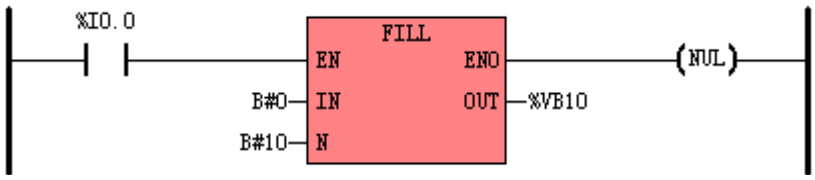
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, то эта инструкция выполняется, и не влияет на CR.

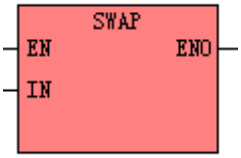
★ Примеры

	Использование	Описание
LD		%SM0.0 всегда включен, поэтому FILL всегда выполняется: 10 переменных от %VB10 до %VB19 установятся в B # 0.
		Если %I0.0 равно 0, FILL не выполняется. Если %I0.0 равен 1: 10 переменных от %VB10 до %VB19 установятся в B # 0.

IL	LD %SM0.0 (* CR создается с %SM0.0 *) FILL B#0, %VB10, B#10 (* 10 переменных от %VB10 до %VB19 установятся в B#0 *)
	LD %I0.0 (* CR создается с %I0.0 *) FILL B#0, %VB10, B#10 (* Если CR = 0, это утверждение не выполняется *) (* Если CR = 1, 10 переменных от %VB10 до %VB19 установятся в B#0 *)
Результат	VB10 VB11 VB12 VB13 ... VB18 VB19 0 0 0 0 ... 0 0

6.3.4 SWAP

★ Описание

	Название	Использование	Группа	
LD	SWAP			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SWAP	SWAP IN	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход / Выход	WORD, DWORD	Q, M, V, L, SM

В SWAP инструкции обмен наиболее значимого байта начиная с самого младшего байта в слово (IN), или обмен наиболее значимого слова начиная с младшего слова в двойное слово (IN).

■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

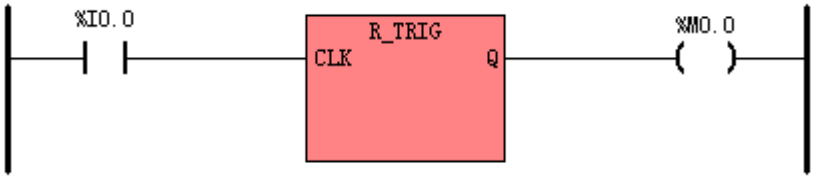
Если CR 1, эта инструкция выполняется, и это не влияет на CR.

★ Примеры

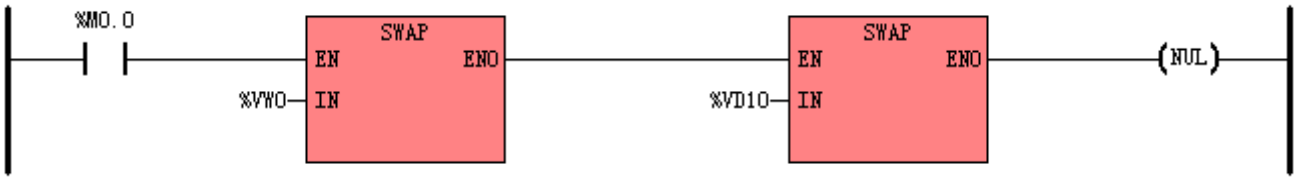
	Использование
IL	(* Network 0 *) LD %I0.0 R_TRIG (* По нарастающему фронту %I0.0, *) SWAP %VW0 (* старший байт начиная с самого младшего байта %VW0 обмениваются, *) SWAP %VD10 (* и старшее слово начиная с самого младшего слова %VD10 обмениваются. *)

LD

(* Network 0 *)
(* По нарастающему фронту I0.0, следующая программа обменивает наиболее значимый байт начиная с самого младшего байта %VW0 и обменивает наиболее значимое слово с младшим словом %VD10. *)

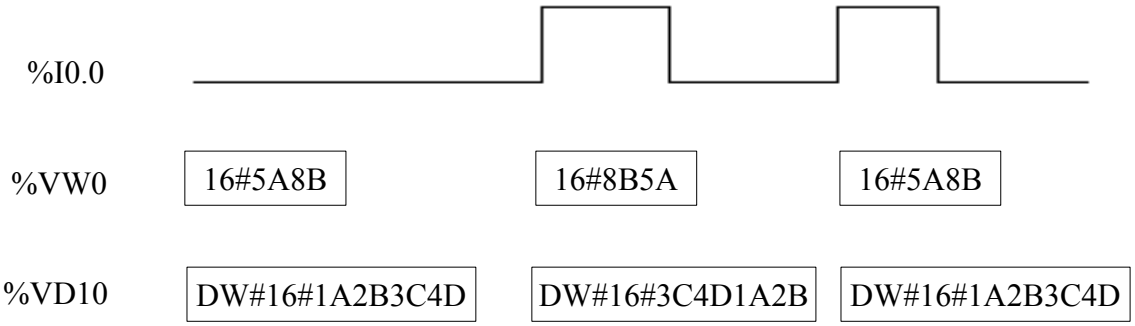


(* Network 1 *)



Результат

Предположим, что начальное значение %VW0 является W # 16 # 5A8B,
и начальное значение %VD10 является DW # 16 # 1A2B3C4D.

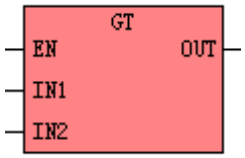


6.4 Инструкции сравнения

Для всех команд сравнения, BYTE сравнения является без знаковым. INT, DINT и REAL сравнения являются знаковыми.

6.4.1 GT (больше чем)

★ Описание

	Название	Использование	Группа	
LD	GT			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	GT	GT *IN1, IN2*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT (LD)	Выход	BOOL	Поток мощности

IN1 и IN2 должны быть одного и того же типа данных.

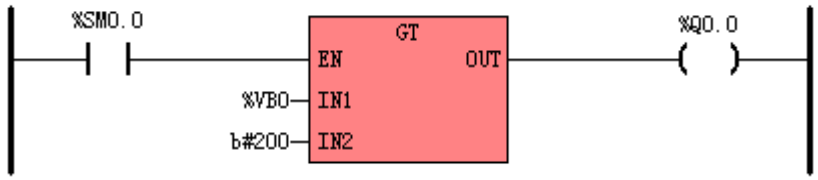
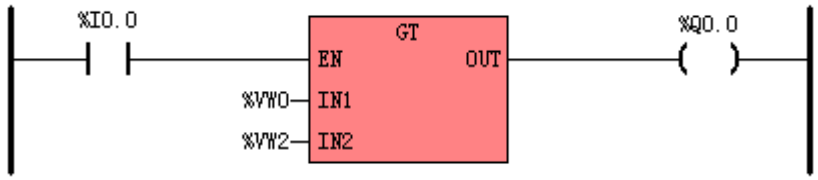
■ LD

Если EN равен 1, то эта инструкция сравнивает IN1 больше, чем IN2 и Логический результат присваивается в OUT. Если EN равен 0, эта инструкция не выполняется, и OUT устанавливается равным 0.

■ IL

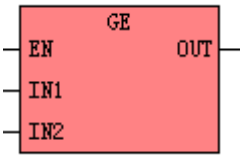
Если CR равен 1, эта инструкция сравнивает IN1 больше, чем IN2 и Логический результат присваивает CR; Если CR равен 0, эта инструкция не выполняется, и CR остаётся равным 0.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому GT всегда выполняется: если значение VB0 больше, чем B#200, то Q0.0 устанавливается равным 1, в противном случае Q0.0 устанавливается равным 0.
		Если I0.0 = 1: если значение VW0 больше, чем VW2, то Q0.0 = 1, в противном случае Q0.0 = 0. Если I0.0 = 0: GT не выполняется, и Q0.0 устанавливается равным 0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) GT %VB0, B#200 (* Если VB0 > B#200, CR = 1, в противном случае CR устанавливается = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	
	LD %I0.0 (* CR создается с I0.0 *) GT %VW0, %VW2 (* Если CR = 1: Если VW0 > VW2, CR устанавливается = 1, в противном случае CR устанавливается = 0 *) (* Если CR = 0: GT не выполняется, CR остаётся = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	

6.4.2 GE (больше или равно)

★ Описание

	Название	Использование	Группа	
LD	GE			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	GE	GE *IN1, IN2*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT (LD)	Выход	BOOL	Поток мощности

IN1 и IN2 должны быть одного и того же типа данных.

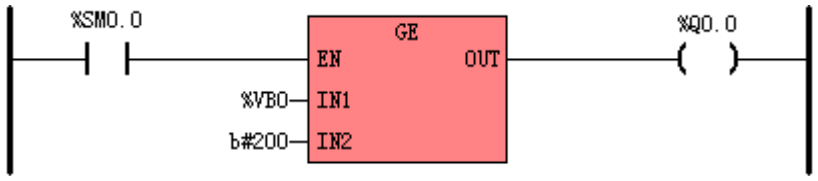
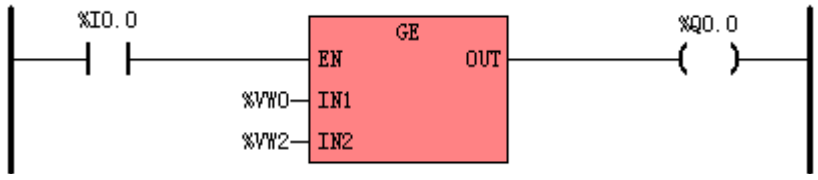
■ LD

Если EN равен 1, то эта инструкция сравнивает IN1 больше или равна IN2 и Логический результат присваивается OUT. Если EN равен 0, эта инструкция не выполняется, и OUT устанавливается равным 0.

■ IL

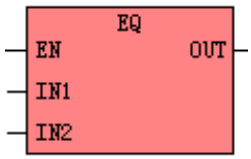
Если CR равен 1, эта инструкция сравнивает IN1 больше или равную IN2 и Логический результат присваивается CR. Если CR равен 0, эта инструкция не выполняется, и CR остаётся равным 0.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому GE всегда выполняется: если VB0 больше или равно B # 200, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0.
		Если I0.0 = 1: Если VW0 больше или равна VW2, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0. Если I0.0 равно 0: GE не выполняется, и Q0.0 = 0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) GE %VB0, B#200 (* Если VB0 ≥ B#200, CR = 1, в противном случае CR устанавливается = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	
IL	LD %I0.0 (* CR создается с I0.0 *) GE %VW0, %VW2 (* Если CR = 1: Если VW0 больше или равно VW2, CR устанавливается = 1, в противном случае CR устанавливается = 0 *) (* Если CR = 0: GE не выполняется, CR остаётся = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	

6.4.3 EQ (равно)

★ Описание

	Название	Использование	Группа	
LD	EQ			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	EQ	EQ *IN1, IN2*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT (LD)	Выход	BOOL	Поток мощности

IN1 и IN2 должны быть одного и того же типа данных.

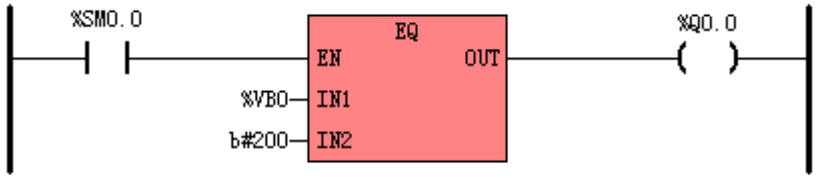
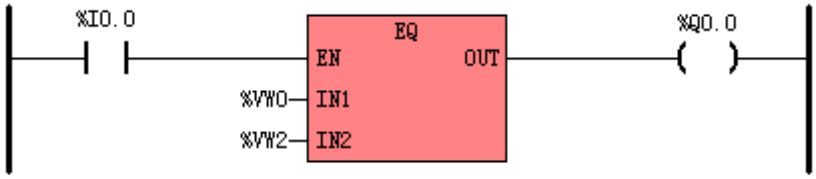
■ LD

Если EN равен 1, то эта инструкция сравнивает IN1 равное IN2 и Логический результат присваивается OUT;
Если EN равен 0, эта инструкция не выполняется, и OUT устанавливается равным 0.

■ IL

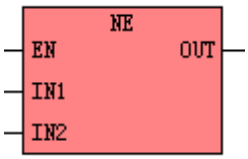
Если CR равен 1, эта инструкция сравнивает IN1 равное IN2 и Логический результат присваивается CR;
Если CR равен 0, эта инструкция не выполняется, и CR остается 0.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому EQ всегда выполняется: если VB0 равно B # 200, Q0.0 устанавливается равным 1, в противном случае Q0.0 устанавливается равным 0.
		Если I0.0 = 1: Если VW0 равно VW2, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0. Если I0.0 равно 0: EQ не выполняется, и Q0.0 = 0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) EQ %VB0, B#200 (* Если VB0 = B#200, CR = 1, в противном случае CR устанавливается = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	
IL	LD %I0.0 (* CR создается с I0.0 *) EQ %VW0, %VW2 (* Если CR = 1: Если VW0 равно VW2, CR устанавливается = 1, в противном случае CR устанавливается = 0 *) (* Если CR = 0: EQ не выполняется, CR остаётся = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	

6.4.4 NE (не равно)

★ Описание

	Название	Использование	Группа	
LD	NE			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	NE	NE *IN1, IN2*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT (LD)	Выход	BOOL	Поток мощности

IN1 и IN2 должны быть одного и того же типа данных.

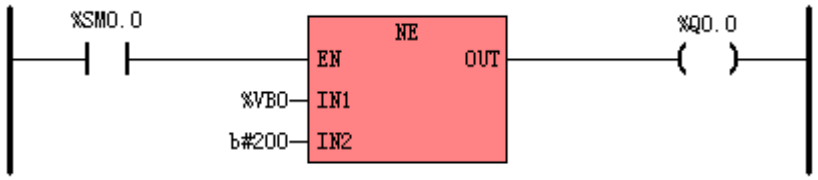
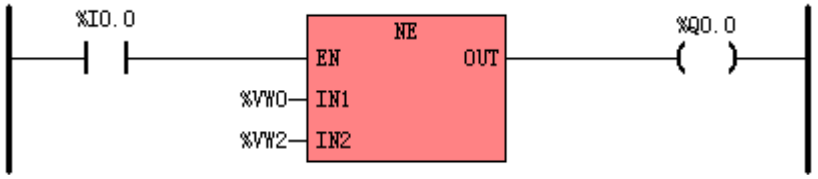
■ LD

Если EN = 1, то эта инструкция сравнивает IN1 не равное IN2 и Логический результат присваивается OUT;
Если EN равен 0, эта инструкция не выполняется, и OUT устанавливается равным 0.

■ IL

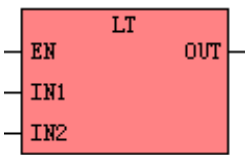
Если CR равен 1, эта инструкция сравнивает IN1 не равное IN2 и Логический результат присваивается CR;
Если CR равен 0, эта инструкция не выполняется, и CR остается 0.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому NE всегда выполняется: если VB0 не равно B # 200, Q0.0 = 1, в противном случае Q0.0 устанавливается равным 0.
		Если I0.0 = 1: Если VW0 не равно VW2, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0. Если I0.0 равно 0: NE не выполняется, и Q0.0 = 0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) NE %VB0, B#200 (* Если VB0 ≠ B#200, CR = 1, в противном случае CR устанавливается = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	
	LD %I0.0 (* CR создается с I0.0 *) NE %VW0, %VW2 (* Если CR = 1: Если VW0 не равно VW2, CR устанавливается = 1, в противном случае CR устанавливается = 0 *) (* Если CR = 0: NE не выполняется, CR остаётся = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	

6.4.5 LT (меньше чем)

★ Описание

	Название	Использование	Группа	
LD	LT			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	LT	LT *IN1, IN2*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT (LD)	Выход	BOOL	Поток мощности

IN1 и IN2 должны быть одного и того же типа данных.

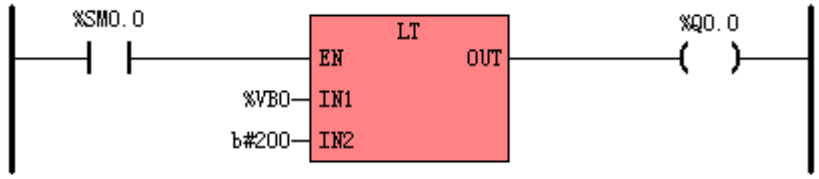
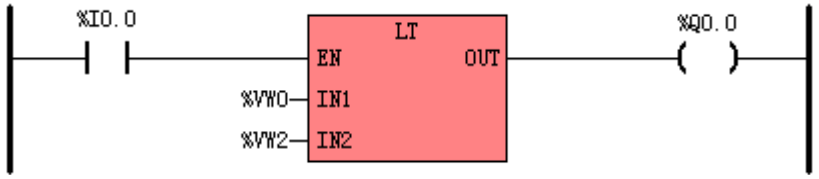
■ LD

Если EN = 1, эта инструкция сравнивает IN1 меньше, чем IN2 и Логический результат присваивается OUT;
Если EN равен 0, эта инструкция не выполняется, и OUT устанавливается равным 0.

■ IL

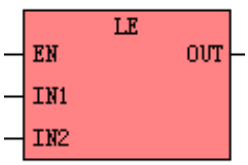
Если CR = 1, эта инструкция сравнивает IN1 меньше, чем IN2 и Логический результат присваивается CR;
Если CR равен 0, эта инструкция не выполняется, и CR остаётся 0.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому LT всегда выполняется: если VB0 меньше чем B # 200, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0.
		Если I0.0 = 1: Если VW0 меньше VW2, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0. Если I0.0 равно 0: LT не выполняется, и Q0.0 = 0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) LT %VB0, B#200 (* Если VB0 < B#200, CR = 1, в противном случае CR устанавливается = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	
	LD %I0.0 (* CR создается с I0.0 *) LT %VW0, %VW2 (* Если CR = 1: Если VW0 меньше VW2, CR устанавливается = 1, в противном случае CR устанавливается = 0 *) (* Если CR = 0: NE не выполняется, CR остаётся = 0 *) ST %Q0.0 (* Q0.0 устанавливается равным CR *)	

6.4.6 LE (меньше или равно)

★ Описание

	Название	Использование	Группа	
LD	LE			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	LE	LE *IN1, IN2*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT, REAL	I, Q, M, V, L, SM, AI, AQ, T, C, HC, константа, курсор
OUT (LD)	Выход	BOOL	Поток мощности

IN1 и IN2 должны быть одного и того же типа данных.

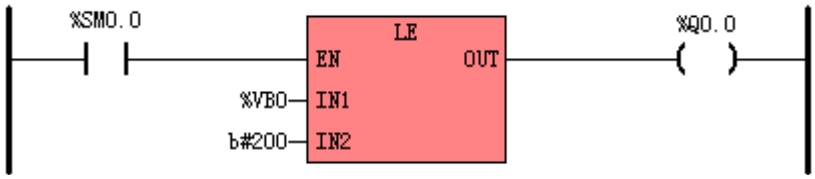
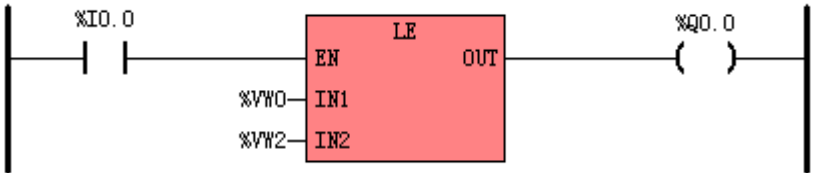
■ LD

Если EN равен 1, то эта инструкция сравнивает IN1 меньше или равно IN2 и Логический результат присваивается OUT; Если EN равен 0, эта инструкция не выполняется, и OUT устанавливается равным 0.

■ IL

Если CR равен 1, эта инструкция сравнивает IN1 меньше или равно IN2 и Логический результат присваивается CR; Если CR равен 0, эта инструкция не выполняется, и CR остаётся 0.

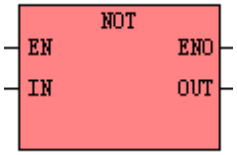
★ Примеры

<table>
<tr><th></th><th>Использование</th><th>Описание</th></tr>
<tr><td rowspan="2">LD</td><td> 
</td><td>SM0.0 всегда включен, поэтому LE всегда выполняется: если VB0 меньше или равно B # 200, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0.</td></tr>
<tr><td> 
</td><td>Если I0.0 = 1: Если VW0 меньше или равно VW2, Q0.0 устанавливается равным 1, в противном случае Q0.0 = 0.
Если I0.0 равно 0: LE не выполняется, и Q0.0 = 0.</td></tr>
<tr><td rowspan="2">IL</td><td colspan="2">LD %SM0.0 (* CR создается с SM0.0 *)
LE %VB0, B#200 (* Если VB0 ≤ B#200, CR = 1, в противном случае CR устанавливается = 0 *)
ST %Q0.0 (* Q0.0 устанавливается равным CR *)</td></tr>
<tr><td colspan="2">LD %I0.0 (* CR создается с I0.0 *)
LE %VW0, %VW2 (* Если CR = 1: Если VW0 меньше или равно VW2, CR устанавливается = 1, в противном случае CR устанавливается = 0 *)
(* Если CR = 0: LE не выполняется, CR остаётся = 0 *)
ST %Q0.0 (* Q0.0 устанавливается равным CR *)</td></tr>
</table>

6.5 Логические операции

6.5.1 NOT

★ Описание

	Название	Использование	Группа	
LD	NOT			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	NOT	NOT *OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

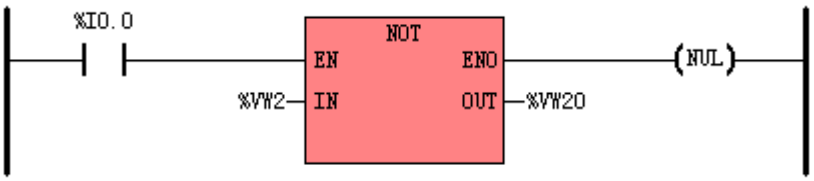
Вход и выход должны быть одинакового типа данных.

Если EN равен 1, то эта инструкция инвертирует каждый бит IN и присваивает результат в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

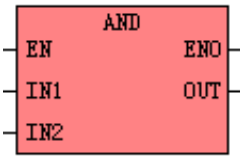
Если CR равен 1, эта инструкция инвертирует каждый бит OUT и сохраняет результат в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: NOT не выполняется. Если I0.0 = 1: NOT инвертирует каждый бит VW2 и присваивает результат VW20.
IL	LD %I0.0 (* CR создается с I0.0 *) NOT %VW20 (* Если CR = 1: NOT инвертирует каждый бит VW2 и сохраняет результат в VW20 *) (* Если CR = 0: инструкция NOT не выполняется *)	
Результат	Для примера в LD, если инструкция NOT выполняется, результат будет следующим: Адрес VW2 Значение W#16#5555 Адрес VW20 Значение W#16#AAAA	

6.5.2 AND

★ Описание

	Название	Использование	Группа	
LD	AND			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	AND	AND *IN1, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
IN2	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть одинакового типа данных.

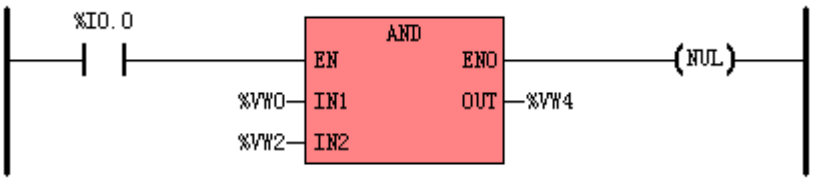
Если EN равен 1, инструкции AND сравнивает соответствующие биты из IN1 и из IN2 и присваивает результат в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

IN и OUT должны быть одинакового типа данных.

Если CR равен 1, инструкции AND сравнивает соответствующие биты из IN и OUT и присваивает результат OUT, и не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: AND не выполняется. Если I0.0 = 1: Инструкции AND сравнивает соответствующие биты из VW0 и VW2, и присваивает результат в VW4.
IL	LD %I0.0 (* CR создается с I0.0 *) AND %VW0, %VW2 (* Если CR = 1: Инструкции AND сравнивает соответствующие биты из VW0 и VW2, и присваивает результат VW4. *) (* Если CR = 0: инструкция NOT не выполняется *)	
Результат	Для примера в LD, если инструкция AND выполняется, результат будет следующим: Address VW0 VW2 Value W#16#129B W#16#960F Address VW4 Value W#16#120B	

6.5.3 ANDN

★ Описание

	Название	Использование	Группа	
LD	ANDN			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ANDN	ANDN *IN1, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
IN2	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

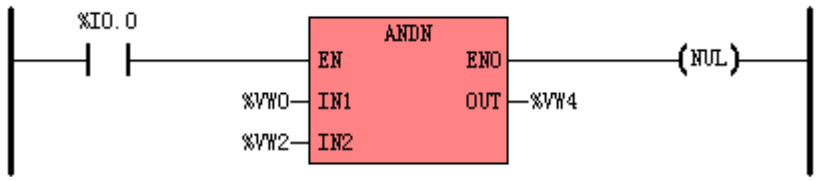
Если EN равен 1, инструкция AND сравнивает соответствующие биты IN1 и IN2, инвертирует каждый бит результата, и конечный результат назначает в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

IN и OUT должны быть того же типа данных.

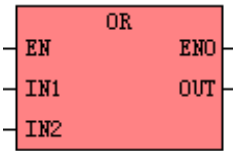
Если CR равен 1, инструкция AND сравнивает соответствующие биты IN и OUT, затем инвертирует каждый бит результата, и конечный результат назначает в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: ANDN не выполняется. Если I0.0 = 1: Инструкция ANDN сравнивает соответствующие биты VW0 и VW2, инвертирует каждый бит результата, и конечный результат назначает в VW4.
IL	LD %I0.0 (* CR создается с I0.0 *) ANDN %VW0, %VW2 (* Если CR = 1: Инструкции ANDN сравнивает соответствующие биты из VW0 и VW2, инвертирует каждый бит результата и конечный результат назначает в VW4. *) (* Если CR = 0: инструкция NOT не выполняется *)	
Результат	Для примера в LD, если инструкция ANDN выполняется, результат будет следующим: Адрес VW0 VW2 Значение W#16#129B W#16#960F Адрес VW4 Значение W#16#EDF4	

6.5.4 OR

★ Описание

	Название	Использование	Группа	
LD	OR			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	OR	OR *IN1, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
IN2	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

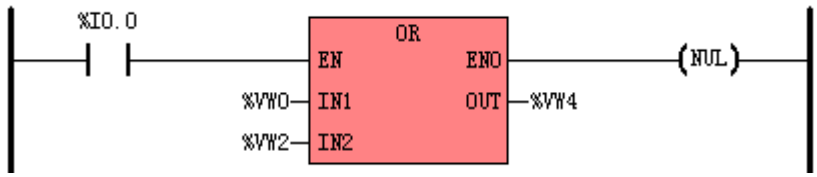
Если EN равен 1, то инструкция OR сравнивает соответствующие биты IN1 и IN2 и присваивает результат в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

Вход и выход должны быть одинакового типа данных.

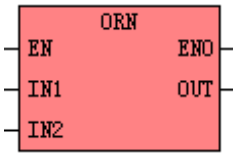
Если CR равен 1, то инструкция OR сравнивает соответствующие биты IN и OUT и присваивает результат в OUT, и не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: OR не выполняется. Если I0.0 = 1: Инструкция OR сравнивает соответствующие биты VW0 и VW2, и результат назначает в VW4.
IL	LD %I0.0 (* CR создается с I0.0 *) OR %VW0, %VW2 (* Если CR = 1: Инструкции OR сравнивает соответствующие биты из VW0 и VW2, и результат назначает в VW4. *) (* Если CR = 0: инструкция OR не выполняется *)	
Результат	Для примера в LD, если инструкция OR выполняется, результат будет следующим: Address VW0 VW2 Value W#16#5555 W#16#AAAA Address VW4 Value W#16#FFFF	

6.5.5 ORN

★ Описание

	Название	Использование	Группа	
LD	ORN			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ORN	ORN *IN1, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
IN2	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

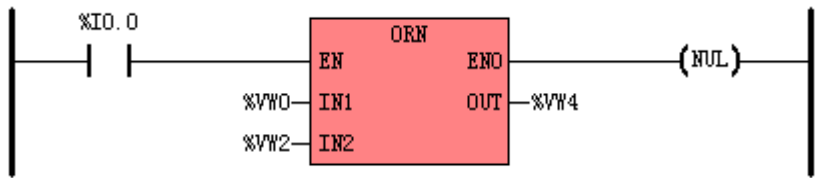
Если EN равен 1, то инструкция OR сравнивает соответствующие биты IN1 и IN2, инвертирует каждый бит результата, и конечный результат назначает в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

IN и OUT должны быть того же типа данных.

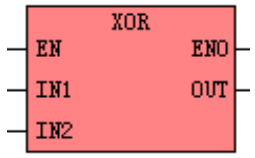
Если CR равен 1, инструкция OR сравнивает соответствующие биты IN и OUT, затем инвертирует каждый бит результата и конечный результат назначает в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: ORN не выполняется. Если I0.0 = 1: Инструкция ORN сравнивает соответствующие биты VW0 и VW2, инвертирует каждый бит результата, и конечный результат назначает в VW4.
IL	LD %I0.0 (* CR создается с I0.0 *) ORN %VW0, %VW2 (* Если CR = 1: Инструкции ORN сравнивает соответствующие биты из VW0 и VW2, инвертирует каждый бит результата, и конечный результат назначает в VW4. *) (* Если CR = 0: инструкция ORN не выполняется *)	
Результат	Для примера в LD, если инструкция ORN выполняется, результат будет следующим: Address VW0 VW2 Value W#16#129B W#16#960F Address VW4 Value W#16#6960	

6.5.6 XOR (исключающее ИЛИ)

★ Описание

	Название	Использование	Группа	
LD	XOR			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	XOR	XOR *IN1, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
IN2	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

Если EN=1, то инструкция XOR сравнивает соответствующие биты IN1 и IN2 и присваивает результат OUT.

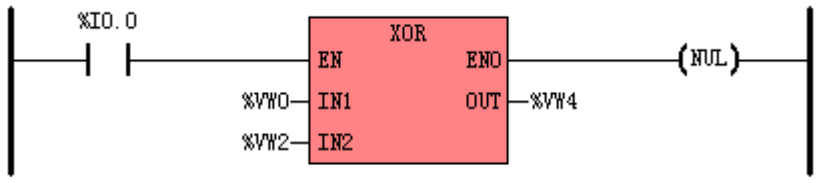
Если EN=0, эта инструкция не выполняется.

■ IL

Вход и выход должны быть одинакового типа данных.

Если CR равен 1, эта инструкция XOR сравнивает соответствующие биты IN и OUT и присваивает результат в OUT, и не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

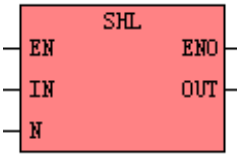
★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: XOR не выполняется. Если I0.0 = 1: Инструкция XOR сравнивает соответствующие биты VW0 и VW2, и результат назначает в VW4.
IL	LD %I0.0 (* CR создается с I0.0 *) XOR %VW0, %VW2 (* Если CR = 1: Инструкции XOR сравнивает соответствующие биты из VW0 и VW2, и результат назначает в VW4 *) (* Если CR = 0: инструкция XOR не выполняется *)	
Результат	Для примера в LD, если инструкция XOR выполняется, результат будет следующим: Address VW0 VW2 Value W#16#9514 W#16#B9A1 Address VW4 Value W#16#2CB5	

6.6 Инструкции Shift / Rotate

6.6.1 SHL (сдвиг влево)

★ Описание

	Название	Использование	Группа	
LD	SHL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SHL	SHL *OUT, N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
N	Вход	BYTE	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

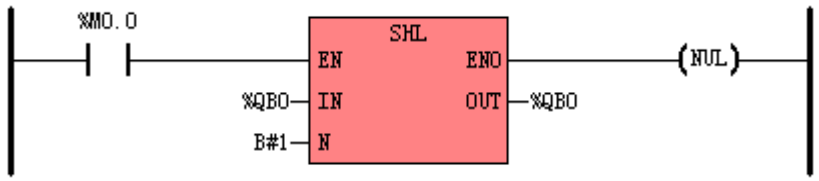
IN и OUT должны быть того же типа данных.

Если EN равен 1, то эта инструкция сдвигает значение IN влево на N битов, и каждый бит заполняется нулем, когда он смещен влево. Результат присваивается в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

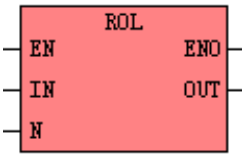
Если CR равен 1, эта инструкция сдвигает значение OUT влево на N битов, и каждый бит заполняется нулем, когда он смещен влево. Результат присваивается в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если M0.0 = 0: SHL не выполняется. Если M0.0 = 1: SHL сдвигает QB0 влево на один бит, а результат все равно назначается в QB0.
IL	LD %M0.0 (* CR создается с M0.0 *) SHL %QB0, B#1 (* Если CR = 1: SHL сдвигает QB0 влево на 1 бит, а результат все равно назначается в QB0 *) (* Если CR = 0: инструкция XOR не выполняется *)	
Результат	Для примера в LD, если инструкция SHL выполняется, результат будет следующим: QB0 B#2#10000001 QB0 После 1-го сдвига B#2#10000010; После 2-го сдвига B#2#00000100; После 3-го сдвига B#2#00001000; После 4-го сдвига B#2#00010000	

6.6.2 ROL (поворот влево)

★ Описание

	Название	Использование	Группа	
LD	ROL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ROL	ROL *OUT, N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
N	Вход	BYTE	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

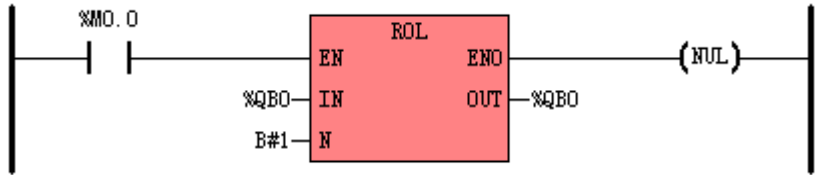
IN и OUT должны быть того же типа данных.

Если EN равен 1, то эта инструкция поворачивает значение IN влево на N битов, и старший бит поворачивается к младшему биту. Результат присваивается в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

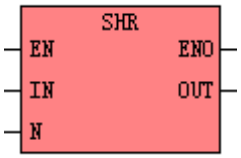
Если CR равен 1, эта инструкция поворачивает значение OUT влево на N битов, и старший бит поворачивается к младшему биту. Результат присваивается в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если M0.0 = 0: ROL не выполняется. Если M0.0 = 1: ROL вращается QB0 влево на 1 бит, и старший бит поворачивается младшему биту. Результат присваивается в QB0.
IL	LD %M0.0 (* CR создается с M0.0 *) ROL %QB0, B#1 (* Если CR = 1: ROL поворачивает QB0 влево на 1 бит, а результат все равно назначается в QB0 *) (* Если CR = 0: инструкция ROL не выполняется *)	
Результат	Для примера в LD, если инструкция ROL выполняется, результат будет следующим: QB0 B#2#10100001 После 1-го сдвига: QB0 B#2#01000011 После 2-го сдвига: B#2#10000110 После 3-го сдвига: B#2#00001101 После 4-го сдвига: B#2#00011010	

6.6.3 SHR (сдвиг вправо)

★ Описание

	Название	Использование	Группа	
LD	SHR			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SHR	SHR *OUT, N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
N	Вход	BYTE	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

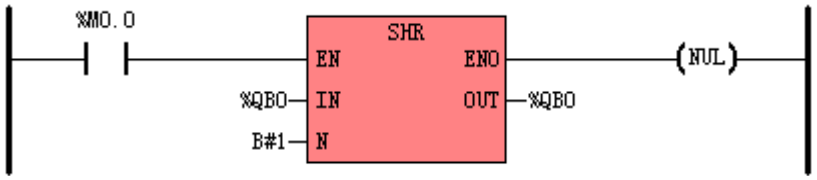
IN и OUT должны быть того же типа данных.

Если EN равен 1, то эта инструкция сдвигает значение IN вправо на N битов, а каждый бит заполняется нулем, когда она смещается вправо. Результат присваивается в OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

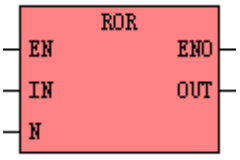
Если CR равен 1, эта инструкция сдвигает значение OUT вправо на N битов, а каждый бит заполняется нулем, когда он смещается вправо. Результат всё равно присваивается в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если M0.0 = 0: SHR не выполняется. Если M0.0 = 1: SHR сдвигает QB0 вправо на один бит, а результат все равно назначается QB0.
IL	LD %M0.0 (* CR создается с M0.0 *) SHR %QB0, B#1 (* Если CR = 1: SHR сдвигает QB0 вправо на 1 бит, а результат все равно назначается в QB0 *) (* Если CR = 0: инструкция SHR не выполняется *)	
Результат	Для примера в LD, если инструкция SHR выполняется, результат будет следующим: QB0 B#2#10000001 После 1-го сдвига: QB0 B#2#01000000 После 2-го сдвига: B#2#00100000 После 3-го сдвига: B#2#00010000 После 4-го сдвига: B#2#00001000	

6.6.4 ROR (поворот вправо)

★ Описание

	Название	Использование	Группа	
LD	ROR			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ROR	ROR *OUT, N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE, WORD, DWORD	I, Q, M, V, L, SM, константа, курсор
N	Вход	BYTE	I, Q, M, V, L, SM, константа, курсор
OUT	Выход	BYTE, WORD, DWORD	Q, M, V, L, SM, курсор

■ LD

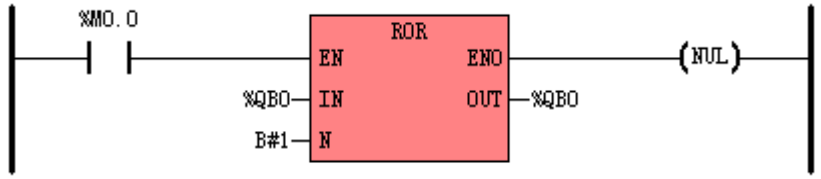
IN и OUT должны быть того же типа данных.

Если EN равен 1, то эта инструкция поворачивает значение IN вправо на N битов, и младший бит поворачивается к старшему биту. Результат присваивается OUT. Если EN равен 0, эта инструкция не выполняется.

■ IL

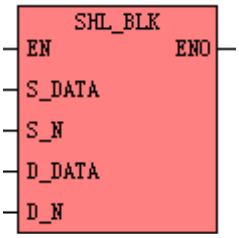
Если CR равен 1, эта инструкция поворачивает значение OUT вправо на N битов, и младший бит поворачивается к старшему биту. Результат присваивается в OUT. Это не влияет на CR. Если CR равен 0, эта инструкция не выполняется.

★ Примеры

	Использование	Описание
LD		Если M0.0 = 0: ROR не выполняется. Если M0.0 = 1: ROR вращает QB0 вправо на один бит, и младший бит поворачивается к старшему биту. Результат присваивается в QB0.
IL	LD %M0.0 (* CR создается с M0.0 *) ROR %QB0, B#1 (* Если CR = 1: ROR вращает QB0 вправо на 1 бит, и результат присваивается в QB0 *) (* Если CR = 0: инструкция ROR не выполняется *)	
Результат	Для примера в LD, если инструкция ROR выполняется, результат будет следующим: QB0 B#2#10100001 После 1-го сдвига: QB0 B#2#11010000 После 2-го сдвига: B#2#01101000 После 3-го сдвига: B#2#00110100 После 4-го сдвига: B#2#00011010	

6.6.5 SHL_BLK (сдвиг влево битовой строки)

★ Описание

	Название	Использование	Группа	
LD	SHL_BLK			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SHL_BLK	SHL_BLK *S_DATA, S_N, D_DATA, D_N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
S_DATA	Вход	BOOL	I, Q, M, V, L
S_N	Вход	INT	I, Q, M, V, L, SM, T, C, AI, AQ, константа, курсор
D_DATA	Вход / Выход	BOOL	Q, M, V, L
D_N	Вход	INT	I, Q, M, V, L, SM, T, C, AI, AQ, константа, курсор

Эта инструкция перемещает число D_N непрерывных битов, начиная с D_DATA, влево на S_N битов. Между тем, число S_N непрерывных битов, начиная с S_DATA, помещают в крайний правый бит D_DATA.

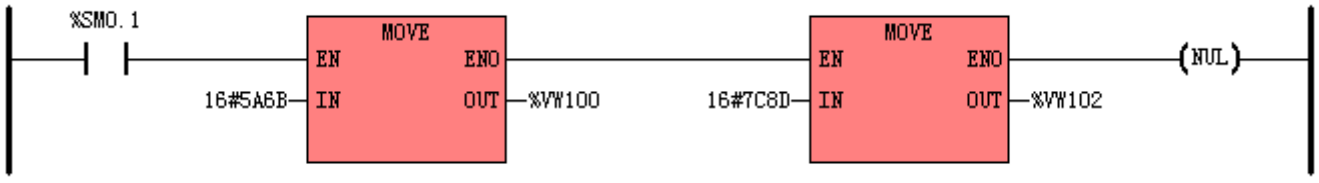
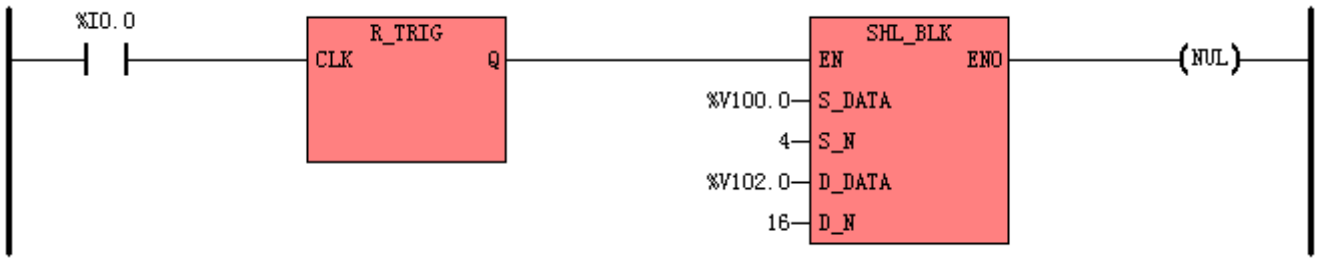
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование
LD	(* Network 0 *) (* Инициализация переменных *)  (* Network 1 *) (* переместить один раз в каждый фронт I0.0 *) 

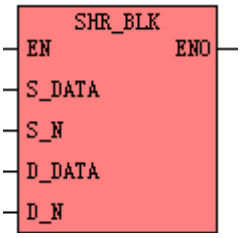
IL	(* Network 0 *)
	(* Инициализация переменных *)
	LD %SM0.1
	MOVE 16#5A6B, %VW100
	MOVE 16#7C8D, %VW102
	(* Network 1 *)
	(* переместить один раз в каждый фронт I0.0 *)
	LD %I0.0
	R_TRIG
	SHL_BLK %V100.0, 4, %V102.0, 16

Результат

	VW102				VW100			
	V103.7			V102.0	V101.7			V100.0
Начальное значение	0111	1100	1000	1101	0101	1010	0110	1011
После 1-го исполнения	1100	1000	1101	1011				
После 2-го исполнения	1000	1101	1011	1011				
После 3-го исполнения	1101	1011	1011	1011				

6.6.6 SHR_BLK (сдвиг вправо битовой строки)

★ Описание

	Название	Использование	Группа	
LD	SHR_BLK			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SHR_BLK	SHR_BLK *S_DATA, S_N, D_DATA, D_N*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
S_DATA	Вход	BOOL	I, Q, M, V, L
S_N	Вход	INT	I, Q, M, V, L, SM, T, C, AI, AQ, константа, курсор
D_DATA	Вход / Выход	BOOL	Q, M, V, L
D_N	Вход	INT	I, Q, M, V, L, SM, T, C, AI, AQ, константа, курсор

Эта команда перемещает число D_n непрерывных битов, начиная с D_DATA, вправо на S_N битов. Между тем, число S_N непрерывных битов, начиная с S_DATA, помещают в самый левый бит D_DATA.

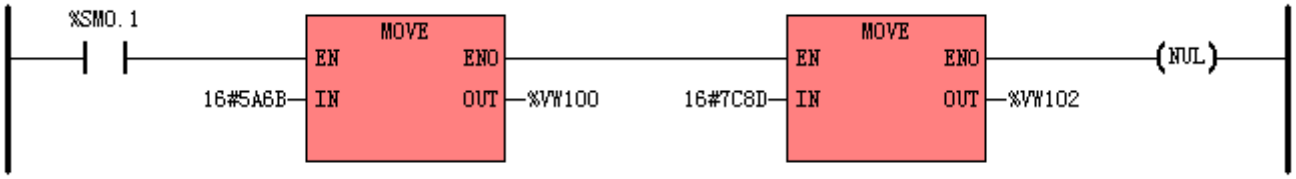
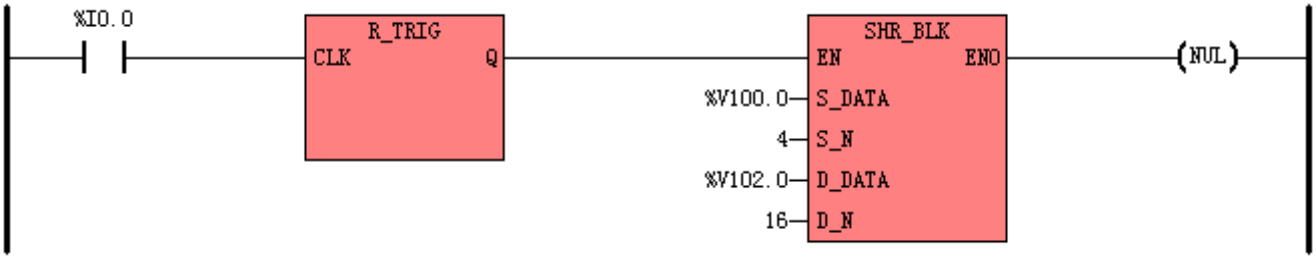
■ LD

Если EN равен 1, то эта инструкция выполнена.

■ IL

Если CR 1, эта инструкция выполняется, и это не влияет на CR.

★ Примеры

	Использование
LD	(* Network 0 *) (* Инициализация переменных *)  (* Network 1 *) (* переместить один раз в каждый фронт I0.0 *) 

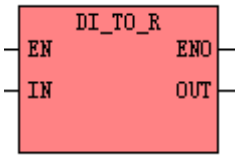
IL	(* Network 0 *)
	(* Инициализация переменных *)
	LD %SM0.1
	MOVE 16#5A6B, %VW100
	MOVE 16#7C8D, %VW102
	(* Network 1 *)
	(* переместить один раз в каждый фронт I0.0 *)
	LD %I0.0
	R_TRIG
	SHR_BLK %V100.0, 4, %V102.0, 16

Результат	**VW102** V103.7 … V102.0 **VW100** V101.7 … V100.0 Начальное значение: 0111 1100 1000 1101 | 0101 1010 0110 1011 После 1-го исполнения: 1011 0111 1100 1000 После 2-го исполнения: 1000 1101 0111 1100 После 3-го исполнения: 1101 1011 1011 0111

6.7 Инструкции конвертирования

6.7.1 DI_TO_R (DINT в REAL)

★ Описание

	Название	Использование	Группа	
LD	DI_TO_R			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	DI_TO_R	DI_TO_R *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	DINT	I, Q, M, V, L, SM, HC, константа
OUT	Выход	REAL	V, L

Эта инструкция преобразует значение DINT (IN) в значение REAL и присваивает результат в OUT.

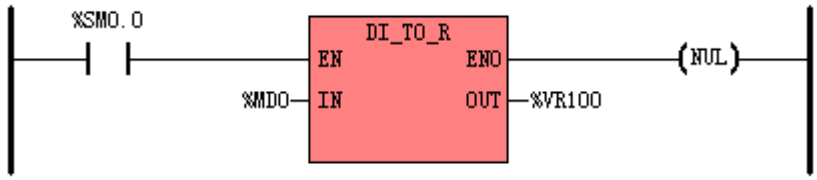
■ LD

Если EN равен 1, то эта инструкция выполнена.

■ IL

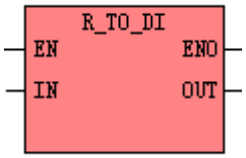
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому DI_TO_R всегда выполняется: значение MD0 преобразуется в значение REAL и назначается в VR100.
IL	LD %SM0.0 (* CR создается с SM0.0 *) DI_TO_R %MD0, %VR100 (* Значение MD0 преобразуется в значение REAL и назначается в VR100 *)	
Результат	**MD0** / **VR100** DI#123 / 123.0 DI#-9876 / -9876.0	

6.7.2 R_TO_DI (REAL в DINT)

★ Описание

	Название	Использование	Группа	
LD	R_TO_DI			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	R_TO_DI	R_TO_DI *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа
OUT	Выход	DINT	M, V, L, SM

Эта инструкция преобразует значение REAL (IN) в значение DINT и присваивает результат в OUT. Во время преобразования, десятичная дробь отбрасывается.

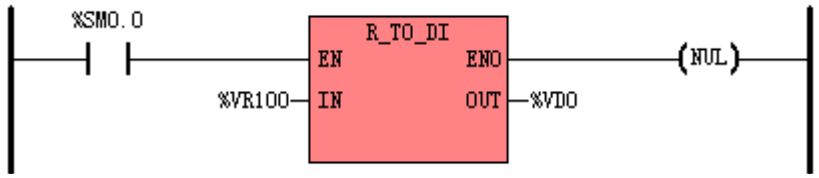
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

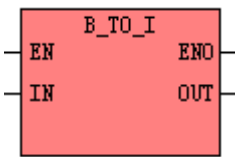
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому R_TO_DI всегда выполняется: значение VR100 преобразуется в значение DINT и назначается в VD0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) R_TO_DI %VR100, %VD0 (* Значение VR100 преобразуется в значение DINT и назначается в VD0 *)	
Результат	**VR100** / **VD0** 123.4 / DI#123 5213.6 / DI#5214	

6.7.3 B_TO_I (BYTE в INT)

★ Описание

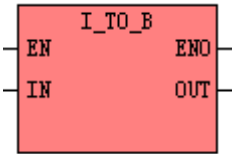
	Название	Использование	Группа	
LD	B_TO_I			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	B_TO_I	B_TO_I *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	I, Q, M, V, L, SM, константа
OUT	Выход	INT	Q, M, V, L, SM, AQ

Эта инструкция преобразует входной байт IN в целое число и присваивает результат в OUT.
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

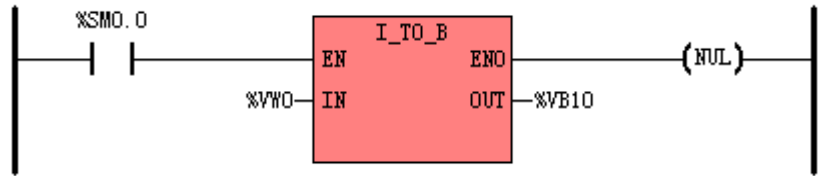
6.7.4 I_TO_B (INT в BYTE)

★ Описание

	Название	Использование	Группа	
LD	I_TO_B			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	I_TO_B	I_TO_B *IN, OUT*	U	

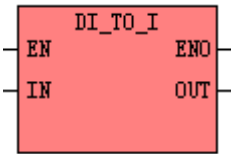
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	INT	I, Q, M, V, L, SM, AI, AQ, T, C, константа
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция присваивает младший входной байт IN в OUT.
■ LD
Если EN равен 1, то эта инструкция выполнена.
■ IL
Если CR равен 1, эта инструкция выполняется, и это не влияет на CR.
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому I_TO_B всегда выполняется: значение VW0 преобразуется в значение BYTE и назначается в VB10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) I_TO_B %VW0, %VB10 (* Значение VW0 преобразуется в значение BYTE и назначается в VB10 *)	
Результат	**VW0** / **VB10** 24 / B#24 255 / B#255 I#16#FFFD / B#16#FD	

6.7.5 DI_TO_I (DINT в INT)

★ Описание

	Название	Использование	Группа	
LD	DI_TO_I			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	DI_TO_I	DI_TO_I *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	DINT	I, Q, M, V, L, SM, HC, константа
OUT	Выход	INT	Q, M, V, L, SM, AQ

Эта инструкция присваивает наименьшее входное слово IN в OUT.

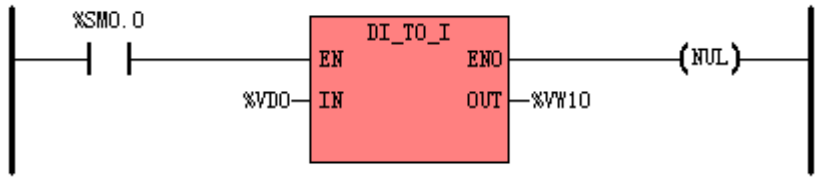
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

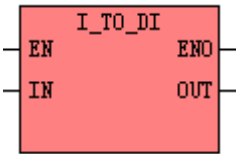
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому DI_TO_I всегда выполняется: младшее слово из VD0, преобразуется в значение INT и назначается VW10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) DI_TO_I %VD0, %VW10 (* Значение VD0 преобразуется в значение INT и назначается в VW10 *)	
Результат	**VD0** / **VW10** DI#12345 / 12345 DI#-234 / -234 DI#16#7A8B9C1D / I#16#9C1D	

6.7.6 I_TO_DI (INT в DINT)

★ Описание

	Название	Использование	Группа	
LD	I_TO_DI			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	I_TO_DI	I_TO_DI *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	INT	I, Q, M, V, L, SM, AI, AQ, T, C, константа
OUT	Выход	DINT	Q, M, V, L, SM

Эта инструкция преобразует входное целое значение IN в DINT и присваивает результат в OUT.

■ LD

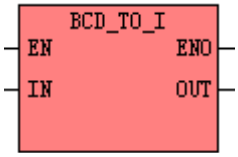
Если EN равен 1, то эта инструкция выполнена.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.7.7 BCD_TO_I (BCD в INT)

★ Описание

	Название	Использование	Группа	
LD	BCD_TO_I			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	BCD_TO_I	BCD_TO_I *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	WORD	I, Q, M, V, L, SM, константа
OUT	Выход	INT	Q, M, V, L, SM, AQ

Эта инструкция преобразует входное двоично-десятичного значение (IN) в целое число и присваивает результат в OUT.

Примечание: код 8421 будет принят для кода BCD. Допустимый диапазон IN: от 0 до 9999 BCD.

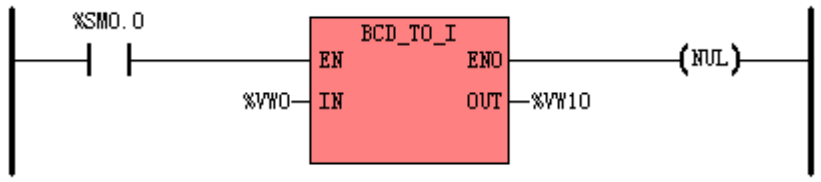
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

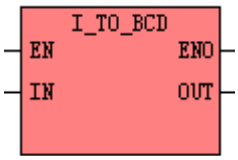
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому BCD_TO_I всегда выполняется: BCD преобразует VW0 в целое число и присваивает его в VW10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) BCD_TO_I %VW0, %VW10 (* Значение VW0 преобразуется в целое число и назначается в VW10 *)	
Результат	**VW0** / **VW10** 16#99 / 99 16#4567 / 4567 16#9999 / 9999	

6.7.8 I_TO_BCD (INT в BCD)

★ Описание

	Название	Использование	Группа	
LD	I_TO_BCD			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	I_TO_BCD	I_TO_BCD *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	INT	I, Q, M, V, L, SM, AI, AQ, T, C, константа
OUT	Выход	WORD	Q, M, V, L, SM

Эта инструкция преобразует входное целое число (IN) в значение двоично-десятичного кода и присваивает результат в OUT.

Примечание: код 8421 будет принят для кода BCD. Допустимый диапазон IN: от 0 до 9999.

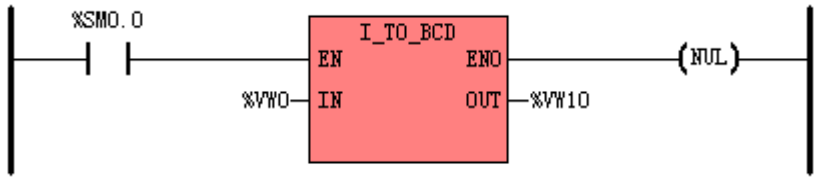
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR 1, эта инструкция выполняется, и это не влияет на CR.

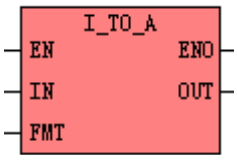
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому I_TO_BCD всегда выполняется: BCD преобразует VW0 в двоично-десятичное значение и присваивает его в VW10.

IL	LD %SM0.0 (* CR создается с SM0.0 *) I_TO_BCD %VW0, %VW10 (* Значение VW0 преобразуется в двоично-десятичное значение и назначается в VW10 *)
Результат	**VW0** / **VW10** 99 / 16#99 4567 / 16#4567 9999 / 16#9999

6.7.9 I_TO_A (INT в ASCII)

★ Описание

	Название	Использование	Группа	
LD	I_TO_A			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	I_TO_A	I_TO_A *IN, OUT, FMT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	INT	I, Q, M, V, L, SM, AI, AQ, T, C, константа
FMT	Вход	BYTE	I, Q, M, V, L, SM
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция преобразует целое значение (IN) в строку ASCII, а затем форматирует строку в соответствии с FMT и помещает результат в выходной буфер начиная с OUT. Результат преобразования положительного значения не включает в себя каких-либо знаков, а результат преобразования отрицательного значения начинается со знака минус (-).
OUT определяет начальный адрес буфера вывода, который занимает область памяти, 8 последовательных байтов. В буфере строки выравниваются по правому краю, а свободные байты заполняются пробелами (ASCII 32).
FMT используется для форматирования строки и правила показаны на рисунке ниже:

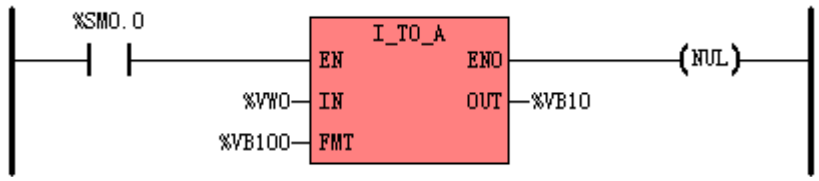
MSB LSB

7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

(1) NNN — Это поле определяет количество цифр дробной части.
Его допустимый диапазон 0 до 5,0 означает отсутствие дробной части.
(2) C — Это поле разделяет целую и дробную части:
0 для ввода десятичной точки (ASCII 46), и 1 для запятой (ASCII 44).
(3) — Высшие четыре бита должны быть равны нулю.

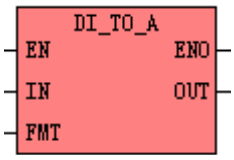
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому I_TO_A всегда выполняется: значение VW0 преобразуется в строку, форматируется и результат помещается в буфер, начиная с VB10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) I_TO_A %VW0, %VB10, %VB100 (* Значение VW0 преобразуется в строку, форматируется и результат помещается в буфер, начиная с VB10 *)	
Результат	**VB100** B#3 **VW0** 12 **Результат** **VB10** … **VB17**: 32 32 32 48 46 48 49 50 (“ ” “ ” “ ” “0” “.” “0” “1” “2”) **VW0** -23456 **Результат** **VB10** … **VB17**: 32 45 50 51 46 52 53 54 (“ ” “-” “2” “3” “.” “4” “5” “6”)	

6.7.10 DI_TO_A (DINT в ASCII)

★ Описание

	Название	Использование	Группа	
LD	DI_TO_A			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	DI_TO_A	DI_TO_A *IN, OUT, FMT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	DINT	I, Q, M, V, L, SM, HC, константа
FMT	Вход	BYTE	I, Q, M, V, L, SM
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция преобразует значение DINT (IN) в строку ASCII, а затем форматирует строку в соответствии с FMT и помещает результат в выходной буфер, начиная с OUT. Результат преобразования положительного значения не включает в себя каких-либо признаков, а результат преобразования отрицательного значения начинается со знака минус (-).
OUT определяет начальный адрес буфера вывода, который занимает в памяти 12 последовательных байтов. В буфере, строки выравниваются по правому краю, а свободные байты заполняются пробелами (ASCII 32).
FMT используется для форматирования строки и правила показаны на рисунке ниже:

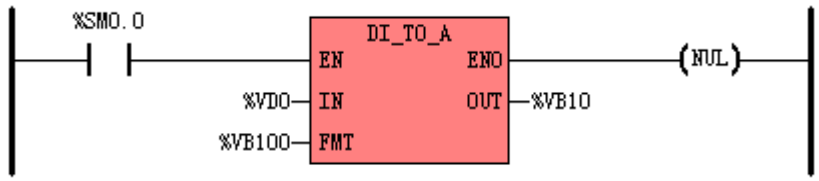
MSB LSB

7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

(1) NNN — Это поле определяет количество цифр дробной части.
Его допустимый диапазон 0 до 5,0 означает отсутствие дробной части.
(2) C — Это поле разделяет целую и дробную части:
0 для ввода десятичной точки (ASCII 46), и 1 для запятой (ASCII 44).
(3) — Высшие четыре бита должны быть равны нулю.

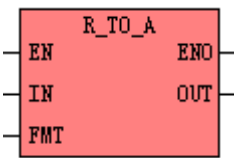
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому DI_TO_A всегда выполняется: значение VD0 преобразуется в строку, форматируется и результат помещается в буфер, начиная с VB10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) DI_TO_A %VD0, %VB10, %VB100 (* Значение VD0 преобразуется в строку, форматируется и результат помещается в буфер, начиная с VB10 *)	
Результат	**VB100**: B#3 / **VD0**: DI#12 / DI#123456 **Результат** VB10 … VB21: 32 32 32 32 32 32 32 48 46 48 49 50 (“ ” “ ” “ ” “ ” “ ” “ ” “ ” “0” “.” “0” “1” “2”) 32 32 32 32 45 49 50 51 46 52 53 54 (“ ” “ ” “ ” “ ” “-” “1” “2” “3” “.” “4” “5” “6”)	

6.7.11 R_TO_A (REAL в ASCII)

★ Описание

	Название	Использование	Группа	
LD	R_TO_A			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	R_TO_A	R_TO_A *IN, OUT, FMT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа
FMT	Вход	BYTE	I, Q, M, V, L, SM
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция преобразует значение REAL (IN) в строку ASCII, а затем форматирует строку в соответствии с FMT и помещает результат в выходной буфер, начиная с OUT. Результат преобразования положительного значения не включает в себя каких-либо признаков, а результат преобразования отрицательного значения начинается со знаком минус (-). Если цифры из десятичной части IN больше, чем NNN в FMT, в котором указываются цифры дробной части строки, то IN округляется перед преобразованием. В противном случае, если она меньше, чем NNN, недостающие цифры дробной части в строке заполнены 0.
OUT определяет начальный адрес буфера вывода, размер которого определяется в FMT. В буфере, строки выравниваются по правому краю, а свободные байты заполняются пробелами (ASCII 32).
FMT используется для форматирования строки и правила показаны на рисунке ниже:

MSB LSB

7	6	5	4	3	2	1	0
s	s	s	s	c	n	n	n

(1) NNN — Это поле определяет количество цифр дробной части.
Его допустимый диапазон 0 до 5,0 означает отсутствие дробной части.
(2) C — Это поле разделяет целую и дробную части:
0 для ввода десятичной точки (ASCII 46), и 1 для запятой (ASCII 44).
(3) SSSS — Это поле определяет размер буфера.
Его допустимый диапазон составляет от 3 до 15, и он должен быть больше, чем NNN

Примечание: Если длина строки результата превышает длину буфера вывода, то весь буфер будет заполнен пробелами (ASCII 32).

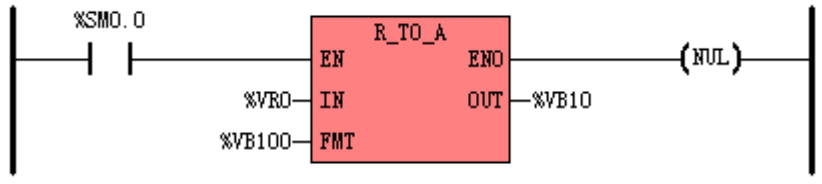
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

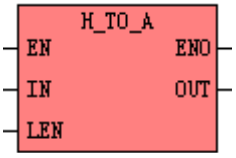
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому R_TO_A всегда выполняется: значение VR0 преобразуется в строку, форматируется и результат помещается в буфер, начиная с VB10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) R_TO_A %VR0, %VB10, %VB100 (* Значение VR0 преобразуется в строку, форматируется и результат помещается в буфер, начиная с VB10 *)	

Результат	**VB100**	**VR0**	**Результат**						
			VB10						**VB17**
	B#16#83	123.4	32	49	50	51	46	52	48	48
			“ ”	“1”	“2”	“3”	“.”	“4”	“0”	“0”
		-123.4567	45	49	50	51	46	52	53	55
			“-”	“1”	“2”	“3”	“.”	“4”	“5”	“7”

6.7.12 H_TO_A (шестнадцатеричное в ASCII)

★ Описание

	Название	Использование	Группа	
LD	H_TO_A			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	H_TO_A	H_TO_A *IN, OUT, LEN*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	I, Q, M, V, L, SM
LEN	Вход	BYTE	I, Q, M, V, L, SM, константа
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция преобразует число LEN шестнадцатеричных цифр, начиная с IN, в строку ASCII, и помещает строку в выходной буфер, начиная с OUT.

Примечание: Каждые 4 двоичных цифры составляют 1 шестнадцатеричную цифру, поэтому каждый входной байт включает в себя 2 шестнадцатеричные цифры, и так размер выходного буфера занимает LEN x 2 байта.

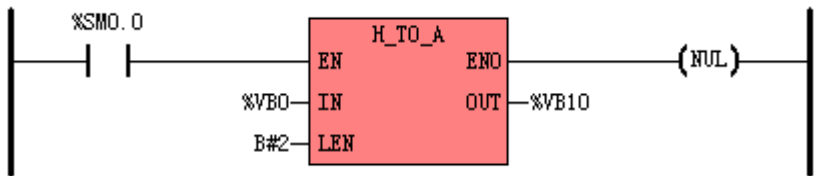
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

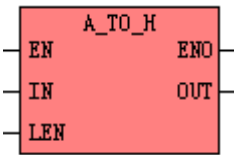
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому H_TO_A всегда выполняется: преобразует 2а байта шестнадцатеричных цифр, начиная с VB0, в строке и помещает результат в буфер, который занимает 4 непрерывных байта, начиная с VB10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) H_TO_A %VB0, %VB10, B#2 (* Преобразует 2а байта шестнадцатеричных цифр, начиная с VB0, в строке и помещает результат в буфер, который занимает 4 байта, начиная с VB10 *)	

Результат	**VB0**		**VB1**		**Результат**		
					VB10		**VB13**
	B#16#1A		B#16#2B		49	65	50	66
					“1”	“A”	“2”	“B”
	B#16#7C		B#16#8D		55	67	56	68
					“7”	“C”	“8”	“D”

6.7.13 A_TO_H (ASCII в шестнадцатеричное)

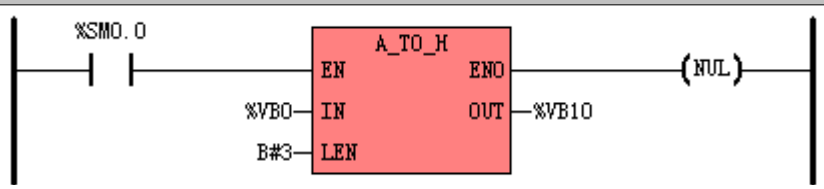
★ Описание

	Название	Использование	Группа	
LD	A_TO_H			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	A_TO_H	A_TO_H *IN, OUT, LEN*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	I, Q, M, V, L, SM
LEN	Вход	BYTE	I, Q, M, V, L, SM, константа
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция преобразует число LEN символов ASCII, начиная с IN, в шестнадцатеричные цифры, и помещает их на выходной буфер, начиная с OUT.
Примечание: Каждые 4 двоичных цифры составляют 1 шестнадцатеричную цифру, поэтому каждый входной байт, который обозначает ASCII символы, занимает 4 двоичных цифры памяти (т.е. половина байта) в выходном буфере.
Допустимый диапазон вводимых значений ASCII является: от B#16#30 до B#16#39 (расшифровываются цифры от 0 до 9), от B#16#41 до B#16#46 (для символов от A до F).
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

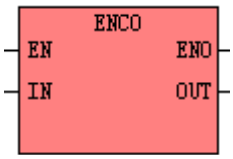
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому A_TO_H всегда выполняется: преобразует 3 байта в строке ASCII, начиная с VB0, в шестнадцатеричные цифры, и помещает результат в выходной буфер, начиная с VB10.

IL	LD %SM0.0 (* CR создается с SM0.0 *) A_TO_H %VB0, %VB10, B#3 (* Преобразует 3 байта в строке ASCII, начиная с VB0, в шестнадцатеричные цифры, и помещает результат в выходной буфер, начиная с VB10 *)
Результат	**VB0 VB1 VB2** — **VB10 VB11** 51 56 54 — B#16#38 B#16#6x "3" "8" "6" 55 65 49 — B#16#7A B#16#1x "7" "A" "1" Примечание: x обозначает для этой половины байта (4 бита) сохранённое исходное значение.

6.7.14 ENCO (кодирование)

★ Описание

	Название	Использование	Группа	
LD	ENCO			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ENCO	ENCO *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	WORD	I, Q, M, V, L, SM, константа
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция проверяет входное слово IN от младшего бита, и записывает номер первого бита равным 1 в выходной байт OUT.

Примечание: Если значение IN = 0, то результат будет бессмысленным.

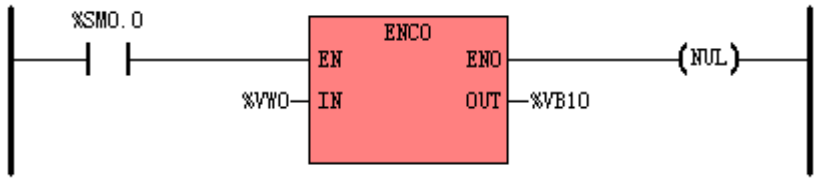
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

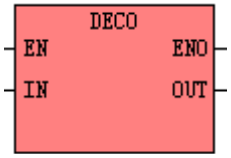
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому ENCO всегда выполняется: пишет номер первого бита, равного 1 в VB10.
IL	LD %SM0.0 (* CR создается с SM0.0 *) ENCO %VW0, %VB10 (* Преобразует 3 байта в строке ASCII, начиная с VB0, в шестнадцатеричные цифры, и помещает результат в выходной буфер, начиная с VB10 *)	

Результат	**VW0**																		**VB10**	
	(MSB)	15				12			9						4				0	(LSB)
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	B#9			
	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	B#4			

6.7.15 DECO (декодирование)

★ Описание

	Название	Использование	Группа	
LD	DECO			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	DECO	DECO *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	I, Q, M, V, L, SM, константа
OUT	Выход	WORD	Q, M, V, L, SM

Эта инструкция устанавливает бит в выходном слове OUT, которое соответствует номеру бита в наименее значимых "полубайтах" (4 бит) входного байта IN. Все остальные биты в OUT, сбрасываются.

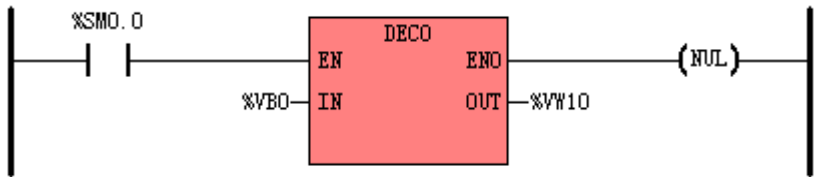
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

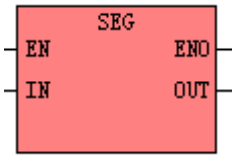
	Использование	Описание
LD		SM0.0 всегда включен, поэтому DECO всегда выполняется: устанавливает бит в VW10, который соответствует числу битов, младшего "полубайта" из VB0.

```
IL    LD      %SM0.0          (* CR создается с SM0.0 *)
      DECO    %VB0, %VW10     (* Преобразует 3 байта в строке ASCII, начиная с VB0, в
                                шестнадцатеричные цифры, и помещает результат в выходной буфер, начиная с VB10 *)
```

Результат	**VB0**								**VW10**								
		(MSB)	15			12			9				4			0	(LSB)
	B#9	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
	B#16#D4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	

6.7.16 SEG (7-сегментный дисплей)

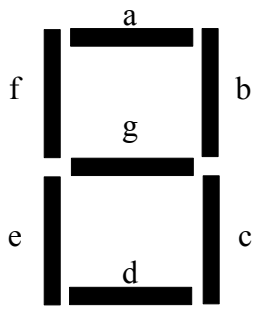
★ Описание

	Название	Использование	Группа	
LD	SEG			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SEG	SEG *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	I, Q, M, V, L, SM, константа
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция генерирует битовую комбинацию 7-сегментного дисплея в соответствии со значением, представленным наименее значимых "полубайтов" (4 бита) входного байта IN, а затем помещает результат в OUT.

IN (LSD)	Дисплей	OUT (- g f e d c b a)
0	0	0 0 1 1 1 1 1 1
1	1	0 0 0 0 0 1 1 0
2	2	0 1 0 1 1 0 1 1
3	3	0 1 0 0 1 1 1 1
4	4	0 1 1 0 0 1 1 0
5	5	0 1 1 0 1 1 0 1
6	6	0 1 1 1 1 1 0 1
7	7	0 0 0 0 0 1 1 1



IN (LSD)	Дисплей	OUT (- g f e d c b a)
8	8	0 1 1 1 1 1 1 1
9	9	0 1 1 0 0 1 1 1
A	A	0 1 1 1 0 1 1 1
B	B	0 1 1 1 1 1 0 0
C	C	0 0 1 1 1 0 0 1
D	D	0 1 0 1 1 1 1 0
E	E	0 1 1 1 1 0 0 1
F	F	0 1 1 1 0 0 0 1

■ LD

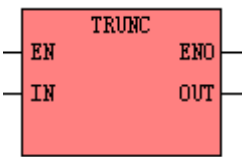
Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.7.17 TRUNC (отбрасывание)

★ Описание

	Название	Использование	Группа	
LD	TRUNC			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	TRUNC	TRUNC *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа
OUT	Выход	DINT	M, V, L, SM

Эта инструкция преобразует REAL значение IN в значение DINT и присваивает результат в OUT. Дробная часть из IN отбрасывается.

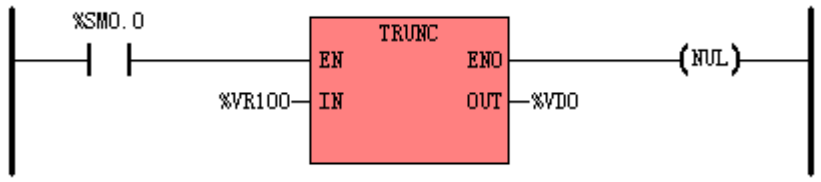
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

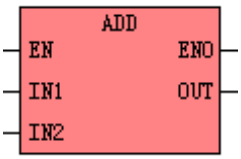
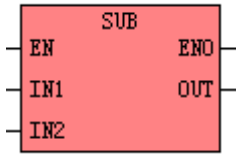
★ Примеры

	Использование	Описание
LD		SM0.0 всегда включен, поэтому TRUNC всегда выполняется: отсекает дробную часть VR100, преобразует результат в значение DINT и присваивает его VD0.
IL	LD %SM0.0 (* CR создается с SM0.0 *) TRUNC %VR100, %VD10 (* Отсекает дробную часть VR100, преобразует результат в значение DINT и присваивает его VD0 *)	
Результат	**VR100** 123.4 → **VD10** DI#123 5213.6 → DI#5213	

6.8 Цифровые инструкции

6.8.1 ADD и SUB (сложение и вычитание)

★ Описание

	Название	Использование	Группа	
LD	ADD			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	SUB			
IL	ADD	ADD *IN1, OUT*	U	
	SUB	SUB *IN1, OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
IN2	Вход	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
OUT	Выход	INT, DINT, REAL	Q, AQ, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

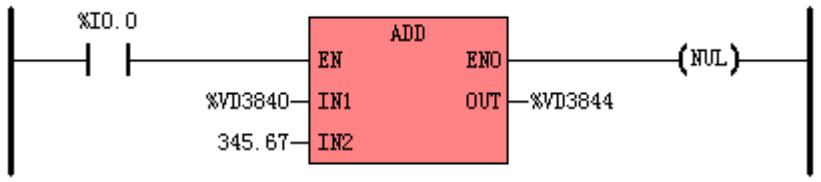
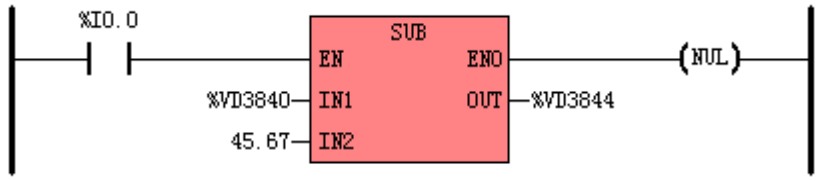
Если EN равен 1, инструкция ADD воспроизводит: OUT = IN1 + IN2, и инструкция SUB воспроизводит: OUT = IN1-IN2.

■ IL

IN1 и OUT должны быть того же типа данных.

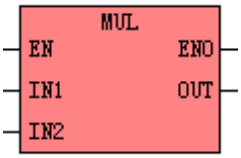
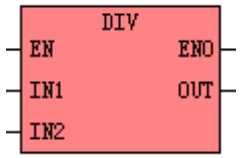
Если CR равен 1, инструкция ADD воспроизводит: OUT = OUT + IN1 и инструкция SUB воспроизводит: OUT = OUT - IN1. Инструкции ADD и SUB не влияют CR.

★ Примеры

<table>
<tr><th></th><th>Использование</th><th>Описание</th></tr>
<tr><td rowspan="2">LD</td><td> 
</td><td>Если I0.0 = 0: ADD не выполняется.
Если I0.0 = 1: команда складывает VD3840 и 345,67, и присваивает результат в VD3844.</td></tr>
<tr><td> 
</td><td>Если I0.0 = 0: SUB не выполняется.
Если I0.0 = 1: инструкция вычитает 45,67 из VD3840, и присваивает результат в VD3844.</td></tr>
<tr><td>IL</td><td colspan="2">LD %I0.0 (* CR создается с I0.0 *)
ADD 345.67, %VD3840 (* Если CR = 1: VD3840 = VD3840 +245.67 *)
(* Если CR = 0: инструкция выполняться не будет *)</td></tr>
<tr><td></td><td colspan="2">LD %I0.0 (* CR создается с I0.0 *)
SUB 45.67, %VD3840 (* If CR = 1: VD3840 = VD3840 - 45.67 *)
(* If CR = 0: инструкция выполняться не будет *)</td></tr>
</table>

6.8.2 MUL и DIV (умножение и деление)

★ Описание

	Название	Использование	Группа	
LD	MUL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	DIV			
IL	MUL	MUL *IN1, OUT*	U	
	DIV	DIV *IN1, OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
IN2	Вход	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
OUT	Выход	INT, DINT, REAL	Q, AQ, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

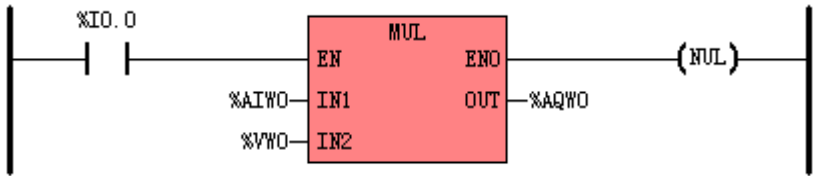
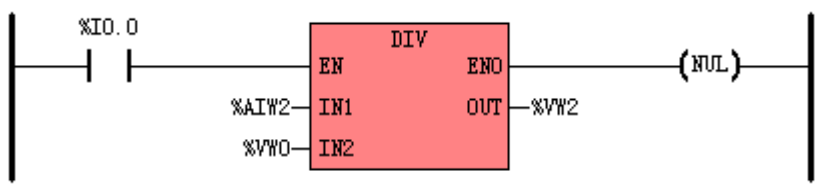
Если EN равен 1, инструкция MUL воспроизводит: OUT = IN1 × IN2, и инструкция DIV воспроизводит: OUT = IN1 ÷ IN2.

■ IL

IN1 и OUT должны быть того же типа данных.

Если CR равен 1, инструкция MUL воспроизводит: OUT = OUT × IN1 и инструкция DIV воспроизводит: OUT = OUT ÷ IN1. Инструкции MUL и DIV не влияют на CR.

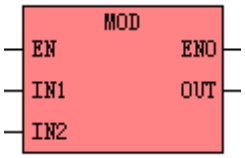
★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: MUL не выполняется. Если I0.0 = 1: инструкция умножает AIW0 и VW0, и присваивает результат в AQW0.
		Если I0.0 = 0: DIV не выполняется. Если I0.0 = 1: инструкция делит AIW2 на VW0, и присваивает результат в VW2.

IL	LD %I0.0 (* CR создается с I0.0 *) MUL %AIW0, %VW0 (* Если CR = 1: VW0 = VW0 × AIW0 *) (* Если CR = 0: инструкция выполняться не будет *)
	LD %I0.0 (* CR создается с I0.0 *) DIV %AIW2, %VW0 (* If CR = 1: VW0 = VW0 ÷ AIW2 *) (* If CR = 0: инструкция выполняться не будет *)

6.8.3 MOD (деление по модулю)

★ Описание

	Название	Использование	Группа	
LD	MOD			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	MOD	MOD *IN1, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
IN2	Вход	BYTE, INT, DINT	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
OUT	Выход	BYTE, INT, DINT	Q, AQ, M, V, L, SM, курсор

■ LD

IN1, IN2 и OUT должны быть того же типа данных.

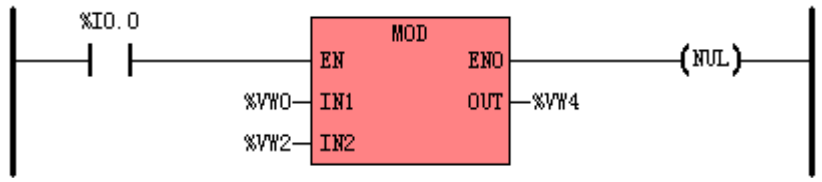
Если EN равен 1, то эта инструкция делит IN1 на IN2 и присваивает остаток на OUT.

■ IL

IN1 и OUT должны быть того же типа данных.

Если CR равен 1, эта инструкция делит OUT на IN1, и присваивает остаток на OUT. Это не влияет на CR.

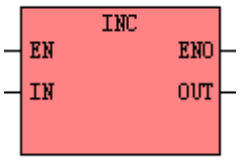
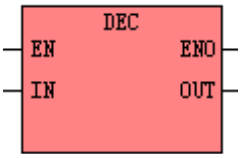
★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: MOD не выполняется. Если I0.0 = 1: VW0 делится на VW2, и остаток назначается в VW4.
IL	LD %SM0.0 (* CR создается с SM0.0 *) MOD %VW0, %VW4 (* Если CR = 1: VW4 делится на VW0, и остаток сохраняется в VW4 *) (* Если CR 0: эта команда не выполняется *)	

Результат	Для примера в LD, если инструкция MOD выполняется, результат показан в следующем виде: Адрес **VW0** **VW2** Значение 8 3 Адрес **VW4** Значение 2

6.8.4 INC and DEC

★ Описание

	Название	Использование	Группа	
LD	INC			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	DEC			
IL	INC	INC *OUT*	U	
	DEC	DEC *OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	BYTE, INT, DINT	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
OUT	Выход	BYTE, INT, DINT	Q, AQ, M, V, L, SM, курсор

■ LD

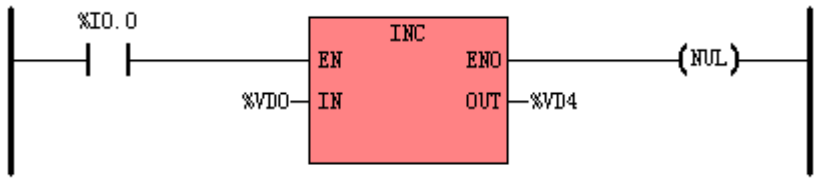
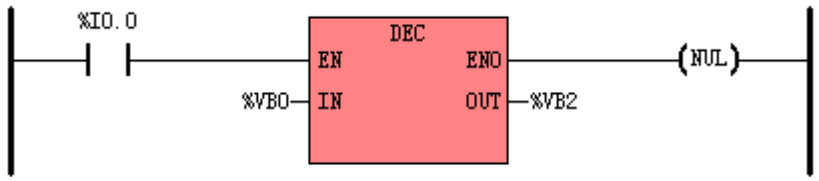
Вход и выход должны быть одинакового типа данных.

Если EN = 1, инструкция INC воспроизводит: OUT = IN + 1, и инструкция DEC воспроизводит: OUT = IN - 1

■ IL

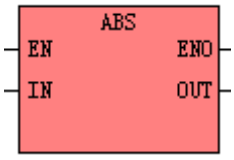
Если CR равен 1, инструкция INC воспроизводит: OUT = OUT + 1, и инструкция DEC воспроизводит: OUT = OUT - 1. Это не влияет на CR.

★ Примеры

	Использование	Описание
LD		Если I0.0 = 0: INC не выполняется. Если I0.0 = 1: VD4 = VD0 + DI#1.
		Если I0.0 = 0: DEC не выполняется. Если I0.0 = 1: VB2 = VB0 – B#1.
IL	LD %I0.0 (* CR создается с I0.0 *) INC %VD4 (* Если CR = 1: VD4 =VD4 + DI#1 *) (* Если CR = 0: инструкция выполняться не будет *)	
	LD %I0.0 (* CR создается с I0.0 *) DEC %VB2 (* If CR = 1: VB2 = VB2 - B#1 *) (* If CR = 0: инструкция выполняться не будет *)	

6.8.5 ABS (абсолютное значение)

★ Описание

	Название	Использование	Группа	
LD	ABS			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	ABS	ABS *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	INT, DINT, REAL	I, Q, AI, AQ, M, V, L, SM, T, C, HC, константа, курсор
OUT	Выход	INT, DINT, REAL	Q, AQ, M, V, L, SM, курсор

IN и OUT должны быть того же типа данных.
Эта инструкция вычисляет абсолютное значение на входе IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = | IN |.

■ LD

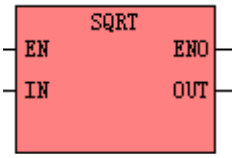
Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.8.6 SQRT (квадратный корень)

★ Описание

	Название	Использование	Группа	
LD	SQRT			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SQRT	SQRT *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа, курсор
OUT	Выход	REAL	V, L, курсор

Эта инструкция вычисляет квадратный корень входа IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = √ IN.

■ LD

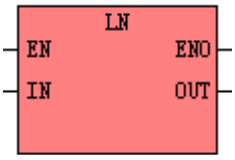
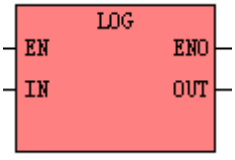
Если EN равен 1, то эта инструкция выполняется.

■ IL

Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.8.7 LN (натуральный логарифм), LOG (логарифм)

★ Описание

	Название	Использование	Группа	
LD	LN			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	LOG			
IL	LN	LN *IN, OUT*	U	
	LOG	LOG *IN, OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа, курсор
OUT	Выход	REAL	V, L, курсор

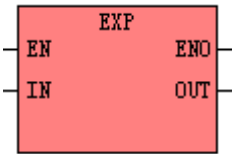
Инструкция LN вычисляет натуральный логарифм входа IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = LN (IN).

Инструкция LOG вычисляет логарифм входа IN, и присваивает результат в OUT, как показано в следующей формуле: $OUT = log_{10} (IN)$.

■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.8.8 EXP (экспонента с базовой "e")

★ Описание

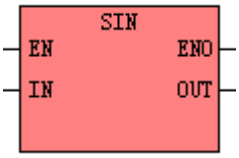
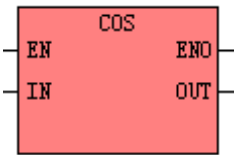
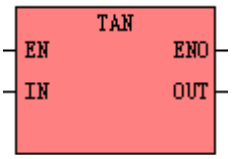
	Название	Использование	Группа	
LD	EXP			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	EXP	EXP *IN, OUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа, курсор
OUT	Выход	REAL	V, L, курсор

Эта инструкция вычисляет экспоненту с базовой "e" входа IN, и присваивает результат в OUT, как показано в следующей формуле: $OUT = e^{IN}$
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.8.9 SIN (синус), COS (косинус), TAN (тангенс)

★ Описание

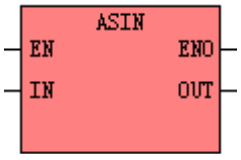
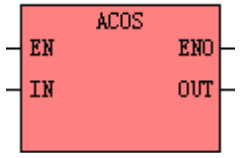
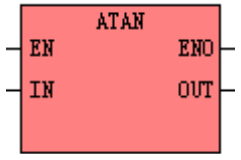
	Название	Использование	Группа	
LD	SIN			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	COS			
	TAN			
IL	SIN	SIN *IN, OUT*	U	
	COS	COS *IN, OUT*		
	TAN	TAN *IN, OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа, курсор
OUT	Выход	REAL	V, L, курсор

Инструкция SIN вычисляет значение синуса на входе IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = SIN (IN).
Инструкция COS вычисляет значение косинуса на входе IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = COS (IN).
Инструкция TAN вычисляет значение тангенса на входе IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = TAN (IN).
■ LD
Если EN равен 1, то эта инструкция выполнена.
■ IL
Если CR равен 1, эта инструкция выполняется, и это не влияет на CR.

6.8.10 ASIN (арксинус), ACOS (арккосинус), ATAN (арктангенс)

★ Описание

	Название	Использование	Группа	
LD	ASIN			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	ACOS			
	ATAN			
IL	ASIN	ASIN *IN, OUT*	U	
	ACOS	ACOS *IN, OUT*		
	ATAN	ATAN *IN, OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	REAL	V, L, константа, курсор
OUT	Выход	REAL	V, L, курсор

Инструкция ASIN вычисляет значение арксинуса на входе IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = ARCSIN (IN).
Инструкция ACOS вычисляет значение арккосинуса на входе IN, и присваивает результат в OUT, как показано в следующей формуле: OUT = ARCCOS (IN).
Инструкция ATAN вычисляет значение арктангенса на входе IN, и присваивает результат в OUT, как

показано в следующей формуле: OUT = ARCTAN (IN).

■ LD

Если EN равен 1, то эта инструкция выполнена.

■ IL

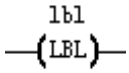
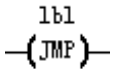
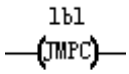
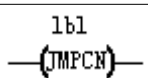
Если CR равен 1, эта инструкция выполняется, и это не влияет на CR.

6.9 Программы контроля

В IL инструкции перехода и инструкции возврата не влияют на CR, поэтому CR остаётся неизменной только после выполнения команды перехода или команды возврата, и вы должны уделять больше внимания при их использовании.

6.9.1 Инструкции LBL и JMP

★ Описание

	Название	Использование	Группа	
LD	LBL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	JMP			
	JMPC			
	JMPCN			
IL	LBL	*lbl:*	U	
	JMP	JMP *lbl*		
	JMPC	JMPC *lbl*		
	JMPCN	JMPCN *lbl*		

Операнд	Описание
lbl	Действующий идентификатор

■ LD

Инструкция **LBL** используется для определения метки на текущей позиции, и метка будет функционировать в качестве цели для инструкций перехода. Переопределение идентификатора метки запрещено. Эта инструкция выполняется безусловно, поэтому вам не нужно добавлять какие-либо элементы с левой стороны от неё. На самом деле, Kinco Builder будет игнорировать все элементы слева от неё.

Инструкция **JMP** используется для безусловного переноса выполнения программы к метке network указанной в LBL.

Инструкция **JMPC** используется для передачи выполнения программы к метке network указанной в LBL, когда состояние горизонтальной линии с лева является true.

Инструкция **JMPCN** используется для передачи выполнения программы к метке network указанной в LBL, когда состояние горизонтальной линии с лева является false.

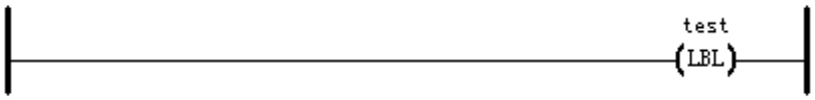
Инструкция перехода и его метка места назначения всегда должны быть в одной и той же POU.

■ IL

Определение формата метки является допустимый идентификатор: определение занимает независимую

линию. Переопределение идентификатора метки запрещено.
Инструкция **JMP** используется для безусловной передачи выполнения программы к метке, указанной в LBL.
Инструкция **JMPC** используется для передачи выполнения программы к метке указанной в LBL, при CR= 1.
Инструкция **JMPCN** используется для передачи выполнения программы к метке указанной LBL, при CR=0.
Инструкция перехода и его метка места назначения всегда должны быть в одной и той же POU.

★ Примеры

LD	IL
(* Network 0 *)  _ _ _ (* Network 4 *)  	(* Network 0 *) test: ... (* Network 4 *) LD %I0.0 JMPC test

6.9.2 Инструкции возврата

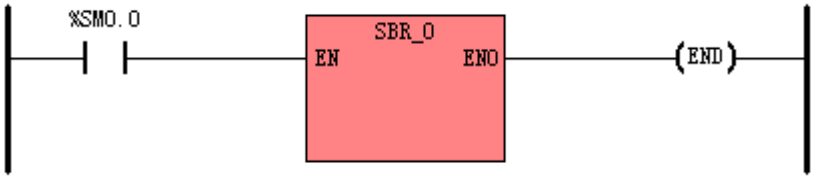
Примечание: Инструкции возврата могут быть использованы только в подпрограммах и прерываниях.

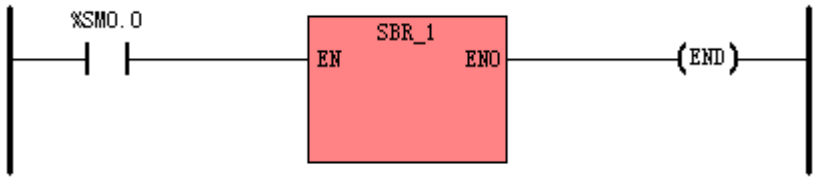
★ Описание

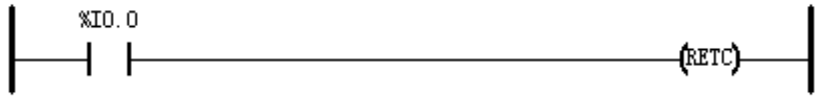
	Название	Использование	Группа	
LD	RETC			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	RETCN			
IL	RETC	RETC	U	
	RETCN	RETCN		

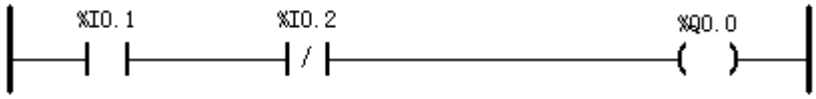
■ LD
Инструкция **RETC** используется для завершения подпрограммы или прерывания и передачи выполнения программы обратно к вызывающему входу, когда состояние горизонтальной линии с лева является true.
Инструкция **RETCN** используется для завершения подпрограммы или прерываний и передачи выполнения программы обратно к вызывающему входу, когда состояние горизонтальной линии с лева является false.
■ IL
Инструкция **RETC** используется для завершения подпрограммы или прерывания и передачи выполнения программы обратно к вызывающему входу, когда CR = 1.
Инструкция **RETCN** используется для завершения подпрограммы или прерывания и передачи выполнения программы обратно к вызывающему входу, когда CR = 0.

★ Примеры

<table>
<tr><th></th><th>Использование</th><th>Описание</th></tr>
<tr><td>LD</td><td>Основная программа:
(* Network 0 *)
 

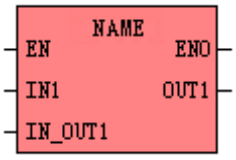
(* Network 1 *)
 

Подпрограмма SBR_0:
(* Network 0 *)
 

(* Network 1 *)
 
</td><td>Для SBR_0:
Если I0.0 равно 0, команды выполняются последовательно. Если I0.0 равно 1, выполнение программы передается обратно к вызывающей записи в основной программе, и Kinco-K5 продолжает выполнять инструкции в Network 1.</td></tr>
<tr><td>IL</td><td colspan="2">Основная программа:
LD %SM0.0 (* CR создается с SM0.0 *)
CAL SBR_0 (* Вызов SBR_0 *)
CAL SBR_1 (* Вызов SBR_1 *)
SBR_0:
LD %I0.0 (* CR создается с I0.0 *)
RETC (* Если CR = 1, SBR_0 должен будет прерван и выполнение программы перейдёт обратно к вызывающему входу в основной программе. *)
LD %I0.1 (* Если RETC не выполняется, то последующие инструкции должны быть выполнены *)
ANDN %I0.2
ST %Q0.0</td></tr>
</table>

6.9.3 CAL (Вызов подпрограммы)

★ Описание

	Название	Использование	Группа	
LD	CAL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CAL	CAL *Имя подпрограммы, IN1, IN_OUT1, OUT1*	U	

Эта инструкция используется для вызова и выполнения подпрограммы с указанным именем. Подпрограмма, которая будет вызываться, уже должна существовать в пользовательской программе.
Вы можете использовать команду CAL с параметрами или без параметров. Если инструкция CAL используется с параметрами, тип данных и тип переменных фактических параметров, должны совпадать с формальными параметрами, которые определены в таблице локальных переменных вызываемой подпрограммы. Кроме того, порядок фактических параметров должны быть таким же, как и у формальных параметров.

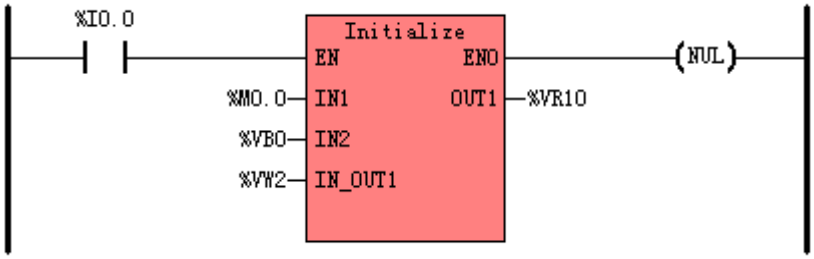
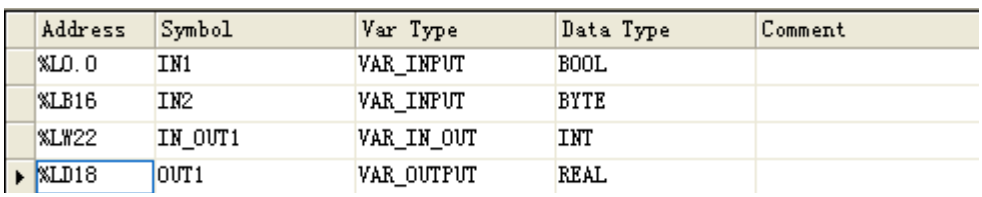
■ LD

Все имена подпрограмм отображаются в группе [SBR] в дереве [LD instructions]. Дважды щелкните на имени, что бы соответствующая подпрограмма добавилась в вашу программу. Если EN равен 1, эта подпрограмма выполняется.

■ IL

Если CR равен 1, подпрограмма будет вызываться и выполняется. Команда CAL не влияет на CR, но CR могут быть изменены в подпрограмме.

★ Примеры

	Использование
LD	**Основная программа** (* Network 0 *) (* вызов подпрограммы "Initialize" *)  **Таблица Локальных переменных подпрограммы "Initialize":** 

	Address	Symbol	Var Type	Data Type	Comment
	%L0.0	IN1	VAR_INPUT	BOOL	
	%LB16	IN2	VAR_INPUT	BYTE	
	%LW22	IN_OUT1	VAR_IN_OUT	INT	
▶	%LD18	OUT1	VAR_OUTPUT	REAL	

IL

Основная программа:

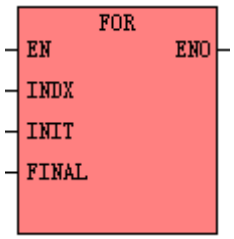
```
(* Network 0 *)
(* вызов подпрограммы 'Initialize' *)
LD      %I0.0
CAL     Initialize, %M0.0, %VB0, %VW2, %VR10
```

Таблица Локальных переменных подпрограммы "Initialize":

Address	Symbol	Var Type	Data Type	Comment
%LO.0	IN1	VAR_INPUT	BOOL	
%LB16	IN2	VAR_INPUT	BYTE	
%LW22	IN_OUT1	VAR_IN_OUT	INT	
%LD18	OUT1	VAR_OUTPUT	REAL	

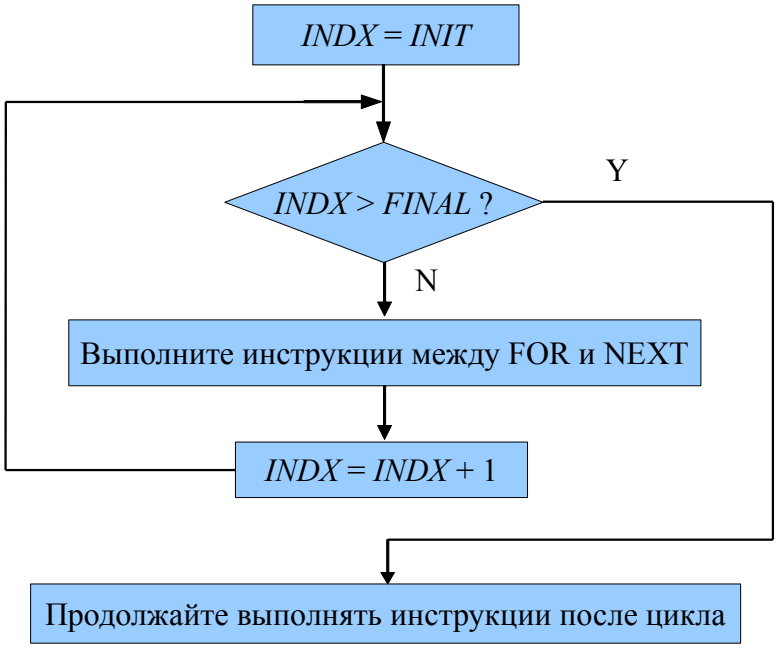
6.9.4 FOR / NEXT (FOR / NEXT цикл)

★ Описание

	Название	Использование	Группа	
LD	FOR			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	NEXT			
IL	FOR	FOR *INDX, INIT, FINAL*	U	
	NEXT	NEXT		

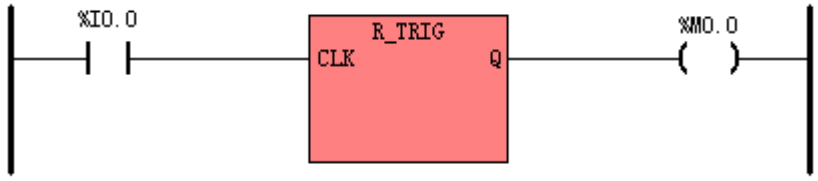
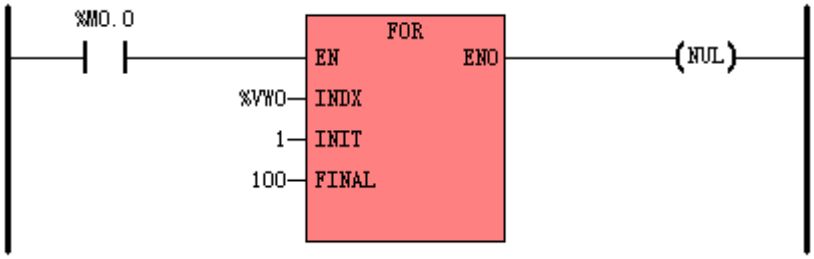
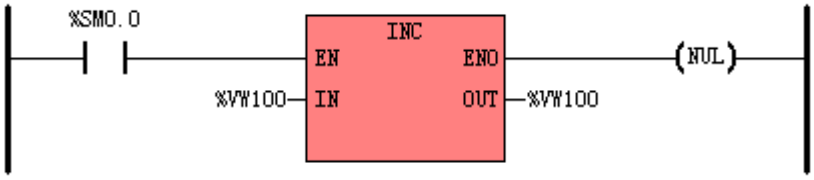
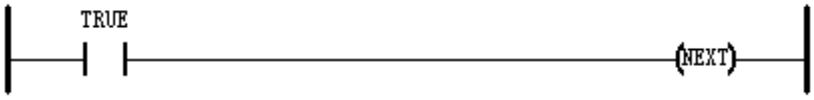
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
INDX	Вход	INT	M, V, L, SM
INIT	Вход	INT	M, V, L, SM, T, C, константа
FINAL	Выход	INT	M, V, L, SM, T, C, константа

Инструкции FOR / NEXT выражают цикл, который повторяется для указанного счетчика. Вы указываете количество циклов (INDX), начальное значение (INIT), и конечное значение (FINAL).
Инструкция NEXT отмечает конец цикла, а инструкция FOR выполняет инструкции между FOR и NEXT. Они должны использоваться в паре, каждая команда FOR требует команду NEXT.
Если цикл FOR / NEXT, существует в другом цикле FOR / NEXT, то это называется вложенный цикл. Вы можете вкладывать FOR / NEXT циклы до восьми раз.
Процесс выполнение FOR / NEXT цикла показано на следующем рисунке:



При использовании инструкции FOR / NEXT, необходимо заметить следующие детали:
★ Инструкция FOR должна быть второй инструкцией в Network.
★ Инструкция NEXT должна монополизировать Network.
★ Вы можете изменить окончательное значение внутри самого цикла, чтобы изменить условие окончания цикла.
★ Цикл, который должен выполняться в течение длительного времени, которое превышает время сторожевого таймера CPU, может приводить к перезагрузке CPU.
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет CR.

★ Примеры

	Использование
LD	(* Network 0 *) (* По переднему фронту I0.0, цикл выполняется 100 раз *)  (* Network 1 *)  (* Network 2 *)  (* Network 3 *) 
IL	(* Network 0 *) (* По переднему фронту I0.0, цикл выполняется 100 раз *) LD %I0.0 R_TRIG ST %M0.0 (* Network 1 *) LD %M0.0 FOR %VW0, 1, 100 (* Network 2 *) LD %SM0.0 INC %VW100 (* Network 3 *) LD TRUE NEXT

6.9.5 END (Завершение цикла сканирования)

★ Описание

	Название	Использование	Группа	
LD	END	—(END)—		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	END	END	U	

Эта команда может использоваться только в главной программе, для завершения текущего цикла сканирования. В конце основной программы, Kinco Builder автоматически вызывает команду END.

■ LD

Если на горизонтальной линии связи с слева присутствует 1, эта инструкция выполняется. В противном случае, эта команда не вступит в силу.

■ IL

Если CR равно 1, эта инструкция будет выполнена. В противном случае, эта команда не вступит в силу. Эта инструкция не влияет CR.

6.9.6 STOP (Стоп CPU)

★ Описание

	Название	Использование	Группа	
LD	STOP	—(STOP)—		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	STOP	STOP	U	

Эта инструкция прекращает выполнение вашей программы и CPU сразу же переходит из режима RUN в режим STOP.

■ LD

Если на горизонтальной линии связи с слева присутствует 1, эта инструкция выполняется. В противном случае, эта команда не вступит в силу.

■ IL

Если CR равен 1, эта инструкция выполняется. В противном случае, эта команда не вступили в силу. Эта инструкция не влияет CR.

6.9.7 WDR (Сторожевой таймер сброса)

★ Описание

	Название	Использование	Группа	
LD	WDR	—(WDR)—		☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	WDR	WDR	U	

Эта инструкция повторно запускает систему сторожевого таймера в CPU.

Используя команду WDR можно увеличить время цикла сканирования, не приводя к ошибке сторожевого таймера, поэтому программа, которой нужно больше времени, может быть успешно выполнена. Но вы должны использовать эту инструкция осторожно, потому что следующие процессы блокируются до тех пор,

пока цикл сканирования не будет завершён:
★ Само диагностика CPU
★ Чтение входов (замеряет все физические входные каналов и записывает эти значения в области ввода изображения)
★ Связь
★ Запись выходов (записывает значения, хранящиеся в области результата изображения с физическими выходными каналами)
★ Время для таймеров 10 мс и 100 мс
■ LD
Если на горизонтальной линии связи с слева присутствует 1, эта инструкция выполняется. В противном случае, эта команда не вступит в силу.
■ IL
Если CR равен 1, эта инструкция выполняется. В противном случае, эта команда не вступит в силу. Эта инструкция не влияет на CR.

6.10 Инструкции прерываний

Целью использования методики прерывания, является повышение эффективности исполнения Kinco-K5 быстро реагировать на специальные внутренние или внешние предопределенные события. Kinco-K5 поддерживает десятки событий, каждому из которых присваивается уникальный номер события.

6.10.1 Как Kinco-K5 обрабатывает прерывания

Подпрограмма обработки прерывания выполняется только один раз в каждом случае события прерывания, связанного с ним. После того, как последняя команда программы обработки прерывания была выполнена, выполнение программы передается обратно в основную программу. Вы можете выйти из подпрограммы, выполнив инструкции RETC или RETCN.
Методика прерывания делает Kinco-K5 способным быстро реагировать на специальные события, поэтому следует оптимизировать процедуры обработки прерываний, они должны быть короткими и эффективными.

6.10.2 Приоритеты прерывания и очереди

Разные события с разными уровнями приоритета. Когда происходят события прерывания, они будут стоять в очереди в соответствии с их уровнями приоритета и временной последовательности: события прерывания в одной очереди группы обрабатываются по принципу "первый пришел, первый обслужен"; события с более высоким приоритетом группы обрабатываются преимущественно. Только одна процедура прерывания может быть выполнена в один момент времени. После того, как программа обработки прерывания начинает выполняться, она не может быть прервана другой процедурой прерывания. Событие прерывания, которое происходит в то время, как другая подпрограмма прерывания выполняется, ставится в очередь для более позднего обращения.

6.10.3 Типы событий прерываний, поддерживаемых Kinco-K5

Kinco-K5 поддерживает следующие типы событий прерывания:
★ Прерывания порта связи
Этот тип прерываний имеет самый высокий приоритет.
Он используется в режиме свободного протокола связи. Прием и передача прерывания облегчает вам полный контроль коммуникации.
★ Прерывания I / O
Этот тип прерываний имеет средний приоритет.

Эти прерывания включают прерывания TRIG, прерывания HSC и PTO прерывания.
В прерываниях TRIG могут быть захвачены только первые четыре канала DI (%I0.0 ~ %I0.3) корпуса CPU. Каждый из них может быть использован, для уведомления, что состояние сигнала изменилось и PLC должен ответить немедленно .
Прерывания HSC возникают, когда значение счетчика достигает заданного значения, изменения направления счета или счетчик сброшен извне. Каждый из них позволяет PLC реагировать в режиме реального времени высокоскоростных событий, на которые не может сразу же ответить на скорости сканирования.
PTO прерывания возникают сразу же после завершения вывода заданного количества импульсов. Типичное применение состоит в управлении шаговым двигателем.
★ Прерывания времени
Этот тип прерываний имеет самый низкий приоритет.
Это прерывание включает в себя своевременные прерывания, и прерывания таймер T2 и T3.
Своевременные прерывания возникают периодически (единица измерения: мс), и могут быть использованы для периодических задач.
Прерывание от таймера происходит сразу, когда текущее значение T2 или T3 достигает заданного значения. Оно может быть использовано, чтобы своевременно реагировать на конце указанного интервала времени.

6.10.4 Список событий прерываний

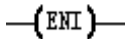
Таблица 6-1 События прерывания

Номер события	Описание	Тип
34	PORT 2: XMT полный	Прерывания порта связи
33	PORT 2: RCV полный	
32	PORT 1: XMT полный	
31	PORT 1: RCV полный	
30	PORT 0: XMT полный	
29	PORT 0: RCV полный	
28~27	Reserved	Прерывания I/O
26	I0.0, по заднему фронту	
25	I0.0, по переднему фронту	
24	I0.1, по заднему фронту	
23	I0.1, по переднему фронту	
22	I0.2, по заднему фронту	
21	I0.2, по переднему фронту	
20	I0.3, по заднему фронту	
19	I0.3, по переднему фронту	
18	HSC0 CV=PV	
17	Изменилось направление HSC0	
16	Внешний сброс HSC0	
15	HSC1 CV=PV	
14	Изменилось направление HSC1	
13	Внешний сброс HSC1	

12~5	Reserved	
4	Циклическое прерывание 1. Период, указан в SMW16, ед. изм.: мс.	Временные прерывания
3	Циклическое прерывание 0. Период, указан в SMW12, ед. изм.: мс.	
2	Timer T3 ET=PT	
1	Timer T2 ET=PT	

6.10.5 ENI (включение прерывания), DISI (отключение прерывания)

★ Описание

	Название	Использование	Группа	
LD	ENI			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	DISI			
IL	ENI	ENI	U	
	DISI	DISI		

Инструкция ENI позволяет глобальную обработку всех подключенных событий прерывания.
Инструкция DISI запрещает глобальную обработку всех событий прерывания.
Когда вы включаете процессор в режиме RUN, прерывание обрабатывается по умолчанию.

■ LD

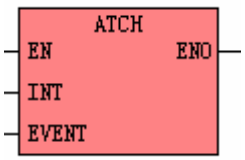
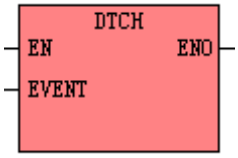
Если на горизонтальной линии связи с слева присутствует 1, команда выполняется. В противном случае, команда не вступит в силу.

■ IL

Если CR равно 1, команда выполняется. В противном случае, команда не вступит в силу. Инструкция не влияет на CR.

6.10.6 ATCH и DTCH инструкции

★ Описание

	Название	Использование	Группа	
LD	ATCH			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	DTCH			
IL	ATCH	ATCH *INT, EVENT*	U	
	DTCH	DTCH *EVENT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
INT	Вход		Имя существующей подпрограммы прерывания
EVENT	Вход	INT	Константа, № события прерывания

■ LD

Если EN равен 1, инструкция ATCH присоединяет событие прерывания (определяется номером события EVENT) к прерыванию (имя подпрограммы указано в INT) и включает событие прерывания. После этого команда выполняется, программа обработки прерывания будет автоматически вызываться при наступлении события прерывания. Вы можете прикрепить несколько событий одного обработчика прерывания, но одно событие может быть присоединено только к одной подпрограмме прерывания.

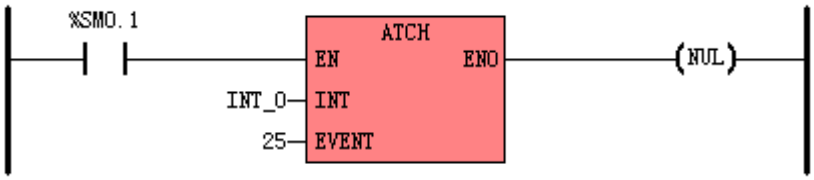
Если EN равен 1, инструкция DTCH разрывает соединение между событием прерывания (определяется номером события EVENT) и его прерыванием, и делает событие прерывания возврата, отключенным.

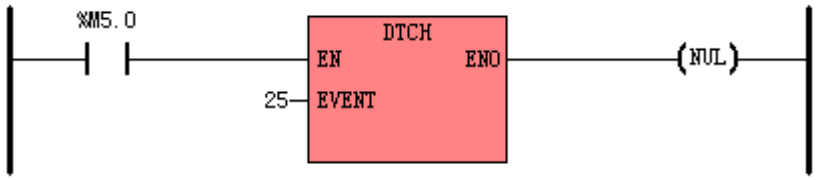
■ IL

Если CR равен 1, инструкция ATCH присоединяет событие прерывания (определяется номером события EVENT) к прерыванию (имя подпрограммы указано в INT) и включает событие прерывания. Эта инструкция не влияет на CR.

Если CR равен 1, инструкция DTCH разрывает соединение между событием прерывания (определяется номером события EVENT) и его прерыванием, и делает событие прерывания возврата, отключенным. Эта инструкция не влияет на CR.

★ Примеры

<table>
<tr><th></th><th>Использование</th></tr>
<tr><td>LD</td><td>(* Network 0 *)
(* На первом цикле, событие №25 включено и присоединено к подпрограмме INT_0 *)
 

(* Network 1 *)
(* Если M5.0 равно 1, событие №25 отключено *)
 
</td></tr>
<tr><td>IL</td><td>(* Network 0 *)
LD %SM0.1
ATCH INT_0, 25 (* На первом цикле, событие №25 включено и присоединено к подпрограмме INT_0 *)
LD %M5.0 (* CR создается с M5.0 *)
DTCH 25 (* Если CR равно 1, событие №25 отключено *)</td></tr>
</table>

6.11 Инструкции часов

Часы реального времени (RTC) построены в модуле CPU в режиме реального времени отображения часов / календаря. Часы реального времени / календарь примет BCD - формат кодирования от секунды до года, автоматически выполняет настройки високосного года и использует конденсатор большой емкости в качестве резервного питания. При нормальной температуре, продолжительность работы супер конденсатора 72 часа.

6.11.1 Настройка RTC онлайн

Вы должны настроить RTC на текущую фактическую дату и время, прежде чем использовать его. Перед настройкой, значение RTC может быть случайным.
Выполните [PLC]> [Time of Day Clock…] в меню команд, чтобы открыть "Time of Day Clock…" диалог для настройки RTC онлайн, как показано на следующем рисунке.

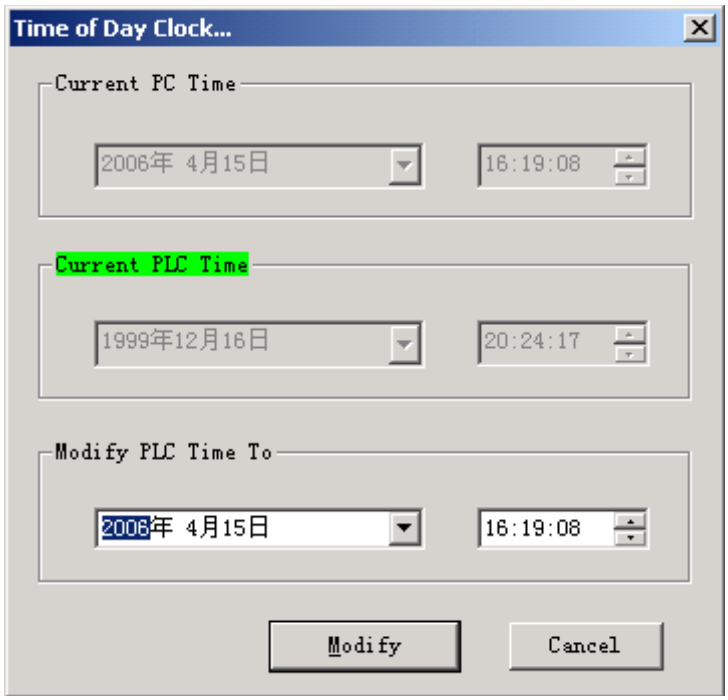
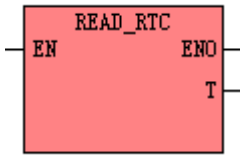
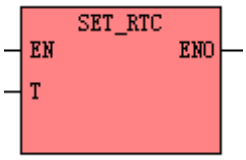


Рисунок 6-1 Настройка RTC

★ **Current PC Time**: Указывает текущую дату и время текущего ПК.
★ **Current PLC Time**: Указывает текущую дату и время RTC онлайн-модуля CPU. Зеленый фон означает, что модуль CPU общается с компьютером успешно, а фон желтого цвета означает, что модуль CPU не может общаться с компьютером.
★ **Modify PLC Time To**: Вы можете ввести желаемую дату и время для RTC здесь. Введите их с клавиатуры, или щелкните на стрелке в правой части соответствующего окна, чтобы выбрать дату или отрегулировать время.
★ **Adopt Summer Time**: Вы можете нажать на этот элемент, когда необходимо принять летнее время.
★ **Modify**: Нажмите эту кнопку, что бы записать дату и время, которую вы ввели, в модуль CPU, а затем RTC должна быть скорректирована до желаемой даты и времени.
Примечание: Если будет принято Летнее время, то требуется перезагрузить Kinco builder, что бы изменения вступили в силу.

6.11.2 READ_RTC и SET_RTC

★ Описание

	Название	Использование	Группа	
LD	READ_RTC			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	SET_RTC			
IL	READ_RTC	READ_RTC *T*	U	
	SET_RTC	SET_RTC *T*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
T	Вход (SET_RTC)	BYTE	V
	Выход (READ_RTC)		

Инструкция READ_RTC используется для чтения текущей даты и времени из RTC и записи их на буфера времени 8 байтов, начинающийся с адреса ***T***.
Инструкция SET_RTC используется для записи даты и времени, указанного в 8 байтах буфера времени начиная с адреса ***T*** в RTC.
Формат хранения даты и времени в буфере времени показан в следующей таблице.
Примечание: Все значения имеют кодирование BCD.

Таблица 6-2 Буфер времени

V Байт	Значение	Примечание
T	день недели	Диапазон: 1 ~ 7 1 означает понедельник, 7 означает воскресенье
T+1	секунда	Диапазон: 0 ~ 59
T+2	минута	Диапазон: 0 ~ 59
T+3	час	Диапазон: 0 ~ 23
T+4	день	Диапазон: 1 ~ 31
T+5	месяц	Диапазон: 1 ~ 12
T+6	год	Диапазон: 0 ~ 99
T+7	век	Фиксированное как 20

Примечание:
■ Рекомендуется настроить RTC, с помощью [PLC]> [Time of Day Clock…] в меню команд перед его использованием.
■ Поскольку модуль CPU не будет проверять даты и время, когда вы вошли и недействительные данные (например, 30 февраля) будут приняты. Таким образом, вы должны обеспечить достоверность даты /

времени, которые вы уже ввели.

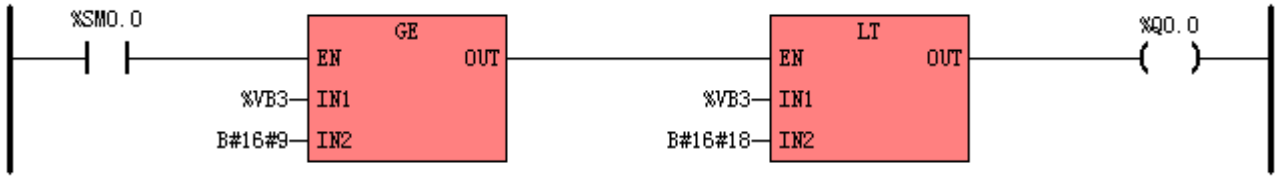
■ LD

Если EN равен 1, то эта инструкция выполняется.

■ IL

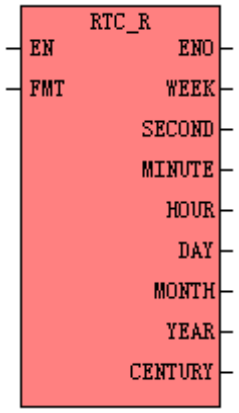
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

★ Примеры

	Использование
LD	(* Network 0 *) (* Читать RTC каждую 1 секунду *)  (* Network 1 *) (* Включить Q0.0 на время 9:00 - 18:00 каждый день, и выключить его на другое время. *) 
IL	(* Network 0 *) (* Читать RTC каждую 1 секунду *) LD %SM0.3 R_TRIG READ_RTC %VB0 (* Network 1 *) (* Включить Q0.0 на время 9:00 - 18:00 каждый день, и выключить его на другое время. *) LD %SM0.0 GE %VB3, B#16#9 LT %VB3, B#16#18 ST %Q0.0

6.11.3 RTC_R

★ Описание

	Название	Использование	Группа	
LD	RTC_R			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	RTC_R	RTC_R *FMT, WEEK, SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, CENTURY*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
FMT	Вход	BYTE	L, M, V, константа
WRRK	Выход	BYTE	L, M, V
SECOND	Выход	BYTE	L, M, V
MINUTE	Выход	BYTE	L, M, V
HOUR	Выход	BYTE	L, M, V
DAY	Выход	BYTE	L, M, V
MONTH	Выход	BYTE	L, M, V
YEAR	Выход	BYTE	L, M, V
CENTURY	Выход	BYTE	L, M, V

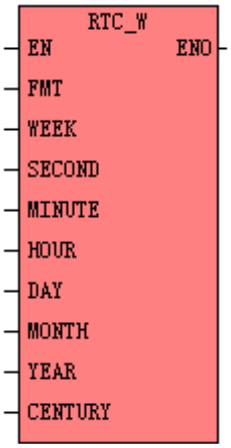
Вы можете обратиться к приведенной ниже таблице, чтобы увидеть параметры:

Операнд	Описание
EN	Включить
FMT	Формат вывода. 0 — десятичный, а 1 — BCD код
WRRK	День недели 1-7 означает Пн - Вс
SECOND	Секунда в пределах от 0-59
MINUTE	Минута в пределах от 0-59
HOUR	Час в пределах от 0-23
DAY	День в пределах от 1-31
MONTH	Месяц в пределах от 1-12
YEAR	Год в пределах от 0-99
CENTURY	Век, фиксированное значение 20

RTC_R может быть использован для загрузки текущего времени и даты, с часами реального времени и, соответственно, сохранять каждый входной параметр.
FMT представляет собой формат каждого параметра; 0 представляет собой десятичный формат, а 1 представляет собой BCD — код.
■ LD
Если EN равно 1, то команда RTC_R будут выполнена.
■ IL
Если CR равен 1, то команда RTC_R будут выполнена. Команда RTC_R не повлияет значение CR.

6.11.4 RTC_W

★ Описание

	Название	Использование	Группа	
LD	RTC_W			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	RTC_W	RTC_W *FMT, WEEK, SECOND, MINUTE, HOUR, DAY, MONTH, YEAR, CENTURY*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
FMT	Вход	BYTE	L, M, V, константа
WRRK	Вход	BYTE	L, M, V, константа
SECOND	Вход	BYTE	L, M, V, константа
MINUTE	Вход	BYTE	L, M, V, константа
HOUR	Вход	BYTE	L, M, V, константа
DAY	Вход	BYTE	L, M, V, константа
MONTH	Вход	BYTE	L, M, V, константа
YEAR	Вход	BYTE	L, M, V, константа
CENTURY	Вход	BYTE	L, M, V, константа

Вы можете обратиться к приведенной ниже таблице, чтобы увидеть параметры:

Операнд	Описание
EN	Включить
FMT	Формат вывода. 0 — десятичный, а 1 — BCD код
WRRK	День недели 1-7 означает Пн - Вс

SECOND	Секунда в пределах от 0-59
MINUTE	Минута в пределах от 0-59
HOUR	Час в пределах от 0-23
DAY	День в пределах от 1-31
MONTH	Месяц в пределах от 1-12
YEAR	Год в пределах от 0-99
CENTURY	Век, фиксированное значение 20

RTC_R может быть использован для загрузки текущего времени и даты, с часами реального времени и, соответственно, сохранить каждый входной параметр.
FMT представляет собой формат каждого параметра; 0 представляет собой десятичной и 1 представляет собой BCD — код.
■ LD
Если EN значение равно 1, то команда RTC_R будет выполнена.
■ IL
Если значение CR равно 1, то команда RTC_R будет выполнена. Команда RTC_R не повлияет CR значение.

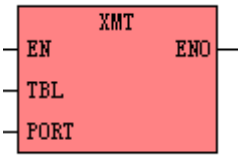
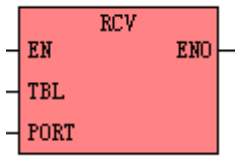
6.12 Инструкции связи

6.12.1 Свободный протокол связи

Эти инструкции используются для свободного протокола связи. Режим свободного протокола связи позволяет вашей программе полностью контролировать порты связи CPU. Вы можете использовать режим свободного протокола связи для реализации пользовательских протоколов связи для общения со всеми видами интеллектуальных устройств. Поддерживаются ASCII и бинарные протоколы.
Модуль CPU интегрирован с 1, 2 или 3 портами связи, каждый из который служит по умолчанию Modbus RTU ведомого устройства. После выполнения настроек связи, режим свободного протокола связи должен активироваться автоматически.
Вы можете настроить параметры связи (например, скорость передачи, четность и т.д.) для каждого порта в окне Hardware. Пожалуйста, обратитесь к пункту 4.3.4.1 “Параметры CPU” для получения подробной информации.
Общий порядок для выполнения программирования свободного протокола связи:
★ Установите параметры порта связи (в том числе номера станции, скорости передачи данных, четность проверки и т.д.). Для получения более подробной информации, пожалуйста, обратитесь к пункту 4.3.4.1 “Parameters of the CPU”.
★ Установите свободную связь регистра управления (см определение в следующих разделах).
Пимечание: прежде всего должен быть установлен регистр управления.
★ Вызовите команды XMT и RCV что бы запрограммировать регистр состояния и прерывания связи.

6.12.2 XMT и RCV

★ Описание

	Название	Использование	Группа	
LD	XMT			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	RCV			
IL	XMT	XMT *TBL, PORT*	U	
	RCV	RCV *TBL, PORT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
TBL	Вход	BYTE	I, Q, M, V, L, SM
PORT	Вход	INT	константа (от 0 до 2)

Инструкция **XMT** используется для передачи данных, сохраненных в буфере данных через порт связи, заданный в PORT в режиме свободного протокола связи. Буфер данных начинается с адреса TBL, и первый байт указывает число байтов, которые должны передаваться, а затем следуют действующие данные. Если SM87.1 = 1, когда CPU передал последний символ в буфере данных, то автоматически будет происходить в XMT - событие завершения прерывания (номер события 30 для PORT_0, и 32 для PORT_1). Если число байтов, которые будут переданы устанавливается равным 0, инструкция XMT не будет выполнять какие-либо действия, и событие прерывания не произойдет.
Инструкция **RCV** используется для получения данных через порт связи, заданного в PORT в режиме свободного протокола связи, и полученные данные хранятся в буфере данных. Буфер данных начинается с адреса TBL, и первый байт указывает число полученных байтов, а затем следует действующие данные.
Вы должны указать время начала и окончания и условия для работы RCV. Если SM87.1 = 1, когда CPU завершает прием (без учета нормального или ненормального завершения), будет автоматически происходить RCV - событие завершения прерывания (номер события 29 для PORT_0 и 31 для PORT_1).

■ LD

Ввод EN решает, следует ли выполнять инструкции XMT и RCV. Если EN равен 1, то инструкции XMT и RCV будут выполняться и наоборот.

■ IL

CR решает, следует ли выполнять инструкции XMT и RCV. Если CR равно 1, то инструкции XMT и RCV будут выполнены, и наоборот. Они не будут влиять на CR.

★ Регистры состояния и Регистры управления SM области для свободного протокола связи.
Кроме того, инструкции XMT и RCV некоторых регистров состояния и регистров управления в SM области предоставляются для свободного протокола связи. Ваша программа может читать и писать в эти регистры, интерпретировать состояние связи и управление связи. Ниже краткое описание байтов состояния и управляющих слов.

★ SMB86 --- Получение регистра состояния

Бит (только для чтения)			Состояние	Описание
PORT 0	**PORT 1**	**PORT 2**		
SM86.0	SM186.0	SM286.0	1	Обнаружена ошибка четности, но получение не прекращается.
SM86.1	SM186.1	SM286.1	1	Получение было прекращено из-за получения максимально количества символов.
SM86.2	SM186.2	SM286.2	1	Получение было прекращено из-за получения символов чрезмерно длительное время.
SM86.3	SM186.3	SM286.3	1	Получение было прекращено из-за системного превышения времени.
SM86.4	SM186.4	SM286.4	-	Зарезервировано
SM86.5	SM186.5	SM286.5	1	Получение было прекращено из-за приема определенного пользователем символа End.
SM86.6	SM186.6	SM286.6	1	Получение было прекращено из-за ошибок в параметрах или отсутствующих условий Start или End.
SM86.7	SM186.7	SM286.7	1	Получение было прекращено из-за команды отключения пользователя.

★ SMB87 --- Получение регистра управления

Бит			Состояние	Описание
PORT 0	**PORT 1**	**PORT 2**		
SM87.0	SM187.0	SM287.0	-	Зарезервировано
SM87.1	SM187.1	SM287.1	0	Прерывания XMT-complete и RCV-complete недействительны.
			1	Разрешить XMT-complete и RCV-complete прерывания.
SM87.2	SM187.2	SM287.2	0	Игнорировать SMW92 / SMW192 / SMW292.
			1	Завершить получения, если время в SMW92 / SMW192 / SMW292 превышает время приема символа.
SM87.3	SM187.3	SM287.3	-	Зарезервировано
SM87.4	SM187.4	SM287.4	0	Игнорировать SMW90 / SMW190 / SMW290.
			1	Обратитесь к эффективным получением, если интервал времени, в SMW90 / SMW190 / SMW290 превышен.
SM87.5	SM187.5	SM287.5	0	Игнорировать SMB89 / SMB189 / SMB289.
			1	Включить определяемый пользователем символ END в SMB89 / SMB189 / SMB289.
SM87.6	SM187.6	SM287.6	0	Игнорировать SMB88 / SMB188 / SMB288.
			1	Включить определяемый пользователем символ Start в SMB88 / SMB188 / SMB288.
SM87.7	SM187.7	SM287.7	0	Функция RCV недействительна. Это условие преобладает над любым другим условием.
			1	Разрешить функцию RCV.

★ Другие регистры управления

PORT 0	PORT 1	PORT 2	Описание
SMB88	SMB188	SMB288	Чтобы сохранить определенный пользователем полученный символ Start. После выполнения команды RCV, CPU переходит в действующее состояние получения, когда принимается символ Start, а ранее полученные данные не будут приняты. CPU принимает символ Start в качестве первого эффективного полученного байта. SM87.6 / SM187.6 / SM287.6 должны быть установлена 1 для включения SMB88 / SMB188 / SMB288.
SMB89	SMB189	SMB289	Чтобы сохранить определенный пользователем полученный символ END. CPU примет этот символ как последний действующий полученный байт. При получении символа, CPU немедленно прекращает получать без учета каких-либо других условий END. SM87.5 / SM187.5 / SM287.5 должны быть установлена 1 для включения SMB89 / SMB189 / SMB289.
SMW90	SMW190	SMW290	Чтобы сохранить определенное пользователем время готовности получения символов. (Диапазон: 1 ~ 60,000ms). После выполнения команды RCV и пройдя через этот промежуток времени, CPU автоматически перейдёт в состояние действующего получения без учета того, получен ли символ Start или нет. После этого, полученные данные должны быть эффективными. SM87.4 / SM287.4 / SM287.4 должны быть установлена 1 для включения SMW90 / SMW190 / SMW290.
SMW92	SMW192	SMW292	Чтобы сохранить определенное пользователем ограничение времени получения символов. (диапазон: 1 ~ 60,000ms). После выполнения команды RCV и перехода в состояние действующего получения, если символ не получен в течение этого интервала времени, CPU прекращает принимать игнорируя любые другие состояния END. SM87.2 / SM187.2 / SM287.2 должны быть установлена 1 для того, чтобы SMW92 / SMW192 / SMW292.
SMW94	SMW194	SMW294	Чтобы сохранить максимальное количество символов, которые будут получены (1 ~ 255). CPU немедленно прекратит получать, как только максимальное количество эффективных символов будет получено, без учета каких-либо других условий END. Если это значение установлено равным 0, инструкция RCV вернется сразу.

В режиме свободного протокола связи, по умолчанию есть система ограничения времени получения (90 секунд). Эти значения ограничения времени функций, следующие: После выполнения команды RCV, CPU немедленно прекращает получать, если данные не получены в течение этого интервала времени. Кроме того, когда CPU переходит в состояние действующего получения, оно будет использовать значение времени ограничения получения символов, определенное сначала в SMW92, и если не действующее значение находится в SMW92, значение системы получения ограничения времени будет использоваться в качестве замены.

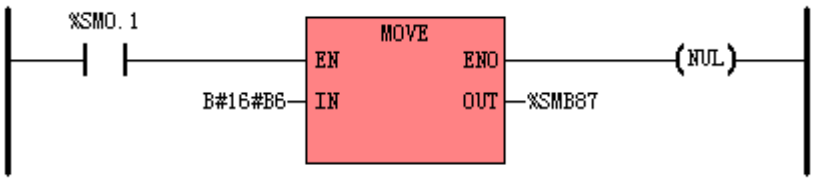
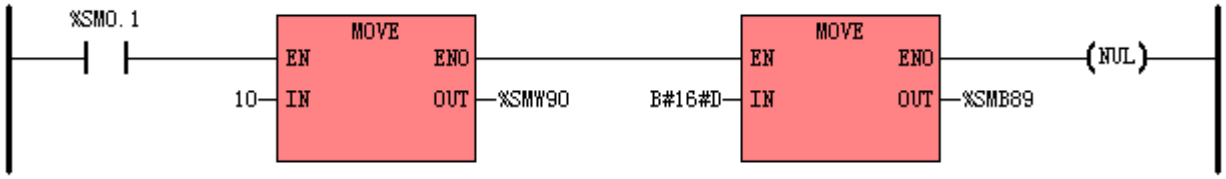
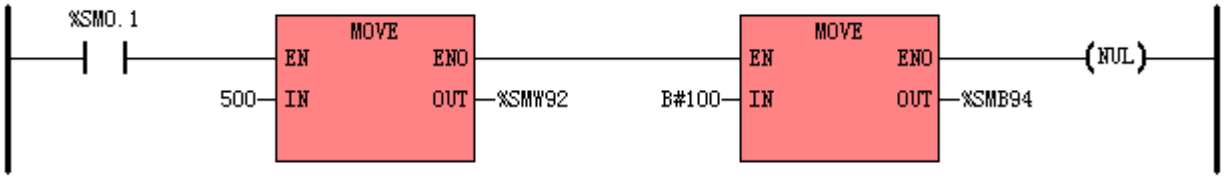
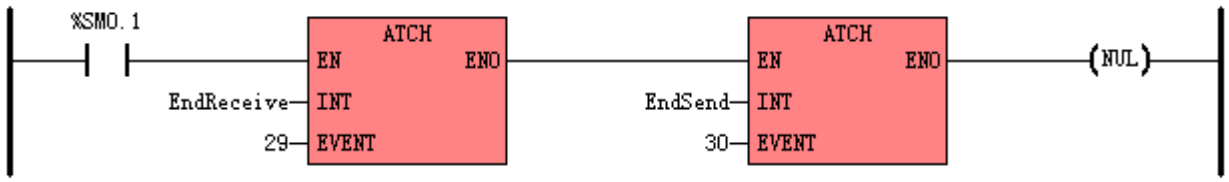
★ Прерывание связи

Kinco-K5 предлагает различные прерывания для свободной связи. Если вам нужна подробная информация, пожалуйста, обратитесь к пункту 6.10.1 «Как Kinco K5 обрабатывает прерывания».

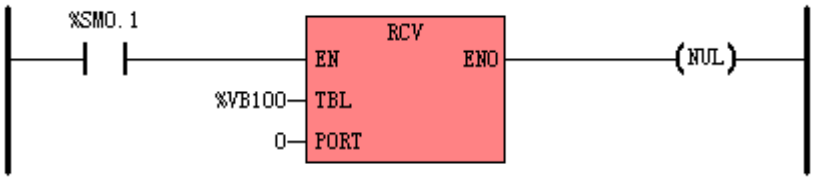
Вы можете использовать SM87.1, SM187.1 или SM287.1, что бы разрешить или запретить CPU прерывания связи. Управление прерывания устанавливается в 1, означающее разрешение подпрограммы; CPU будет генерировать прерывание завершения отправки после завершения отправки последнего слова в буферной области; CPU будет генерировать прерывание завершение получения, когда прекратит получение (независимо от того, нормальное или ненормальное прекращение).

★ Примеры

Ниже приведены примеры для объяснения применения режима свободного протокола связи. В примере, CPU получает строку символов, принимает символ RETURN, как получение символа END; если получение завершается нормально, полученные данные передаются обратно и получение возобновляется, если получение завершено ненормально (например, из-за ошибок связи, времени ожидания, и т.д.), полученные данные будут игнорироваться, и получение будет перезагружено.

	Использование
LD	**Основная программа:** (* Network 0 *) (* Следующая программа для инициализации свободного протокола связи. *) (* Во-первых, настройте условия Start и End эффективного приема информации. *)  (* Network 1 *) (* Установить время получения равным 10 мс, *) (* Установить получение символа END как символ RETURN, чей код в таблице ASCII равен 13. *)  (* Network 2 *) (* Получать символ дополнительное время, установлено в 500 мс, *) (* Максимальное количество символов, которые будут получены установлено в 100. *)  (* Network 3 *) (* Установите завершение события RCV в подпрограмме EndReceiver *) (* Установите завершение события XMT в подпрограмме EndSend *) 

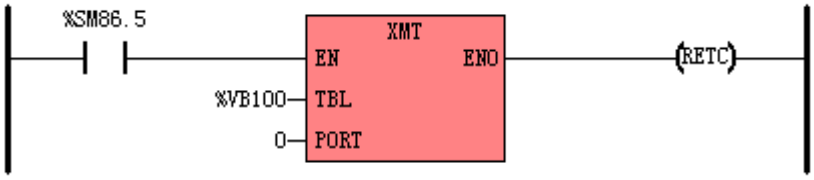
(* Network 4 *)
(* Start получения в первом цикле. *)



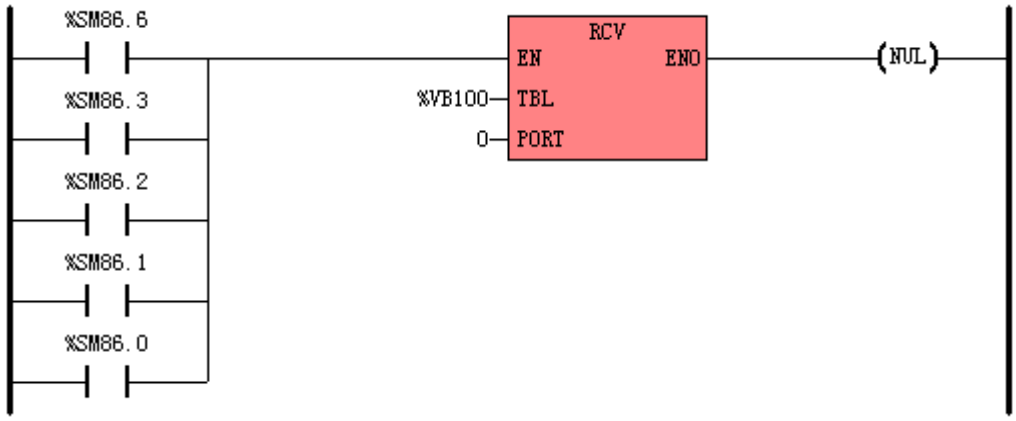
LD

EndReceive (INT00): завершение подпрограммы прерывания RCV

(* Network 0 *)
(* При приеме, получить символ END, затем полученные данные передать обратно и вернуться. *)



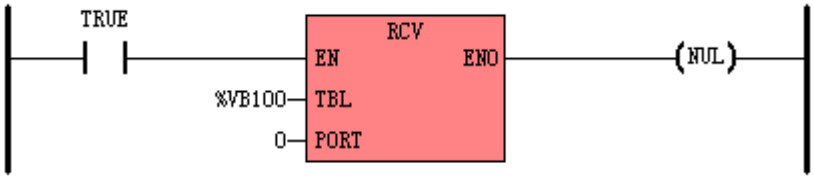
(* Network 1 *)
(* Если получение завершится аварийно, перезагрузите получение. *)



LD

EndSend (INT01): завершение подпрограммы прерывания XMT

(* Network 0 *)
(* Перезагрузка получения после завершения передачи. *)



IL

MAIN Program:

```
(* Network 0 *)
(* Следующая программа для инициализации свободного протокола связи. *)
(* Во-первых, настройте условия Start и End эффективного приема информации. *)
LD              %SM0.1
MOVE            B#16#B6, %SMB87
(* Network 1 *)
(* Установить время получения равным 10 мс, *)
(* Установить получение символа END как символ RETURN, чей код в таблице ASCII равен 13. *)
LD              %SM0.1
MOVE            10, %SMW90
MOVE            B#16#D, %SMB89
(* Network 2 *)
(* Получать символ дополнительное время, установлено в 500 мс, *)
(* Максимальное количество символов, которые будут получены установлено в 100. *)
LD              %SM0.1
MOVE            500, %SMW92
MOVE            B#100, %SMB94
(* Network 3 *)
(* Установите завершение события RCV в подпрограмме EndReceiver *)
(* Установите завершение события XMT в подпрограмме EndSend *)
LD              %SM0.1
ATCH            EndReceive, 29
ATCH            EndSend, 30
(* Network 4 *)
(* Start получения в первом цикле. *)
LD              %SM0.1
RCV             %VB100, 0
```

IL

EndReceive (INT00): завершение подпрограммы прерывания RCV

```
(* Network 0 *)
(* При приеме, получить символ END, затем полученные данные передать обратно и вернуться. *)
LD          %SM86.5
XMT         %VB100, 0
RETC
(* Network 1 *)
(* Если получение завершится аварийно, перезагрузите получение. *)
LD          %SM86.6
OR          %SM86.3
OR          %SM86.2
OR          %SM86.1
OR          %SM86.0
RCV         %VB100, 0
```

IL

EndSend (INT01): завершение подпрограммы прерывания XMT

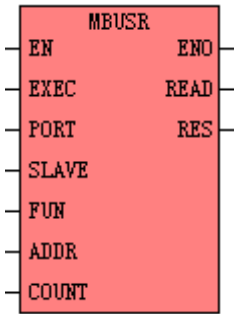
```
(* Network 0 *)
(* Перезагрузка получения после завершения передачи. *)
LD          TRUE
RCV         %VB100, 0
```

6.12.3 Инструкции Modbus RTU Master

Протокол Modbus RTU широко используется в промышленной сфере. Kinco-K5 предоставляет инструкции Modbus RTU Master, и вы можете вызывать их напрямую, чтобы сделать Kinco-K5 в качестве ведущего Modbus RTU.
Примечание: эти инструкции поддерживаются только PORT1.
Основные этапы программирования Modbus Master описаны далее:
★ Настройка параметров связи PORT1 в окне Hardware. Пожалуйста, обратитесь к разделу 2.6 “Как изменить параметры связи CPU” и 4.3.4.1 “Параметры CPU” для более подробной информации.
★ Вызовите в программе инструкции MBUSR и MBUSW.

6.12.3.1 MBUSR (Чтение Modbus RTU Master)

★ Описание

	Название	Использование	Группа	
LD	MBUSR			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	MBUSR	MBUSR *EXEC, PORT, SLAVE, FUN, ADDR,COUNT, READ, RES*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
PORT	Вход	INT	константа (от 0 до 2)
SLAVE	Вход	BYTE	I, Q, M, V, L, SM, константа
FUN	Вход	INT	константа (MODBUS код функции)
ADDR	Вход	INT	I, Q, M, V, L, SM, AI, AQ, константа
COUNT	Вход	INT	I, Q, M, V, L, SM, AI, AQ, константа
READ	Выход	BOOL, WORD, INT	Q, M, V, L, SM, AQ
RES	Выход	BYTE	Q, M, V, L, SM

Эта инструкция используется для чтения данных из ведомого устройства. Доступные функциональные коды включают в себя 1 (чтение состояния DO), 2 (прочитать статус DI), 3 (чтение данных AO) и 4 (чтение данных AI).
Параметр PORT определяет используемый порт связи. SLAVE определяет адрес целевого ведомого устройства, чей доступный диапазон составляет 1 ~ 31. FUN определяет действительный код функции. ADDR определяет начальный адрес Modbus регистра чтения. COUNT определяет количество (макс. 32) регистров чтения.
Передний фронт EXEC используется для начала связи. Хотя инструкция MBUSR выполняется, он будет общаться один раз по переднему фронту EXEC: организует сообщение Modbus RTU в соответствии с

параметрами SLAVE, FUN, ADDR и COUNT, затем передаёт его и ждёт ответа ведомого; При получении ответного сообщения ведомого устройства, проверьте CRC, номер ведомого и код функции, чтобы решить, корректное ли сообщения или нет, если правильно, то полезные данные будут записаны в буфер, начиная с READ, в противном случае, полученное сообщение будет отброшено.
READ определяет начальный адрес буфера, в котором хранятся полученные данные. Тип данных чтения должен соответствовать коду функции. Если код функции представляет 1 или 2, READ имеет тип BOOL; а если код функции 3 или 4, READ имеет тип INT или WORD.
RES сохраняет состояние связи и информации о неисправности текущего выполнения, только для чтения. Он описан в следующем рисунке.

MSB LSB

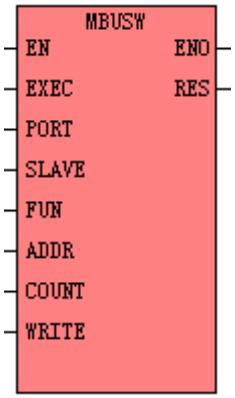
7	6	5	4	3	2	1	0

Bit 7 --- Указывает, была ли закончена связь или нет: 0 = не закончена, 1 = закончена.
Bit 6 --- Зарезервировано
Bit 5 --- Невозможный *SLAVE*.
Bit 4 --- Невозможный *COUNT*.
Bit 3 --- Невозможный *ADDR*.
Bit 2 --- 1 = Указанный порт занят.
Bit 1 --- 1 = Время вышло
Bit 0 --- 1 = Полученное сообщение неверно из-за ошибки CRC, ошибки кадров и т.д.

■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.12.3.2 MBUSW (Запись Modbus RTU Master)

★ Описание

	Название	Использование	Группа	
LD	MBUSW			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	MBUSW	MBUSW *EXEC, PORT, SLAVE, FUN, ADDR, COUNT, READ, RES*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
PORT	Вход	INT	константа (от 0 до 2)
SLAVE	Вход	BYTE	I, Q, M, V, L, SM, константа
FUN	Вход	INT	константа (MODBUS код функции)
ADDR	Вход	INT	I, Q, M, V, L, SM, AI, AQ, константа
COUNT	Вход	INT	I, Q, M, V, L, SM, AI, AQ, константа
READ	Вход	BOOL, WORD, INT	I, Q, RS, SR, V, M, L, SM, T, C, AI, AQ
RES	Выход	BYTE	Q, M, V, L, SM

Эта инструкция используется для записи данных на ведомое устройство. Доступные функциональные коды включают в себя (запись в DO), 6 (запись в AO), 15 (написать несколько DoS) и 16 (написать несколько AOS).
Параметр PORT определяет используемый порт связи. SLAVE определяет адрес целевого ведомого устройства, чей доступный диапазон составляет 1 ~ 31. FUN определяет действительный код функции. ADDR определяет начальный адрес Modbus регистра чтения. COUNT определяет количество (макс. 32) регистров чтения.
WRITE определяет начальный адрес буфера, в котором хранятся данные, которые будут записаны в ведомое устройство. Тип данных WRITE должен соответствовать коду функции. Если код функции 5 или 15, WRITE имеет тип BOOL; а если код функции 6 или 16, WRITE имеет тип INT или WORD.
Передний фронт EXEC используется для начала связи. Хотя инструкция MBUSW выполняется, он будет общаться один раз по переднему фронту EXEC: организует сообщение Modbus RTU в соответствии с параметрами SLAVE, FUN, ADDR, COUNT и WRITE, затем передаёт его и ждёт ответа ведомого ;
При получении ответного сообщения ведомого устройства, проверьте CRC, номер ведомого и кода функции, чтобы решить, выполняет ли ведомый команду правильно или нет.
RES сохраняет состояние связи и информации о неисправности текущего выполнения, только для чтения. Он описан в следующем рисунке.

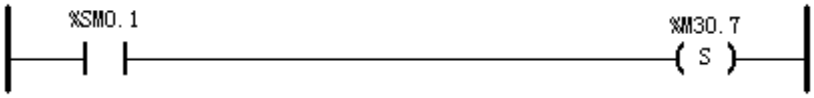
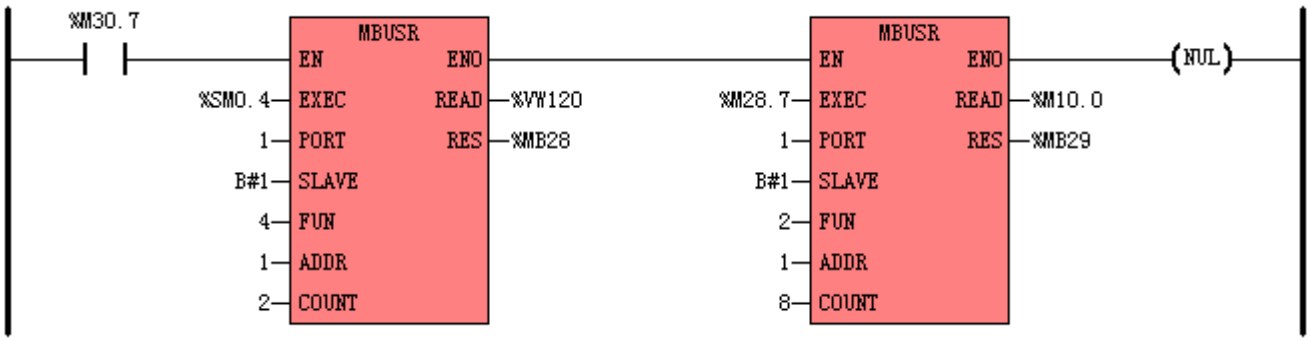
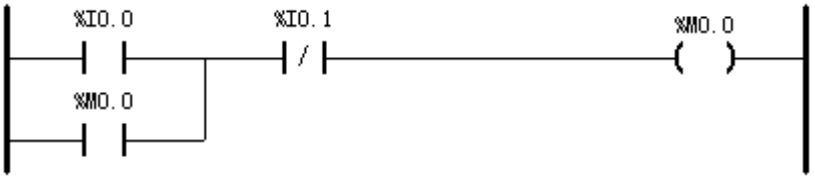
MSB LSB

7	6	5	4	3	2	1	0

Bit 7 --- Указывает, была ли закончена связь или нет: 0 = не закончена, 1 = закончена.
Bit 6 --- Зарезервировано
Bit 5 --- Невозможный *SLAVE*.
Bit 4 --- Невозможный *COUNT*.
Bit 3 --- Невозможный *ADDR*.
Bit 2 --- 1 = Указанный порт занят.
Bit 1 --- 1 = Время вышло
Bit 0 --- 1 = Полученное сообщение неверно из-за ошибки CRC, ошибки кадров и т.д.

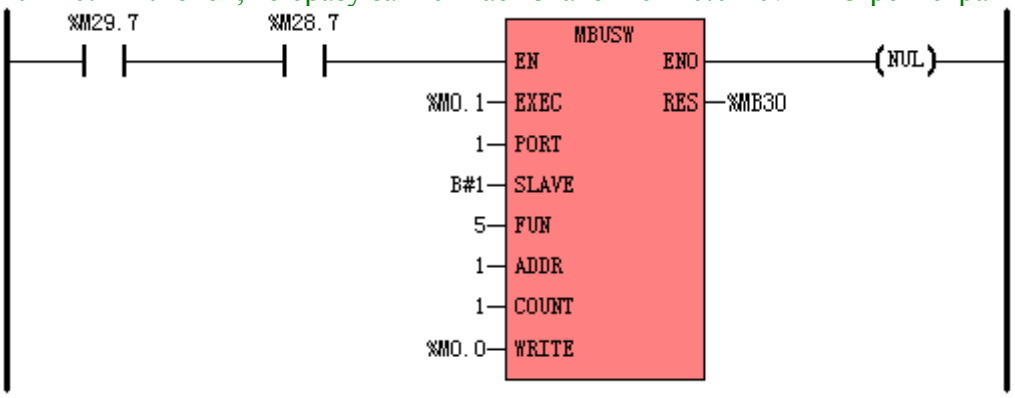
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.12.3.3 Пример MBUSR и MBUSW

	Использование
LD	(* Network 0 *) (* M30.7 показывает, закончил ли MBUSW общение или нет. *)  (* Network 1 *) (* Если PORT1 свободен в настоящее время, то MBUSR будет выполняться: каждые 2 секунды, считывает данных с подчиненного 1. Сначала читает №1 и №2 AI регистры, затем считывает DI регистры №1 - №8. *)  (* Network 2 *)  (* Network 3 *) 

LD

(* Network 4 *)
(* Если PORT1 свободен в настоящее время, то MBUSW будет выполняться: после того, как I0.0 или I0.1 включен, то сразу записывает значение M0.0 в №1 DO регистра ведомого устройства 1. *)



IL

```
(* Network 0 *)
(* M30.7 показывает, закончил ли MBUSW общение или нет *)
LD          %SM0.1
S           %M30.7
(* Network 1 *)
(* Если PORT1 свободен в настоящее время, то MBUSR будет выполняться: *)
(* каждые 2 секунды, считывает данные с подчиненного 1. *)
(* Сначала читает №1 и №2 AI регистры, затем считывает DI регистры №1 - №8. *)
LD          %M30.7
MBUSR       %SM0.4, 1, B#1, 4, 1, 2, %VW120, %MB28
MBUSR       %M28.7, 1, B#1, 2, 1, 8, %M10.0, %MB29
(* Network 2 *)
LD          %I0.0
OR          %M0.0
ANDN        %I0.1
ST %M0.0
(* Network 3 *)
LD          %I0.0
OR          %I0.1
ST          %M0.1
(* Network 4 *)
(* Если PORT1 свободен в настоящее время, то MBUSW будет выполняться: *)
(* после того, как I0.0 или I0.1 включен, то сразу записывает значение M0.0 в №1 DO регистра *)
(* ведомого устройства 1. *)
LD          %M29.7
AND         %M28.7
MBUSW       %M0.1, 1, B#1, 5, 1, 1, %M0.0, %MB30
```

6.12.4 CANOpen и SDO

★ CANopen основные функции

• Поддержка сообщения управления NMT.
• Поддержка режима предварительного подключения CANopen, общее число PDO составляет 255 TxPDO и 255 RxPDO, каждый ведомый можно настроить 1 ~ 8 TxPDO и 1 ~ 8 RxPDOl.
• Поддержка до 100 ведомых устройств, номер ведомой станции доступен с 10 ~ 126.
• Поддержка Emergency Message, Node Guide и Heartbeat Message.

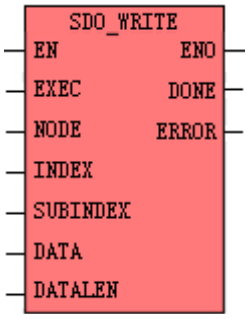
• Поддержка настройки процесса загрузки ведомых устройств .
• Поддержка функции импортирования EDS файла.
• Использование SDO_WRITE и SDO_READ для работы SDO.
• Поддержка 254 и 255 режима передачи PDO, поддержка отправки PDO по таймеру времени, он может отправить данные 8 PDO в то же время, и по времени можно установить время.
• Поддержка различной скорости передачи данных: 10K / 20K / 50K / 125K / 250K / 500K / 800K / 1M.
• Проверка состояния ведущего и ведомого с помощью системного регистра PLC.
• Поддержка нескольких PLC по CAN шине.
• Поддержка функции обработки ошибок ведомого при появлении ошибки тремя возможными способами обработки ошибок: STOP NODE, STOP NETWORD и NO OPERATION.

★ Инструкция SDO:
SDO используется для передачи данных с низким приоритетом, типичное приложение используется для настройки / управления ведомого устройства. Например, оно используется для изменения параметров текущего цикла / контура скорости / контура положения, параметров конфигурации PDO и так далее.
Этот вид передачи данных такой же, как Modbus, ведомый должен возвращать данные мастеру. Эта передача подходит только для установки параметров, но не подходит для передачи данных в режиме реального времени. Когда HMI или PLC осуществляет связь с сервоприводом, они используют этот метод для настройки параметров связи.

★ Формат данных SDO:
SDO это сокращение от Object Service Data в протоколе CANopen. Он обеспечивает доступ к данным в словаре одного устройства через индекс и субиндекс. Запрашивающее устройство называется клиентом, а одно устройство, к которому обращаются, называется сервер. Запрос клиента обязательно будет в отвечен сервером. SDO в основном используется для передачи данных с низким приоритетом между устройствами. Он подходит для настройки и управления устройств, но не передачи данных, которым требуется высокая производительность в режиме реального времени.
CAN мастер отправляет следующую инструкцию загрузки: 01 40 FF 60 00 00 00 00 00 60, эта инструкция для чтения скорости ведомого(60FF0020).
Ответ ведомого: 01 43 FF 60 00 00 20 4E 00 EF

6.12.4.1 SDO_WRITE

★ Описание

	Название	Использование	Группа	
LD	SDO_WRITE			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SDO_WRITE	SDO_WRITE *EXEC, NODE, INDEX, SUBINDEX, DATA, DATALEN, DONE, ERROR*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
NODE	Вход	BYTE	I, Q, V, M, L, SM, константа
INDEX	Вход	WORD	I, Q, V, M, L, SM, константа
SUBINDEX	Вход	BYTE	I, Q, V, M, L, SM, константа
DATA	Вход	BYTE	I, Q, V, M, L, SM
DATALEN	Вход	BYTE	I, Q, V, M, L, SM, константа
DONE	Выход	BOOL	Q, V, M, L, SM
ERROR	Выход	DWORD	Q, V, M, L, SM

NODE, INDEX, SUBINDEX, DATALEN должны быть константами или переменными.

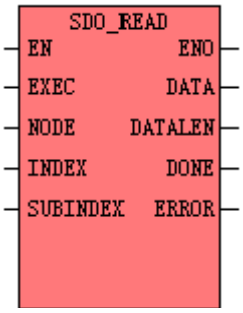
Операнд	Описание
EN	Включение. Если EN равен 1, то эта команда будет выполнена.
EXEC	Выполнение. Появление из EXEC будет принято для начала связи SDO. Лучше, чтобы EN предшествовал EXEC.
NODEID	Адрес узла.
INDEX	Индекс объекта в OD.
SUBINDEX	Субиндекс объекта в OD.
DATA	Сохраненные начальное слово данных для передачи.
DATALEN	Длина данных. Единица измерения: слово.
DONE	Выполнение отображения результата. Если SDO в настоящее время выполнено, то DONE = 0; если SDO заканчивается (ответ получен либо предел времени) DONE = 1.
ERROR	Ошибки. Пожалуйста, смотрите в таблице ниже.

Код	Описание
0	Без ошибок.
1	Количество команд SDO в K5 ограничено, общее количество команд SDO_WRITE и SDO_READ в одном проекте максимально 72.
2	Главная станция не в рабочем состоянии, поэтому отчет SDO не будет отправлен.
4	Узел не существует или остановлен из-за ошибки, поэтому отчет SDO не будет отправлен.
5	Последняя аналогичная команда не получила ответ.
6	Ошибка параметра командной строки.
8	Не получен отчет из-за ограничения по времени. Время SDO может быть установлено в [Main Station и Whole Configuration]
9	Данные ответа сообщают об ошибке.
10	Ответное сообщение не такое, как ожидалось.

■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.12.4.2 SDO_READ

★ Описание

	Название	Использование	Группа	
LD	SDO_READ			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SDO_READ	SDO_READ *EXEC, NODE, INDEX, SUBINDEX, DATA, DATALEN, DONE, ERROR*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
NODE	Вход	BYTE	I, Q, V, M, L, SM, константа
INDEX	Вход	WORD	I, Q, V, M, L, SM, константа
SUBINDEX	Вход	BYTE	I, Q, V, M, L, SM, константа
DATA	Выход	BYTE	I, Q, V, M, L, SM
DATALEN	Выход	BYTE	I, Q, V, M, L, SM
DONE	Выход	BOOL	Q, V, M, L, SM
ERROR	Выход	DWORD	Q, V, M, L, SM

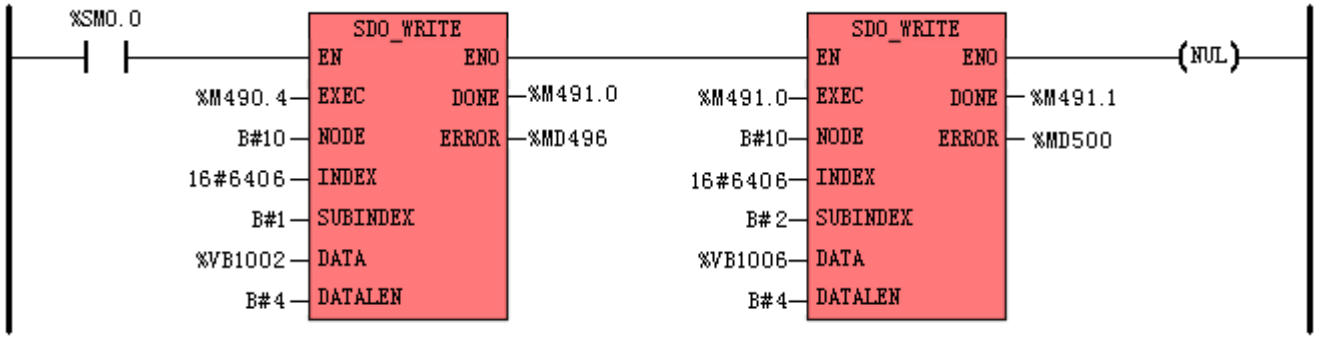
Операнд	Описание
EN	Включение. Если EN равен 1, то эта команда будет выполнена.
EXEC	Выполнение. Появление из EXEC будет принято для начала связи SDO. Лучше, чтобы EN предшествовал EXEC.
NODEID	Адрес узла.
INDEX	Индекс объекта в OD.
SUBINDEX	Субиндекс объекта в OD.
DATA	Сохраненные начальное слово данных для передачи.
DATALEN	Длина данных. Единица измерения: слово.
DONE	Выполнение отображения результата. Если SDO в настоящее время выполнено, то DONE = 0; если SDO заканчивается (ответ

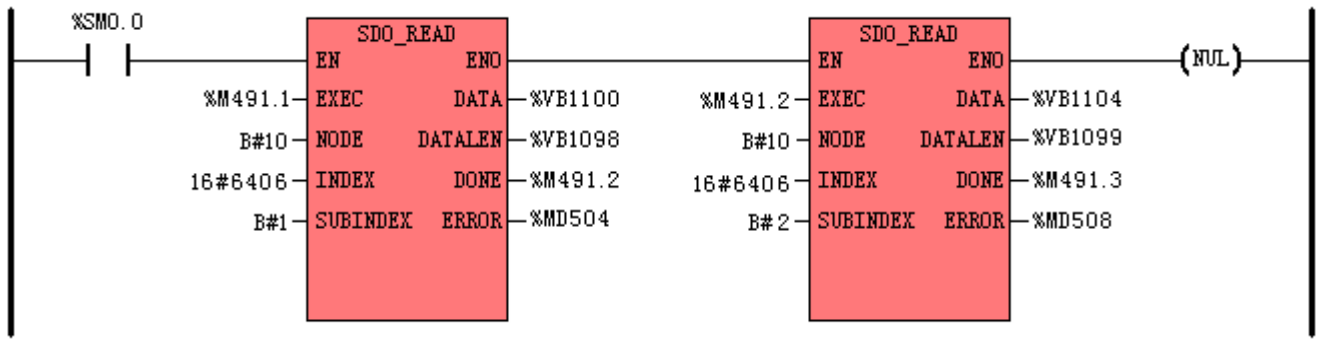
	получен либо предел времени) DONE = 1.
ERROR	Ошибки. Пожалуйста, смотрите в таблице ниже.

Код	Описание
0	Без ошибок.
1	Количество команд SDO в K5 ограничено, общее количество команд SDO_WRITE и SDO_READ в одном проекте максимально 72.
2	Главная станция не в рабочем состоянии, поэтому отчет SDO не будет отправлен.
4	Узел не существует или остановлен из-за ошибки, поэтому отчет SDO не будет отправлен.
5	Последняя аналогичная команда не получила ответ.
6	Ошибка параметра командной строки.
8	Не получен отчет из-за ограничения по времени. Время SDO может быть установлено в [Main Station и Whole Configuration]
9	Данные ответа сообщают об ошибке.
10	Ответное сообщение не такое, как ожидалось.

■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет на CR.

6.12.4.3 Пример SDO_WRITE и SDO_READ

	Использование
LD	(* Network 0 *) (* Изменить значение объекта 0x6406sub01 и 0x6406sub02 узла № 10 *)  (* Network 1 *) (* После редактирования и чтения значение объекта 0x6406sub01 и 0x6406sub02 в узле № 10 *)

	
IL	(* Network 0 *) (* Изменить значение объекта 0x6406sub01 и 0x6406sub02 узла №10 *) LD %SM0.0 SDO_WRITE %M490.4, B#10, 16#6406, B#1, %VB1002, B#4, %M491.0, %MD496 SDO_WRITE %M491.0, B#10, 16#6406, B#2, %VB1006, B#4, %M491.1, %MD500 (* Network 1 *) (* После редактирования и чтения значение объекта 0x6406sub01 и 0x6406sub02 в узле №10 *) LD %SM0.0 SDO_READ %M491.1, B#10, 16#6406, B#1, %VB1100, %VB1098, %M491.2, %MD504 SDO_READ %M491.2, B#10, 16#6406, B#2, %VB1104, %VB1099, %M491.3, %MD508

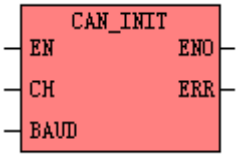
6.12.5 Команда CAN связи

K5 обеспечивать функции CANopen главной станции и CAN свободной связи, которая должна быть принята с модулем расширения K541.
Две команды могут быть использованы одновременно. Пожалуйста, обратите внимание, что при одновременном использовании нескольких команд скорость передачи данных всех узлов должна быть одинаковой.
Команда CAN связи поддерживает CAN2.0A и CAN2.0B. Команда поддерживает только кадр данных, но не большой длинны. Формат выглядит следующим образом:

ID	Слово 1-8
11 цифр (CAN2.0A, стандартный блок) или 29 цифр (CAN 2.0B, блок расширения)	Длина данных 1-8 слов

6.12.5.1 CAN_INI

★ Описание

	Название	Использование	Группа	
LD	CAN_INIT			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CAN_INIT	CAN_INIT *CH, BAUD*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
EN	Вход	BOOL	I, Q, V, M, L, SM
CH	Вход	INT	Константа (2)
BAUD	Вход	INT	V, M, L, константа
ERR	Выход	BOOL	V, M, L, константа

Операнд	Описание
EN	Включение.
CH	CAN порт. Значение 2 означает модуль K541.
BAUD	Скорость передачи данных CAN 8 --- 1000K 7 --- 800K 6 --- 500K 5 --- 250K 4 --- 125K 3 --- 50K 2 --- 20K 1 --- 10K
ERR	Выполнение команд. 0 означает успех, а 1 означает ошибку.

Изменение по переднему фронту входа EN вызовет эту команду. Может быть использован для CAN порта (CH), чтобы назначить скорость передачи данных CAN в значение BAUD.

■ LD

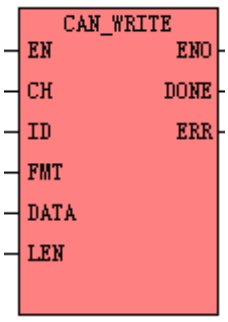
Изменение повышение EN вызовет эту команду, и наоборот.

■ IL

Изменение повышение CR вызовет эту команду, и наоборот. Команда не влияет на значение CR.

6.12.5.2 CAN_WRITE

★ Описание

	Название	Использование	Группа	
LD	CAN_WRITE			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CAN_WRITE	CAN_WRITE *CH, ID, FMT, DATA, LEN, DONE, ERR*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
CH	Вход	INT	Константа (2)
ID	Вход	DWORD	V, M, L, константа
FMT	Вход	BYTE	V, M, L, константа
DATA	Вход	BYTE	L, V, M
LEN	Вход	BYTE	V, M, L, константа
DONE	Выход	BOOL	L, V, M
ERR	Выход	BOOL	L, V, M

Операнд	Описание
EN	Включение.
CH	CAN порт. Значение 2 означает модуль K541.
FMT	Формат отчета. 0 означает стандартный кадр и 1 означает расширенный кадр.
DATA	Сохраненный начальный адрес слова отправки данных.
LEN	Длина данных. Единица измерения: слово.
DONE	Выполнение отображение результата. Если выполнен, то DONE = 0; Если заканчивается (ответ получен или ограничение по времени) DONE = 1.
ERR	Индикация состояния отчетов. Если передать сообщение не удалось (обычно вызвано переполненной областью буфера отправки), то ERR будет сброшена на 1.

Отправка отчета CAN будет принято в ID, формат (FMT, стандартный кадр или расширенный кадр), данные (DATA, адрес памяти для хранения) и длина (LEN).
Изменение по переднему фронту на входе EN вызовет команду один раз и отправит отчет в область буфера отправки с помощью CAN порта (CH).
Если команда успешно отправила отчет в буфер, выполнение будет сделано, и DONE будет установлен в 1.
Если буферная область переполнена, то отправка не удастся, DONE и ERR будет установлен в 1.

■ LD

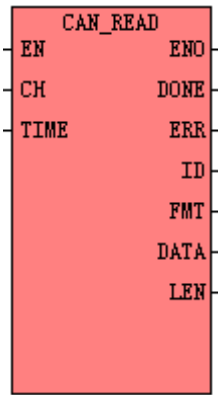
Изменение по переднему фронту EN вызовет эту команду, и наоборот.

■ IL

Изменение по переднему фронту CR вызовет эту команду, и наоборот. Команда не влияет на значение CR.

6.12.5.3 CAN_READ

★ Описание

	Название	Использование	Группа	
LD	CAN_READ			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CAN_READ	CAN_READ *CH, TIME, DONE, ERR, FMT, DATA, LEN*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
CH	Вход	INT	Константа (только 2)
TIME	Вход	INT	V, M, L, константа
DONE	Выход	BOOL	L, V, M
ERR	Выход	BOOL	L, V, M
ID	Выход	DWORD	L, V, M
FMT	Выход	BYTE	L, V, M
DATA	Выход	BYTE	V, M
LEN	Выход	BYTE	L, V, M

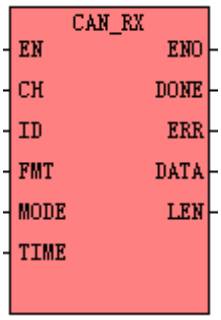
Операнд	Описание
EN	Включение.
CH	CAN порт. Значение 2 означает модуль K541.
FMT	Формат отчета. 0 означает стандартный кадр и 1 означает расширенный кадр.
DATA	Сохраненный начальный адрес слова отправки данных.
LEN	Длина данных. Единица измерения: слово.
DONE	Выполнение отображение результата. Если выполнен, то DONE = 0; Если заканчивается (ответ получен или ограничение по времени) DONE = 1.
ERR	Индикация состояния отчетов. Если передать сообщение не удалось (обычно вызвано переполненной областью буфера отправки), то ERR будет сброшена на 1.

Изменение по переднему фронту входа EN вызовет эту команду: начало приема и получение любого отчета от верного CAN порта (CH).

Во время начала получении, если PLC получает отчет CAN в пределах ограничения по времени, то отчет будет помещён в выходные параметры ID, FMT, DATA и LEN соответственно, затем DONE будет установлен на 1 и закончит прием. Если он не получает ни одного отчета в пределах ограничения по времени, он закончит прием и установит DONE и ERR равным 1.
После начала командных CAN_READ, PLC будет получать доклад какой-либо определенной CAN порта.
Пожалуйста, обратите внимание при использовании вместе с другими протоколами (например, CANopen).
■ LD
Изменение по переднему фронту EN вызовет эту команду, и наоборот.
■ IL
Изменение по переднему фронту CR вызовет эту команду, и наоборот. Команда не влияет на значение CR.

6.12.5.4 CAN_RX

★ Описание

	Название	Использование	Группа	
LD	CAN_RX			☐ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CAN_RX	CAN_RX *CH, ID, FMT, MODE, TIME, DONE, ERR, FMT, DATA, LEN*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
CH	Вход	INT	Константа (только 2)
ID	Вход	DWORD	L, M,V, константа
FMT	Вход	INT	L, M,V, константа
MODE	Вход	INT	L, M,V, константа
TIME	Вход	INT	L, M,V, константа
DONE	Выход	BOOL	L, M,V
ERR	Выход	BOOL	L, M,V
DATA	Выход	BYTE	M,V
LEN	Выход	BYTE	L, M,V

Операнд	Описание
EN	Включение.
CH	CAN порт. Значение 2 означает модуль K541.
ID	ID ожидает получения отчетов.
FMT	Формат отчета. 0 означает стандартный кадр и 1 означает расширенный кадр.

MODE	Режим приема. 0 означает постоянную передачу и 1 означает собой получение.
TIME	Прием вне ограничений по времени. Единица измерения: мс
DONE	Выполнение отображение результата. Если выполнен, то DONE = 0; Если заканчивается (ответ получен или ограничение по времени) DONE = 1.
ERR	Индикация состояния отчетов. Если передать сообщение не удалось (обычно вызвано переполненной областью буфера отправки), то ERR будет сброшена на 1.
DATA	Начальный адрес последнего приема данных отчета.
LEN	Длина данных. Единица измерения: слово.

Изменение по переднему фронту входа EN вызовет эту команду: начало приема и получение любого отчета от верного CAN порта (CH).
Во время начала получении, если PLC получает отчет CAN в пределах ограничения по времени, то отчет будет помещён в выходные параметры ID, FMT, DATA и LEN соответственно, затем DONE будет установлен на 1 и закончит прием. Если он не получает ни одного отчета в пределах ограничения по времени, он закончит прием и установит DONE и ERR равным 1.
После начала командных CAN_READ, PLC будет получать доклад какой-либо определенной CAN порта.
Пожалуйста, обратите внимание при использовании вместе с другими протоколами (например, CANopen).

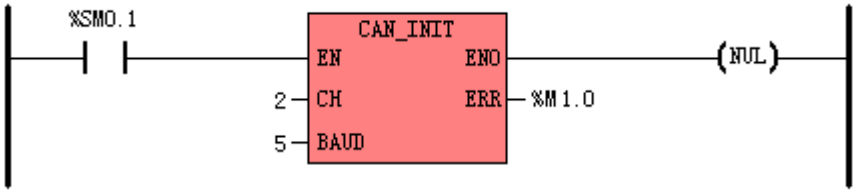
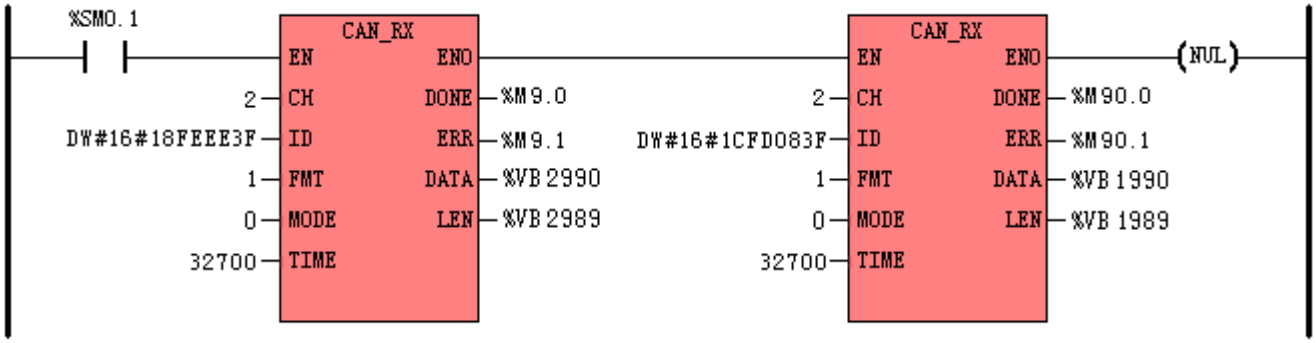
■ LD

Изменение по переднему фронту EN вызовет эту команду, и наоборот.

■ IL

Изменение по переднему фронту CR вызовет эту команду, и наоборот. Команда не влияет на значение CR.

★ Пример

	Использование
LD	(* Network 0 *) (* Установите скорость передачи данных K541 *)  (* Network 1 *) (* Назначьте ID и формат отчета, который будет автоматически приниматься. Прием является постоянным. PLC будет сохранять данные отчетов в установленные области памяти после получения отчета. *) 

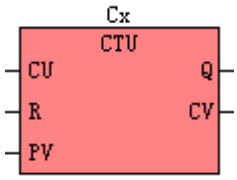
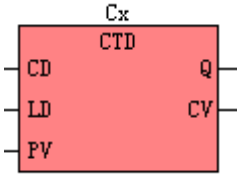
IL	(* Network 0 *) (* Установите скорость передачи данных K541 *) LD %SM0.1 CAN_INIT 2, 5, %M1.0 (* Network 1 *) (* Назначьте ID и формат отчета, который будет автоматически приниматься. Прием является постоянным. PLC будет сохранять данные отчетов в установленные области памяти после получения отчета. *) LD %SM0.1 CAN_RX 2, DW#16#18FEEE3F, 1, 0, 32700, %M9.0, %M9.1, %VB2990, %VB2989 CAN_RX 2, DW#16#1CFD083F, 1, 0, 32700, %M90.0, %M90.1, %VB1990, %VB1989

6.13 Счетчики

6.13.1 CTU (счетчик вверх) и CTD (счетчик вниз)

Счетчик является одним из функциональных блоков, определенных в стандарте IEC61131-3, всех трех типов, то есть CTU, CTD и CTUD. Пожалуйста, обратитесь к пункту 3.6.5 “Функциональные блоки и их примеры”, для получения более подробной информации.

★ Описание

	Название	Использование	Группа	
LD	CTU			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	CTD			
IL	CTU	CTU *Cx, R, PV*	P	
	CTD	CTD *Cx, LD, PV*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
Cx	-	Пример счетчика	C
CU	Вход	BOOL	Питание
R	Вход	BOOL	I, Q, M, V, L, SM, T, C
CD	Вход	BOOL	Питание
LD	Вход	BOOL	I, Q, M, V, L, SM, T, C
Q	Выход	BOOL	Питание
CV	Выход	INT	Q, M, V, L, SM, AQ

■ LD

CTU счетчик считает вверх по нарастающему фронту на входе CU. Когда текущее значение CV равно или больше, чем заданное значение PV, и выход счетчика Q и бит состояния Cx устанавливаются в 1. Cx сбрасывается, когда вход сброса R включен. Когда счетчик достигает PV, он продолжает считать, пока он не достигнет и удержится на максимальном значении INT (т.е. 32767).

CTD счетчик считает в обратном направлении по нарастающему фронту на входе CD. Когда текущее значение CV равно или больше, чем заданное значение PV, и выход счетчика Q и бит состояния Cx устанавливаются в 1. Cx сбрасывается и PV загружается в CV, когда вход LD включен. Когда счетчик достигает PV, он продолжает считать, пока он не достигнет и удержится на 0.

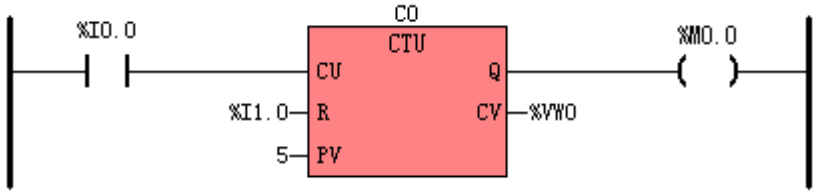
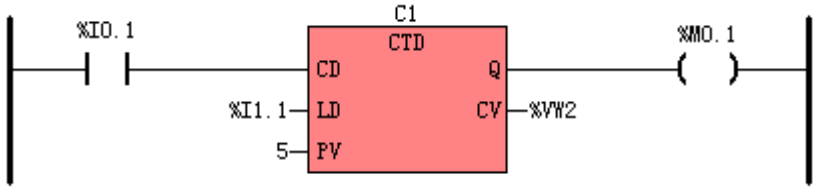
■ IL

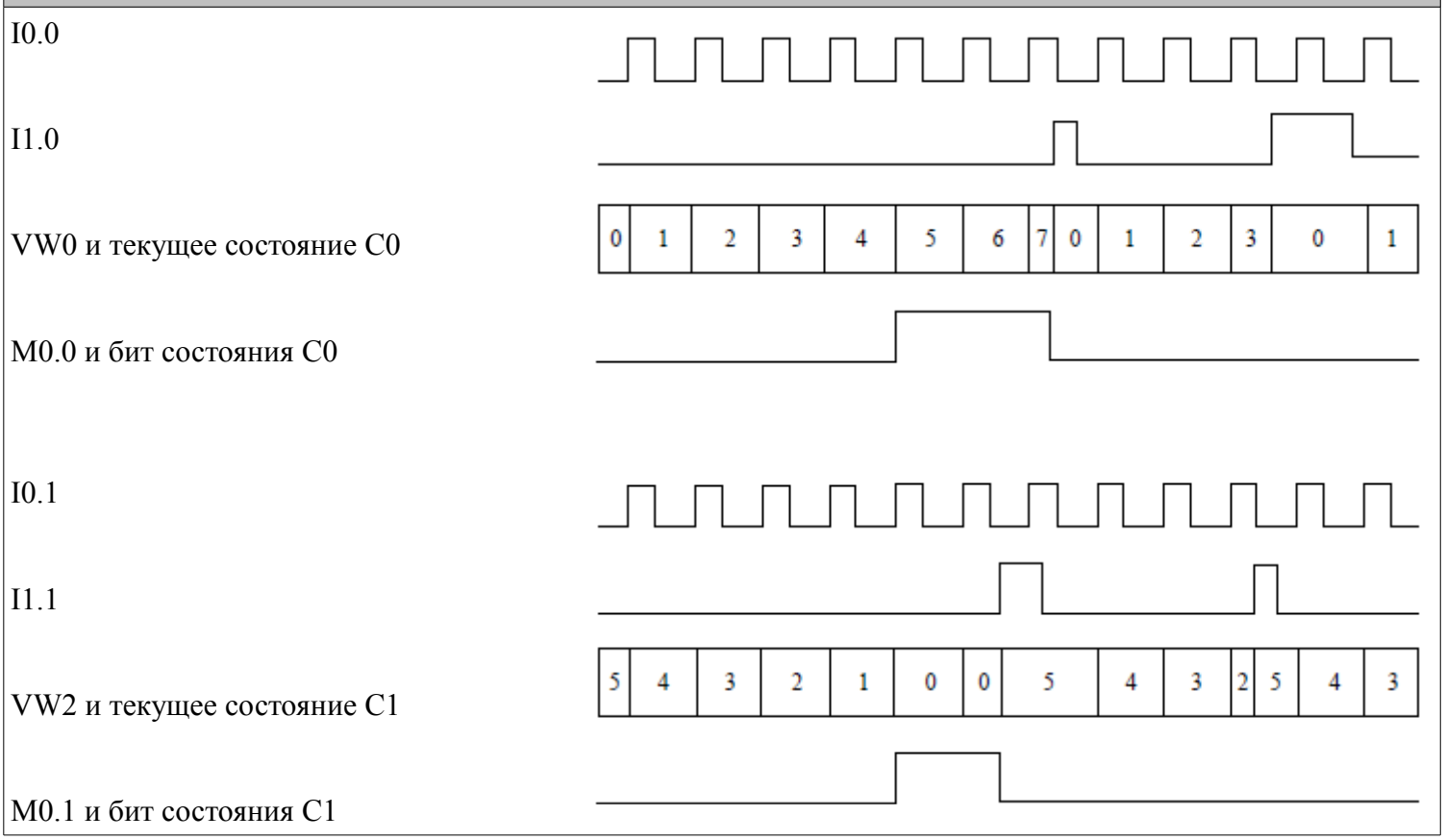
CTU счетчик считает вверх по переднем фронту CR. Когда текущее значение Cx равено или больше, чем заданное значение PV, бит состояния счетчика устанавливается равным 1. Cx сбрасывается, когда вход сброса R включен. Когда счетчик достигает PV, он продолжает считать, пока он не достигнет и удержится на максимальном значении INT (т.е. 32767).

После каждого сканирования, CR устанавливается равным бита состояния значения Cx.

CTD счетчик считает в обратном направлении по переднему фронту CR. Когда текущее значение Cx равно или больше, чем заданное значение PV, бит состояния счетчика устанавливается равным 1.Cx сбрасывается и PV загружается в текущее значение, когда вход нагрузки LD включен. Когда счетчик достигает PV, он продолжает считать, пока он не достигнет и удержится на 0. После каждого сканирования, CR устанавливается равным бита состояния значения Cx.

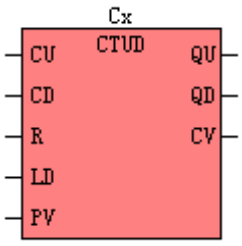
★ Пример

LD	IL
(* Network 0 *)  (* Network 1 *) 	(* Network 0 *) LD %I0.0 CTU C0, %I1.0, 5 ST %M0.0 (* Network 1 *) LD %I0.1 CTD C1, %I1.1, 5 ST %M0.1

Временная диаграмма


6.13.2 CTUD (счетчик вверх-вниз)

★ Описание

	Название	Использование	Группа	
LD	CTUD			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CTUD	CTUD *Cx, CD, R, LD, PV, QD*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
Cx	-	Экземпляр счетчика	C
CU	Вход	BOOL	Питание
CD	Вход	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
R	Вход	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
LD	Вход	BOOL	I, Q, M, V, L, SM, T, C, RS, SR
PV	Вход	INT	I, Q, M, V, L, SM, AI, AQ, константа

QU	Выход	BOOL	Питание
QD	Выход	BOOL	Q, M, V, L, SM
CV	Выход	INT	Q, M, V, L, SM, AQ

■ LD

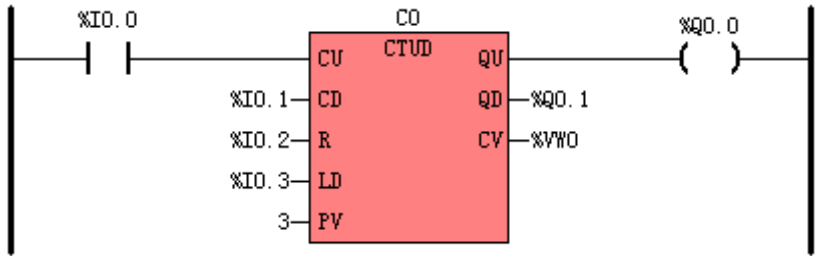
Счетчик CTUD считает в прямом направлении по переднему фронту на входе CU и в обратном направлении по нарастающему фронту сигнала CD и текущее значение счетчика Cx присваивается в CV. Когда CV равна или больше, чем заданное значение PV, и QU и бит состояния Cx установлены на 1, в противном случае они установлены в 0. Когда CV равен 0, QD устанавливается в 1, в противном случае он установлен в 0. Когда вход сброса R включен, Cx и CV сбрасывается. Когда вход LD включен, PV загружается в Cx и CV. Если R и LD являются 1, в то же время, R принимает высокий приоритет.

■ IL

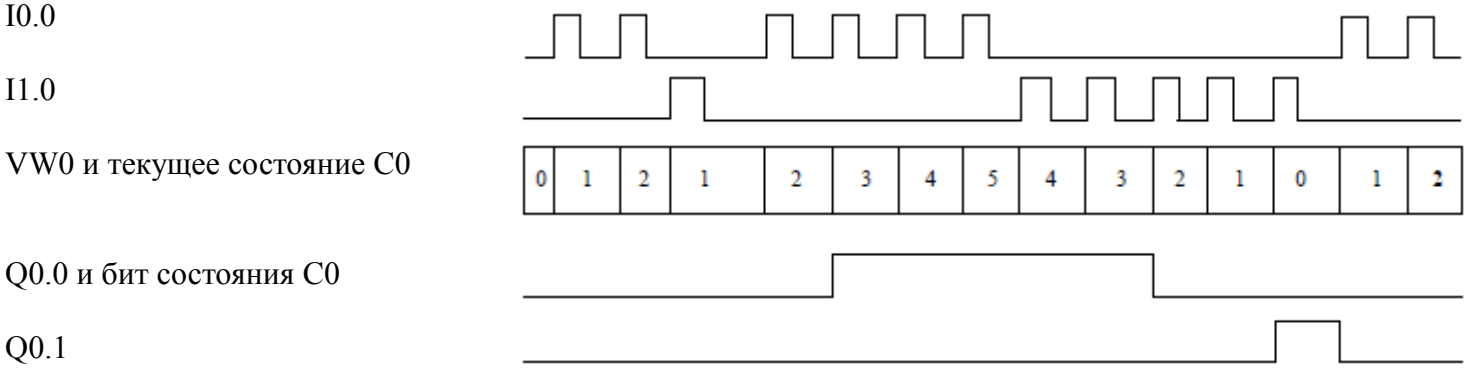
Счетчик CTUD считает в прямом направлении по переднему фронту CR и считает в обратном направлении по нарастающему фронту сигнала CD и текущее значение счетчика Cx присваивается в CV. Когда CV равна или больше, чем заданное значение PV, и QU и бит состояния Cx установлены на 1, в противном случае они установлены в 0. Когда CV равен 0, QD устанавливается в 1, в противном случае он установлен в 0. Когда вход сброса R включен, Cx и CV сбрасывается. Когда вход LD включен, PV загружается в Cx и CV. Если R и LD являются 1, в то же время, R принимает высокий приоритет.

После каждого сканирования, CR устанавливает бит состояния значение Cx.

★ Пример

LD	IL
	LD %I0.0 CTUD C1, %I0.1, %I0.2, %I0.3, 4 ,%Q0.1 ST %Q0.0

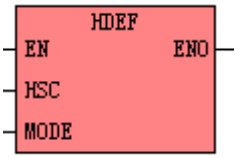
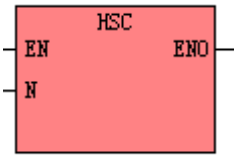
Временная диаграмма



6.13.3 Инструкции высокоскоростных счётчиков

Высокоскоростные счетчики считают высокоскоростные импульсные входы, которые не могут контролироваться на скорости сканирования CPU.

★ Описание

	Название	Использование	Группа	
LD	HDEF			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	HSC			
IL	HDEF	HDEF *HSC, MODE*	U	
	HSC	HSC *N*		

Операнд	Вход / Выход	Тип данных	Описание
HSC	Вход	INT константа (0~5)	Номер HSC
MODE	Вход	INT константа (0~11)	Режим эксплуатации
N	Вход	INT константа (0~5)	Номер HSC

Команда HDEF (определение высокоскоростных счетчиков) используется для определения режима работы (MODE) высокоскоростного счетчика (HSC). Эта команда предназначена для каждого высокоскоростного счетчика. Высокоскоростной счетчик может быть сконфигурирован в одном из 11 различных режимов работы. Режим определяет тактовый вход, направление счёта, запуск и сброс свойств высокоскоростного счетчика.

Команд HSC (высокоскоростной счетчик) настраивает и управляет высокоскоростным счетчиком, номер которого указан в N в соответствии со значениями соответствующих регистров SM.

В IL, CR решает, следует ли выполнять команды HDEF и HSC. Они не будут влиять на CR.

■ LD

Если значение EN равно 1, то HDEF и HSC выполняется.

■ IL

Если значение CR равно 1, то HDEF и HSC выполняется. Выполнение HDEF и HSC не влияет на CR.

6.13.3.1 Высокоскоростные счетчики, поддерживаемые Kinco-K5

Особенность	CPU504, CPU504EX, CPU506, CPU506EA, CPU508
Высокоскоростные счетчики Однофазные Двухфазные	2 счетчика (HSC0 и HSC1) 2 на частоте 60 кГц 2 на частоте 20 кГц

Прежде всего, в высокоскоростном счетчике должен быть назначен режим работы по команде HDEF. Все высокоскоростные счетчики имеют те же функции в том же режиме. Каждый вход высокоскоростного счетчика имеет следующие функции:

6.13.3.2 Режимы работы и входы высокоскоростных счетчиков

Входные сигналы высокоскоростного счетчика входят: счёт (входной импульс), направление, запуск и сброс. В различных режимах работы входные сигналы отличаются. Пожалуйста, смотрите ниже:

HSC 0				
Режим	Описание	I0.1	I0.0	I0.5
0	Однофазный счетчик вверх / вниз с внутренним управлением направления: SM37.3	Счёт		
1			Сброс	
2			Сброс	Старт
3	Однофазный счетчик вверх / вниз с внешним управлением направления	Счёт		Направление
4			Сброс	Направление
6	Двухфазный счетчик вверх / вниз с тактовыми входами	Счёт Up	Счёт Down	
9	A / B квадратурные импульсы счетчика	Счёт A	Счёт B	

HSC 1					
Режим	Описание	I0.4	I0.6	I0.3	I0.2
0	Однофазный счетчик вверх / вниз с внутренним управлением направления: SM47.3			Счёт	
1		Сброс			
2		Сброс	Старт		
3	Однофазный счетчик вверх / вниз с внешним управлением направления			Счёт	Направление
4		Сброс			Направление
5		Сброс	Старт		Направление
6	Двухфазный счетчик вверх / вниз с тактовыми входами			Счёт Down	Счёт Up
7		Сброс			
8		Сброс	Старт		
9	A / B квадратурные импульсы счетчика			Счёт A	Счёт B
10		Сброс			
11		Сброс	Старт		

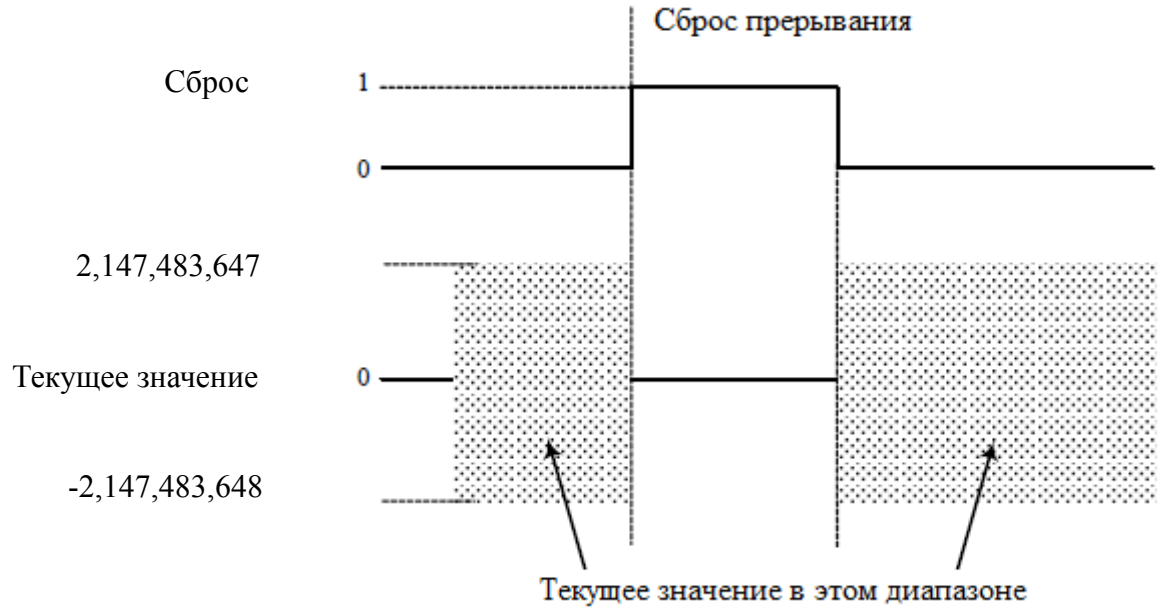
6.13.3.3 Временная диаграмма высокоскоростных счетчиков

Для того, чтобы помочь вам хорошо понять работу высокоскоростного счетчика, на следующих диаграммах показаны различные последовательности времени.

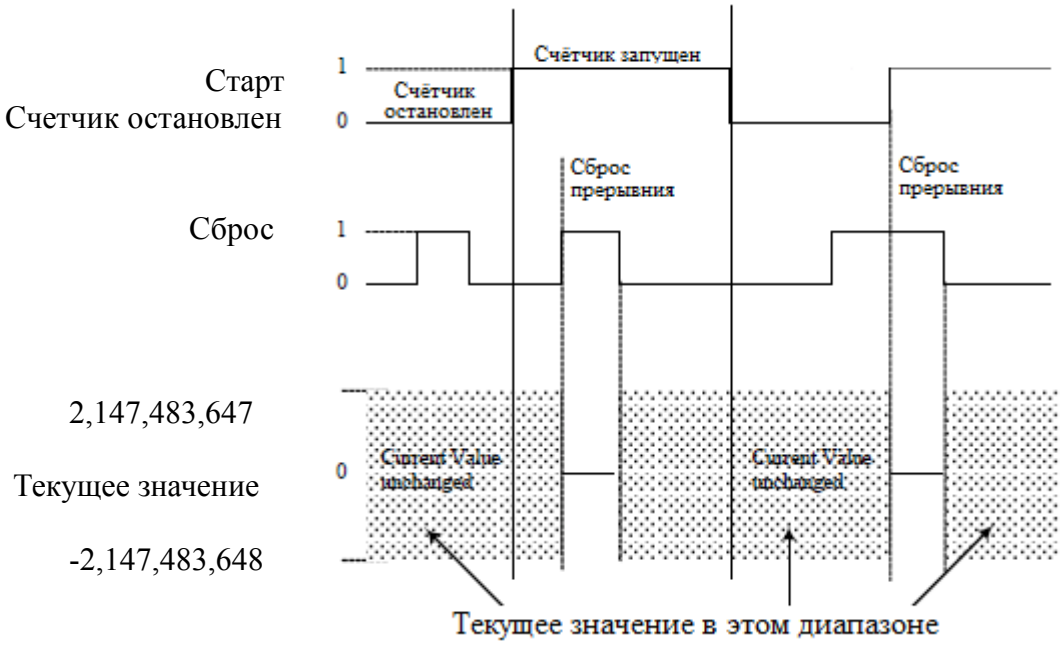
★ Сброс и пуск

Действия на следующих рисунках подходят для всех режимов, которые используют сброс и пуск входов.

Временная диаграмма со сбросом и без пуска

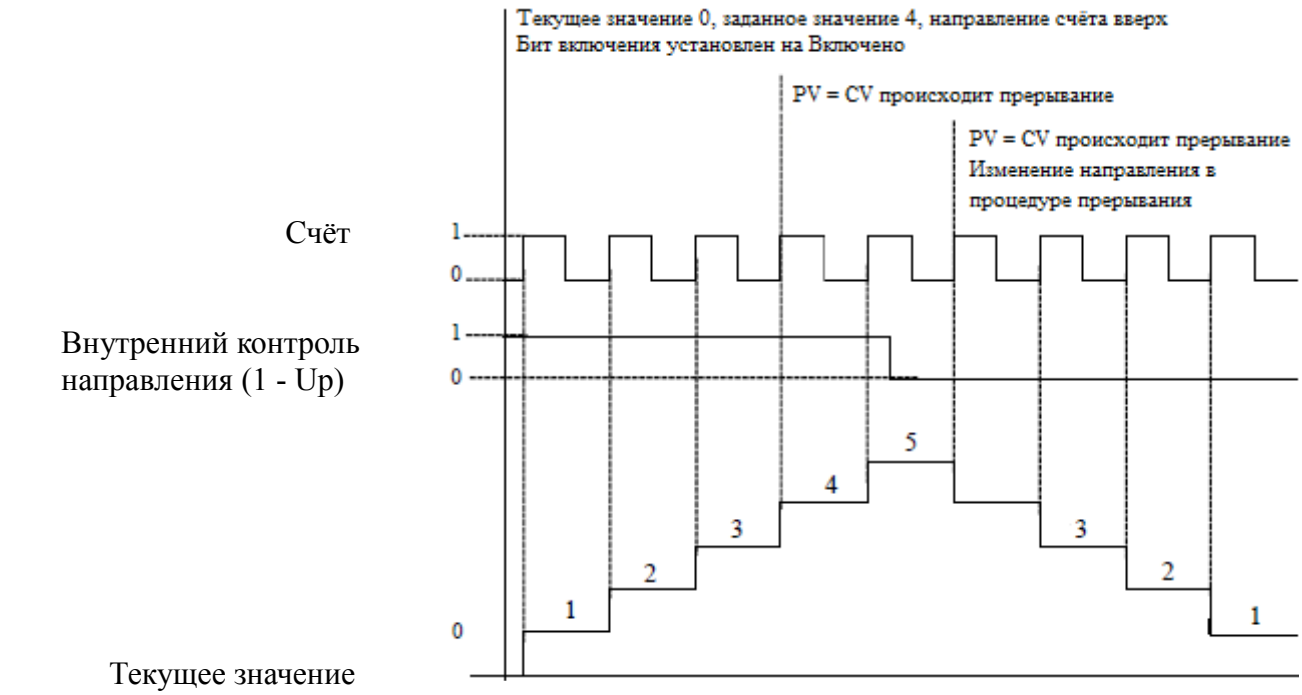


Временная диаграмма со сбросом и пуском

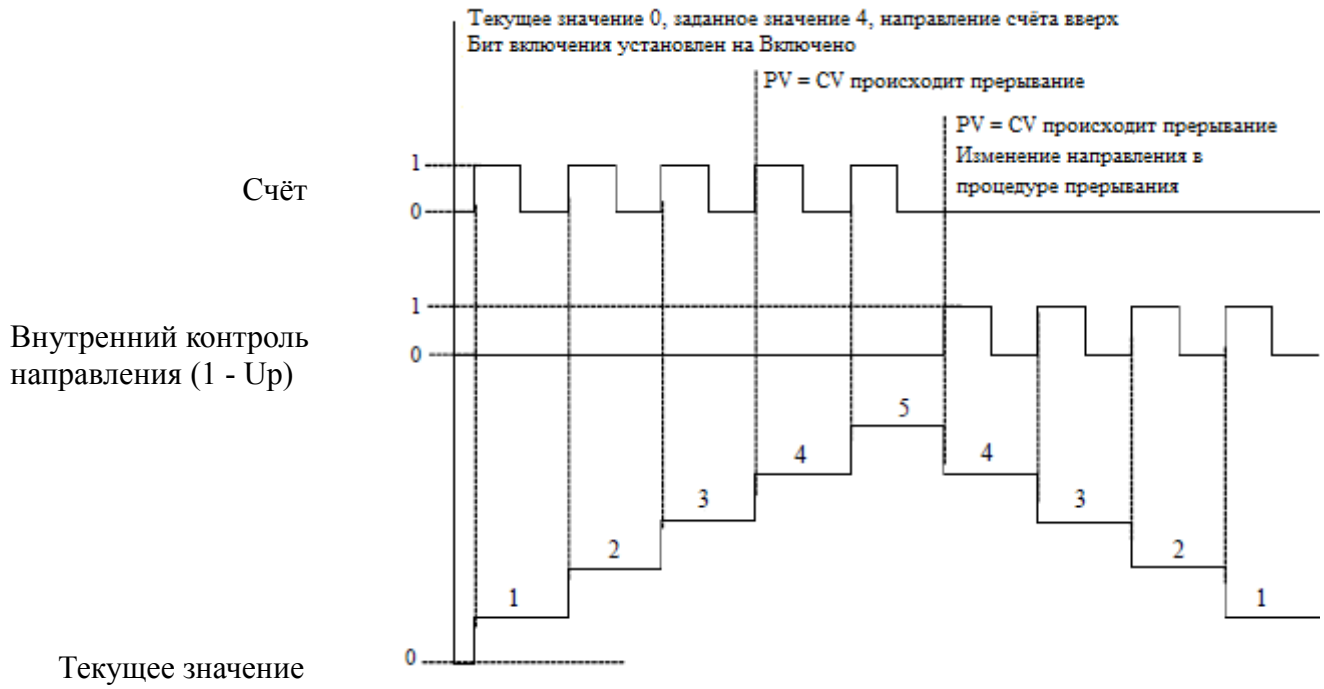


Временная диаграмма режимов 0, 1 или 2

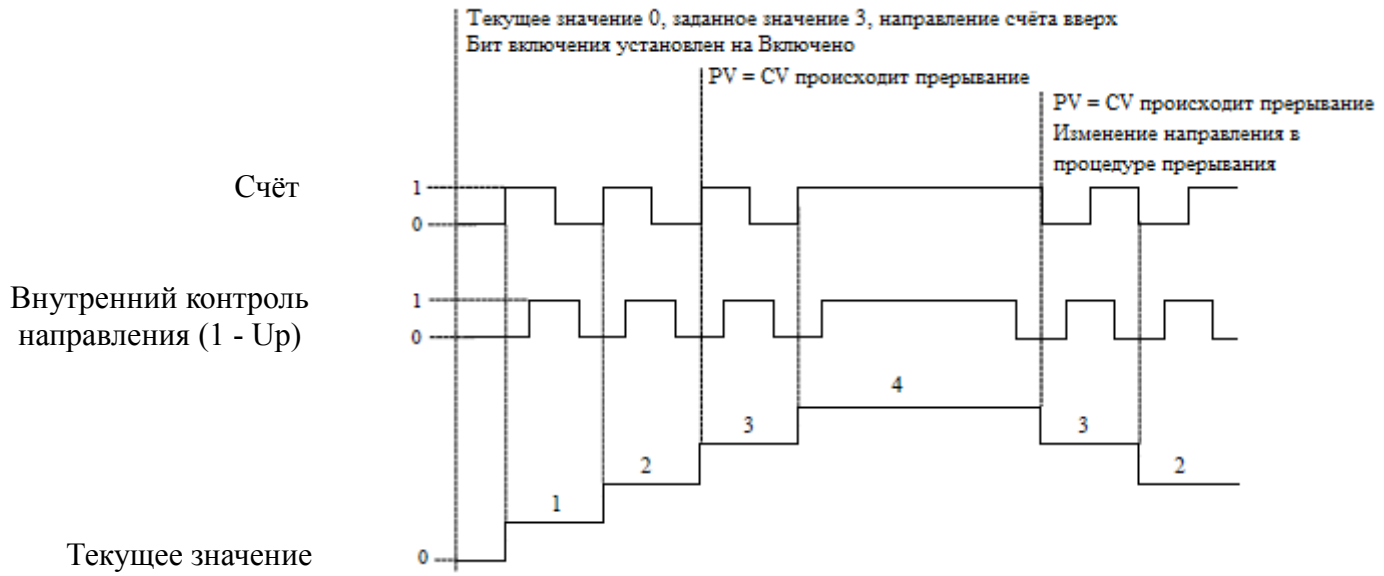
Временная диаграмма режимов 3, 4 или 5



Временная диаграмма режимов 6, 7 или 8



Временная диаграмма режимов 9, 10 или 11



6.13.3.4 Байт управления и Байт состояния

★ Байт управления

В SM области каждого высокоскоростного счетчика задается контрольный байт, чтобы сохранить свои данные конфигурации: одно управляющее слово (8 бит), текущее значение и предварительно установленное (двойное число с 32 бит). Начальное значение текущего заданного значения.

Если написано текущее значение высокоскоростного счетчика, он будет начинать отсчет от этой величины. Пожалуйста, смотрите ниже:

HSC0	HSC1	Описание
SM37.0	SM47.0	Эффективный уровень сигнала сброса: 0 = высокий, 1 = низкий
SM37.1	SM47.1	Эффективный уровень сигнала пуска: 0 = высокий, 1 = низкий
SM37.2	SM47.2	Скорость счетчика: 0 = 1x скорость; 1 = 4x скорость*
SM37.3	SM47.3	Направление счета 0 = минус; 1 = плюс
SM37.4	SM47.4	Написать направления счета в HSC? 0 = нет; 1 = да
SM37.5	SM47.5	Написать новое предустановленное значение в HSC? 0 = нет; 1 = да
SM37.6	SM47.6	Написать новое текущее значение в HSC? 0 = нет; 1 = да
SM37.7	SM47.7	Разрешить этот высокоскоростной счетчик? 0 = нет; 1 = да
SMD38	SMD48	Текущее значение
SMD42	SMD52	Предустановленное значение

Только после определения высокоскоростного счетчика и определения его режима, можно запрограммировать динамические параметры счетчика. Байт управления предусмотрен для каждого высокоскоростного счетчика, и вы можете работать следующим образом:

★ Включить или отключить HSC.

★ Управление направлением счета (ограничивается режиме 0, 1 и 2), либо начальное направление всех других режимов.

★ Загрузить текущее значение.

★ Загрузить заданное значение.

Контрольный байт, соответствующий текущему значению и заданное значение должны быть загружены до выполнения инструкции HSC.

Следующая таблица описывает каждый из этих управляющих битов.

★ Байт состояния

В области SM, каждый высокоскоростной счетчик имеет байт состояния, в котором некоторые биты указывают текущее направление счета, и является ли текущее значение больше или равно, чем заданное значение. Определение битов состояния для каждого скоростного счетчика показано в следующей таблице.

HSC0	HSC1	Описание
SM36.0	SM46.0	Зарезервировано
SM36.1	SM46.1	Зарезервировано
SM36.2	SM46.2	Зарезервировано
SM36.3	SM46.3	Зарезервировано
SM36.4	SM46.4	Зарезервировано
SM36.5	SM46.5	Текущее направление счета 0 = минус; 1 = плюс
SM36.6	SM46.6	Текущее значение равно заданному значению: 0 = не равны; 1 = равно
SM36.7	SM46.7	Текущее значение больше, чем заданное значение: 0 = не больше; 1 = больше

★ Настройка текущей и предустановленной величины

Каждый высокоскоростной счетчик имеет 32-разрядное текущее значение (т.е. начальное значение) и 32-разрядное заданное значение. Любое текущее значение или заданное значение подписаны двойным целым. Для того, чтобы написать новое текущее значение и заданное значение в высокоскоростной счетчик, управляющий байт и байт SM, которые хранят текущее значение и / или заданное значение, должны быть настроены в первую очередь, а затем инструкция HSC должна быть выполнена так, что бы новые значения

могли быть записаны в высокоскоростной счетчик. Следующая таблица показывает байты SM, которые хранят новое текущее значение и заданное значение.

	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
Новое текущее значение	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
Новое заданное значение	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

★ **Доступ к текущему значению высокоскоростного счетчика**
Текущее значение счёта высокоскоростного счетчика только для чтения и может быть представлено только в виде двойного целого числа (32-бит). Текущее значение счёта высокоскоростного счетчика доступна с использованием типа памяти (HC) и номера счетчика; Например, HC0 представляет текущее значение HSC0, как показано на следующей схеме.

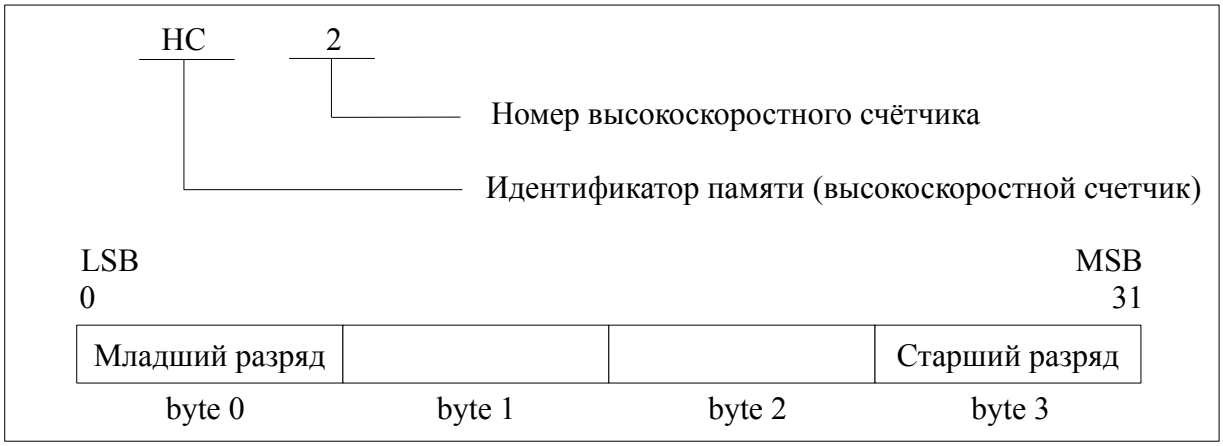


Рисунок 6-8 Доступ к текущему значению высокоскоростного счетчика

6.13.3.5 Назначение прерываний

Каждый режим поддерживает PV = CV (текущее значение равно заданному значению) прерывания. Режим, который использует внешний вход сброса, поддерживает прерывание External Reset. Режим, который использует внешний вход управления направлением, поддерживает прерывание Direction Changed. У каждого из этих прерываний условия могут быть включены или отключены отдельно.
Пожалуйста, обратитесь к главе 6.10.3 "Типы событий прерываний, поддерживаемых Kinco-K5" для подробной информации.

6.13.3.6 Программирование Высокоскоростных счетчиков

Вы можете запрограммировать высокоскоростной счетчик следующим образом:
★ Назначить управляющий байт.
★ Назначить текущее значение (т.е, начальное значение) и установленное значение.
★ (Необязательно) Назначьте процедуру прерываний с помощью команды ATCH.
★ Определите счетчик и его режим с помощью инструкции HDEF.
Примечание: Инструкция HDEF может быть выполнена только один раз для каждого высокоскоростного счетчика после перехода CPU в режим RUN.
★ Запустите высокоскоростной счетчик с помощью команды HSC.

Ниже подробное описание шагов для инициализации и работы, принимающих HSC0 в качестве примера. Рекомендуется сделать подпрограмму, которая содержит команду HDEF и другие инструкции

инициализации и вызвать эту подпрограмму в основной программе с помощью SM0.1, чтобы сократить время цикла CPU.

★ **Использование HSC**
Следующий пример использует режим 9. Другие режимы имеют аналогичные шаги.
★ В подпрограмме инициализации загрузите нужный режим управления в SMB37.
Например (1x скорость счета), SMB37 = B # 16 # F0 означает:
★ Включить HSC0
★ Написать новое текущее значение в HSC0
★ Написать новое заданное значение для HSC0
★ Установить вход пуска и вход сброса активными
★ Загрузите желаемое текущее значение (32-бит) в SMD38. Если загружен 0, то SMD38 очищается.
★ Загрузите нужное заданное значение (32-бит) в SMD42.
★ (Необязательно) Установите событие CV = PV (событие 18) в программе обработки прерываний для ответа в режиме реального времени, когда текущее значение равно предустановленному значению.
★ (Необязательно) Установите событие изменения направления (событие 17) в программе обработки прерываний для ответа в режиме реального времени для события изменения направления.
★ (Необязательно) Установите событие внешнего сброса (событие 16) в программе обработки прерываний для ответа в режиме реального времени на событие внешнего сброса.
★ Выполните команду HDEF с входом HSC, установленным в 0, и входом MODE, установленным в 9.
★ Выполните команду HSC, чтобы CPU настроил HSC0 и запустил его.

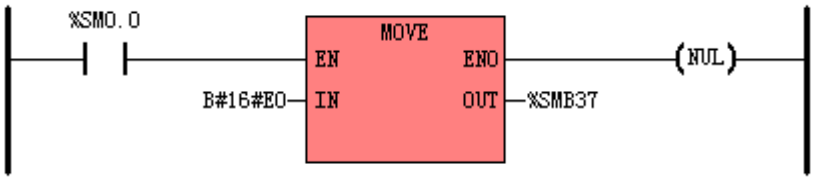
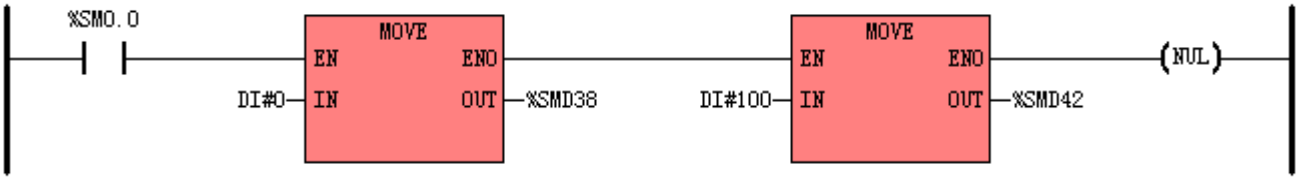
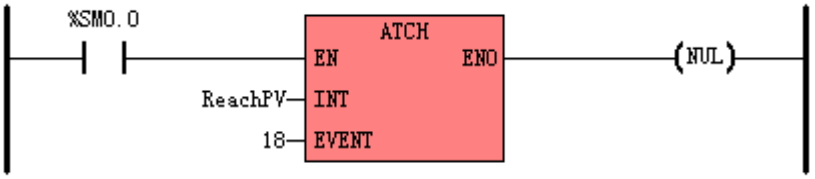
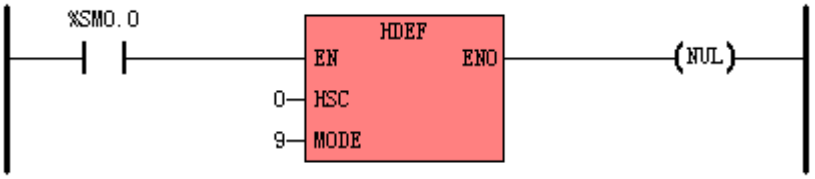
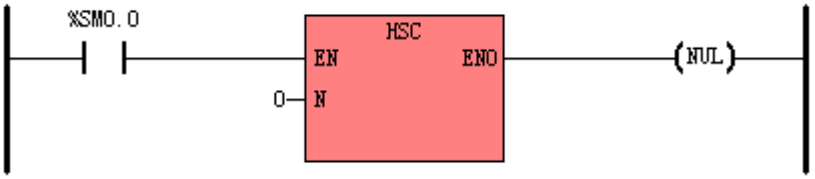
★ **Изменение направление счета в режиме 0, 1 и 2:**
Ниже показано, как изменить направление HSC0 (режим 0, 1 и 2).
★ Загрузите нужный режим управления в SMB37:
SMB37 = B # 16 # 90: Включить счетчик,
Установить новое направление, счет Down
★ Выполните команду HSC, чтобы CPU настроил HSC0 и запустил его.

★ **Загрузка нового текущего значения (во всех режимах)**
Ниже показано, как изменить текущее значение (т.е. начальное значение) HSC0.
★ Загрузите нужный режим управления в SMB37:
SMB37 = B # 16 # C0 Включить счетчик,
Разрешить запись нового текущего значения в HSC0.
★ Загрузите требуемое текущее значение в SMD38. Если загружен 0, то SMD38 очищается.
★ Выполните команду HSC, чтобы CPU настроил HSC0 и запустил его.
★ **Загрузка нового заданного значения (во всех режимах)**
Ниже показано, как изменить заданное значение HSC0.
★ Загрузите нужный режим управления в SMB37:
SMB37 = B # 16 # A0 Включить счетчик,
Разрешить запись нового заданного значения для HSC0.
★ Загрузите нужное заданное значение в SMD42.
★ Выполните команду HSC, чтобы CPU настроил HSC0 и запустил его.

★ **Отключение высокоскоростного счетчика (во всех режимах)**
Ниже приводятся порядок отключения HSC0.
★ Загрузите нужный режим управления в SMB37:
SMB37 = B # 16 # 00 Отключить счетчик;
★ Выполните команду HSC, чтобы CPU отключил счетчик.

6.13.3.7 Примеры

Следующий пример также использует HSC0.

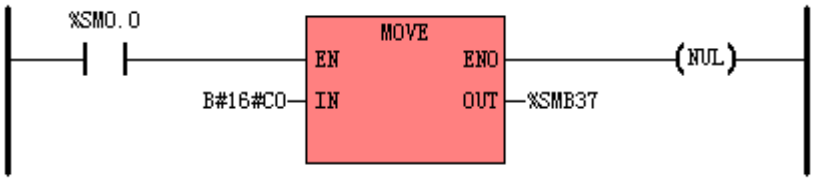
	Использование
LD	**Подпрограмма инициализации: Initialize** (* Network 0 *) (* Скорость счета 1x; Включить HSC0; Разрешить обновление текущего значения и заданного значения; Установить вход пуска и входа сброса активными *)  (* Network 1 *) (* Установите новое текущее значение и новое заданное значение *)  (* Network 2 *) (* Установите CV = PV событие (событие 18) ReachPV подпрограммы прерываний *)  (* Network 3 *) (* Определить HSC0 режим 9 *)  (* Network 4 *) (* Настройка и запуск HSC0 *) 

LD

Подпрограмма обработки прерывания: ReachPV

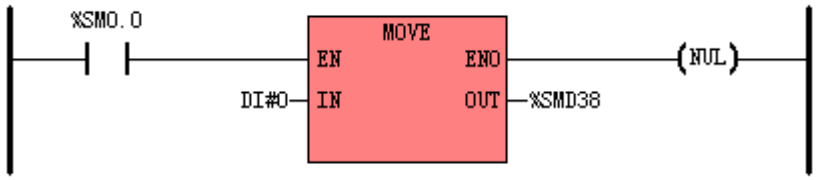
(* Network 0 *)

(* Разрешить обновление текущего значения *)



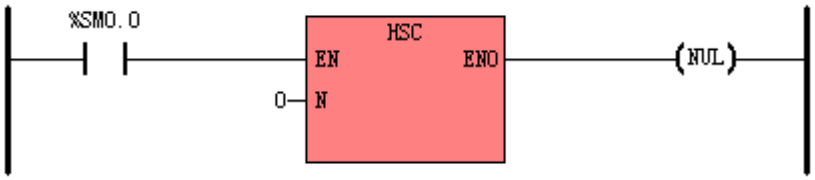
(* Network 1 *)

(* Установите новое текущее значение равным 0, чтобы пересчитать *)



(* Network 2 *)

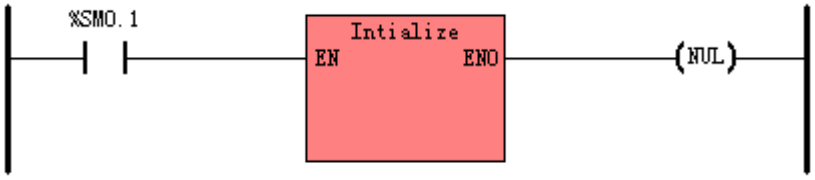
(* Настройка и перезапуск HSC0 *)



Основная программа:

(* Network 0 *)

(* Вызов подпрограммы Initialize *)



IL

Подпрограмма инициализации: Initialize

```
(* Network 0 *)
(* Скорость счета 1x; Включить HSC0; Разрешить обновление текущего значения и заданного значения; Установить вход пуска и входа сброса активными *)
LD          %SM0.0
MOVE      B#16#E0, %SMB37

(* Network 1 *)
(* Установите новое текущее значение и новое заданное значение *)
LD          %SM0.0
MOVE      DI#0, %SMD38
MOVE      DI#100, %SMD42

(* Network 2 *)
(* Установите CV = PV событие (событие 18) ReachPV подпрограммы прерываний *)
LD          %SM0.0
ATCH       ReachPV, 18

(* Network 3 *)
(* Определить HSC0 режим 9 *)
LD          %SM0.0
HDEF       0, 9

(* Network 4 *)
(* Настройка и запуск HSC0 *)
LD          %SM0.0
HSC         0
```

Подпрограмма обработки прерывания: ReachPV

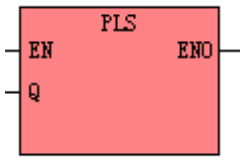
```
(* Network 0 *)
(* Разрешить обновление текущего значения *)
LD          %SM0.0
MOVE       B#16#C0, %SMB37

(* Network 1 *)
(* Установите новое текущее значение равным 0, чтобы пересчитать *)
LD          %SM0.0
MOVE      DI#0, %SMD38

(* Network 2 *)
(* Настройка и перезапуск HSC0 *)
LD          %SM0.0
HSC         0
```

Основная программа:

```
(* Network 0 *)
(* Вызов подпрограммы Initialize *)
LD         %SM0.1
CAL       Intialize
```

6.13.4 Инструкции высокоскоростных импульсных выходов

Здесь высокоскоростной импульсный выход подразумевает серию выходных импульсов (PTO) или широтно-импульсной модуляции (PWM).
K5 обеспечивает два генератора импульсов PWM, используя Q0.0 и Q0.1, которые называются PWM0 и PWM1. Генератор и изображение регистра DO имеют один и тот же адрес Q0.0 и Q0.1. Если PWM запускается в Q0.0 или Q0.1, генератор будет контролировать выходной канал и запретит выход общей функции. Когда функция PWM запрещена, Q0.0 и Q0.1 будет контролироваться регистром изображения DO.
Наибольшая выходная скорость K504 составляет 20 кГц, другие модули CPU имеют скорость 200 кГц.

	Название	Использование	Группа	
LD	PLS			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PLS	PLS *Q*	U	

Операнд	Вход / Выход	Тип данных	Описание
Q	Вход	INT	Константа (0 или 1)

Инструкция PLS используется для загрузки соответствующих конфигураций PTO / PWM, указанных Q от указанных регистров SM, а затем эксплуатирует генератор PTO / PWM.
В LD, вход EN решает, следует ли выполнить команду PLS.
В IL, значение CR решает, следует ли выполнять инструкции PLS. Это не влияет на CR.

6.13.4.1 Настройка и управление PWM

Каждый генератор PWM снабжен некоторыми регистрами в SM области для хранения своих настроек или индикации его состояния.
Характеристики сигнала PWM могут быть изменены путем изменения соответствующих регистров SM, а затем выполнения команды PLS. Следующая таблица описывает управляющие регистры.

Q0.0	**Q0.1**	Описание
SM67.0	SM77.0	Следует ли обновлять время цикла: 0 = не обновлять; 1 = обновлять
SM67.1	SM77.1	Следует ли обновить ширину импульсов: 0 = не обновлять; 1 = обновлять
SM67.2	SM77.2	Зарезервировано
SM67.3	SM77.3	Время базы: 0 = 1 мкс; 1 = 1 мс
SM67.4	SM77.4	Зарезервировано
SM67.5	SM77.5	Зарезервировано
SM67.6	SM77.6	1 = PWM
SM67.7	SM77.7	Включение: 0 = отключено; 1 = включено
SMW68	SMW78	Значение времени цикла, диапазон: от 2 до 65535
SMW70	SMW80	Значение длительности импульса, диапазон: от 0 до 65535

Следующая таблица описывает биты состояния PTO / PWM генераторов.

Q0.0	Q0.1	Биты состояния
SM66.4	SM76.4	Профиль PTO прекращён в связи с возрастанием ошибки в расчетах: 0 = нет ошибки; 1 = прекращён
SM66.5	SM76.5	Профиль PTO прекращён в связи с командой пользователя: 0 = не прекращено; 1 = прекращено
SM66.6	SM76.6	Канал PTO переполнен / не наполнен: 0 = нет; 1 = переполнен / не наполнен
SM66.7	SM76.7	PTO в режиме ожидания: 0 = в работе; 1 = в режиме ожидания

Бит ожидания PTO (SM66.7 или SM76.7) указывает на завершение вывода серии импульсов. Кроме того, как только серия импульсов завершена, вызывается соответствующая программа обработки прерывания. Если используется работа нескольких сегментов, обработка прерываний вызывается, как только таблица профиля завершена.

6.13.4.2 Операции PWM

Ниже на примере PWM0 показано, как сконфигурировать и эксплуатировать генератор PTO / PWM в программе пользователя.

★ **Инициализация PWM выхода**
Используйте SM0.1 (первое сканирование бита памяти), чтобы вызвать подпрограмму, которая содержит инструкции инициализации. Так как используется SM0.1, подпрограмма должна быть вызвана только один раз, это уменьшает время сканирования и обеспечивает лучшую структуру программы.
Следующие шаги описывают, как настроить PWM0 в подпрограмме инициализации:
★ Загрузите нужный режим управления в SMB67:
Например, SMB67 = B # 16 # D3 означает:

- ◆ Включение функции PWM
- ◆ Выберите операцию PWM
- ◆ Выберите 1 мкс в качестве времени базы
- ◆ Разрешите обновление значения ширины импульса и времени значение цикла
- ◆ Выберите Синхронный способ обновления

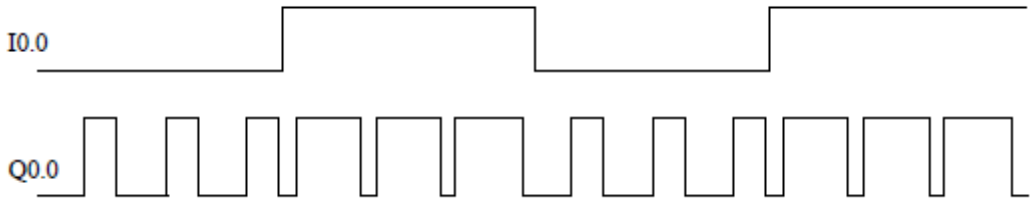
★ Загрузите значение времени цикла в SMW68.
★ Загрузите значение ширины импульса в SMW70.
★ Выполните команду PLS, чтобы CPU настроил PWM0 и запустите его.

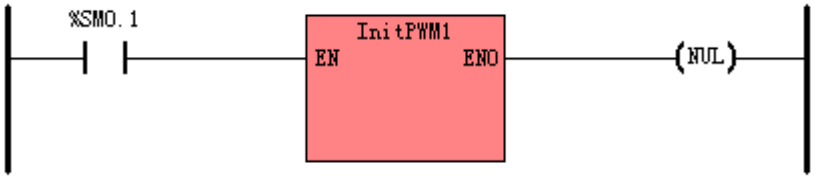
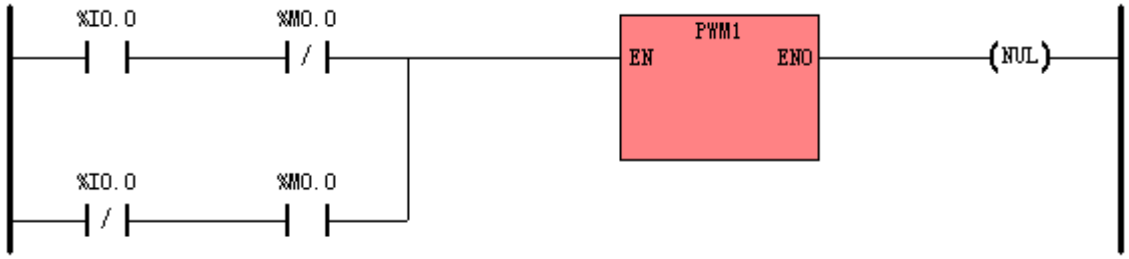
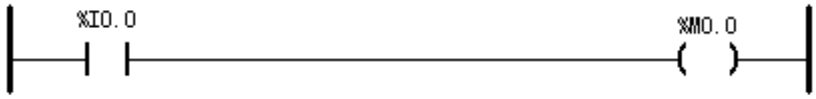
■ **Изменение длительности импульса для выхода PWM**
Следующие шаги описывают, как изменить ширину выходного импульса PWM (предположим, что SMB67 был с предустановленной B # 16 # D2 или B # 16 # DA.):
★ Загрузите широтно-импульсное значение (16-бит) в SMW70.
★ Выполните команду PLS, чтобы CPU настроил PWM и запустите его.

6.13.4.3 Пример

★ PWM

В этом примере используется PWM1 (выход через Q0.1).
Если I0.0 является ложным, изменить ширину импульса до 40% рабочего цикла; если I0.0 истина, изменить ширину импульса до 80% рабочего цикла. Временная диаграмма отображается следующим образом:

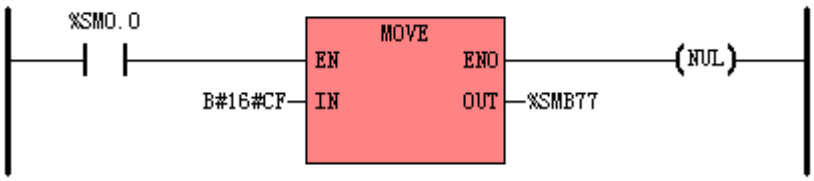


	Использование
LD	**Основная программа:** (* Network 0 *) (* Использование SM0.1 для вызова подпрограммы InitPWM1 для инициализации PWM1 *)  (* Network 1 *) (* Если состояние I0.0 изменится, подпрограмма PWM1 будет изменять длительность импульса. *)  (* Network 2 *) 

Подпрограмма InitPWM1:

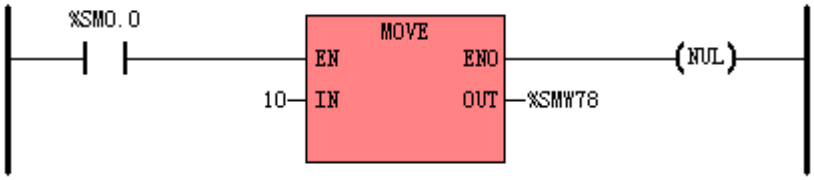
(* Network 0 *)

(* Выбор PWM1; Выберите 1 мс как время базы; Разрешите обновление значения времени цикла и длительности импульса *)



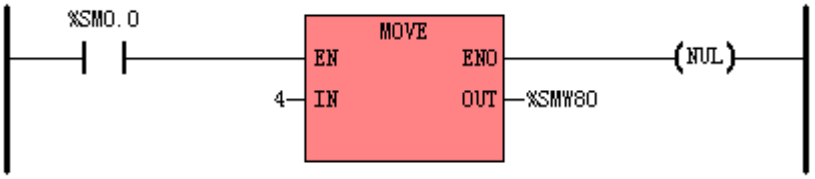
(* Network 1 *)

(* Установите время цикла PWM1 = 10 мс *)



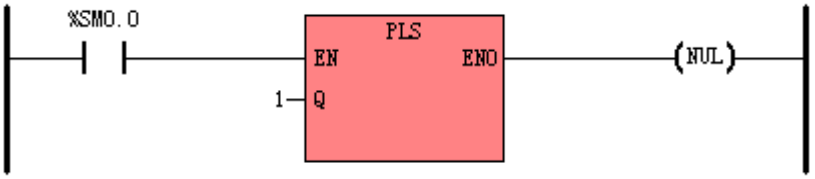
(* Network 2 *)

(* Установите ширину импульса PWM1 = 4 мс *)



(* Network 3 *)

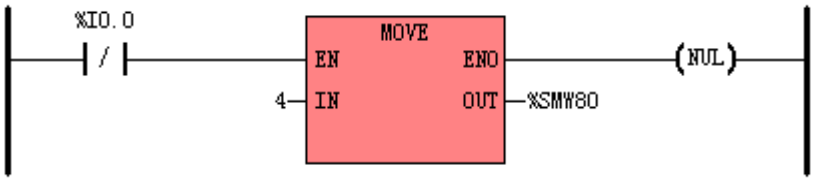
(* Выполнить PWM1 *)



Подпрограмма PWM1:

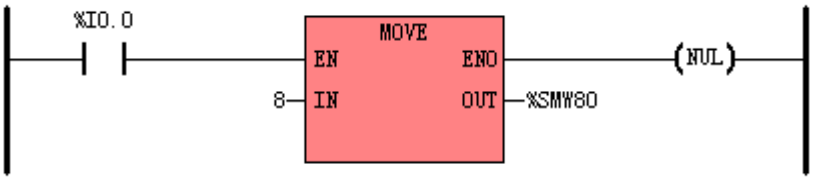
(* Network 0 *)

(* Если I0.0 ложно, ширина импульса PWM1 установлена 4 мс *)



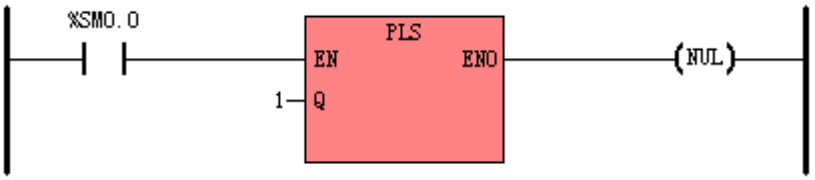
(* Network 0 *)

(* Если I0.0 истина, ширина импульса PWM1 установлена 8ms *)



(* Network 0 *)

(* Выполнить PWM1 *)



IL

Основная программа:

```
(* Network 0 *)
(* Использование SM0.1 для вызова подпрограммы InitPWM1 для инициализации PWM1 *)
LD          %SM0.1
CAL         InitPWM1
(* Network 1 *)
(* Если состояние I0.0 изменится, подпрограмма PWM1 будет изменять длительность импульса. *)
LD          %I0.0
ANDN        %M0.0
OR(
LDN         %I0.0
AND         %M0.0
)
CAL PWM1
(* Network 2 *)
LD          %I0.0
ST          %M0.0
```

Подпрограмма InitPWM1:

```
(* Network 0 *)
(* Выбор PWM1; Выберите 1 мс как время базы; Разрешите обновление значения времени цикла и длительности импульса *)
LD          %SM0.0
MOVE        B#16#CF, %SMB77
```

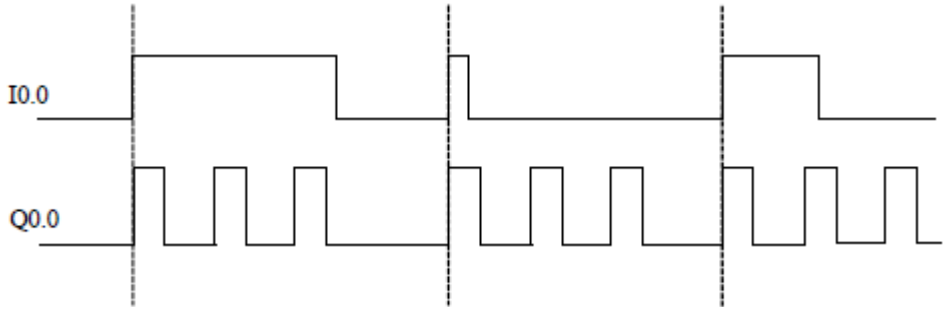
```
(* Network 1 *)
(* Установите время цикла PWM1 = 10 мс *)
LD          %SM0.0
MOVE      10, %SMW78
(* Network 2 *)
(* Установите ширину импульса PWM1 = 4 мс *)
LD          %SM0.0
MOVE      4, %SMW80
(* Network 3 *)
(* Выполнить PWM1 *)
LD          %SM0.0
PLS         1
```

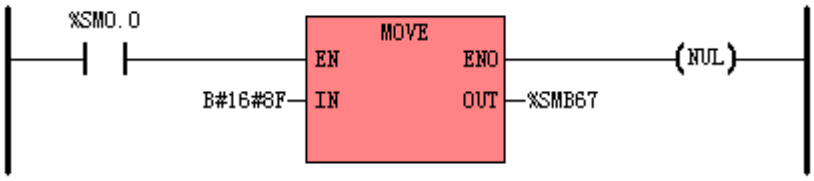
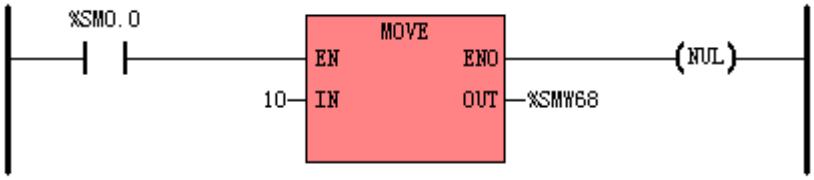
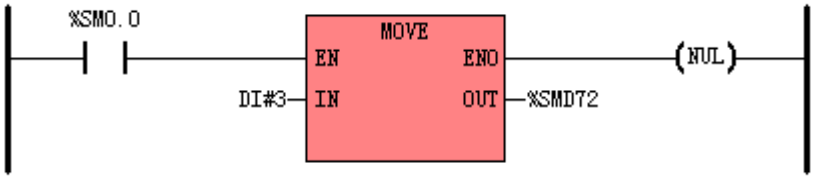
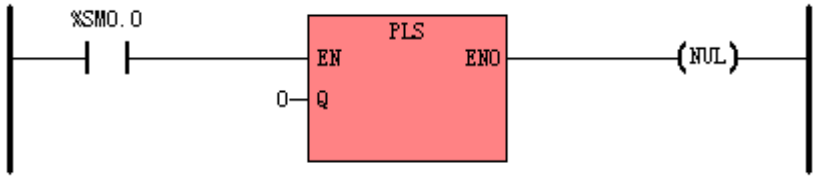
Subroutin *PWM1*:

```
(* Network 0 *)
(* Если I0.0 ложно, ширина импульса PWM1 установлена 4 мс *)
LDN         %I0.0
MOVE      4, %SMW80
(* Network 1 *)
(* Если I0.0 истина, ширина импульса PWM1 установлена 8ms *)
LD          %I0.0
MOVE      8, %SMW80
(* Network 2 *)
(* Выполнить PWM1 *)
LD          %SM0.0
PLS         1
```

★ Работа PTO (одно - сегментный)

В этом примере используется PTO0 (выход через Q0.0).
Запуск PTO0 и выход 3 импульсов каждый раз по переднему фронту I0.0.
Временная диаграмма отображается следующим образом:



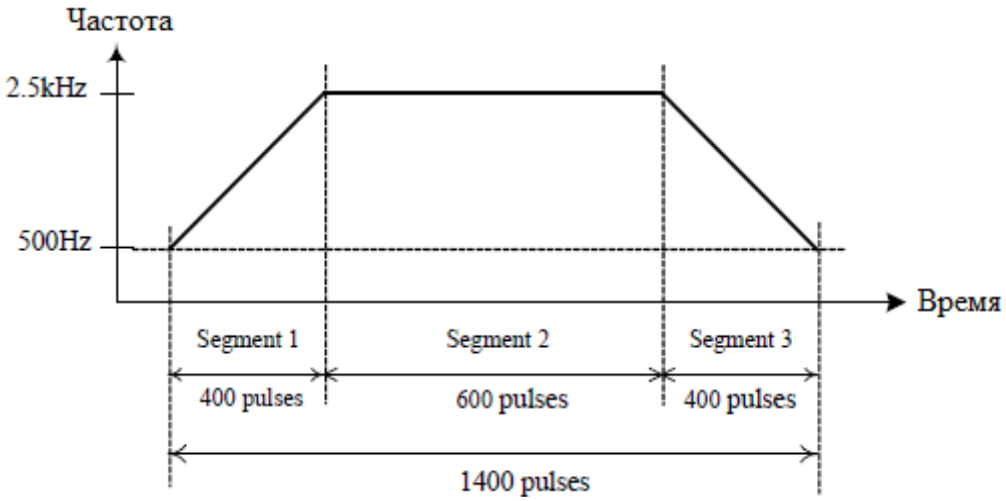
	Использование
LD	**Основная программа:** (* Network 0 *) (* Пуск PTO0 по переднему фронту I0.0 *)  **Подпрограмма PTO0:** (* Network 0 *) (* Выберите операцию одного сегмента для PTO0; Выберите 1 мс как время базы; Разрешите обновление времени цикла и количества импульсов *)  (* Network 1 *) (* Установите время цикла 10 мс *)  (* Network 2 *) (* Установите количество импульсов равным 3 импульса *)  (* Network 3 *) (* Выполнить PTO0 *) 

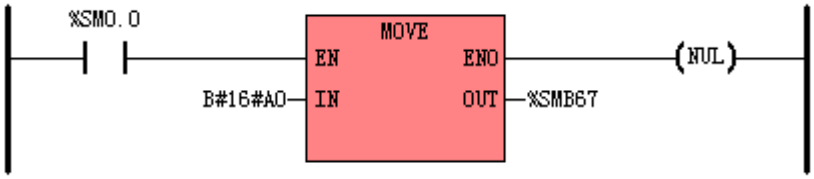
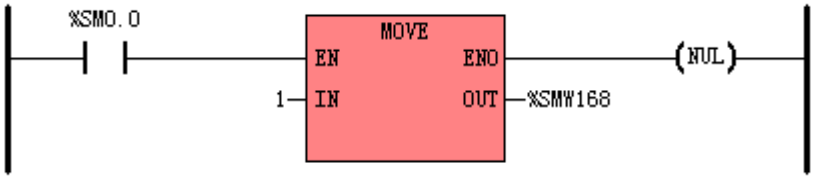
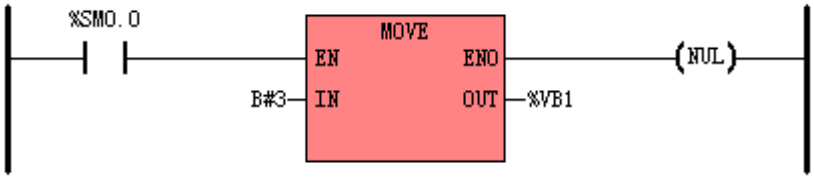
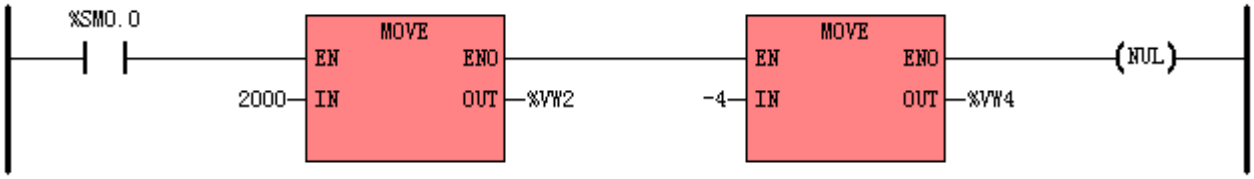
IL	**Основная программа:** (* Network 0 *) (* Пуск PTO0 по переднему фронту I0.0 *) LD %I0.0 R_TRIG CAL PTO0

IL	**Подпрограмма PTO0:**
	(* Network 0 *)
	(* Выберите операцию одного сегмента для PTO0; Выберите 1 мс как время базы; Разрешите обновление времени цикла и количества импульсов *)
	LD %SM0.0
	MOVE B#16#8F, %SMB67
	(* Network 1 *)
	(* Установите время цикла 10 мс *)
	LD %SM0.0
	MOVE 10, %SMW68
	(* Network 2 *)
	(* Установите количество импульсов будет 3 импульсов *)
	LD %SM0.0
	MOVE DI#3, %SMD72
	(* Network 3 *)
	(* Выполнить PTO0 *)
	LD %SM0.0
	PLS 0

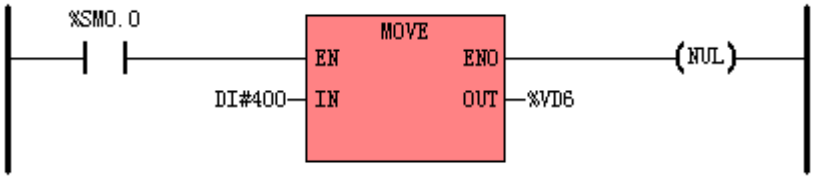
★ Работа PTO (много-сегментный)

В этом примере используется PTO0 (выход через Q0.0).
Пуск PTO0 по переднему фронту I0.0.
Рассчитать значения много-сегментного профиля в соответствии с приведенной ниже диаграммой.

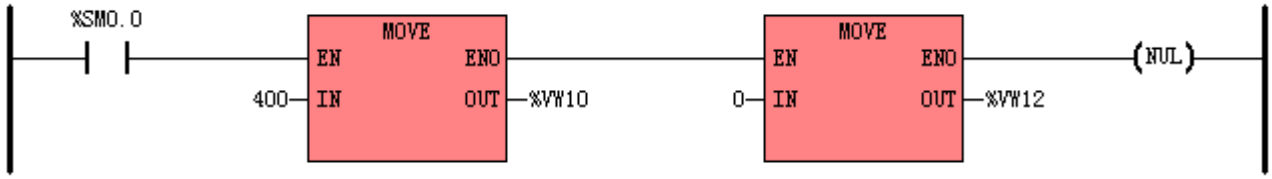


	Использование
LD	**Основная программа:** (* Network 0 *) (* Пуск PTO0 по переднему фронту I0.0 *)  **Подпрограмма PTO0:** (* Network 0 *) (* Включить PTO0; Выбрать много-сегментную операцию; Установка времени базы, 1мкс *)  (* Network 1 *) (* Используйте VB1 в качестве исходной позиции таблицы профилей *)  (* Network 2 *) (* Установить количество сегментов значение 3 *)  (* Network 3 *) (* Сегмент 1: Установите начальное время цикла равным 2000мкс; Установите время цикла, приращения равным -4мкс *) 

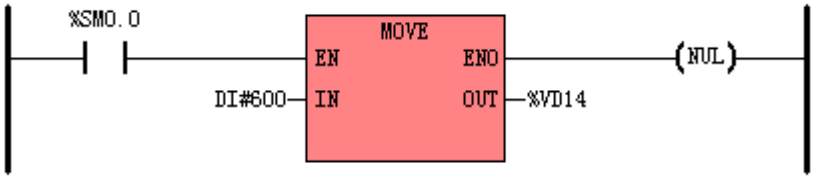
(* Network 4 *)
(* Сегмент 1: Установить количество импульсов до 400 *)



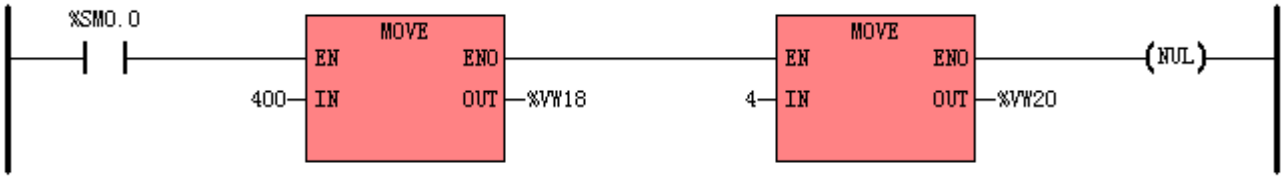
(* Network 5 *)
(* Сегмент 2: Установите начальное время цикла для 400мкс; Установите время цикла приращение равным 0 *)



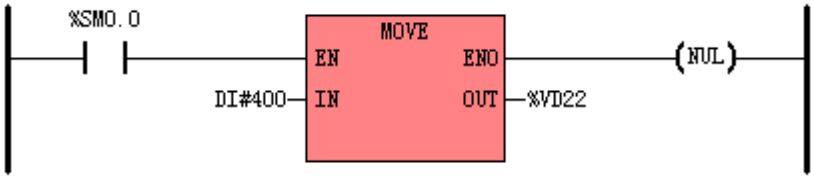
(* Network 6 *)
(* Сегмент 2: Установите количество импульсов на 600 *)



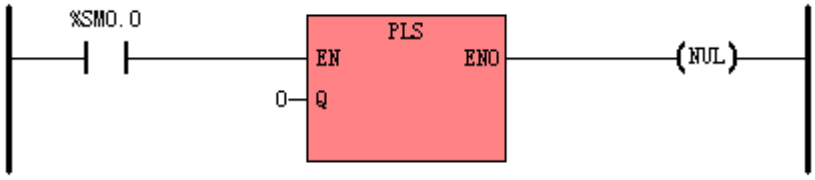
(* Network 7 *)
(* Сегмент 3: Установите начальное время цикла для 400мкс; Установите время цикла приращения до 4мкс *)



(* Network 8 *)
(* Сегмент 3: Установите количество импульсов на 400 *)



(* Network 9 *)
(* Выполнить PTO0 *)



```
IL  MAIN Program:
    (* Network 0 *)
    LD %I0.0
    R_TRIG
    CAL PTO0                    (* Пуск PTO0 по переднему фронту I0.0 *)

    Subroutine PTO0:
    (* NETWORK 0 *)
    LD %SM0.0
    MOVE B#16#A0, %SMB67        (* Включить PTO0; Выбрать много-сегментную операцию;
                                 Установка времени базы, 1мкс *)
    MOVE 1, %SMW168             (* Используйте VB1 в качестве исходной позиции таблицы
                                 профилей  *)
    MOVE B#16#03, %VB1          (* Установить количество сегментов значение 3 *)
    (* Segment 1 *)
    MOVE 2000, %VW2             (* Установите начальное время цикла для 2000мкс *)
    MOVE -4, %VW4               (* Установите время цикла приращения до -4мкс *)
    MOVE DI#400, %VD6           (* Установите количество импульсов на 400 *)
    (* Segment 2 *)
    MOVE 400, %VW10             (* Установите начальное время цикла для 400мкс *)
    MOVE 0, %VW12               (* Установите время цикла приращение равным 0 *)
    MOVE DI#600, %VD14          (* Установите количество импульсов на 600 *)
    (* Segment 3 *)
    MOVE 400, %VW18             (* Установите начальное время цикла для 400мкс *)
    MOVE 4, %VW20               (* Установите время цикла приращения до 4мкс *)
    MOVE DI#400, %VD22          (* Установите количество импульсов на 400 *)
    PLS 0                       (* Выполнить PTO0 *)
```

6.14 Таймеры

Таймер является одним из функциональных блоков, определенных в стандарте IEC61131-3, всех трех типов т.е. TON, TOF и TP. Пожалуйста, обратитесь к главе 3.6.5 "Функциональные блоки и их примеры" для получения более подробной информации.

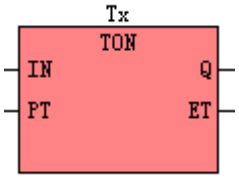
6.14.1 Разрешение таймера

Есть таймеры трёх разрешений. Число таймера определяет его разрешение. Заданное значение и текущее значение таймера кратные этому разрешению, например, значение 100, на таймере 10 мс, означает 1000 мс. Максимальное значение таймера составляет 32767 × разрешение. Предустановленное значение и текущее значение таймера кратные разрешению этого таймера, например, значение 100 на таймере 10 мс составит 1000 мс. PLC будет обновлять значение времени таймера только тогда, когда выполняется команда таймера. Это будет зависеть от цикла сканирования.

	CPU504	CPU504EX, CPU506	CPU508
Разрешение	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T63: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T127: 100ms	T0 --- T3: 1ms T4 --- T19: 10ms T20 --- T255: 100ms
Максимальное время	32767 * Разрешение	32767 * Разрешение	32767 * Разрешение

6.14.2 TON (Таймер задержки включения)

★ Описание

	Название	Использование	Группа	
LD	TON			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	TON	TON *Tx, PT*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
Tx	-	Экземпляр таймера	T
IN	Вход	BOOL	Питание
PT	Вход	INT	I, AI, AQ, M, V, L, SM, константа
Q	Выход	BOOL	Питание
ET	Выход	INT	Q, M, V, L, SM, AQ

Tx является примером функционального блока TON.

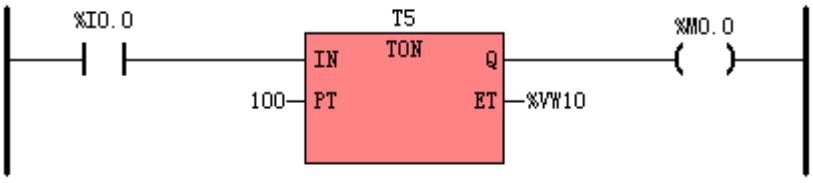
■ LD

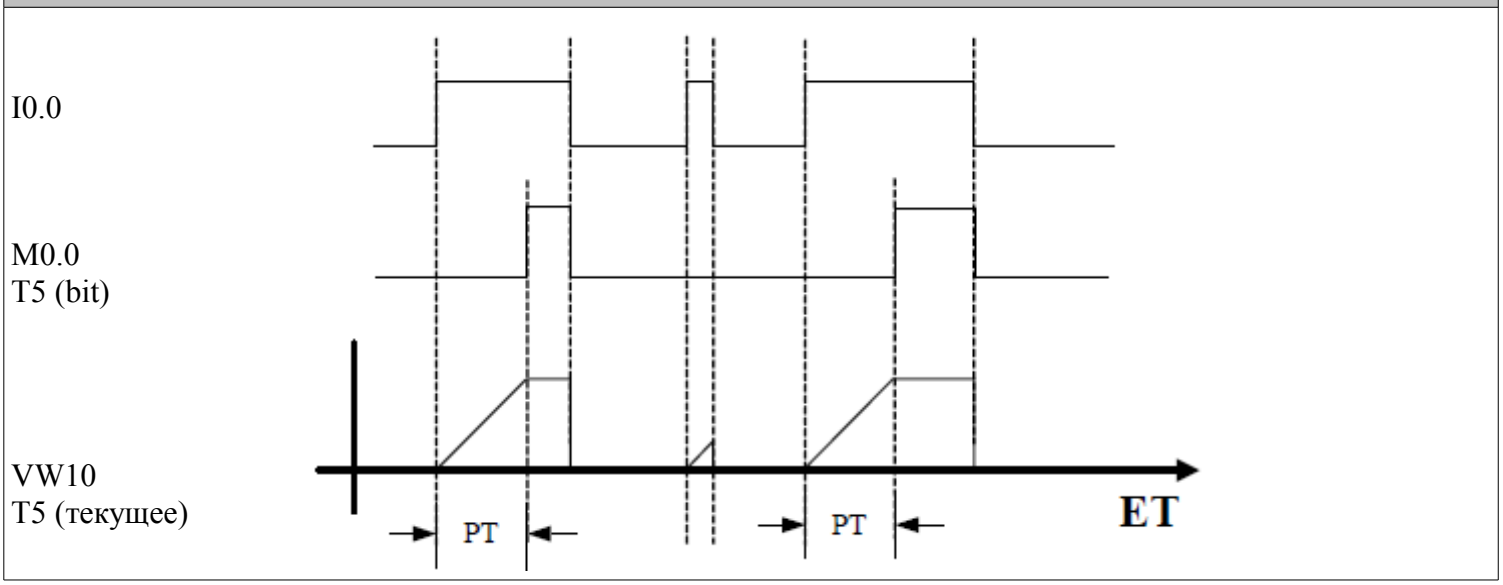
Tx начинает отсчёт времени по фронту на входе IN. Когда текущее время (то есть текущее значение) ET больше или равно заданному времени PT, и выход Q и бит статуса Tx установлены в TRUE. Если вход IN превращается в FALSE, Tx сбрасывается, то и выход Q и его значение бита состояния устанавливается FALSE, в то же время его текущее значение сбрасывается в 0.

■ IL

Tx начинает отсчёт времени по переднему фронту CR. Когда текущее значение больше или равно заданному значению PT, бит состояния Tx устанавливается в TRUE. Если CR превращается в FALSE, Tx сбрасывается, и его бит состояния устанавливается в FALSE, в то же время его текущее значение сбрасывается в 0. После каждого сканирования, CR устанавливается равным значению бита состояния Tx.

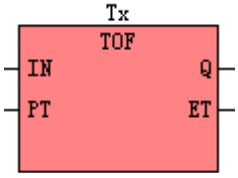
★ Пример

LD	IL
(* Network 0 *) (* T5 является экземпляром TON, а его заданное время 1000 мс (100 * 10 мс) *) 	(* Network 0 *) LD %I0.0 TON T5, 100 ST %M0.0

Временная диаграмма


6.14.3 TOF (Таймер задержки выключения)

★ Описание

	Название	Использование	Группа	
LD	TOF			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	TOF	TOF *Tx, PT*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
Tx	-	Экземпляр таймера	T
IN	Вход	BOOL	Питание
PT	Вход	INT	I, AI, AQ, M, V, L, SM, константа
Q	Выход	BOOL	Питание
ET	Выход	INT	Q, M, V, L, SM, AQ

Tx является примером функционального блока TOF.

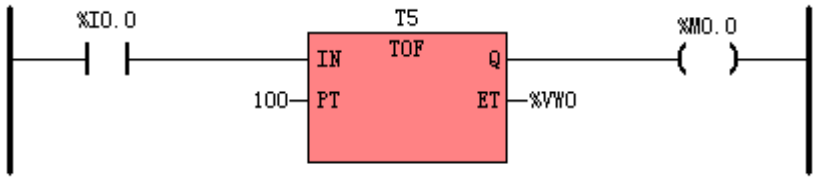
■ LD

Tx начинает отсчёт времени по заднему фронту на входе IN. Когда истекшее время (то есть текущее значение) ET больше или равно заданному времени PT, то выход Q и бит состояния Tx устанавливаются в FALSE. Если вход IN переходит в TRUE, Tx сбрасывается, и выход Q и бит состояния устанавливаются, в TRUE, тем временем его текущее значение сбрасывается в 0.

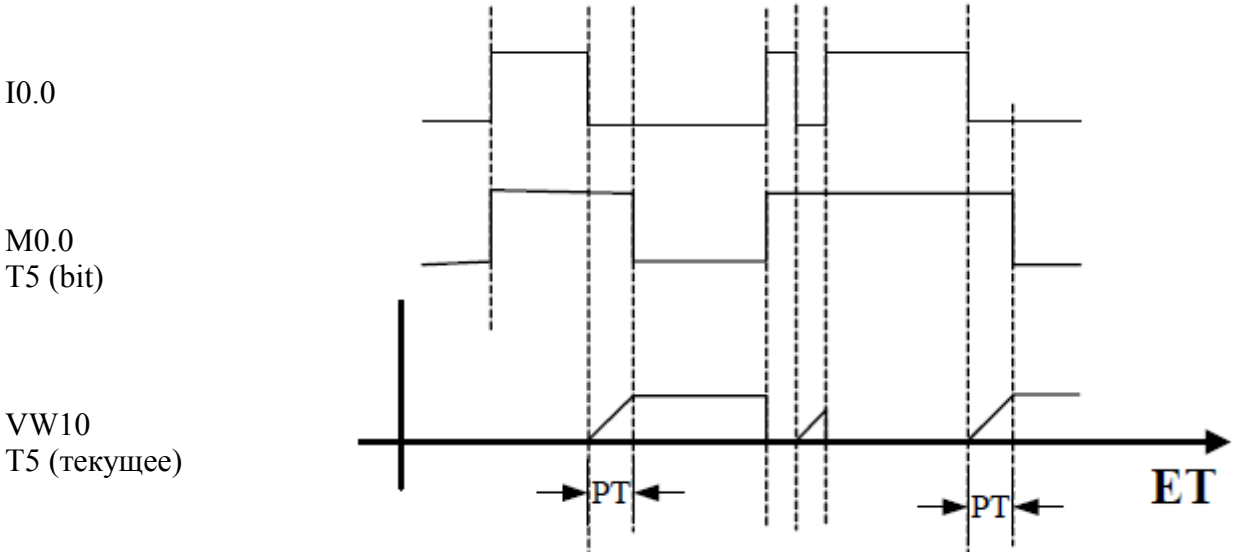
■ IL

Tx начинает отсчёт времени по заднему фронту CR. Когда текущее значение больше или равно заданному значению PT, бит состояния Tx устанавливается в FALSE. Если CR переходит в состояние TRUE, Tx сбрасывается, и состояние бита устанавливается в TRUE, в то же время его текущее значение сбрасывается в 0. После каждого сканирования, CR устанавливается равным значению бита состояния Tx.

★ Пример

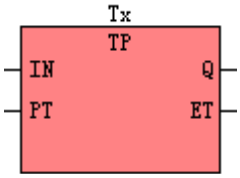
LD	IL
(* Network 0 *) (* T5 является экземпляром TOF, а его заданное время 1000 мс (100 * 10 мс) *)  	(* Network 0 *) LD %I0.0 TOF T5, 100 ST %M0.0

Временная диаграмма



6.14.4 TP (Импульсный таймер)

★ Описание

	Название	Использование	Группа	
LD	TP			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	TP	TP *Tx, PT*	P	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
Tx	-	Экземпляр таймера	T
IN	Вход	BOOL	Питание
PT	Вход	INT	I, AI, AQ, M, V, L, SM, константа
Q	Выход	BOOL	Питание
ET	Выход	INT	Q, M, V, L, SM, AQ

Tx является примером функционального блока TP. Инструкция TP используется для генерирования импульса в течение заданного времени.

■ LD

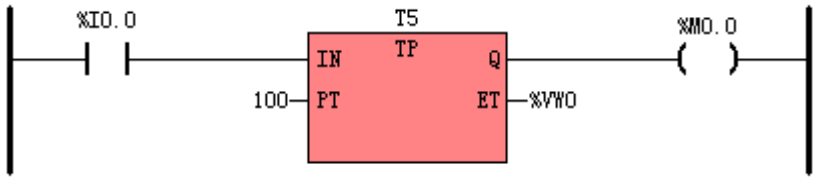
По переднему фронту входного сигнала IN, Tx начинает отсчёт времени, и выход Q и бит состояния Tx устанавливаются в положение TRUE. Выход Q и бит состояния остаются TRUE в течение заданного времени PT. Как только прошедшее время (т.е. текущего значения) ET достигает PT, то выход Q и бит состояния устанавливаются в FALSE.

■ IL

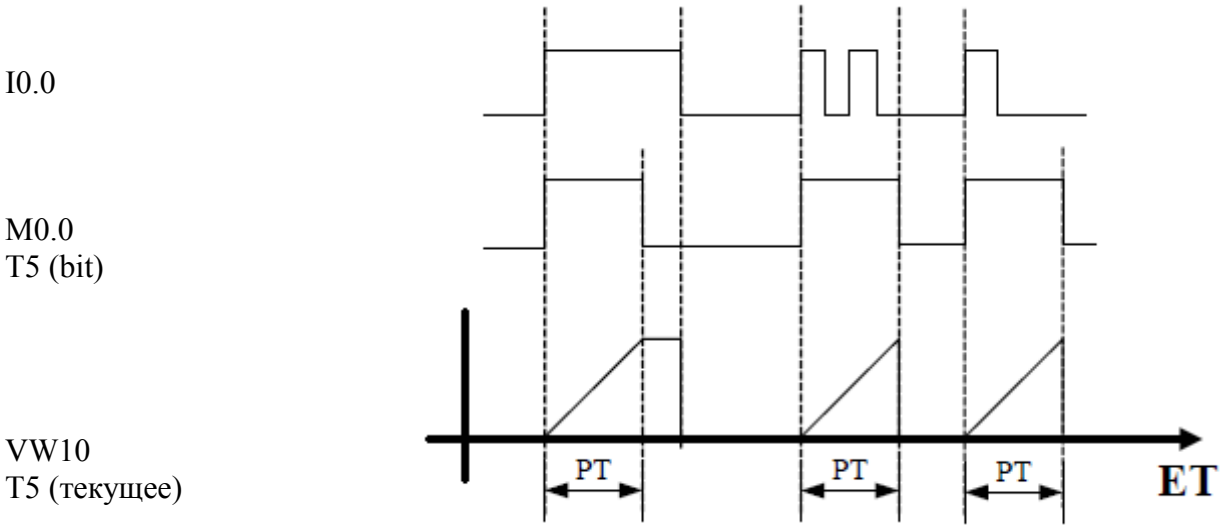
По переднему фронту CR, Tx начинает отсчёт времени, и бит состояния Tx устанавливается в TRUE. Бит состояния остается в TRUE в течение заданного времени PT. Как только текущее значение достигает PT, бит состояния становится FALSE.

После каждого сканирования, CR установлен быть бит состояния значение Tx.

★ Примеры

LD	IL
(* Network 0 *) (* T5 является экземпляром TP, а его заданное время 1000 мс (100 * 10 мс) *) 	(* Network 0 *) LD %I0.0 TP T5, 100 ST %M0.0

Временная диаграмма



6.15 PID

В Kinco-K5 предусмотрена инструкция PID и алгоритм положения. Вы можете использовать его в качестве PID регулятора установленной неподвижной точки с непрерывным входом и выходом, и вы можете использовать до 8 PID контуров в одном CPU.

★ Описание

	Название	Использование	Группа	
LD	PID			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PID	PID *AUTO, PV, SP, XO, KP, TR, TD,* PV_H, PV_L, XOUTP_H, XOUTP_L, CYCLE, XOUT, XOUTP	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти	Комментарий
AUTO	Вход	BOOL	I, Q, V, M, SM, L, T, C	Manual / Auto. 0 = Ручной, 1 = Авто.
PV	Вход	INT	AI, V, M, L	Переменная процесса
SP	Вход	INT	V	Уставка
XO	Вход	REAL	V	Значение вручную, диапазон [0.0, 1.0]
KP	Вход	REAL	V	коэффициент пропорциональности
TR	Вход	REAL	V	Сброс времени отклика интегратора. (Ед. Изм.: сек)
TD	Вход	REAL	V	Производные времени отклика производной . (Ед. Изм.: сек)
PV_H	Вход	INT	V	Значение верхнего предела PV
PV_L	Вход	INT	V	Значение нижнего предела PV
XOUTP_H	Вход	INT	V	Значение верхнего предела XOUTP
XOUTP_L	Вход	INT	V	Значение нижнего предела XOUTP
CYCLE	Вход	DINT	V	Период измерения. (Ед. Изм.: мс)
XOUT	Выход	REAL	V	Управляющее значение, диапазон [0.0, 1.0].

XOUTP	Выход	INT	AQ, V	Управляющее значение вспомогательное. Это нормализующее значение результата XOUT.

■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если EN равен 1, то эта инструкция выполняется, и не влияет на CR.

★ Другая информация
★ **Manual / Auto**
Это возможность переключаться между ручным и автоматическим режимом с помощью входа Auto.
Если Auto равен 0, то PID находится в ручном режиме, и в настоящее время значение входа XO должно быть установлено непосредственно в качестве управляющей переменной (XOUT).
Если Auto равен 1, то PID находится в автоматическом режиме, и теперь он должен выполнить расчеты PID в соответствии с входами и установить окончательный результат, как управляющей переменной (XOUT).

★ **Нормализация PV и SP**
PV и SP могут быть введены в периферическом формате (целое число). Но PID алгоритм нуждается в значение с плавающей точкой от 0.0 до 1.0, поэтому нормализация необходима.
Kinco-K5 автоматически завершает нормализации в соответствии с входным PV, SP, PV_H и PV_L. Вы можете присваивать любую линейную зависимость их значения, но входы должны быть той же размерности. Например, вы хотите контролировать давление на ожидаемую величину 25 Мпа. Преобразователь давления используется для измерения давления и измерительный диапазон преобразователя является 0-40MPa и его выходной диапазон 4-20. Выход преобразователя подключен к каналу модуля AI, и этот канал настроен следующим образом: адрес AIW0 и тип измерения "4-20", чьё измеренное значение "4000-20000". Теперь вы можете задавать следующие значения на входы PID:

	Фактический параметр	Комментарий
PV	AIW0	AIW0 может быть установлен как PV из-за их линейной зависимости.
SP	14000	14 мА. Потому что 14 мА означает, что значение 25 МПа реальное давление.
PV_L	4000	Нижнее предельное значение выхода преобразователя
PV_H	20000	Верхнее предельное значение выхода преобразователя

★ **Управляющее значение**
PID имеет два управляющих значения: XOUT и XOUTP.
XOUT является значением между 0,0 и 1,0 (то есть между 0,0 и 100,0%).
XOUTP является целым числом, определяемое пользователем, и является результатом нормализации XOUT в соответствии с входным XOUTP_H и XOUTP_L:

$$XOUTP = (XOUTP\_H - XOUTP\_L) * XOUT + XOUTP\_L$$

Это удобно для пользователя, чтобы передать XOUT_P к каналу AO. Например, результаты PID должны быть отправлены в регулирующий клапан через AQW0 модуля AO, тогда соответствующие параметры настройки будут следующие:

	Фактический параметр	Комментарий
XOUTP	AQW0	AQW0 может быть напрямую приниматься в качестве ПИД выхода для линейной зависимости между значением AQW0 и открытием клапана.
XOUTP_L	4000	Нижний предел значения AQW0.
XOUTP_H	20000	Верхний предел значения AQW0.

• Пропорциональная составляющая

Пропорциональная составляющая вносит изменения на выходе, который пропорционален текущей ошибке (E), которая представляет собой разность между заданным значением (SP) и переменной процесса (PV). Пропорциональный ответ можно регулировать путем умножения погрешности от пропорционального усиления (KP).

Пропорциональная формула:

$$MPn = KP * En = KP * (SPn - PVn)$$

где: MPn - это выходное значение пропорциональной составляющей в момент времени ***n***

KP - это пропорциональное усиление

SPn - это значение уставки в момент времени ***n***

PVn - это значение переменной в момент времени ***n***

Высокий коэффициент пропорционального усиления (KP) приводит к большому изменению выхода для данного изменения погрешности. Если коэффициент пропорционального усиления слишком высок, система может стать нестабильной. В противном случае, малый коэффициент усиления приводит к малому отклику выхода на большой вход погрешности, и менее чувствителен. Если коэффициент пропорционального усиления слишком мал, управляющее воздействие может быть слишком мало, отвечая на возмущение системы.

Цикл прямого действия, если KP положителен и обратного действия, если KP отрицателен.

• Интегральная составляющая

Вклад от интегрального звена пропорционален величине ошибки и длительности ошибки. Суммируя мгновенную ошибку с течением времени (интеграция ошибки) дает накопленное смещение, которое должно быть исправлено. Затем, накопленная ошибка делится на время интегрирования (TR) и добавляется к выходу контроллера. Величина вклада интегральной составляющей для общего управления определяет пропорциональное усиление (KP), период выборки (Ts) и интеграл времени (TR), который использует время для управления влияния из интегральной составляющей в расчете выходного (TR). Интегральное уравнение выглядит следующим образом:

$$MIn = KP * Ts * En / TR = KP * Ts * (SPn – PVn) / TR$$

где: **MIn** выходное значение интегральной составляющей в момент времени **n**

KP - это пропорциональное усиление

TR - это время интеграции

Ts - это цикл выборки

SPn - это значение уставки в момент времени **n**

PVn - это значение переменной в момент времени **n**

Интегральная составляющая (при добавлении к пропорциональной составляющей) ускоряет движение процесса к заданному значению и исключает остаточную стационарную ошибку, которая возникает только при пропорциональном контроллере. Если TR равен 0, интегральная составляющая отменяется.

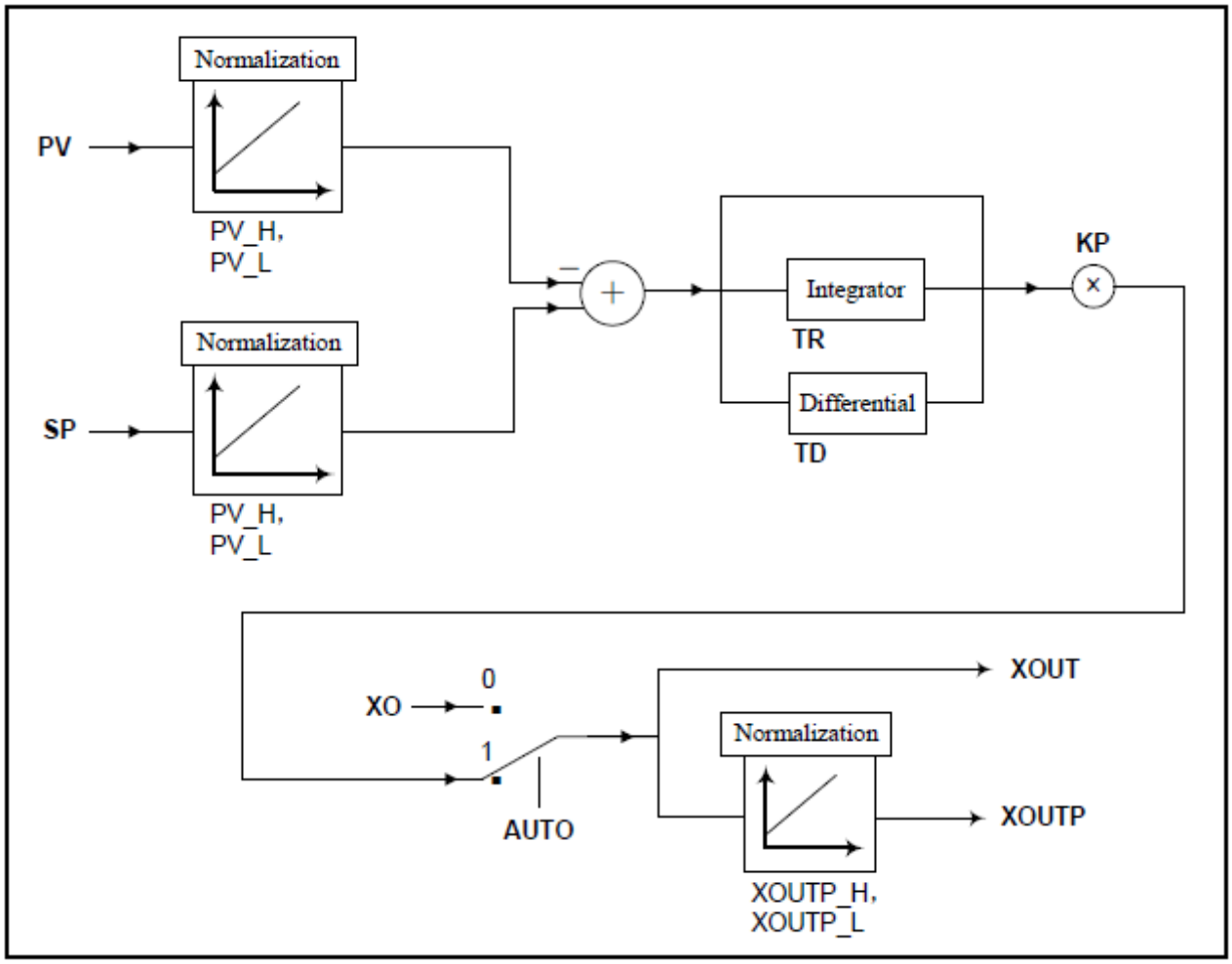
• Производная составляющая

Скорость изменения ошибки рассчитывается путем определения крутизны ошибки с течением времени (т.е. первой производной по времени) и умножением этой скорости изменения на производную времени (KD). Производная составляющая используется для уменьшения величины броска, возникающего в интегральной составляющей и улучшения стабильности процесса управления. Тем не менее, дифференциация сигнала усиливает шумы и, таким образом эта составляющая, является весьма чувствительна к шуму в составляющей ошибки, и может стать причиной нестабильности, если шум и дифференциальный коэффициент достаточно велики. Уравнение производной выглядит следующим образом:

$$MDn = KP * TD / Ts * (En – En-1) = KP * TD / Ts * ((SPn – PVn) - (SPn-1 – PVn-1))$$

где: MDn - это выходное значение производной составляющей в момент времени **n**
KP - это пропорциональное усиление
TD - это производная времени
Ts - это цикл выборки
SPn - это значение уставки в момент времени **n**
PVn - это значение переменной в момент времени **n**
PVn-1 - это значение уставки в момент времени **n-1**
PVn1 - это значение переменной в момент времени **n-1**

★ PID Схема

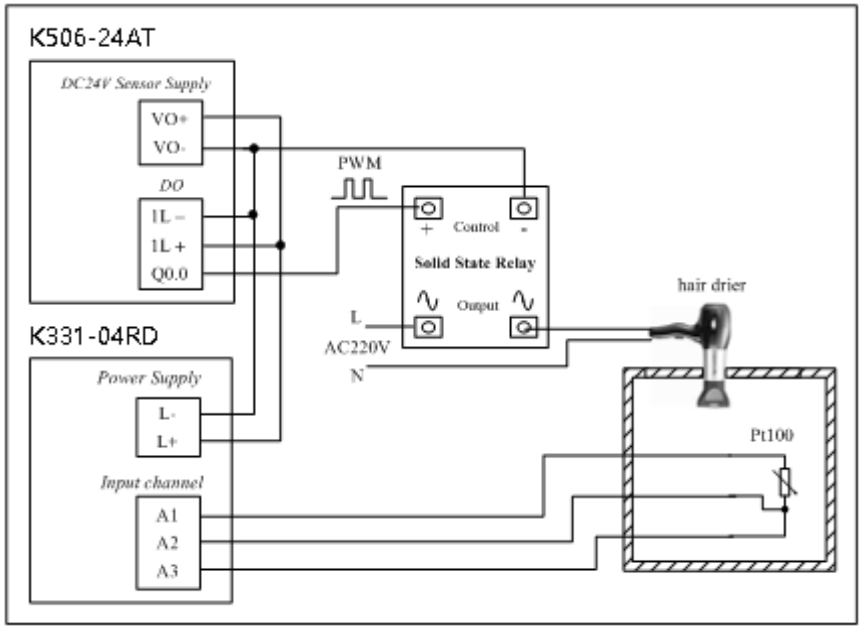


★ Пример

В этом примере, мы создаем систему управления, и эта система будет поддерживать постоянную температуру внутри шкафа, используя электрический нагреватель для нагрева, но с естественным охлаждением. Мы используем датчик Pt100 для измерения температуры, а также полупроводниковое реле для управления питанием нагревателя.

Мы используем K506-24AT и K531-04RD. PT100 подключен к каналу AIW0 в K531-04RD. Q0.0 подключен к управляющему контакту твердотельного реле. Q0.0 выводит импульсы PWM и результат ПИД используется для регулировки ширины импульса, и контроля мощности нагревателя.

Ниже приведена диаграмма этой системы:



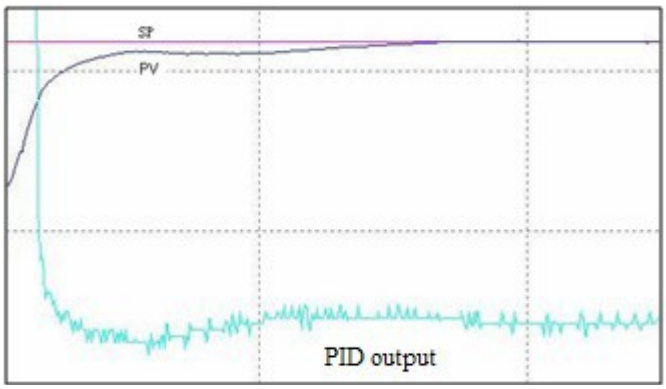
Идея этого примера программы выглядит следующим образом:

1) Использование таймера TON для генерирования простого выхода PWM.

2) Для того, чтобы улучшить эффект управления, мы поступим следующим образом: если погрешность (SP-PV) более чем 5 ºC, ПИД регулятор принимает ручной режим и его значение устанавливается на 100%; если погрешность меньше или равна 5 ºC, ПИД регулятор переключиться на автоматический режим.

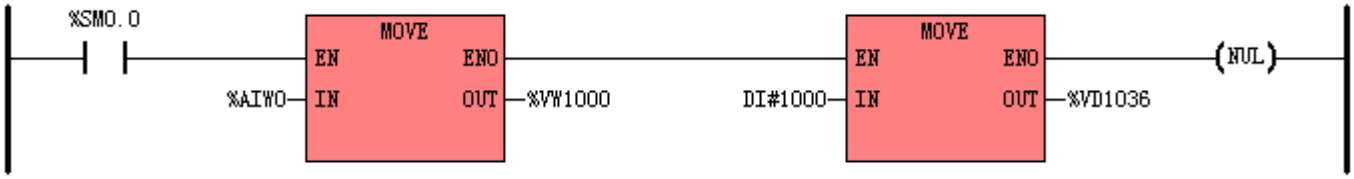
Если погрешность слишком большая, автоматический режим может привести к чрезмерному перерегулированю и более длительному времени регулировки из-за эффекта накопления интегральной составляющей, поэтому ПИД регулятор принимает ручной режим.

На следующем рисунке результат, когда SP установлен на 50 ºC.

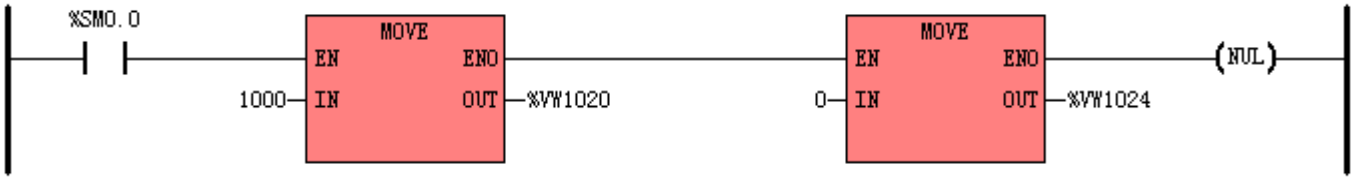


LD

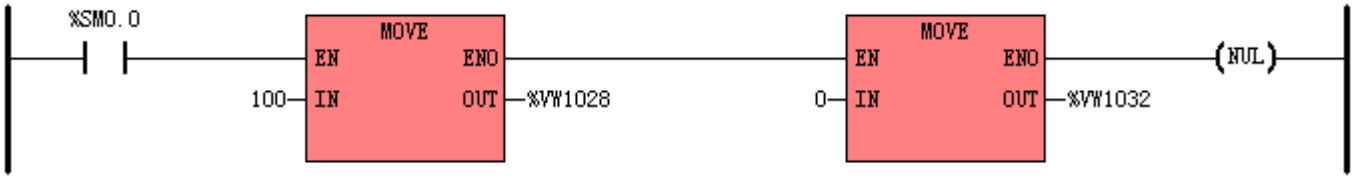
(* Network 0 *)
(* Установка PV и периода выборки *)
(* Период выборки может влиять на эффект управления, поэтому он должен быть установлен в соответствии с фактическими характеристиками контролируемого объекта. *)



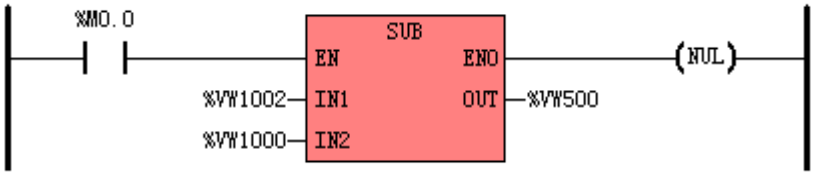
(* Network 1 *)
(* Установка верхнего и нижнего предельного значения PV *)



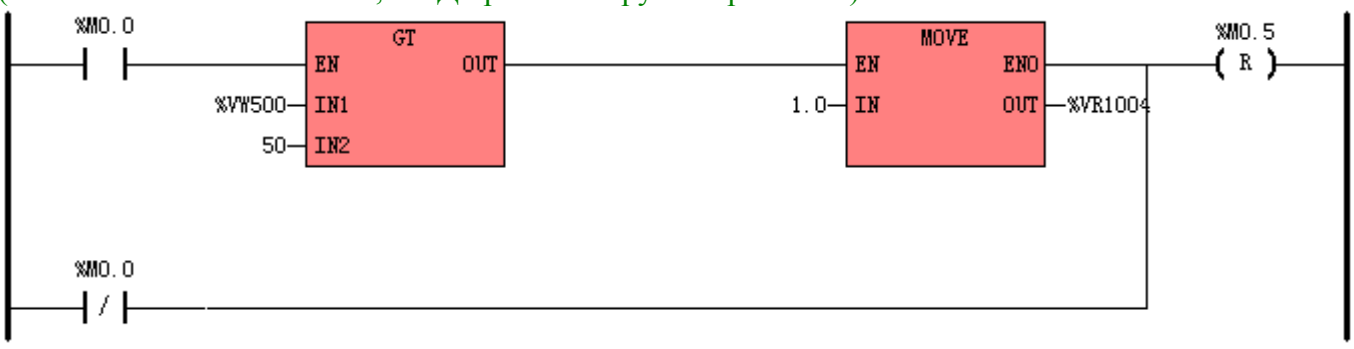
(* Network 2 *)
(* Установка верхнего и нижнего предельного значения XOUT_P *)



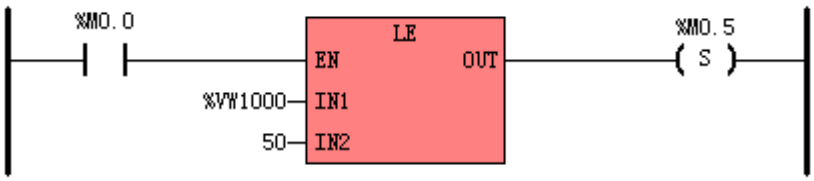
(* Network 3 *)
(* M0.0 - A кнопка в HMI для выбора ручного или автоматического управления системой. *)
(* Если пользователь выбирает автоматический способ, PLC вычисляет ошибку и определяет режим ПИД регулятора. *)



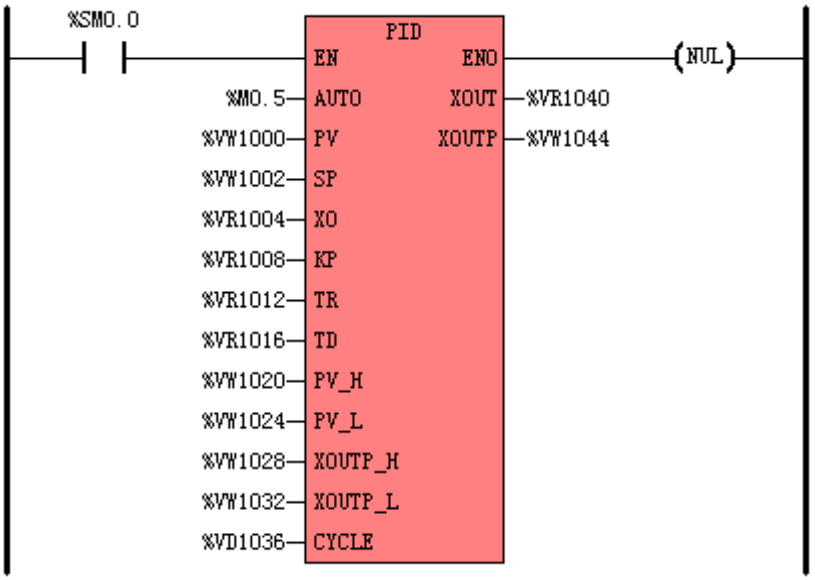
(* Network 4 *)
(* Если ошибка больше 5ºC, ПИД принимает ручной режим. *)



(* Network 5 *)

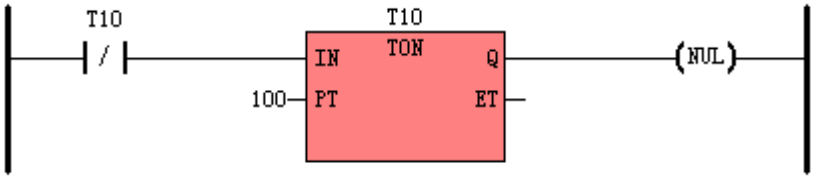


(* Network 6 *)



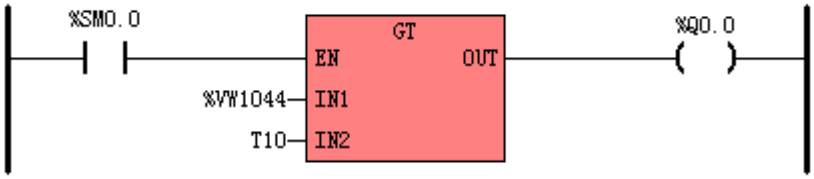
(* Network 7 *)

(* Использование таймера для генерации ШИМ выхода с периодом 1с. *)



(* Network 8 *)

(* Выход PID используется для регулировки ширины импульса ШИМ. *)



IL

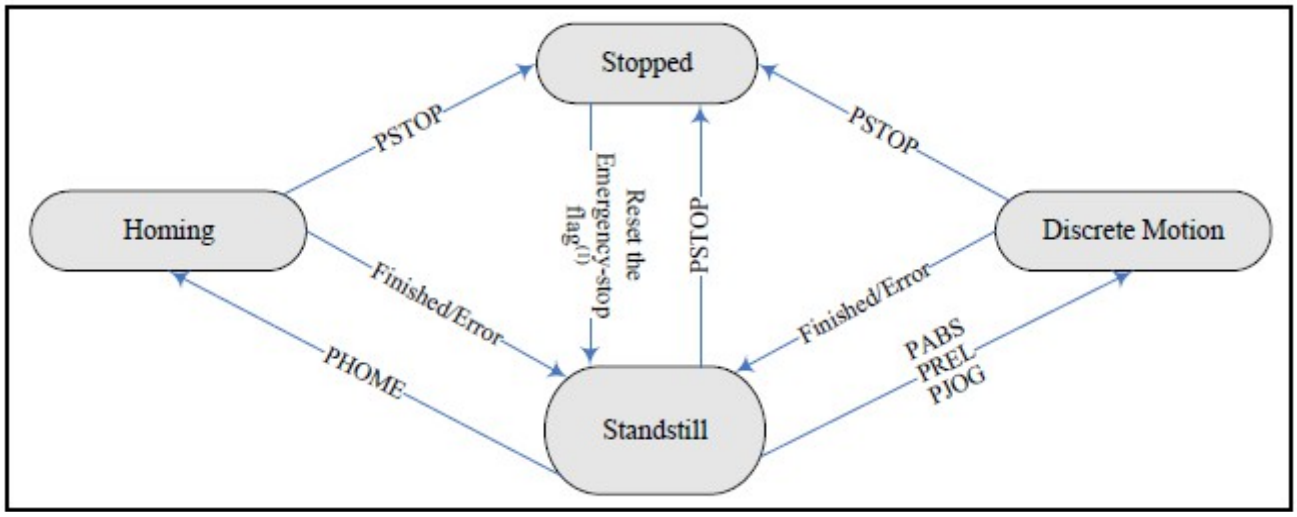
```
(* Network 0 *)
(* Установка PV и периода выборки *)
(* Период выборки может влиять на эффект управления, поэтому он должен быть установлен в
соответствии с фактическими характеристиками контролируемого объекта. *)
LD      %SM0.0
MOVE    %AIW0, %VW1000
MOVE    DI#1000, %VD1036
(* Network 1 *)
(* Установка верхнего и нижнего предельного значения PV *)
LD      %SM0.0
MOVE    1000, %VW1020
MOVE    0, %VW1024
(* Network 2 *)
(* Установка верхнего и нижнего предельного значения XOUT_P *)
MOVE    100, %VW1028
MOVE    0, %VW1032
(* Network 3 *)
(* M0.0 - A кнопка в HMI для выбора ручного или автоматического управления системой. *)
(* Если пользователь выбирает автоматический способ, PLC вычисляет ошибку и определяет режим
ПИД регулятора. *)
LD      %M0.0
MOVE    %VW1002, %VW500
SUB     %VW1000, %VW500
(* Network 4 *)
(* Если ошибка больше 5ºC, ПИД принимает ручной режим. *)
LD      %M0.0
GT      %VW500, 50
MOVE    1.0, %VR1004
ORN     %M0.0
R       %M0.5
(* Network 5 *)
LD      %M0.0
LE      %VW1000, 50
S       %M0.5
(* Network 6 *)
LD      %SM0.0
PID     %M0.5, %VW1000, %VW1002, %VR1004, %VR1008, %VR1012, %VR1016, %VW1020,
        %VW1024, %VW1028, %VW1032, %VD1036, %VR1040, %VW1044
(* Network 7 *)
(* Использование таймера для генерации ШИМ выхода с периодом 1с. *)
LDN     T10
TON     T10, 100
(* Network 8 *)
(* Выход PID используется для регулировки ширины импульса ШИМ. *)
LD      %SM0.0
GT      %VW1044, T10
ST      %Q0.0
```

6.16 Управление положением

Kinco-K5 предусматривает 2 высокоскоростных импульсных выходных канала: Q0.0 и Q0.1, они могут быть использованы для контроля положения для 2 осей. В главе 6.13.4 "Инструкция по Высокоскоростным импульсным выходам", использование ШИМ и обучение PLS описывается подробно.
Инструкция управления положением, описанная в этой главе, является еще одним применением функции высокоскоростного импульсного выхода. По сравнению с инструкцией PLS, инструкция контроля положения более удобна для управления положением. Точно так же, частота импульсного выхода находится в диапазоне от 20 кГц - 200 кГц.

6.16.1 Модель

Следующая схема ориентирована на одной оси, и определяет поведение оси на высоком уровне, когда команды управления положения активированы. Основное правило заключается в том, что команды положения всегда принимаются последовательно.
Ось всегда в одном определенном состоянии (см схему ниже). Любая команда изменения положения, которая изменяет состояние оси, в результате, способствует изменению текущего вычисленного положения.



★ Флажком аварийного останова являются SM201.7 / SM231.7. Он будет установлен в 1 автоматически во время выполнения команды PSTOP. Пожалуйста, обратитесь к подробному описанию в следующем разделе.

6.16.2 Корреляционные переменные

6.16.2.1 Выходной канал направления

Для команд управления позиционирования, Kinco-K5 определяет выходной канал направления для каждого импульсного высокоскоростного выхода, и бит управления в области SM для направления выхода. Пожалуйста, обратитесь к следующей таблице.

Высокоскоростной импульсный выход	**Q0.0**	**Q0.1**
Направление выхода	Q0.2	Q0.3
Бит управления направлением	SM201.3	SM231.3

Выходной канал направления используется для обеспечения сигнала направления, который контролирует

направление вращения электродвигателей: 0 означает, вращение вперёд, а 1 означает вращение в обратном направлении.
Бит управления направлением используется для отключения или включения соответствующих выходных каналов направления. Бит управления направлением имеет самый высокий приоритет. Если сигнал направления не требуется, то соответствующий выходной канал направления может быть использован в качестве обычного дискретного выхода.

6.16.2.2 Регистры состояния и регистры управления

Для команд управления положением, Kinco-K5 определяет управляющий байт для каждого выходного высокоскоростного канала для хранения своих настроек.
Регистр состояния также указывается для хранения текущего значения (число выходных импульсов, DINT). Текущее значение увеличивается при вращении вперед, и уменьшается при вращении в обратном направлении. Следующая таблица подробно описывает эти регистры.
Примечание: После окончания инструкции управления положением, текущее значение не сбрасывается автоматически, и вы можете его удалить в вашей программе.
Следующая таблица описывает управляющий байт и текущее значение.

Q0.0	**Q0.1**	Описание
SM201.7	SM231.7	Флажок аварийной остановки. Если этот бит равен 1, никакие инструкции контроля положения не могут быть выполнены. При выполнении инструкции PSTOP, этот бит устанавливается в 1 автоматически, и он должен быть сброшен вашей программой.
SM201.0~SM201.2	SM201.0~SM201.2	Зарезервировано
SM201.3	SM231.3	Бит управления направлением. 1 --- Отключение выходного канала направления. 0 --- Включение выходного канала направления.
SM201.0~SM201.2	SM201.0~SM201.2	Зарезервировано
SMD212	SMD242	Текущее значение

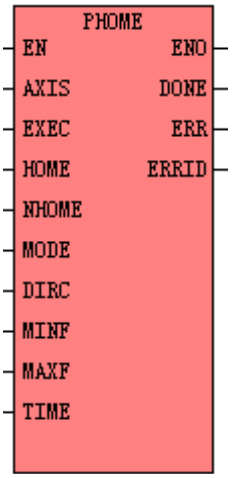
6.16.2.3 Идентификация ошибок

Во время выполнение команд управления положения, могут возникнуть не фатальные ошибки, тогда CPU будет генерировать коды ошибок, которые записываются в параметр ERRID. Следующая таблица описывает эти коды ошибок.

Код ошибки	**Описание**
0	Нет ошибок
1	Значение AXIS не равно 0 или 1.
2	Значение MINF больше, чем значение MAXF (200 кГц).
3	Значение MINF меньше допустимого низкой частоты (20 Гц).
4	Значение времени (ускорения / торможения) не совпадает со значением MINF и MAXF.
5	Значение MINF больше, чем MAXF

6.16.3 PHOME (самонаведение)

★ Описание

	Название	Использование	Группа	
LD	PHOME			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PHOME	PHOME *AXIS, EXEC, HOME, NHOME, MODE, DIRC, MINF, MAXF, TIME, DONE, ERR, ERRID*	U	

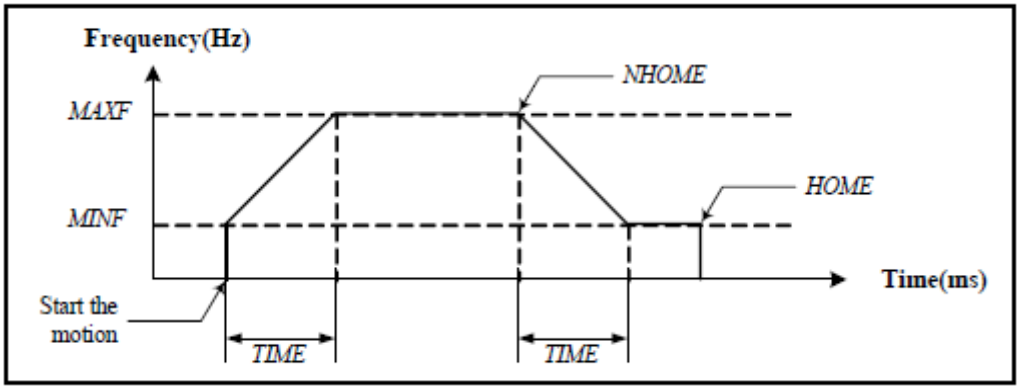
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 or 1)
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
HOME	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
NHOME	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
MODE	Вход	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Константа
DIRC	Вход	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Константа
MINF	Вход	WORD	I, Q, M, V, L, SM, Константа
MAXF	Вход	DWORD	I, Q, M, V, L, SM, Константа
TIME	Вход	WORD	I, Q, M, V, L, SM, Константа
DONE	Выход	BOOL	Q, M, V, L, SM
ERR	Выход	BOOL	Q, M, V, L, SM
ERRID	Выход	BYTE	Q, M, V, L, SM

Следующая таблица подробно описывает все операнды.

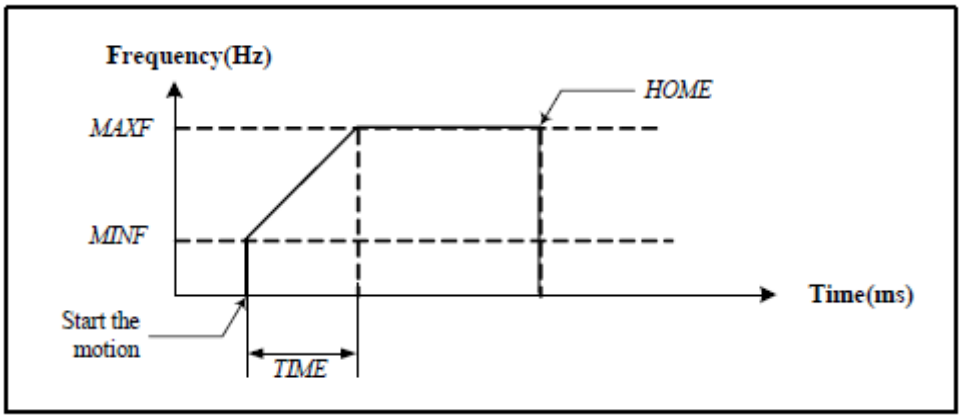
Операнд	Описание
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
EXEC	Если EN равен 1, EXEC начинает движение по переднему фронту.
HOME	Сигнал HOME от датчика HOME
NHOME	Сигнал NEAR HOME от датчика NEAR HOME
MODE	Задает режим самонаведения: 0 означает, что используются сигналы HOME сигнал и NHOME; 1 означает, что используется только HOME сигнал.
DIRC	Указывает направление вращения электродвигателя: 0 означает вращение вперед; 1 означает вращение в обратном направлении. Пожалуйста, обратитесь к главе 6.16.2.1 "Выходной канал направления" для получения дополнительной информации.
MINF	Определяет начальную скорость (т.е. начальную частоту) выходных импульсов. Единица измерения: Гц. Примечание: значение MINF должно быть равна или меньше, чем 2 кГц.
MAXF	Задает максимальную скорость (т.е. наибольшая частота) выхода импульсов. Единица измерения: Гц. Доступный диапазон MAXF является 20 Гц ~ 20 кГц. MAXF должны быть больше или равны MINF.
TIME	Определяет время разгона / торможения. Единица измерения: мс. В инструкции управления положением, время ускорения является таким же, как время торможения. Время разгона это время для скорости разгона из состояния MINF в MAXF. Время торможения это время для скорости замедления из состояния MAXF в MINF.
DONE	Указывает, что инструкция успешно завершена. 0 = не завершена; 1 = завершена.
ERR	Указывает, что произошла ошибка во время выполнения. 0 = нет ошибки; 1 = ошибка.
ERRID	Идентификация ошибок.

Эта инструкция контролирует выполнение AXIS используя комбинацию сигналов HOME и NHOME. MODE определяет режим самонаведения. При выполнении движения, если DIRC установлен равным 0 (вращение вперед), текущее значение (SMD212 / SMD242) увеличивается; если DIRC установлен равным 1 (вращение в обратном направлении), текущее значение (SMD212 / SMD242) уменьшается.

★ Если MODE равно 0 (с использованием сигналов HOME и NHOME), инструкция PHOME будет контролировать AXIS, замедляться, как только NHOME становится 1, и остановится, как только HOME становится 1.
Временная диаграмма выглядит следующим образом:



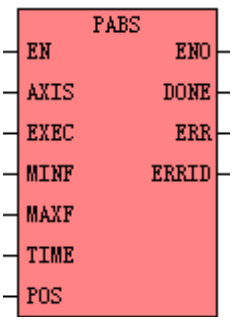
★ Если MODE равно 1 (с использованием только сигнала HOME), инструкция PHOME будет контролировать AXIS, чтобы остановить, как только HOME становится равным 1.
Временная диаграмма выглядит следующим образом:



■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет CR.

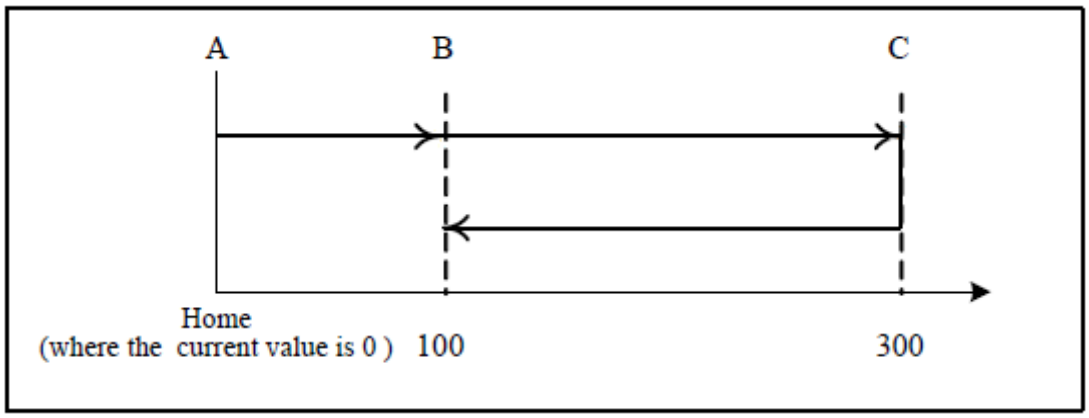
6.16.4 PABS (абсолютное перемещение)

★ Описание

	Название	Использование	Группа	
LD	PABS			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PABS	PABS *AXIS, EXEC, MINF, MAXF, TIME, POS, DONE, ERR, ERRID*	U	

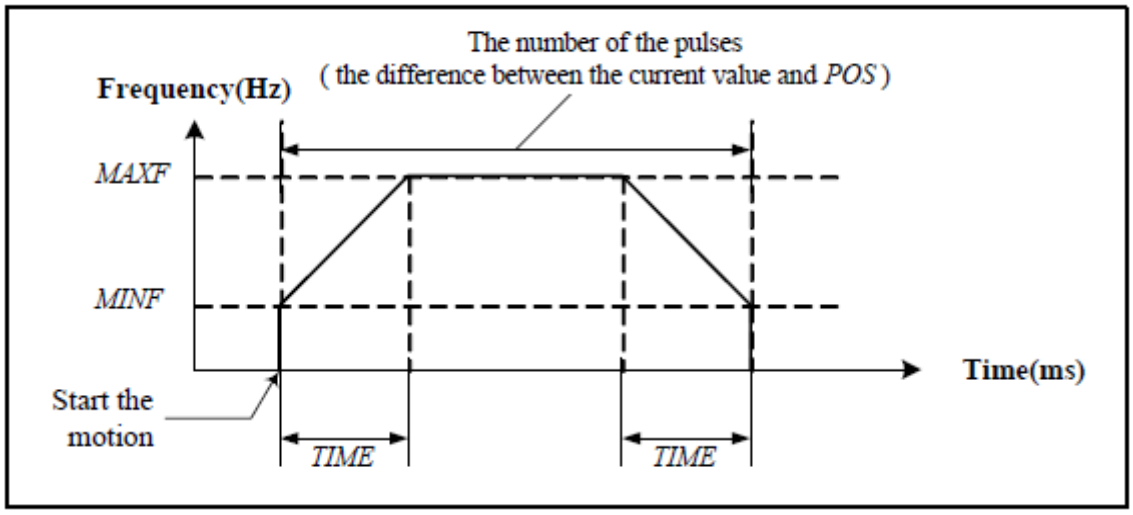
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 or 1)
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Вход	WORD	I, Q, M, V, L, SM, Константа
MAXF	Вход	DWORD	I, Q, M, V, L, SM, Константа
TIME	Вход	WORD	I, Q, M, V, L, SM, Константа
POS	Вход	DINT	I, Q, M, V, L, SM, HC, Константа
DONE	Выход	BOOL	Q, M, V, L, SM
ERR	Выход	BOOL	Q, M, V, L, SM
ERRID	Выход	BYTE	Q, M, V, L, SM

Следующая таблица подробно описывает все операнды.

Операнд	Описание
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
EXEC	Если EN равен 1, EXEC начинает абсолютное движение по переднему фронту.
MINF	Определяет начальную скорость (т.е. начальную частоту) выходных импульсов. Единица измерения: Гц. Примечание: значение MINF должно быть равна или меньше, чем 2 кГц.
MAXF	Задает максимальную скорость (т.е. наибольшая частота) выхода импульсов. Единица измерения: Гц. Доступный диапазон MAXF является 20 Гц ~ 20 кГц. MAXF должны быть больше или равны MINF.
TIME	Определяет время разгона / торможения. Единица измерения: мс. В инструкции управления положением, время ускорения является таким же, как время торможения. Время разгона это время для скорости разгона из состояния MINF в MAXF. Время торможения это время для скорости замедления из состояния MAXF в MINF.
POS	Указывает конечное значение. Представляется количеством импульсов между исходным положением, в котором текущее значение равно 0, и конечным положением. Как показано на следующем рисунке, если объект перемещается из пункта A в пункт Б, POS должен быть установлен как "100"; Если он перемещается от B к C, POS должен быть установлен в '300'; Если он перемещается от C к B, POS должен быть установлен в '100'. 

DONE	Указывает, что инструкция успешно завершена. 0 = не завершена; 1 = завершена.
ERR	Указывает, что произошла ошибка во время выполнения. 0 = нет ошибки; 1 = ошибка.
ERRID	Идентификация ошибок.

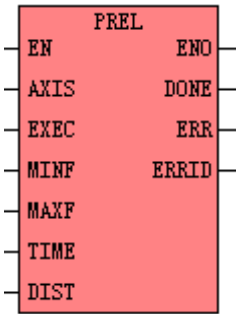
Эта команда управляет AXIS движением в указанную абсолютную позицию (POS), и это обеспечивает вывод последовательности импульсов, до тех пор, пока текущее значение не станет равно заданному значению.
Если бит контроля направления (SM201.3 / SM231.3) установлен в 0, инструкция PABS будет генерировать направление выходного сигнала соответствующим направлению выходного канала (Q0.2 / Q0.3): Если заданное значение больше чем текущее значение, он генерирует направление выхода вращения вперед, тогда текущее значение (SMD212 / SMD242) увеличивается; Если заданное значение меньше, чем текущее значение, он генерирует направление выхода вращения в обратном направлении, и затем текущее значение (SMD212 / SMD242) уменьшается.
Временная диаграмма выглядит следующим образом:



■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет CR.

6.16.5 PREL (относительное перемещение)

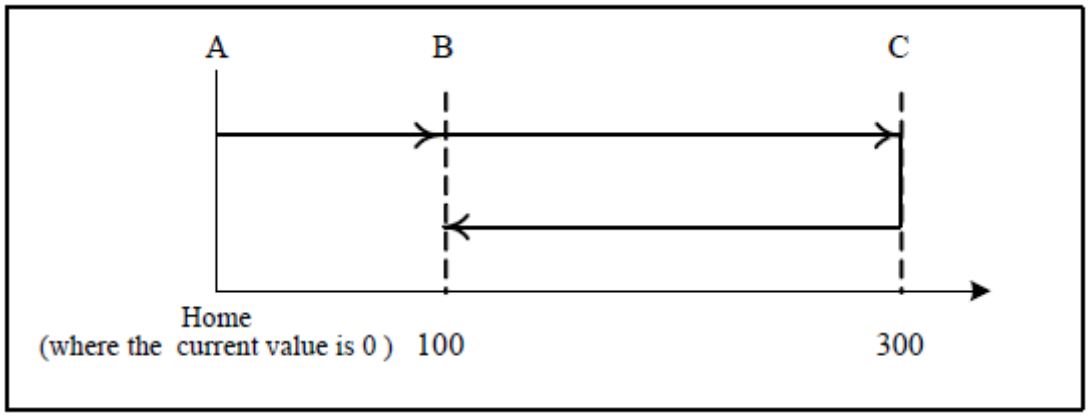
★ Описание

	Название	Использование	Группа	
LD	PREL			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PREL	PREL *AXIS, EXEC, MINF, MAXF, TIME, DIST, DONE, ERR, ERRID*	U	

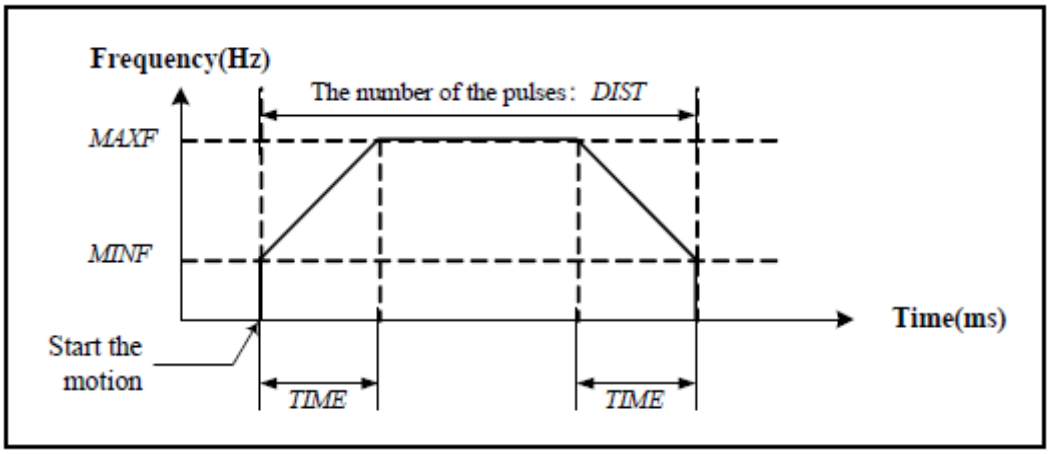
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 or 1)
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
MINF	Вход	WORD	I, Q, M, V, L, SM, Константа
MAXF	Вход	DWORD	I, Q, M, V, L, SM, Константа
TIME	Вход	WORD	I, Q, M, V, L, SM, Константа
DIST	Вход	DINT	I, Q, M, V, L, SM, HC, Константа
DONE	Выход	BOOL	Q, M, V, L, SM
ERR	Выход	BOOL	Q, M, V, L, SM
ERRID	Выход	BYTE	Q, M, V, L, SM

Следующая таблица подробно описывает все операнды.

Операнд	Описание
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
EXEC	Если EN равен 1, EXEC начинает относительное движение по переднему фронту.
MINF	Определяет начальную скорость (т.е. начальную частоту) выходных импульсов. Единица измерения: Гц. Примечание: значение MINF должно быть равна или меньше, чем 2 кГц.
MAXF	Задает максимальную скорость (т.е. наибольшая частота) выхода импульсов. Единица измерения: Гц. Доступный диапазон MAXF является 20 Гц ~ 20 кГц. MAXF должны быть больше или равны MINF.
TIME	Определяет время разгона / торможения. Единица измерения: мс. В инструкции управления положением, время ускорения является таким же, как время торможения. Время разгона это время для скорости разгона из состояния MINF в MAXF. Время торможения это время для скорости замедления из состояния MAXF в MINF.

DIST	Задает расстояние. Представлено количеством импульсов между текущей позицией и заданным положением. Как показано на следующем рисунке, если объект перемещается из пункта A в пункт Б, DIST должен быть установлен как "100"; Если он перемещается от B к C, DIST должен быть установлен в '200'; Если он перемещается от C к B, DIST должен быть установлен в "-200". 
DONE	Указывает, что инструкция успешно завершена. 0 = не завершена; 1 = завершена.
ERR	Указывает, что произошла ошибка во время выполнения. 0 = нет ошибки; 1 = ошибка.
ERRID	Идентификация ошибок.

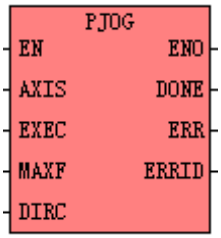
Эта инструкция контролирует AXIS выполнение движение на заданное расстояние (DIST) по отношению к текущему значению на момент выполнения.
Если бит контроля направления (SM201.3 / SM231.3) установлен в 0, инструкция PREL будет генерировать выходной сигнал направления выходного канала с соответствующим направлением (Q0.2 / Q0.3): Если DIST является положительным, он генерирует направление выхода вращения вперед, тогда текущее значение (SMD212 / SMD242) увеличивается; Если DIST отрицательный, он генерирует направление выхода вращения в обратном направлении, и затем текущее значение (SMD212 / SMD242) уменьшается.
Временная диаграмма выглядит следующим образом:



■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет CR.

6.16.6 PJOG (регулятор Jog)

★ Описание

	Название	Использование	Группа	
LD	PJOG			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PJOG	PJOG *AXIS, EXEC, MINF, DIRC, DONE, ERR, ERRID*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 or 1)
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
MAXF	Вход	DWORD	I, Q, M, V, L, SM, Константа
DIRC	Вход	INT	I, Q, M, V, L, SM, AI, AQ, T, C, Константа
DONE	Выход	BOOL	Q, M, V, L, SM
ERR	Выход	BOOL	Q, M, V, L, SM
ERRID	Выход	BYTE	Q, M, V, L, SM

Следующая таблица подробно описывает все операнды.

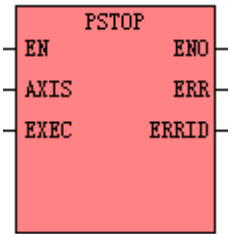
Операнд	Описание
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
EXEC	Если EN равен 1, EXEC начинает движение jog по переднему фронту.
MAXF	Определяет скорость (т.е. начальную частоту) выходных импульсов. Единица измерения: Гц.
DIRC	Указывает направление вращения электродвигателя: 0 означает вращение вперед, и 1 означает вращение в обратном направлении.
DONE	Указывает, что инструкция успешно завершена. 0 = не завершена; 1 = завершена.
ERR	Указывает, что произошла ошибка во время выполнения. 0 = нет ошибки; 1 = ошибка.
ERRID	Идентификация ошибок.

Эта инструкция контролирует AXIS выполнение шагового движения: Генерация длительной последовательности импульсов на выходе, частота которого MINF.
Если бит контроля направления (SM201.3 / SM231.3) установлен в 0, инструкция PJOG будет генерировать направление выходного сигнала соответствующее направлению выходного канала (Q0.2 / Q0.3): если DIRC равен 0 (вращение вперед), текущее значение (SMD212 / SMD242) увеличивается; если DIRC равен 1 (вращение в обратном направлении), текущее значение (SMD212 / SMD242) уменьшается.

■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет CR.

6.16.7 PSTOP (Стоп)

★ Описание

	Название	Использование	Группа	
LD	PSTOP			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PSTOP	PSTOP *AXIS, EXEC, ERR, ERRID*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 or 1)
EXEC	Вход	BOOL	I, Q, V, M, L, SM, RS, SR
ERR	Выход	BOOL	Q, M, V, L, SM
ERRID	Выход	BYTE	Q, M, V, L, SM

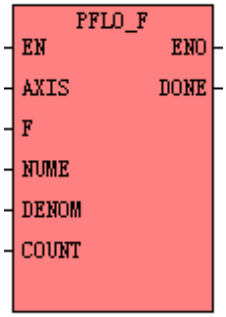
Следующая таблица подробно описывает все операнды.

Операнд	Описание
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
EXEC	Если EN равен 1, EXEC останавливает текущее движение по переднему фронту.
ERR	Указывает, что произошла ошибка во время выполнения. 0 = нет ошибки; 1 = ошибка.
ERRID	Идентификация ошибок.

Эта инструкция останавливает текущую движение AXIS. При этом, флаг аварийной остановки (SM201.7 / SM231.7) установлен в 1, и никакие инструкции управления положением не могут быть выполнены, пока этот флаг не будет сброшен вашей программе.
■ LD
Если EN равен 1, то эта инструкция выполняется.
■ IL
Если CR равен 1, эта инструкция выполняется, и не влияет CR.

6.16.8 PFLO_F

★ Описание

	Название	Использование	Группа	
LD	PFLO_F			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PFLO_F	PFLO_F *AXIS, F, NUME, DENOM, COUNT, DONE*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 или 1)
F	Вход	DINT	L, M, V, Константа
NUME	Вход	INT	L, M, V, Константа
DENOM	Вход	INT	L, M, V, Константа
DONE	Выход	BOOL	Q, M, V, L
COUNT	Вход	DWORD	L, M, V, Константа

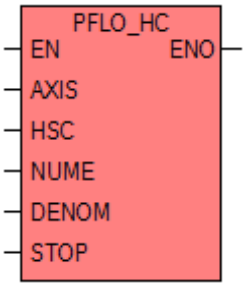
Следующая таблица подробно описывает все операнды.

Операнд	Описание
EN	Включение. Если EN равен 1, то будет выполняться последовательность импульсов на выходе. В противном случае останов.
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
F	Скорость входа. Единица измерения: Гц
NUME	Числитель электронного редуктора скорости импульсного выхода.
DENOM	Знаменатель электронного редуктора скорости импульсного выхода.
DONE	Готовность. Если импульсный выход существует, то он будет 0, в противном случае будет 1.
COUNT	Количество импульсов.

Примечание: F, NUME, DENOM и COUNT должно быть либо все константы, либо все переменные.
Эта инструкция контролирует AXIS генерирует на выходе серию импульсов (PTO), а в общей сложности генерирует импульсы COUNT. Частота PTO равна F × (NUME / DENOM), и его абсолютное значение должно быть больше или равно 30 Гц, в противном случае K5 будет останавливать PTO до тех пор, пока его частота снова превысит 30 Гц.
DONE указывает на завершение серии импульсов на выходе, он устанавливается равным 0, как только начинается PFLO_F, и устанавливается равным 1, как только серия импульсов завершена.
Если бит контроля направления (SM201.3 / SM231.3) установлен в 0, PFLO_F будет генерировать

направление выходного сигнала соответствующее направлению выходного канала (Q0.2 / Q0.3) в соответствии со знаком F: Если F положительно, PFLO_F генерирует направление выхода вращения вперед, и текущее значение (SMD212 / SMD242) увеличивается; Если F является отрицательным, он генерирует направление выхода вращения в обратном направлении, и текущее значение (SMD212 / SMD242) уменьшается.
Если EN (или CR) является 1, эта инструкция выполняется, и запускает PTO, если EN (или CR) изменится в 0, то PTO останавливается, и если EN (или CR) вновь изменится в 1, то PTO продолжит работать и сгенерирует оставшиеся импульсы.
■ LD
Если EN равен 1, то эта инструкция выполняется, в противном случае она останавливается.
■ IL
Если CR равен 1, эта инструкция выполняется, в противном случае она останавливается и не влияет на CR.

6.16.9 PFLO_HC

	Название	Использование	Группа	
LD	PFLO_HC			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	PFLO_HC	PFLO_HC *AXIS, HSC, NUME, DENOM, STOP*	U	

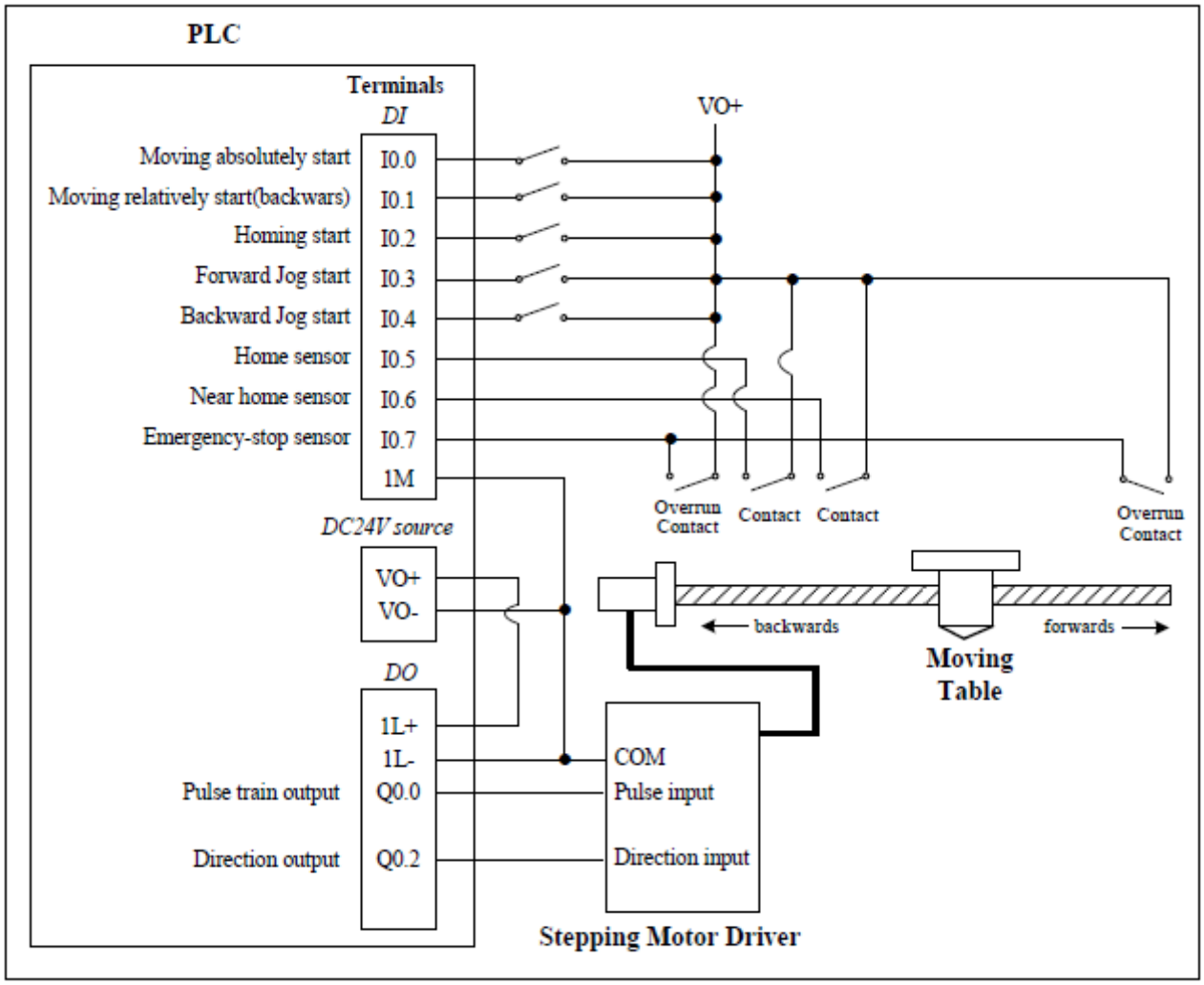
Операнд	Вход / Выход	Тип данных	Допустимые области памяти
AXIS	Вход	INT	Константа (0 или 1)
HSC	Вход	INT	Константа (0 или 1)
NUME	Вход	INT	L, M, V, Константа
DENOM	Вход	INT	L, M, V, Константа
STOP	Вход	WORD	I, Q, M, V, L, SM, Константа

Операнд	Описание
EN	Включение. Если EN равен 1, то будет выполняться последовательность импульсов на выходе. В противном случае останов.
AXIS	Высокоскоростной выходной канал, 0 означает Q0.0; 1 означает Q0.1.
HSC	Номер высокоскоростного счётчика. Поддерживает только 0 или 1.
NUME	Числитель электронного редуктора скорости импульсного выхода.
DENOM	Знаменатель электронного редуктора скорости импульсного выхода.
STOP	Время останова, единица измерения: мс

Эта инструкция генерирует на выходе *AXIS* импульсы в соответствии с импульсным входом HSC.

6.16.10 Примеры

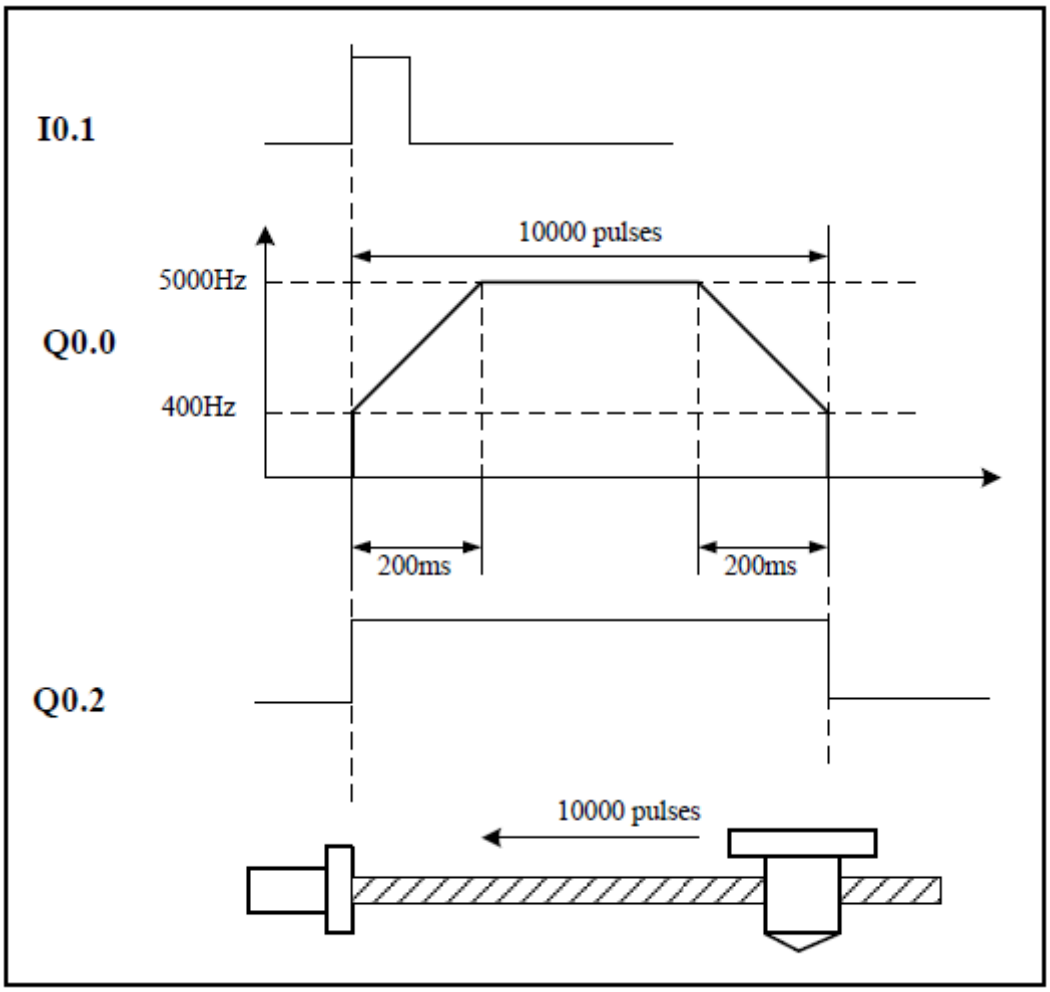
★ Подключение

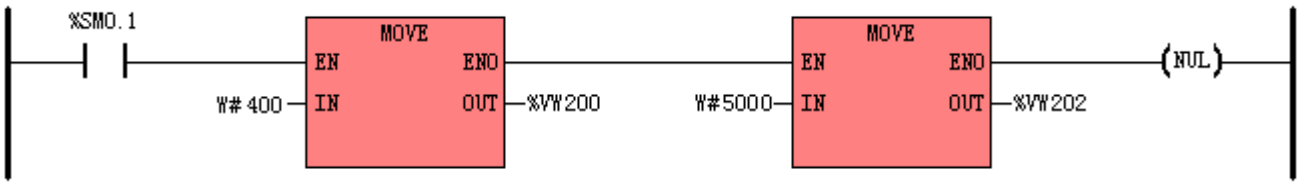
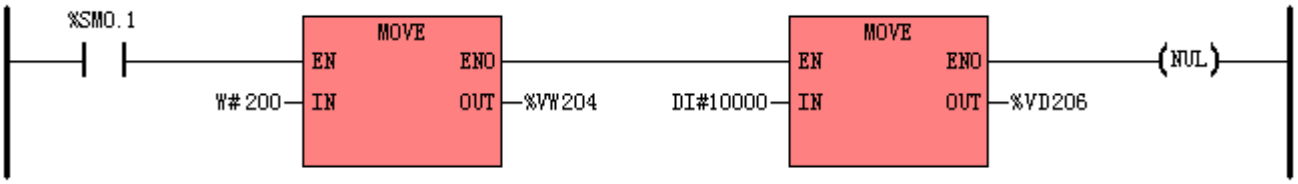


Данная система принимается в качестве примера, как использовать инструкции PREL, PABS, PHOME, PJOG и PSTOP.

★ Перемещение относительно

I0.1 используется для начала движения относительно (в обратном направлении).

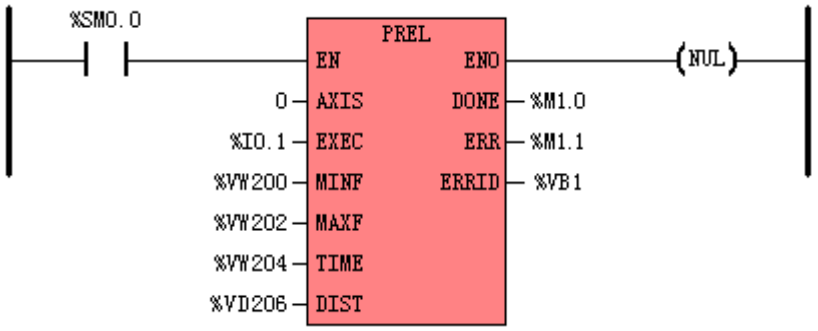
Временная диаграмма


	Использование
LD	(* Network 0 *) (* Установить исходную частоту и максимальную частоту *)  (* Network 1 *) (* Установить время ускорения / торможения и расстояние *) 

LD

(* Network 2 *)
(* Сброс флага аварийного останова *)

(* Network 3 *)
(* Вызов инструкции PREL *)

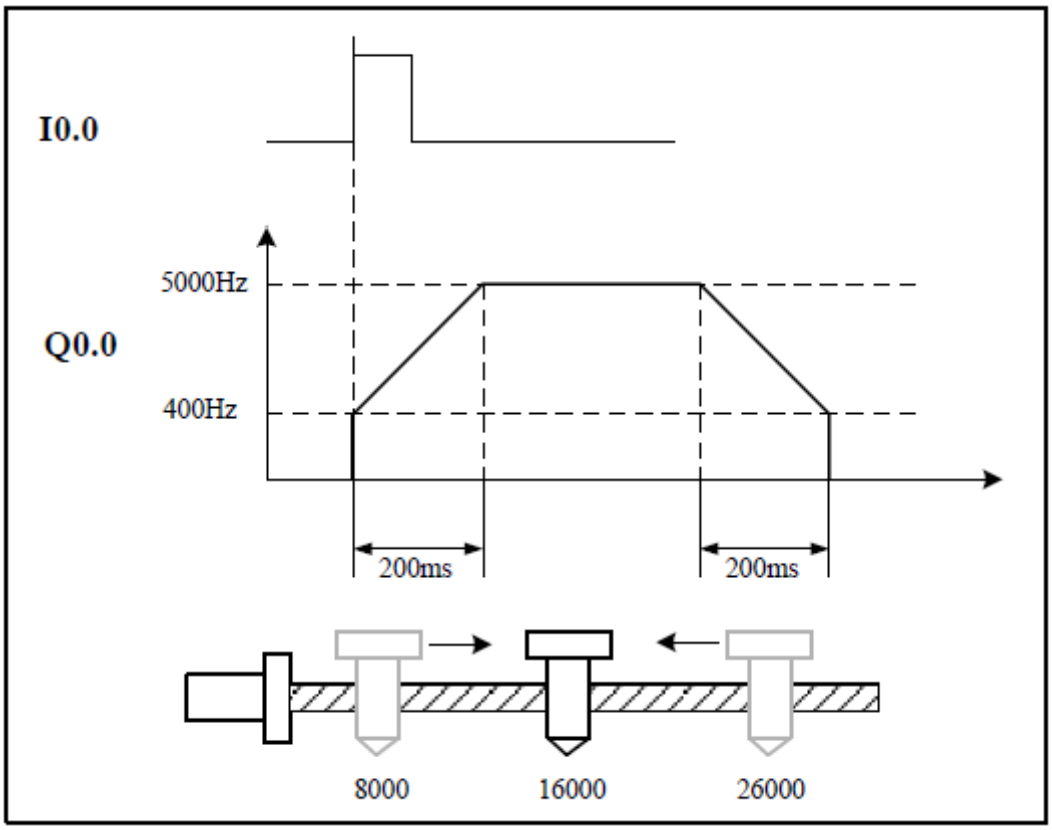


IL

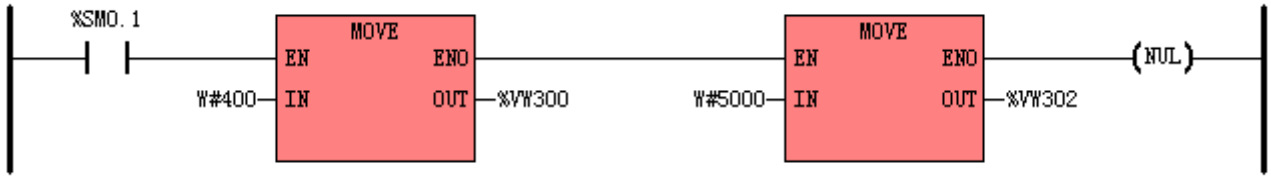
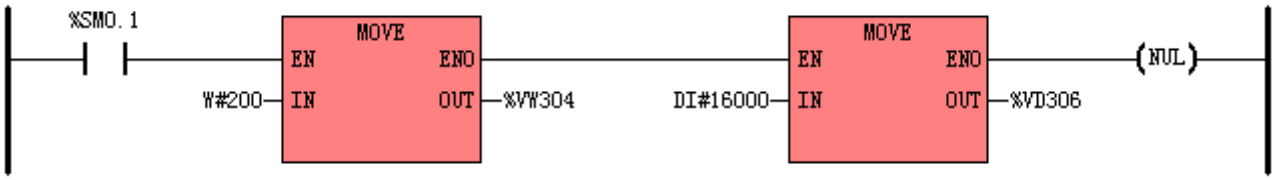
```
(* Network 0 *)
(* Установить исходную частоту и максимальную частоту *)
LD          %SM0.1
MOVE        W#400, %VW200
MOVE        W#5000, %VW202
(* Network 1 *)
(* Установить время ускорения / торможения и расстояние *)
LD          %SM0.1
MOVE        W#200, %VW204
MOVE        DI#-10000, %VD206
(* Network 2 *)
(* Сброс флага аварийного останова *)
LD          %I0.1
R           %SM201.7
(* Network 3 *)
(* Вызов инструкции PREL *)
LD          %SM0.0
PREL        0, %I0.1, %VW200, %VW202, %VW204, %VD206, %M1.0, %M1.1, %VB1
```

★ Перемещение абсолютно

I0.0 используется для запуска движения абсолютно.

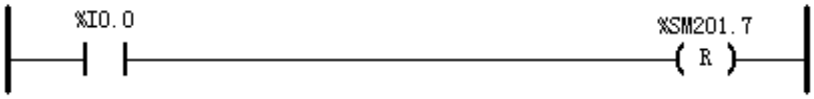
Временная диаграмма


Требуется переместить стол в положение "16000".

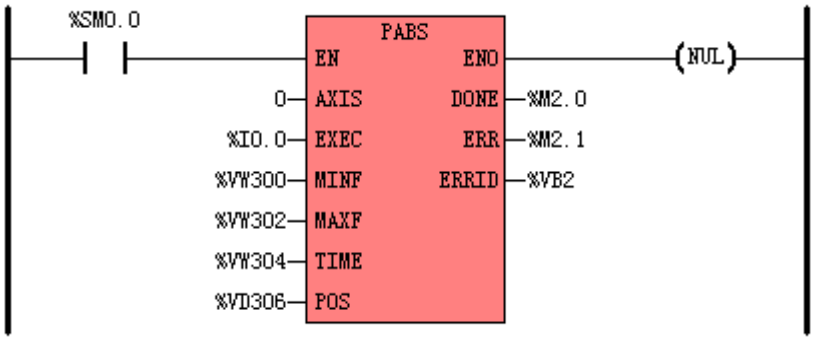
	Использование
LD	(* Network 0 *) (* Установить исходную частоту и максимальную частоту *)  (* Network 1 *) (* Установить время ускорения / торможения и расстояние *) 

LD

(* Network 2 *)
(* Сброс флага аварийного останова *)



(* Network 3 *)
(* Вызов инструкции PABS *)



IL

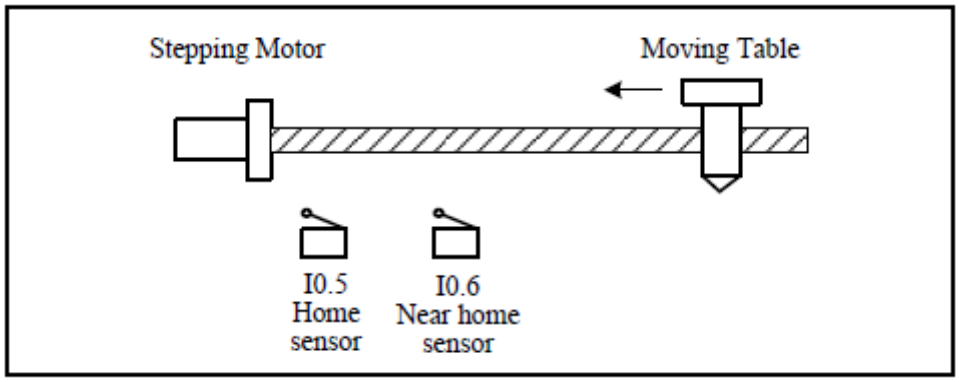
```
(* Network 0 *)
(* Установить исходную частоту и максимальную частоту *)
LD          %SM0.1
MOVE        W#400, %VW300
MOVE        W#5000, %VW302
(* Network 1 *)
(* Установить время ускорения / торможения и расстояние *)
LD          %SM0.1
MOVE        W#200, %VW304
MOVE        DI#16000, %VD306
(* Network 2 *)
(* Сброс флага аварийного останова *)
LD          %I0.0
R           %SM201.7
(* Network 3 *)
(* Вызов инструкции PABS *)
LD          %SM0.0
PABS        0, %I0.0, %VW300, %VW302, %VW304, %VD306, %M2.0, %M2.1, %VB2
```

★ Исходное положение

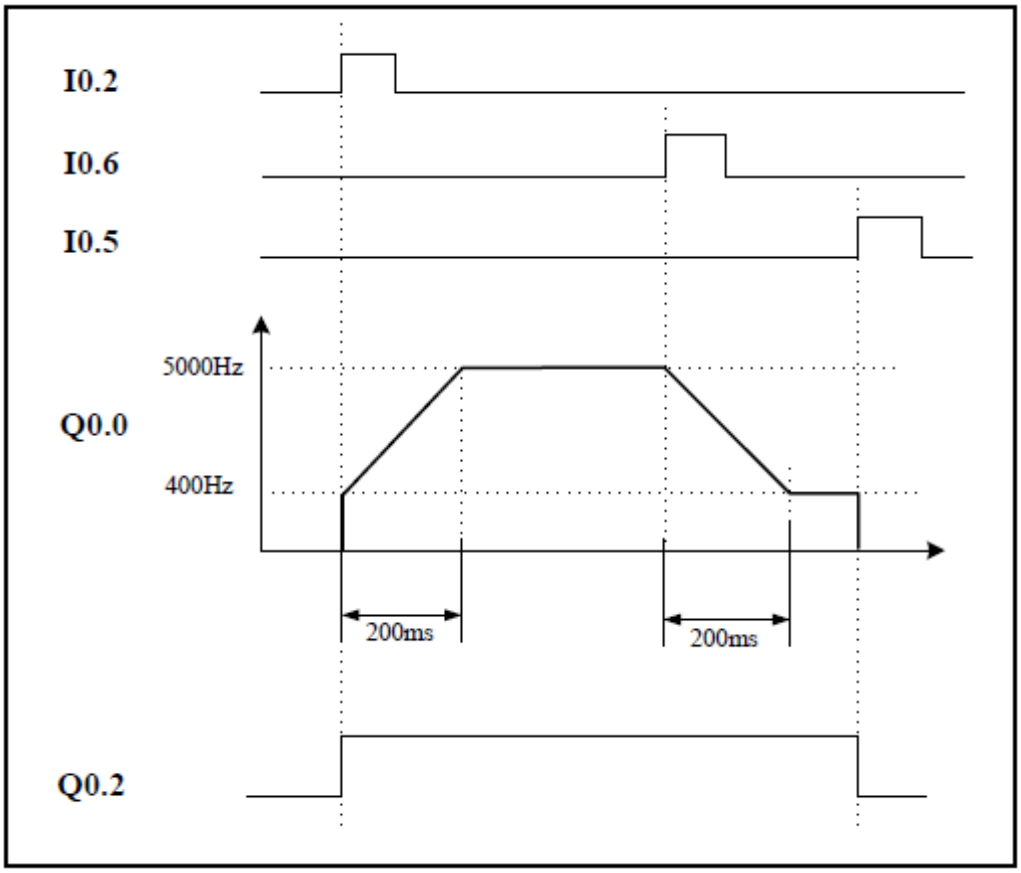
I0.2 используется для запуска возврата в исходное положение.

Временная диаграмма

Предположим, что перемещение в следующем исходном состоянии:



Во время движения Q0.2 равно 1, т.к. движение осуществляется в обратном направлении.



IL

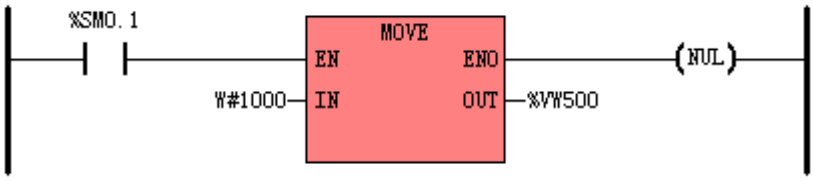
```
(* Network 0 *)
(* Использовать входы HOME и NHOME ; двигаться в обратном направлении *)
LD          %SM0.1
MOVE        0, %VW396
MOVE        1, %VW398
(* Network 1 *)
(* Установить начальную частоту, максимальную частоту и время разгона / торможения *)
LD          %SM0.1
MOVE        W#400, %VW400
MOVE        W#5000, %VW402
MOVE        W#200, %VW404
(* Network 2 *)
(* Сброс флага аварийного останова *)
LD          %I0.2
R           %SM201.7
(* Network 3 *)
LD          %SM0.0
PHOME       0, %I0.2, %I0.5, %I0.6, %VW396, %VW398, %VW400, %VW402, %VW404, %M3.0,
            %M3.1, %VB3
```

★ Толчковый

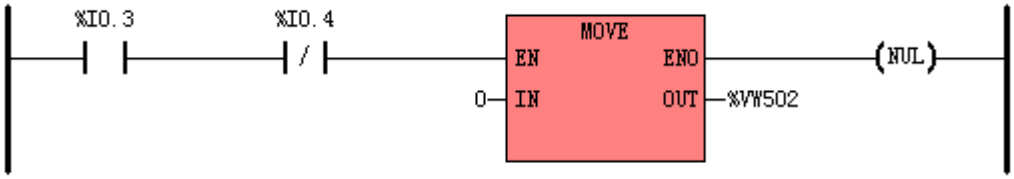
I0.3 используется для запуска вперед пробежку. I0.4 используется для запуска назад пробежку.
Если I0.3 и I0.4 являются все 1, то последнее направление следования.

LD

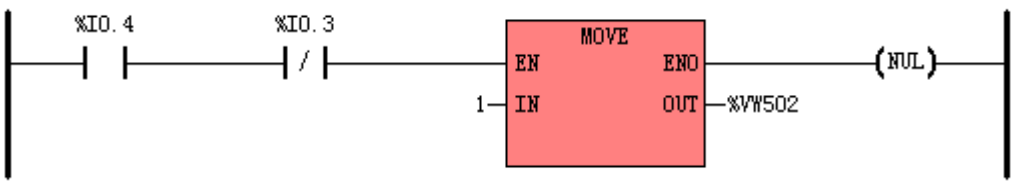
```
(* Network 0 *)
(* Установите частоту PTO *)
```




```
(* Network 1 *)
(* Выберите направление *)
```



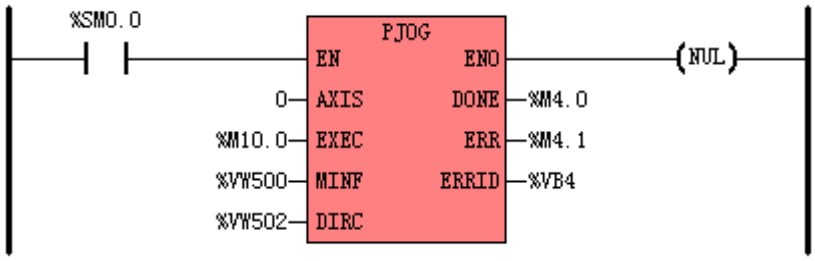

```
(* Network 2 *)
```



LD

(* Network 3 *)
(* Jog *)

(* Network 4 *)

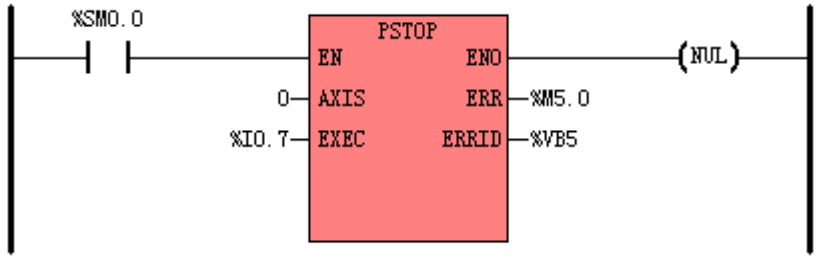


IL

```
(* Network 0 *)
(* Установите частоту PTO *)
LD          %SM0.1
MOVE        W#1000, %VW500
(* Network 1 *)
(* Выберите направление *)
LD          %I0.3
ANDN        %I0.4
MOVE        0, %VW502
(* Network 2 *)
LD          %I0.4
ANDN        %I0.3
MOVE        1, %VW502
(* Network 3 *)
(*Jog*)
LD          %I0.3
OR          %I0.4
ST          %M10.0
R           %SM201.7
(* Network 4 *)
LD          %SM0.0
PJOG        0, %M10.0, %VW500, %VW502, %M4.0, %M4.1, %VB4
```

★ **Стоп**

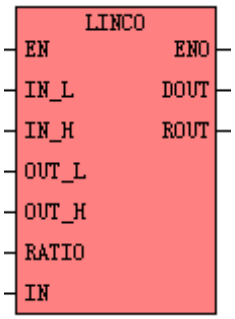
Существует два концевика на двух концах подающего винта, и они соединены параллельно, чтобы I0.7 работал как сигнал аварийного останова.

LD	(* Network 0 *) 
IL	(* Network 0 *) LD %SM0.0 PSTOP 0, %I0.7, %M5.0, %VB5

6.17 Дополнительные команды

6.17.1 LINCO (линейный расчет)

★ Описание

	Название	Использование	Группа	
LD	LINCO			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	LINCO	LINCO *IN_L, IN_H, OUT_L, OUT_H, RATIO, IN, DOUT, ROUT*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN_L	Вход	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Константа
IN_H	Вход	INT	I, Q, V, M, L, SM, T, C, AI, AQ, Константа
OUT_L	Вход	REAL	V, L, Константа
OUT_H	Вход	REAL	V, L, Константа
RATIO	Вход	REAL	Константа
IN	Вход	INT	I, Q, V, M, L, SM, T, C, AI, AQ
DOUT	Выход	DINT	Q, M, V, L, SM
ROUT	Вход	REAL	V, L

Примечание: IN_L, IN_H, OUT_L и OUT_H должны быть либо все константы, либо все переменные.
Эта инструкция рассчитывает вход IN в соответствии с указанной линейной зависимостью, и умножает результат с коэффициентом RATIO, а затем присваивает новый результат в ROUT. Кроме того, усеченное значение DINT из ROUT (путем отбрасывания дробной части) передаёт в DOUT. Линейная зависимость определяется в соответствии с методом "2-х точек", а двумя точками являются (IN_L, OUT_L) и (IN_H, OUT_H).
Функция LINCO может быть описана с помощью следующей формулы:

$$ROUT = RATIO * (k*IN + b)$$

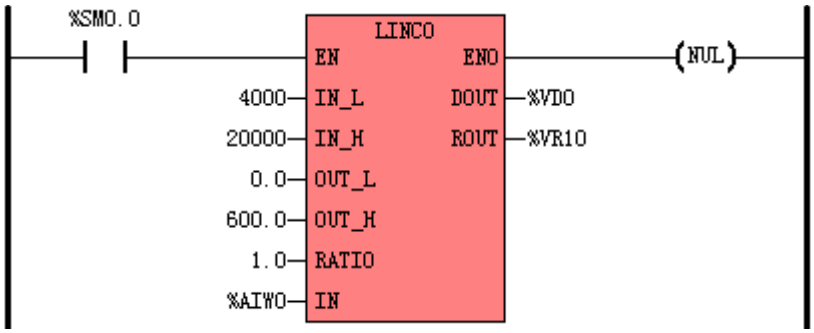
$$DOUT = TRUNC(ROUT)$$

где, $k = (OUT\_H — OUT\_L) / (IN\_H — IN\_L)$

$b = OUT\_L - k \cdot IN\_L$.

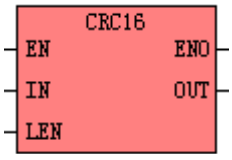
■ LD
Если EN равен 1, то эта инструкция выполняется, в противном случае она останавливается.
■ IL
Если CR равен 1, эта инструкция выполняется, в противном случае она останавливается и не влияет на CR.

★ Примеры
Предположим, что диапазон измерения датчика температуры 0 ~ 600 ° C, а на выходе диапазон 4 ~ 20 мА. Выходной сигнал датчика подключен к каналу AIW0 в Kinco-K5. Теперь необходимо рассчитать фактическое значение температуры.

LD	(* Network 0 *) 
IL	(* Network 0 *) LD %SM0.0 LINCO 4000, 20000, 0.0, 600.0, 1.0, %AIW0, %VD0, %VR10

6.17.2 CRC16 (16-Bit CRC)

★ Описание

	Название	Использование	Группа	
LD	CRC16			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	CRC16	CRC16 *IN, OUT, LEN*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN	Вход	BYTE	I, Q, M, V, L, SM
LEN	Вход	BYTE	I, Q, M, V, L, SM, Константа
OUT	Выход	BYTE	Q, M, V, L, SM

Эта инструкция вычисляет 16-битную CRC (Циклический Избыточный Код) для числа LEN последовательных переменных, начиная с IN, и помещает результат в 2 непрерывных байта переменных, начиная с OUT. Здесь OUT является старшим байтом CRC, а последующие байт переменной после OUT является младшим байтом CRC.

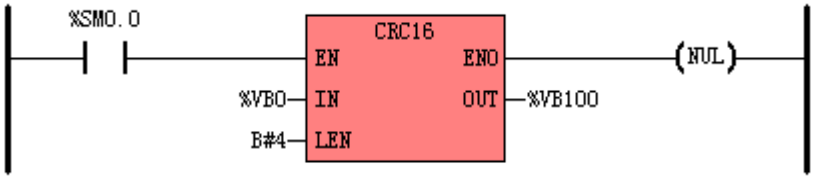
■ LD

Если EN равен 1, то эта инструкция выполняется, в противном случае она останавливается.

■ IL

Если CR равен 1, эта инструкция выполняется, в противном случае она останавливается и не влияет на CR.

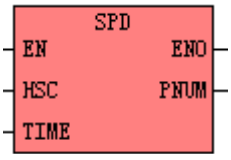
★ Примеры

LD		SM0.0 всегда 1, поэтому CRC16 всегда выполняется: вычисляет CRC для 4 непрерывных байтов, начиная с VB0, затем помещает старший байт результата в VB100, а младший байт в VB101.
IL	LD %SM0.0 CRC16 %VB0, %VB100, B#4	
Результат	**Данные должны быть проверены**	**16-bit CRC**

VB0	VB1	VB2	VB3	VB100	VB101
B#16#1A	B#16#2B	B#16#3C	B#16#4D	B#16#A6	B#16#1

6.17.3 SPD (обнаружение скорости)

★ Описание

	Название	Использование	Группа	
LD	SPD			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
IL	SPD	SPD *HSC, TIME, PNUM*	U	

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
HSC	Вход	INT	Константа (номер HSC)
TIME	Вход	WORD	I, Q, M, V, L, SM, Константа
PNUM	Выход	DINT	Q, M, V, L, SM

Эта инструкция считает количество импульсов, принимаемых указанным высокоскоростным счетчиком, номер которого указан в HSC, за указанный период времени (TIME, ед. изм. мс), и записывает результат на выход PNUM.

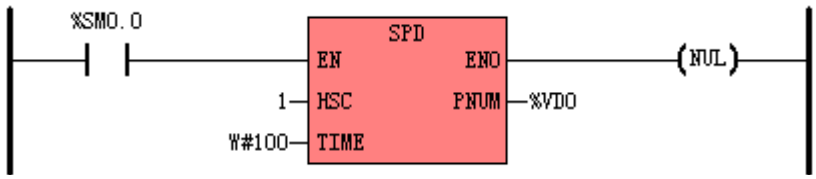
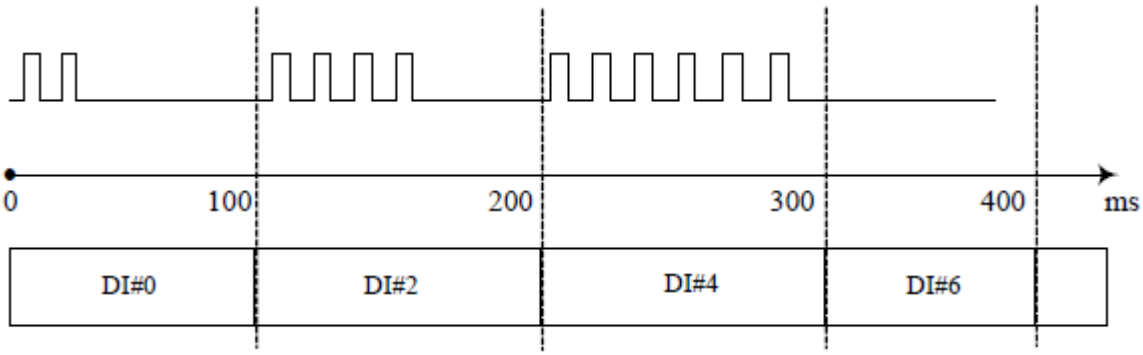
■ LD

Если EN равен 1, то эта инструкция выполняется, в противном случае она останавливается.

■ IL

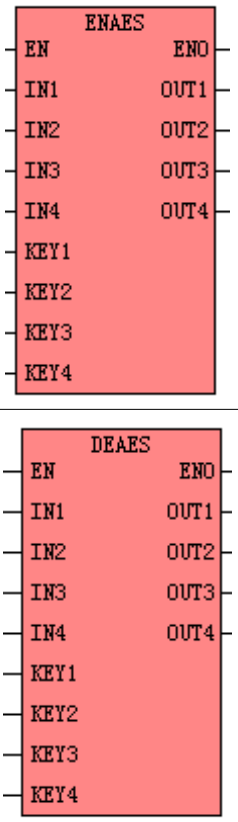
Если CR равен 1, эта инструкция выполняется, в противном случае она останавливается и не влияет на CR.

★ Примеры

LD		SM0.0 всегда 1, поэтому SPD всегда выполняется: считает количество импульсов, полученных в HSC1 каждые 100 мс, и записывает результат в VD0.
IL	LD %SM0.0 SPD 1, W#100, %VD0	
Результат	Последовательность входных импульсов HSC1 Время VD0 	

6.17.4 ENAES (AES-128 шифрование) DEAES (AES-128 дешифрование)

★ Описание

<table>
<tr><th></th><th>Название</th><th>Использование</th><th>Группа</th><th></th></tr>
<tr><td rowspan="2">LD</td><td>ENAES</td><td rowspan="2"> 
</td><td rowspan="3"></td><td rowspan="3">☑ CPU504
☑ CPU504EX
☑ CPU506
☑ CPU506EA
☑ CPU508</td></tr>
<tr><td>DEAES</td></tr>
<tr><td rowspan="2">IL</td><td>ENAES</td><td>ENAES IN1,IN2,IN3,IN4,KEY1,KEY2,
KEY3,KEY4,OUT1,OUT2,OUT3,OUT4</td><td rowspan="2">U</td></tr>
<tr><td>DEAES</td><td>DEAES IN1,IN2,IN3,IN4,KEY1,KEY2,KEY3,
KEY4,OUT1,OUT2,OUT3,OUT4</td></tr>
</table>

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
IN1	Вход	DWORD	I, Q, M, V, L, SM, Константа
IN2	Вход	DWORD	I, Q, M, V, L, SM, Константа
IN3	Вход	DWORD	I, Q, M, V, L, SM, Константа
IN4	Вход	DWORD	I, Q, M, V, L, SM, Константа
KEY1	Вход	DWORD	I, Q, M, V, L, SM, Константа
KEY2	Вход	DWORD	I, Q, M, V, L, SM, Константа
KEY3	Вход	DWORD	I, Q, M, V, L, SM, Константа
KEY4	Вход	DWORD	I, Q, M, V, L, SM, Константа
OUT1	Выход	DWORD	Q, M, V, L, SM
OUT2	Выход	DWORD	Q, M, V, L, SM
OUT3	Выход	DWORD	Q, M, V, L, SM

OUT4	Выход	DWORD	Q, M, V, L, SM

Примечание: IN1, IN2, IN3, IN4, KEY1, KEY2, KEY3, KEY4 должны быть либо все константы, либо все переменные.
ENAES и DEAES являются инструкциями шифрования и дешифрования для AES-128, соответственно.
IN1 / IN2 / IN3 / IN4 являются исходными данными для шифрования / дешифрования; KEY1 / KEY2 / KEY3 / KEY4 секретные ключи, заданные пользователем; OUT1 / OUT2 / OUT3 / OUT4 являются данными после шифрования / дешифрования данных.

■ LD

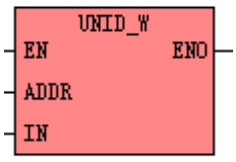
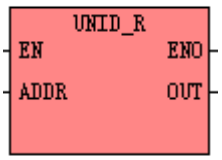
Если EN равен 1, то эта инструкция будет выполнена.

■ IL

Если CR = 1, эта инструкция будет выполнена. Эта инструкция не влияет на CR.

6.17.5 Чтение / запись уникальной области памяти (UNID_R / UNID_W)

Постоянная область памяти K5 обеспечивает 128-байтовую область уникальной памяти, которая разделена на 32 отдельных блока по 4 байта в группе. Пользователи могут читать / писать уникальную область памяти свободно.
Уникальная область памяти очищается перед отправкой PLC с завода. Пользователи могут написать только ненулевые данные в каждом блоке один раз. PLC заблокирует ненулевые данные и не позволит написать снова. Но данные могут быть считаны свободно.
Пожалуйста, обратите внимание, что 4 байта для записи не должны быть все равны нулю. В противном случае ПЛК будет игнорировать операцию записи.
В Kinco Builder выполните [PLC] → [Clear] в меню команд для очистки всех данных в PLC, в том числе проекта пользователя. Данные уникальной области памяти не могут быть удалены по отдельности.
Производители могут использовать уникальную область памяти, чтобы записать серийный номер.

	Название	Использование	Группа	
LD	UNID_W			☑ CPU504 ☑ CPU504EX ☑ CPU506 ☑ CPU506EA ☑ CPU508
	UNID_R			
IL	UNID_W	UNID_W *ADDR, IN*	U	
	UNID_R	UNID_W *ADDR, OUT*		

Операнд	Вход / Выход	Тип данных	Допустимые области памяти
ADDR	Вход	INT	I, Q, V, L, M, T, C, SM, AI, AQ, Константа
OUT	Выход	DWORD	Q, L, M, V, SM
IN	Вход	DWORD	I, Q, L, M, V, SM, Константа

UNID_W для записи данных IN в указанный блок. ADDR определяет номер блока, диапазон 0 ~ 31.
UNID_R для чтения данных из определенного блока и вывод в OUT. ADDR определяет номер блока, диапазон 0 ~ 31.
Примечание: данные в IN должны быть отличными от нуля, в противном случае, PLC будет игнорировать операцию записи.
Примечание: данные уникальной области памяти могут быть очищены только с помощью команды [Clear], не могут быть очищены по отдельности.

■ LD

Если EN равен 1, то эта инструкция будет выполнена.

■ IL

Если CR = 1, эта инструкция будет выполнена. Эта инструкция не влиятельных CR.

Приложение A. Связь с помощью протокола Modbus RTU

По умолчанию, Kinco-K5 служит в качестве ведомого используя протокол Modbus RTU, и может общаться с мастером Modbus RTU напрямую.

1. Область памяти PLC

1.1 Доступные области памяти

Области памяти, которые могут быть доступны с помощью мастера Modbus RTU классифицируются следующим образом:

Тип	Доступные Код функции	Соответствующей области памяти ПЛК
DO (Дискретный выход, 0XXXX)	01, 05, 15	Q, M
DI (Дискретный вход, 1XXXX)	02	I, M
AO (Аналоговый выход, 4XXXX)	03, 06, 16	AQ, V
AI (Аналоговый вход, 3XXXX)	04	AI, V
Запись об ошибке (целое число 16-бит без знака)	03,04	Ошибка PLC области записи

Максимальное количество регистров, которое одна инструкция может посетить:
1. Бит чтения: чтение 1600 бит (200 байт) (код функции 01,02).
2. Бит записи: запись 800 бит (код функции 15).
3. Слово чтения: чтение 100 слов (код функции 03,04).
4. Слово записи: запись 100 слов (код функции 16).
5. Если диапазон памяти меньше, чем максимальное значение, пользователи могут не только прочитать и записать всю память, но не могут прочитать или записать максимальное количество регистров, например, пользователь не может прочитать 90 слов в AI области (аналоговый вход), потому что есть только 32 слова в этой области.

1.2 Номер регистра Modbus

Из-за различий в памяти между CPU504 / CPU 504EX и других CPU, диапазоны, которые разрешено использовать, имеют ограничение. В некотором оборудовании, регистры Modbus начинается с 1, поэтому 1 должна быть добавлена в каждых значениях данных.

★ Для CPU504

Область	Диапазон	Тип	Соответствующий регистр Modbus
I	I0.0 --- I0.7	DI	0 --- 7
Q	Q0.0 --- Q0.5	DO	0 --- 5
M	M0.0 --- M1023.7	DI/DO	320 -- 8511
AI	---	AI	---
AQ	---	AO	---
V	VW0 ---VW4094	AI/AO	100 -- 2147

★ **Для CPU504EX**

Область	Диапазон	Тип	Соответствующий регистр Modbus
I	I0.0 --- I4.7	DI	0 --- 39
Q	Q0.0 --- Q4.7	DO	0 --- 39
M	M0.0 --- M1023.7	DI/DO	320 -- 8511
AI	AIW0 --- AIW14	AI	0 --- 15
AQ	AQW0 --- AQW14	AO	0 --- 15
V	VW0 ---VW4094	AI/AO	100 -- 2147

★ **Для CPU506, CPU506EA и CPU508**

Область	Диапазон	Тип	Соответствующий регистр Modbus
I	I0.0 --- I31.7	DI	0 --- 255
Q	Q0.0 --- Q31.7	DO	0 --- 255
M	M0.0 --- M31.7	DI/DO	320 -- 8511
AI	AIW0 --- AIW62	AI	0 --- 31
AQ	AQW0 --- AQW62	AO	0 --- 31
V	VW0 ---VW4094	AI/AO	100 -- 2147

★ **Запись об ошибке**

Номер регистра Modbus	Описание
9000-9127	Последние 128 типичных кодов ошибок после запуска PLC Среди которых 9000 последняя ошибка и 9001 предпоследняя
9128-9255	Последние 128 типичных кодов ошибок после запуска PLC Среди которых 9128 последняя ошибка и 9129 предпоследняя
9256-9383	Последние 128 типичных кодов ошибок после запуска PLC Среди которых 9256 последняя ошибка и 9383 предпоследняя
9384-9511	Последние 128 типичных кодов ошибок после запуска PLC Среди которых 9384 последняя ошибка и 9511 предпоследняя

2. Основная часть формата отчета Modbus RTU

В идентификационных кодах CRC, как правило, старшие байты будут находиться перед младшими байтами.

Отчет интервала времени, который не более чем 3,5 символа	Номер станции	Код функции	Данные	Идентификационный код CRC
	1 byte	1 byte	N byte	2 byte

2.1 Modbus RTU

Ниже представлен правильный "формат ответа" от ведомого устройства. Если ответ является некорректным, то "код функции" ответа будет иной: старший бит кода функции устанавливается = 1, чтобы получить новое значение. Возьмем код функции 0x01 для примера, если ответ ведомой станции некорректный, то возвращённый код функции ответа в формате 0x81.

2.1.1 Код функции 01: чтение катушки (вкл. / выкл. выход)

Формат запроса:

Номер станции	Код функции	Начальный адрес		Чтение суммы		CRC
1 byte	01	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

Правильный формат ответа:

Номер станции	Код функции	Возвращённое количество байт данных	Возвращённый байт данных 1	Возвращённый байт данных 2	...	CRC
1 byte	01	1 byte	1 byte	1 byte	...	2 byte

2.1.2 Код функции 02: запись / чтение состояния (вкл. / выкл. вход)

Формат запроса:

Номер станции	Код функции	Начальный адрес		Чтение суммы		CRC
1 byte	02	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

Правильный формат ответа:

Номер станции	Код функции	Возвращённое количество байт данных	Возвращённый байт данных 1	Возвращённый байт данных 2	...	CRC
1 byte	02	1 byte	1 byte	1 byte	...	2 byte

2.1.3 Код функции 03: чтение регистра управления (значение аналогового выхода)

Формат запроса:

Номер станции	Код функции	Начальный адрес		Чтение суммы		CRC
1 byte	03	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

Правильный формат ответа:

Номер станции	Код функции	Возвращённое количество байт данных	Регистр 1 старший байт	Регистр 1 младший байт	...	CRC
1 byte	03	1 byte	1 byte	1 byte	...	2 byte

2.1.4 Код функции 04: чтение регистра входа (значение аналогового входа)

Формат запроса:

Номер станции	Код функции	Начальный адрес		Чтение суммы		CRC
1 byte	04	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

Правильный формат ответа:

Номер станции	Код функции	Возвращённое количество байт данных	Регистр 1 старший байт	Регистр 1 младший байт	...	CRC
1 byte	04	1 byte	1 byte	1 byte	...	2 byte

2.1.5 Код функции 05: запись одной катушки (вкл. / выкл. выхода)

Формат запроса: Если установлено успешно, то вернуть исходный отчет.

Номер станции	Код функции	Адрес катушки		Значение		CRC
1 byte	05	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

Примечание: если Значение = 0xFF00, то катушка подключена; если Значение = 0x0000, то катушка отключена.
Формат ответа: Если установлено успешно, то вернуть первоначальный отчет.

2.1.6 Код функции 06: запись одного регистра управления (величина аналогового выхода)

Формат запроса:

Номер станции	Код функции	Адрес регистра		Значение		CRC
1 byte	06	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

Формат ответа: Если установлено успешно, то вернуть первоначальный отчет.

2.1.7 Код функции 15: запись нескольких катушек (вкл. / выкл. выход)

Формат запроса:

Номер станции	Код функции	Начальный адрес		Записать сумму		Значение байта суммы	Значение байта 1	...	CRC
1 byte	15	Старший байт	Младший байт	Старший байт	Младший байт	1 byte	1 byte	...	2 byte

Правильный формат ответа:

Номер станции	Код функции	Адрес катушки		Сумма записи		CRC
1 byte	15	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

2.1.8 Код функции 16: Запись регистра хранения (вкл. / выкл. выхода)

Формат запроса:

Номер станции	Код функции	Начальный адрес		Записать сумму		Значение байта суммы	Значение старшего байта 1	Значение младшего байта 1	...	CRC
1 byte	16	Старший байт	Младший байт	Старший байт	Младший байт	1 byte	1 byte	1 byte	...	2 byte

Правильный формат ответа:

Номер станции	Код функции	Начальный адрес		Сумма записи		CRC
1 byte	16	Старший байт	Младший байт	Старший байт	Младший байт	2 byte

2.2 CRC проверка алгоритма в протоколе Modbus

В протоколе Modbus RTU кадры проверяются CRC. Алгоритмы, следующие:

2.2.1 Прямой расчет CRC

```
/* Параметр: chData —— const BYTE*, direct to the initial address of the memory storage of the data save area to be verified
            uNO —— Байт количества данных, подлежащих проверке
        Return Value: WORD, to calculate the CRC value */
WORD CalcCrc(const BYTE* chData, WORD uNo)
{
        WORD crc=0xFFFF;
        WORD wCrc;
        UCHAR i,j;
        for (i=0; i<uNo; i++)
        {
                crc ^= chData[i];
                for (j=0; j<8; j++)
                        {
                                if (crc & 1)
                                        {
                                                crc >>= 1;
                                                crc ^= 0xA001;
                                        }
                                        else
                                                crc >>= 1;
                        }
        }
        wCrc=( (WORD)LOBYTE(crc) )<<8;
        wCrc=wCrc|( (WORD)HIBYTE(crc) );
        return (wCrc);
}
```

2.2.2 Быстрый расчет CRC со ссылкой на таблицу

```
/ * Таблица старшего байта CRC * /

const UCHAR auchCRCHi[] =
{
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
```

```
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;

/ * Таблица младшего байта CRC * /

const UCHAR auchCRCLo[] =
{
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;
```

```
/* Параметр：puchMsg —— const BYTE*, direct to the initial address of the memory storage of the data
save area to be verified
usDataLen ——  Байт количества данных, подлежащих проверке
Return Value: WORD, to calculate CRC */
WORD CKINCOSerialCom::CalCrcFast(const BYTE* puchMsg , WORD usDataLen)
{
        BYTE uchCRCHi = 0xFF ; /* CRC High Byte Initialization */
        BYTE uchCRCLo = 0xFF ; /* CRC Low Byte Initialization */
        WORD uIndex ;                /* CRC Table Index*/

        while (usDataLen--)
        {
                uIndex = uchCRCHi ^ *puchMsg++ ; /* Calculate CRC */
                uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex];
                uchCRCLo = auchCRCLo[uIndex] ;
        }
        return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Приложение B Динамическое изменение параметров порта RS485

Коммуникационный порт

По умолчанию, они могут вступить в силу только тогда, когда параметры для каждого порта связи будут отредактированы в **[Конфигурации Оборудования PLC]** и загружены в ПЛК.
В то же время, K5 обеспечивает функцию для пользователей, чтобы использовать определенный регистр SM для изменения порта связи RS485 (PORT1 и _PORT2). PORT 0 является портом программирования, который может часто использоваться, следовательно, не может быть изменен.

1. Общее описание

★ Разрешено динамически изменять [Номер станции PLC], [Скорость передачи] и [Проверку четности]
★ Динамически измененное значение параметра хранится в постоянной памяти.
★ Приоритет динамически изменённых параметров связи выше, чем в **[Конфигурации Оборудования PLC]**. Если пользователь повторно загрузит новый проект, K5 будет иметь приоритеты принятия параметров связи изменённых динамически. Пользователь может использовать [PLC] → [Clear ...], чтобы очистить все параметры.
★ После изменения параметров связи, [Номер станции PLC] вступят в силу немедленно, но [Скорость передачи] и [Проверка четности] не наверняка: если коммуникационный порт свободен, то два значения вступят в силу немедленно; в противном случае нет.
Все измененные параметры вступят в силу сразу же после перезагрузки PLC.

2. Команды регистра

K5 будет использовать SMB20 - SMB25 для изменения параметров связи. Здесь Вы можете найти:

★ Параметры: SMB23, SMB24 и SMB25

SMB	Описание
SMB23	Номер станции PLC. Допустимый диапазон: 1-31 Если выполнить запись, SMB23 напишет новый номер станции ПЛК; Если выполнить чтение, SMB23 может только считать номер текущей станции ПЛК; Если выполнить очистку, SMB23 будут игнорироваться.
SMB24	Скорость передачи данных. Допустимый диапазон 0-5: 0 означает 2400, 1 означает 4800, 2 означает 9600 и 3 означает 19200 Если выполнить запись, SMB24 напишет новую скорость передачи данных; Если выполнить чтение, SMB24 только считает текущую скорость передачи данных; Если выполнить очистку, SMB24 будет игнорироваться.
SMB25	Проверка четности. Допустимый диапазон: 0-2. 0 означает отсутствие проверки, 1 означает нечетную проверку, а 2 означает чётную проверку. Если выполнить запись, SMB25 будет проверять четность значения, которое должно храниться; Если выполнить чтение, SMB25 будет читать текущую значение четности; Если выполнить очистку, SMB25 будет игнорироваться.

★ Байт управления: SMB20 и SMB21

Бит	Описание
SMB20: Назначьте порт и работу	
SM20.7	Значение = 1 означает выполнение записи. ПЛК сбросит бит в значение 0 после записи новых параметров.
SM20.6	Значение = 1 означает выполнение чтения. ПЛК сбросит бит в значение 0 после записи новых параметров.
SM20.5	Значение = 1 означает выполнение очистки. ПЛК сбросит бит в значение 0 после записи новых параметров.
SM20.4	Остальные должны считаться как значением 0.
SM20.3 ~ SM20.0	Эти четыре элемента являются номерами портов для работы: 1 означает PORT1 и 2 означает PORT2. Если биты установлены в другое нулевое значение, произойдет ошибка и PLC завершит работу.
SMB21: Назначение параметры связи для работы	
SM21.7 ~ SM21.3	Запасные. Должны оцениваться в 0.
SM21.2	1 означает редактирование или очистку значения четности некоторого порта связи.
SM21.1	1 означает редактирование или очистку значения скорости некоторого порта связи.
SM21.0	1 означает редактирование или очистку сигнала PLC некоторого порта связи.

В то же время, только одно значение SM20.5, SM20.6 или SM20.7 может равняться 1, либо произойдет ошибка и ПЛК завершит работу.
Если выполнить чтение, значение SMB21 будут игнорироваться, и PLC будет читать все параметры связи в одно время.
Когда один параметр сбрасывается, PLC примет соответствующие параметры в аппаратной конфигурации.

★ Бит состояния: SMB22

В SMB22 результат работы этого динамического изменения параметров связи сохраняется.

Бит (только чтение)	Описание
SM22.7	1 означает завершение операции. Если операция будет завершена, независимо от успеха или неудачи, SM22.7 будет установлено в 1 автоматически. Только тогда, когда SM22.7 = 1, другие биты будут действительны в SMB22.
SM22.6	При SM22.7 = 1, если SM22.6 установлен в 1, то операция завершена успешно, а если установлено в 0, то означает сбой.
SM22.5 ~ SM22.0	Если произошёл сбой операции, эти биты выводят коды ошибок, пожалуйста, смотрите ниже.

Код ошибки	Описание ошибки
1	Неправильная команда, например, SM20.7 и SM20.6 будет установлен в 1.
2	Неправильный порт.
3	Неверное значение SMB21.
4	Неверное значение SMB23.
5	Неверное значение SMB24.
6	Неверное значение SMB25.
10	PORT1 не удалось прочитать динамический номер станции PLC, сохраненный в постоянном регистре.
11	PORT1 номер динамической станции PLC еще не установлен.
12	PORT1 не удалось прочитать динамическую скорость передачи данных PLC, сохраненную в постоянном регистре.
13	PORT1 динамическая скорость передачи данных PLC еще не установлена.
14	PORT1 не удалось прочитать PLC динамическое значение чётности, сохраненное в постоянном регистре.
15	PORT1 динамическое значение четности PLC еще не установлено.
20	PORT2 не удалось прочитать динамический номер станции PLC, сохраненный в постоянном регистре.
21	PORT2 номер динамической станции PLC еще не установлен.
22	PORT2 не удалось прочитать динамическую скорость передачи данных PLC, сохраненную в постоянном регистре.
23	PORT2 динамическая скорость передачи данных PLC еще не установлена.
24	PORT2 не удалось прочитать PLC динамическое значение чётности, сохраненное в постоянном регистре.
25	PORT2 динамическое значение четности PLC еще не установлено.
61	Не удалось записать динамические параметры связи в постоянной регистр.

3. Инструкции

★ **Редактирование параметров связи**
★ Установить нижние четыре SMB20 номера портов для работы, например, SMB = B # 1 означает параметры PORT1 для работы
★ Дать соответствующее значение для SMB21 в соответствии с типом параметра
например SMB21 = B # 16 # 03 означает номер станции PLC и передачу значения для редактирования.
★ Дать ожидая новые параметры соответствующих регистров: SMB23 является новым номером станции PLC, SMB24 является новым значением скорости передачи и SMB25 является новым значением проверки четности.
Например SMB23 = B # 03 означает редактирование PLC номера станции №3, SMB24 = B # 3 означает редактирование скорости передачи данных до 19200.
★ (Необязательно) Если бит прошёл через операцию параметров (чтения, записи или очистки), SM22.7 должен быть проверен в первую очередь. Только тогда, когда SM22.7 будет установлен в 1, может начинаться операция.

★ Значение SM20.7 = 1 во время операции записи. ПЛК очистит SM20.7 после записи новых параметров.
★ (Необязательно) Проверьте бит SM22.7 и SM22.6. Оба параметра установленные в 1 означают успешное редактирование параметров.

★ **Чтение параметров связи PLC**
★ Установите нижние четыре SMB20 с номером порта для работы.
Например SMB21 = B # 1 означает параметр PORT1 для чтения.
★ (Необязательно) Если бит прошёл через операцию параметров (чтения, записи или очистки), SM22.7 должен быть проверен в первую очередь. Только тогда, когда SM22.7 будет установлен в 1, может начинаться операция.
★ Значение SM20.6 = 1 во время операции записи. ПЛК очистит SM20.6 после записи новых параметров.
★ Проверьте SM22.7 и SM22.6. Оба параметра установленные в 1 означают успешное редактирование параметров. После чтения параметров, они будут сохранены в регистрах следующим образом: в SMB23 номер станции PLC, в SMB24 скорость передачи данных и SMB25 проверка значения четности.

★ **Удаление параметров связи ПЛК**
★ Установите нижние четыре SMB20 с номером порта для работы.
Например SMB21 = B # 1 означает параметр PORT1 для чтения.
★ Значения в соответствии с типами параметров должны быть очищены.
Например SMB21 = B # 1 означает динамический номер станции PLC и скорость передачи данных должны быть очищены в постоянных регистрах.
★ (Необязательно) Если бит прошел через операцию параметров (чтения, записи или очистки), SM22.7 должны быть проверены в первую очередь. Только тогда, когда SM22.7 будет установлен в 1, может начинаться операция.
★ Значение SM20.5 = 1 во время операции записи. ПЛК очистит SM20.5 после записи новых параметров.
★ (Необязательно) Проверьте SM22.7 и SM22.6. Оба параметра установленные в 1 означают успешное редактирование параметров.

4. Пример

Следующий пример - это вариант динамического номера станции PORT1 и PORT2 на HMI. Его язык программирования IL, вы можете скопировать в редакторе KincoBuilder и выполнить команду [Project] → [LD Language].
VW47 является новым номером станции, который может быть отредактирован на HMI. VW48 также может быть сохранен в VW3690. ПЛК проверит значение VW48 в режиме реального времени и сравнит его с сохранённым в VW3690. Если изменённое значение и VW48 действительное значение, то VW48 будет рассматриваться как новый номер станции PORT1 и PORT2 и редактирование выполнится.

```
(* Network 0 *)
(* Включение используемых значений, которые сохранены, для инициализации VW48 *)
LD          %SM0.1
MOVE      %VW3690, %VW48
(* Network 1 *)
(* Если изменяется значение VB48 и его действительность. Затем сохраните VW48 и начните
редактирование *)
LD          %SM0.0
GE          %VB48, B#1
LE          %VB48, B#31
NE          %VW48, %VW3690
MOVE      %VW48, %VW3690
```

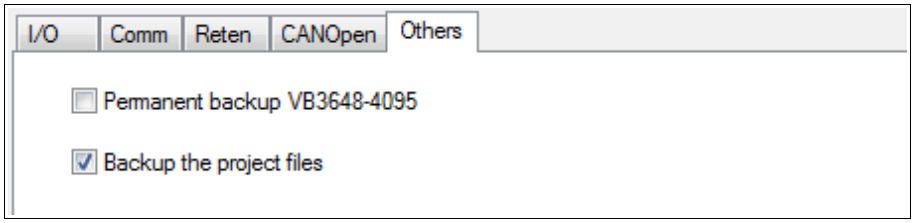
```
ST          %M999.7
(* Network 2 *)
(* Начало редактирования номера станции PORT1 *)
LD          %M999.7
R_TRIG
MOVE        B#1, %SMB20
MOVE        B#1, %SMB21
MOVE        %VB48, %SMB23
S           %SM20.7
S           %M999.6
(* Network 3 *)
(* PORT2 должен быть отредактирован после успешного редактирования PORT1 *)
LD          %M999.6
AND         %SM22.7
R_TRIG
AND         %SM22.6
MOVE        B#2, %SMB20
S           %SM20.7
R           %M999.6
```

Приложение C Резервное копирование данных

Резервное копирование данных означает, запись данных в постоянный реестр, чтобы позволить PLC сохранить данные при потере питания.
Постоянный регистр K5 принимает FARM, который позволяет записывать 10 миллиардов раз. Вы должны отметить, что:
- вы можете обеспечивать только резервное копирование данных, когда это необходимо. Если регистр теряет свою эффективность это приведёт к ошибке CPU.
K5 предусматривает область данных резервного копирования в V области, в которой данные будут автоматически прописаны в постоянном регистре. Пожалуйста, смотрите таблицу ниже:

Длина	448 байт
Диапазон	VB3648 - VB4095

Для совместимости с Kinco-K3, K5 займет VB2684 - VB3902 в области резервного копирования данных, как эффективный при создании нового проекта, который означает, что данные после VB3902 не могут быть сохранены. Если вы хотели бы иметь области резервного копирования после области VB3903, вы можете установить в [PLC Hardware Configuration]→[Others]. Пожалуйста, смотрите на рисунке ниже:



★ [Постоянно сохранять VB3648-4095]
Если будет выбран этот пункт, VB3648-4095 станет областью резервного копирования данных.
★ [Сохранить весь проект в ПЛК Постоянного Регистра]
По умолчанию в K5 сохраняется только информация о конфигурации оборудования и информация по проекту. Если этот пункт будет выбран, вся информация будет сохранена в ПЛК.

Приложение D Диагностика ошибок

K5 имеет три уровня ошибок: критическая ошибка, серьезная ошибка и нормальная ошибка. При возникновении ошибки PLC будет принимать меры в соответствии с уровнем и записывать код ошибки во временной последовательности для последующего анализа. PLC запишет одну и ту же ошибку максимально 4 раза.
Независимо от уровня ошибки, мы настоятельно рекомендуем вам проанализировать и проверить после возникновения ошибки.

1. Уровни ошибок

★ Критическая ошибка
Критическая ошибка возникает, когда в PLC происходит сбой работы микросхем. Эта ошибка может привести к поломке ПЛК и дальнейшим ошибкам. Решением критической ошибки, является переход PLC в безопасное состояние.
При возникновении критической ошибки, ПЛК автоматически выйдет из нормального сканирования и сбросит или войдёт в независимую безопасную подсистему в соответствии с SM2.0. SM2.0 принимает такие меры при возникновении критической ошибки: если значение равно 0, PLC войдет в безопасную подсистему; если значение равно 1, PLC сбросит цикл сканирования и перезагрузится.
Ниже приведены описания состояния безопасности:
★ Все выходы (DO и AO) будут выводить значение определенное в **[PLC Hardware Configuration]**.
★ Индикаторы ERR и STOP мигают, указывая на возникновение критической ошибки.
★ Код и точка записи ошибок позволяют записывать информацию с определенного программного обеспечения.
Примечание: Критические ошибки выводят PLC из состояния нормальной работы, которые могут быть записаны.

★ Серьезная ошибка
Серьезная ошибка может привести PLC в состояние не выполнения нескольких важных функций, но результаты находятся в ожидании. Если возникнет серьезная ошибка PLC автоматически примет меры:
★ Установит PLC в состояние STOP, все выходы (DO и AO) примут выходные значения "Останов и выход" соответственно.
★ Индикаторы ERR и STOP мигают длительно.
★ Записывает код ошибки и позволяет Вам ознакомиться с записью через KincoBuilder и по протоколу Modbus RTU.

★ Нормальная ошибка
Нормальная ошибка происходит, когда ПЛК выполняет несколько функций, но PLC может запускать другие программы. Результаты находятся в ожидании. PLC примет следующие меры:
★ PLC продолжает работать.
★ Индикаторы ERR и STOP мигают длительно.
★ Записывает код ошибки и позволяет Вам ознакомиться с записью через KincoBuilder и по протоколу Modbus RTU.

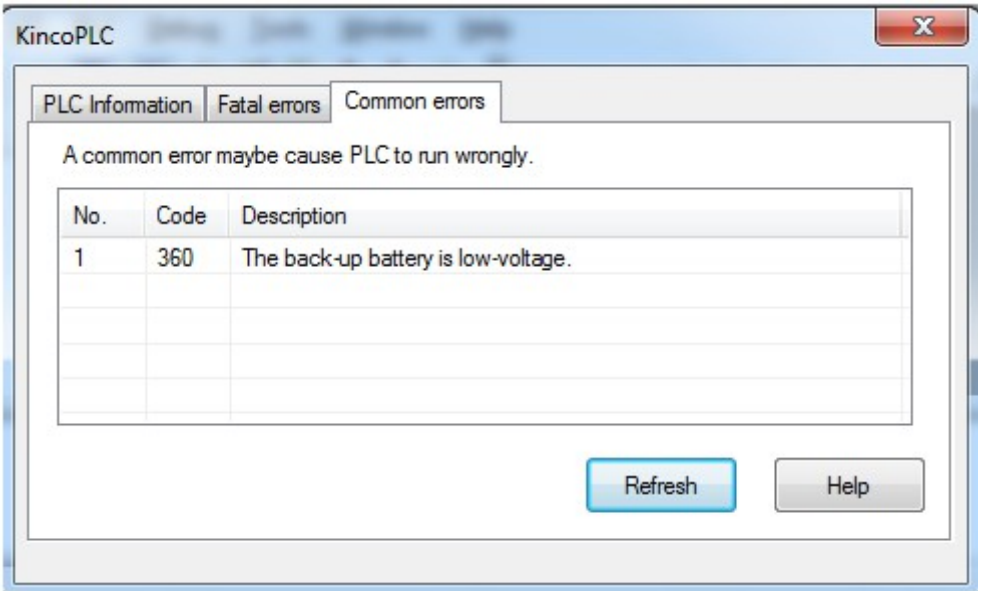
2. Коды ошибок

Код ошибки	Описание
Критические ошибки	
20	Тип CPU в [PLC Hardware Configuration] отличается от типа фактически подключенного CPU
21	Неверный модуль расширения в [PLC Hardware Configuration]
25	При включении не удаётся прочитать тип защиты PLC.
26	При включении не удается прочитать чистый файл.
27	При включении не удается прочитать файл шифра.
28	При включении не удаётся проверить файл конфигурации CRC.
29	При включении, ПЛК обнаруживает неизвестную команду.
30	При включении, число параметров ограничено.
35	При включении, не удается прочитать данные из постоянной.
40	Не удалось выполнить команду JMP.
41	Не удалось вызвать подпрограмму.
42	Не удалось вызвать прерывание подпрограммы.
60	При включении, нет ответа от первого модуля расширения из-за ограничения.
61	При включении, первый модуль расширения отвечает ошибку.
62	1-ый модуль расширения того же типа, что и в аппаратной конфигурации.
65	При включении, нет ответа от второго модуля расширения из-за ограничения.
66	При включении, второй модуль расширения отвечает ошибку.
67	2-ой модуль расширения того же типа, что и в аппаратной конфигурации.
70	При включении, нет ответа от третьего модуля расширения из-за ограничения.
71	При включении, третий модуль расширения отвечает ошибку.
72	3-ий модуль расширения того же типа, что и в аппаратной конфигурации.
75	При включении, нет ответа от четвёртого модуля расширения из-за ограничения.
76	При включении, четвёртый модуль расширения отвечает ошибку.
77	4-ый модуль расширения того же типа, что и в аппаратной конфигурации.
80	При включении, нет ответа от пятого модуля расширения из-за ограничения.
81	При включении, пятый модуль расширения отвечает ошибку.
82	5-ый модуль расширения того же типа, что и в аппаратной конфигурации.
85	При включении, нет ответа от шестого модуля расширения из-за ограничения.
86	При включении, шестой модуль расширения отвечает ошибку.
87	6-ой модуль расширения того же типа, что и в аппаратной конфигурации.
95	При включении, CPU не удается отправить отчет.
96	При включении, шина расширения CPU переходит в состояние пассивной ошибки.

97	При включении, шина расширения CPU переходит в закрытое состояние шины.
Нормальные ошибки	
136	При включении, не удается прочесть регулировку канала AI.
137	При включении, не удается прочесть регулировку канала AO.
138	После настройки, не удается прочесть регулировку канала AI.
139	После настройки, не удается прочесть регулировку канала AO.
300	Во время работы, AI канал выдаёт ошибку DMA.
301	Во время работы, AI канал остановился при преобразовании измерения.
320	Во время работы, шина расширения связи имеет ошибку формата кадра.
321	Во время работы, шина расширения связи вошла в состояние активной ошибки.
322	Во время работы, шина расширения связи вошла в состояние пассивной ошибки.
323	Во время работы, шина расширения была закрыта.
324	Во время работы, шина расширения связи в ошибке: принимающая буферная область переполнена.
325	Во время работы, шина расширения связи в ошибке: буферная область отправки переполнена.
326	Во время работы, шина расширения связи в ошибке: не удалось отправить отчет.
327	Во время работы, CANopen обнаружил ошибку ведомой станции (SDO не отвечает, и т.д.)
329	Во время работы, сообщение об ошибке: деление на ноль.
330	Во время работы, сообщение об ошибке: преобразование команд типа (I_TO_B, DI_TO_I) переполнено.
331	Во время работы, сообщение об ошибке: Команда LN установлена на 0 или отрицательное число.
332	Во время работы, сообщение об ошибке: Команда LOG установлена на 0 или отрицательное число.
333	Во время работы, сообщение об ошибке: Команда SQRT установлена на 0 или отрицательное число.
334	Во время работы, сообщение об ошибке: Команда I_TO_BCD имеет недопустимое значение входного сигнала.
335	Во время работы, сообщение об ошибке: Команда A_TO_H имеет недопустимое значение входного сигнала.
336	Во время работы, сообщение об ошибке: Команда R_TO_A имеет недопустимое значение входного сигнала.
341	Во время работы, сообщение об ошибке: Команда FOR имеет недопустимое значение входного сигнала.
350	Во время работы, сообщение об ошибке: не удалось сохранить постоянные данные.
351	При включении питания сбой данных в оперативной памяти.
360	Низкий уровень заряда резервной батареи.

3. Чтение ошибок

PLC будет автоматически записывать коды ошибок. Вы можете прочитать ошибки следующим способом:



★ С помощью KincoBuilder

Вы можете выполнить [PLC] → [PLC Serious Error] или [PLC Normal Error] в Kincobuilder PORT0 и проверить, как показано на рисунке выше. Вы можете нажать кнопку [Refresh], что бы обновить информацию.

★ С помощью Modbus RTU

Пользователь может использовать команду связи Modbus RTU (код функции 03 и 04), чтобы записать информацию об ошибке через PORT0, PORT1 и PORT2.

Адреса регистров представлены в следующей таблице:

Регистр Modbus	Описание
9000-9127	По истечении времени включения, последние 128 кодов нормальных ошибок. Среди которых 9000 последняя ошибка и 9001 предпоследняя.
9128-9255	По истечении времени включения, последние 128 кодов нормальных ошибок. Среди которых 9128 последняя ошибка и 9129 предпоследняя.
9256-9383	По истечении времени включения, последние 128 кодов нормальных ошибок. Среди которых 9256 последняя ошибка и 9257 предпоследняя.
9384-9511	По истечении времени включения, последние 128 кодов нормальных ошибок. Среди которых 9384 последняя ошибка и 9385 предпоследняя.

4. Регистрация ошибок

K5 обеспечивает регистрацию ошибок в SM и записывает ошибки, когда они происходят. Вы можете сразу прочитать регистр.

★ SMB2: байт управления

Бит (может быть прочтён и записан)	Описание
SM2.0	Его значение определяет, какое действие будет выполнено при возникновении критической ошибки. Исходное значение 0. Если он равен 0: PLC входит в независимом подсистему, чтобы работать безопасно. Если он равен 1: PLC сбрасывается, когда происходит критическая ошибка.

■ SMB0 и SMB1: память и ошибка команды

SM	Описание
SMB0 (только чтение)	
SM0.2	Если ПЛК обнаруживает данных о сбоях питания в ОЗУ, этот бит будет установлен в 1, в противном случае 0.
SMB1 (только чтение)	
SM1.0	1 означает возникновение ошибки: DIV и MOD делится на ноль.
SM1.1	1 означает возникновение ошибки: команды LN, LOG и SQRT являются недействительным (0 или отрицательное число)
SM1.2	1 означает возникновение ошибки: I_TO_B и DI_TO_I переполнены
SM1.3	1 означает возникновение ошибки: I_TO_BCD недействителен BCD код ввода
SM1.4	1 означает возникновение ошибки: A_TO_H входная строка имеет неопределенные байты
SM1.5	1 означает возникновение ошибки: R_TO_A означает переполнение результата преобразования.
SM1.6	1 означает возникновение ошибки: FOR входной параметр недействителен

■ SMB3 и SMB96-SMB110: ошибка модуля расширения.
Если ПЛК обнаружит какую-либо ошибку связи или ошибку отправки отчета в шине расширения, он определит соответствующую шину расширения и разместит код ошибки в регистре ошибок для будущей проверки. Если ошибки не обнаружены, то бит и регистр будут установлены в 0.

Примечание: Если ПЛК обнаружит ошибки в шине расширения или модуле расширения, он перейдёт в состояние STOP и загорится индикатор ERR. Эта ошибка не будет установлена в соответствующий регистр из-за отсутствия выполнения CPU.

SM	Описание
SMB3, SMB5 (только чтение): сигнал ошибки шины расширения	
SMB3.0	Будет установлен в 1, если 1-й модуль расширения имеет ошибки.
SMB3.1	Будет установлен в 1, если 2-й модуль расширения имеет ошибки.
SMB3.2	Будет установлен в 1, если 3-й модуль расширения имеет ошибки.
SMB3.3	Будет установлен в 1, если 4-й модуль расширения имеет ошибки.

SMB3.4	Будет установлен в 1, если 5-й модуль расширения имеет ошибки.
SMB3.5	Будет установлен в 1, если 6-й модуль расширения имеет ошибки.
SMB3.6	Будет установлен в 1, если 7-й модуль расширения имеет ошибки.
SMB3.7	Будет установлен в 1, если 8-й модуль расширения имеет ошибки.
SMB5.0	Будет установлен в 1, если 9-й модуль расширения имеет ошибки.
SMB5.1	Будет установлен в 1, если 10-й модуль расширения имеет ошибки.
SMB5.2	Будет установлен в 1, если 11-й модуль расширения имеет ошибки.
SMB5.3	Будет установлен в 1, если 12-й модуль расширения имеет ошибки.
SMB5.7	Будет установлен в 1, если CPU обнаружит ошибку в шине расширение.
SMB96 - SMB110 (только чтение): сигнал ошибки шины расширения	
SMB96	Если 1-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB97	Если 2-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB98	Если 3-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB99	Если 4-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB100	Если 5-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB101	Если 6-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB102	Если 7-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB103	Если 8-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB104	Если 9-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB105	Если 10-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB106	Если 11-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB107	Если 12-й модуль расширения имеет ошибку, то код будет сохранен здесь.
SMB110	Код ошибки шины расширения CPU сохраняется здесь.

Таблица 1. Ошибка расширения

Код ошибки	Описание
0	Без ошибки
6	Несвоевременный отчёт
7	CPU получает экстренное сообщение в модуле расширения Код ошибки, пожалуйста, обратитесь к следующей таблице 2.

Таблица 2. Код ошибки в аварийном отчете

Код ошибки	Описание
0	Без ошибки
2	CAN шина переходит в состояние пассивной ошибки
3	CAN шина переходит в состояние выкл. шины.
5	PDO ошибка длины

6	Другие отчеты, за исключением ошибок длины PDO
10	ADC ошибка преобразования AI канала
11	Настройка значения сохранения ошибки
12	Регулировка значения чтения ошибки
14	Входной сигнал 1-ого канала модуля расширения вне границ измерения
15	Входной сигнал 2-ого канала модуля расширения вне границ измерения
16	Входной сигнал 3-ого канала модуля расширения вне границ измерения
17	Входной сигнал 4-ого канала модуля расширения вне границ измерения

Таблица 3. Код ошибки CPU шины расширения ошибки связи

Код ошибки	Описание
0	Без ошибки.
1	Ошибка формата кадра связи.
2	Шина расширения в состоянии предупреждения ошибки.
3	Шина расширения в состоянии пассивной ошибки.
4	Шина расширения входит в состояние отключения шины.
5	Получающая буферная зона шины расширения переполнена.
6	Буфер отправки шины расширения переполнен.
7	CPU не удается отправить отчет.

5. Как вернуть CPU к заводским настройкам?

Следующие методы предназначены для возврата CPU к заводским настройкам:
Следующая операция очистит память CPU, в том числе программы пользователя, данных конфигурации, пароль и так далее. CPU возвращается на заводские настройки по умолчанию.
Примечание: После очистки, ПЛК не может снова реализовать необходимую функцию и процесс. Если у вас нет копии программы ПЛК, то вам лучше не очищать его. В противном случае вы должны попросить поставщика оборудования предоставить копию программы и загрузить её снова.

Шаги для очистки:
① Установка параметров связи порта программирования CPU по умолчанию.
Выключите CPU и установите переключатель "STOP". Затем включите CPU, порт программирования восстановит параметры связи по умолчанию: Address: 1; Baudrate: 9600; Parity: None; DataBits:8; StopBits:1.
Примечание: Пожалуйста, не изменяйте положение переключателя до завершения очистки.

② Настройка параметров связи порта ПК.
На странице **[Communication]**, установите параметры **Local** таким же, как порт программирования CPU. Подробнее см. пункт «2.5 Как подключить компьютер с Kinco-K5».

③ Выполните команду "Clear"
В KincoBuilder, выполните [PLC] → [Clear ...] в меню команд, чтобы очистить все сохраненные данные в памяти CPU. После выполнения команды "Clear", CPU восстановит заводские настройки по умолчанию. После очистки, параметры связи порта программирования будет установлены по умолчанию: Address: 1; Baudrate: 9600; Parity: None; DataBits: 8; StopBits: 1.

6. Неисправность: индикаторы RUN или STOP начинают мигать.

Возможные причины и принимаемые меры (подходит только для K5):
★ Установите SM2.0 = 1.
В этом состоянии, когда происходит ошибка, PLC автоматически перезагрузится. Если фатальные ошибки происходит постоянно, то PLC будет постоянно перезагружаться. Индикатор RUN будет постоянно мигать. В этом состоянии индикатор STOP обычно не мигает. Пользователь может установить переключатель в положение STOP, а затем вновь загрузить программу или очистить PLC.
★ Для некоторых индивидуальных K5 PLC, когда пользователь обновляет прошивку до стандартной прошивки K5 PLC, индикатор RUN может мигать. Для этого очистите PLC с помощью KincoBuilder, прежде чем обновлять прошивку. Если индикатор STOP также мигает, пользователи должны восстановить прошивку согласно программе пользователя, а затем очистить PLC или обратиться к нами для технической поддержки, непосредственно.
★ Повреждения оборудования
Существуют различные повреждения оборудования. Повреждение памяти EEPROM может привести к фатальной ошибке, при этом индикаторы RUN / STOP мигать не будут. Повреждения чипа памяти может привести индикаторы RUN / STOP к миганию. При повреждении чипа MCU, индикаторы мигать не будут. Если нет ударов молнии, перенапряжения или вибрации, то при нормальных условиях эксплуатации оборудование повреждается редко.

7. Неисправность: при включения питания K5 PLC, все индикаторы RUN/STOP/ERR включены.

Причина в сбое самоконтроля при включении питания:
★ Полностью повреждена память EEPROM в K5 PLC.
Виды данных сохраняются в памяти EEPROM, в том числе и загружаемые данные. Когда память EEPROM повреждена, PLC не сможет прочитать код загрузки.
★ Пользователь самостоятельно изменил память EEPROM.
Виды данных сохраняются в памяти EEPROM, в том числе и загружаемые данные. Если пользователь изменит память EEPROM, все данные будут потеряны. Тогда PLC не сможет прочитать код загрузки.
★ Пользователь самостоятельно изменил MCU.
Пользователям запрещается изменять MCU или память EEPROM для защиты интеллектуальной собственности производителя.
Если пользователь изменил память EEPROM или MCU, то необходимо изменить их обратно.

Приложение E Определения SM области

В этом приложении описаны определения области системного реестра, SM области. Эта область помогает Kinco-K5 реализовать определенные функции. Вы также можете использовать его для чтения состояния PLC.

1. SMB0: байт состояния системы

SM0.0-SM0.7 используется программой CPU и не может управляться пользователем. Вы можете только вызывать некоторые функции (только для чтения):

Бит (только чтение)	Описание
SM0.0	Всегда "1"
SM0.1	Бит первого сканирования. "1" при первом сканировании, а потом очищается. Используется при инициализации.
SM0.2	Если данные, при сбое питания, отсутствуют в оперативной памяти, то установить этот бит в 1 и очистить.
SM0.3	Серия импульсов с циклом 1с , 50%
SM0.4	Серия импульсов с циклом 2с , 50%
SM0.5	Серия импульсов с циклом 4с , 50%
SM0.6	Серия импульсов с циклом 60с , 50%

2. SMB2: байт управления системой

Бит (только чтение)	Описание
SM2.0	Его значение принимает решение при возникновении критических ошибок. Исходное значение 0. Если он равен 0: PLC входит в подсистему безопасной работы Если он равен 1: сброс PLC
SM2.1	Его значение определяет состояние AI / AO каналов. Начальное значение 0. Если он равен 0: AI / AO канал работает в нормальном режиме Если он равен 1: AI / AO канал в состоянии настройки

3. Сброс порта связи

K5 обеспечивает функцию сброса коммуникационного порта (PORT0, PORT1 и PORT2). После сброса K5 очистит буферную зону коммуникационных портов и начнёт инициализацию. После сброса параметров и функций порта останется то же самое.

★ Регистр управления и Регистр состояния

Бит			Значение	Описание
PORT0	PORT1	PORT2		
SM87.0	SM187.0	SM287.0	1	Установите два бита определенного значения и используйте команду RCV.
SM87.7	SM187.7	SM287.7	0	
SM4.0	SM4.1	SM4.2	-	После удачного сброса, K5 установит значение бита в 1. Он требует ручного сброса.

★ Сброс (PORT0 в качестве примера)
★ (Необязательно) Установите SM4.0 0.
★ Установите SM87.7 = 0 и SM87.0 = 1.
★ Вызовите RCV и задайте в параметрах номер PORT связи. Необходимо установить в 0.
★ (Необязательно) Проверьте SM4.0. Если 1, то сброс выполнен удачно.

★ Пример

```
(* Network 0 *)
(* Используйте изменение I0.0 по переднему фронту для сброса PORT1 *)
(* Команда RCV не повлияет на сброс. *)
LD          %I0.0
R_TRIG
MOVE        B#0, %SMB4
AND         B#16#7F, %SMB187
OR          B#16#1, %SMB187
RCV         %VB222, 1
(* Network 1 *)
(* После сброса вы можете подождать, пока PORT1 не станет стабильным и продолжить работу. *)
(* PORT1 в состоянии получения после сброса *)
LD          %SM4.1
TON         T4, 3
OR          B#16#80, %SMB187
RCV         %VB222, 1
R           %SM4.1
```

4. Другие функциональные переменные

SM	Описание
SMB6	Только для чтения. Сохранить последнее время сканирования PLC. Единица измерения: мс.
SMW10	Только для чтения. Сохранить напряжение резервного аккумулятора. Единица измерения: 0.01V. Если питание резервного аккумулятора постоянно будет ниже, чем 2.6V, PLC предупредит "Low Back-up Battery".
SMB274-SMB285	ID в CPU

5. SMD12 и SMD16: Таймер прерывания цикла

K5 может обеспечить два таймера прерывания с базой 0,1 мс: Таймер прерывание 0 с номером 3; Таймер прерывания 1 с номером 4.
SMD12 используется для определения цикла таймера прерывания 0, с единицей измерения 0,1 мс. Если SMD12 установлен в 0, то таймер прерывания 0 будет запрещён. По умолчанию значение SMD12 = 0;
SMD16 используется для определения цикла таймера прерывания 1, с единицей измерения 0,1 мс. Если SMD12 установлен в 1, то таймер прерывания 0 будет запрещён. По умолчанию значение SMD16 = 0;
Таймер прерывания будет генерировать периодически, и вы можете использовать его для завершения периодических задач. Таймер прерывания не зависит от периода сканирования PLC и может быть использован для точного времени.

Приложение F CANOpen Мастер

CANopen является системой сетевого взаимодействия на основе последовательной шины CAN. Он был первоначально разработан для промышленных систем управления позиционирования, например, системы обработки, но он также может быть использован в других областях, например, транспортные средства, медицинское оборудование и автоматизация зданий.

1. Объекты связи CANOpen

CANOpen Application Layer и Profile Communication (CiA DS-301) подходит для всех устройств CANopen. В DS-301, различные объекты связи определены, и они описываются сервисом и протоколом. Здесь мы представим некоторые, часто используемые объекты связи.

1.1 Управление сетью (NMT)

Управление сетью (NMT) является CANopen ориентированным устройством и применяет структуру ведущий / ведомый. Для этого требуется, что бы одно устройство в сети выполняло функцию NMT Master. Другие узлы NMT Slaves. NMT ведомые устройства однозначно определены в сети по его node-ID, значение, которого, в диапазоне [1..127].
Объекты NMT используются для выполнения сервиса NMT. Через сервис NMT происходит инициализация, пуск, мониторинг, сброс или останов CANopen устройств.

1.1.1 Узел управления NMT

Через сервис контроля узлов, NMT мастер управляет состоянием NMT ведомых устройств. Атрибутом состояния NMT является одно из значений {Stopped, Pre-operational, Operational, Initialisation}.
Формат управляющего сообщения узела выглядит следующим образом.
Master → Slave

COB-ID	Byte 0	Byte 1
0x000	CS (Command Specifier)	Node ID

Node-ID определяет получателя сообщения. Если он равен нулю, то сообщение получат все NMT ведомые.
Command Specifier (CS) представляет следующий сервис:
Запуск удаленного узла (CS = 1),
Останов удаленного узла (CS = 2),
Ввод в предэксплуатационное (CS = 128),
Сброс узла (CS = 129),
Сброс связи (CS = 130).

1.1.2 Контроль ошибок NMT

Сервис контроля ошибок контролирует состояние узлов и сети связи. Существует два варианта для выполнения контроля ошибок.
Охрана узла достигается путем передачи охраняющих запросов от NMT мастера. Если NMT ведомое устройство не отвечает в течение определенного промежутка времени (node life time), или если состояние связи NMT ведомого изменилось, то NMT мастер сообщает главному приложению NMT об этом событии.
Устройство Heartbeat для CANopen устройства устанавливается циклической передачей сообщения heartbeat с помощью производителя heartbeat. Если цикл heartbeat не выполняется производителем heartbeat местного применения, то heartbeat потребитель будет проинформирован об этом событии.

Настоятельно рекомендуется реализовать протокол heartbeat для новых устройств.

★ NMT охрана узла

NMT мастер передает специальный удаленный кадр, в котором нет никаких данных.

COB-ID
0x700 + Node ID

Соответствующий ответ ведомого:

COB-ID	Byte 0
0x700 + Node ID	Bit7: бит переключения. Bit0-6: состояние ведомого. 4 STOPPED; 5 Operational; 127 Pre-Operational.

★ Heartbeat

Производитель heartbeat циклически передает сообщение heartbeat. Один или более потребителей heartbeat получат индикацию. Потребители heartbeat охраняют прием сердцебиения в течение времени heartbeat потребителей. Если heartbeat не получено в течение времени heartbeat потребителя, то будет сгенерировано событие heartbeat.

COB-ID	Byte 0
0x700 + Node ID	Состояние ведомого 4 STOPPED; 5 Operational; 127 Pre-Operational.

1.2 Service Data Object (SDO)

SDO предоставляет прямой доступ к записям объекта CANopen устройства через индекс и субиндекс. С помощью SDO устанавливается канал связи между двумя устройствами CANopen. Клиент всегда инициирует передачу SDO для любого типа передачи. Владельцем словаря объекта является сервер SDO. SDO позволяет передавать данные любого размера. Передача сообщений менее 5 байтов данных, может быть выполнена с помощью передачи "expedited". Передача сообщений более 4 байтов данных должна быть выполнена с помощью передачи "segmented". При передаче "expedited" сообщения выглядят следующим образом.

Запрос Клиент → Сервер:

COB-ID	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x600 + Node ID	CS	index	Sub-index	Data

Ответ Сервер → Клиент:

COB-ID	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x580 + Node ID	CS	index	Sub-index	Data

1.3 Process Data Object (PDO)

Передача данных в режиме реального времени осуществляется с помощью "Process Data Object" (PDO). Связь PDO может быть описана моделью производитель / потребитель. Данные процесса могут быть переданы от одного устройства (производитель) другому устройству (потребитель) или нескольким другим устройствам (широковещание). PDO передаются в неподтвержденном режиме.

Существует два типа использования для PDO. Первый - это передача PDO (TPDO) данных и второй - прием

PDO (RPDO) данных. CANopen устройства, поддерживающие TPDO являются PDO производители, а устройства, поддерживающие CANopen RPDO называются PDO потребители.
PDO выполняется без дополнительных протоколов. Содержание и параметры PDO определяются пользователем через средства конфигурации сети.
PDO описывается параметром связи PDO и параметром отображения PDO. Параметр связи PDO описывает коммуникационные возможности PDO. Параметр отображения PDO содержит информацию о содержании PDO. Здесь мы опишем параметры связи PDO.

★ COB-ID
COB-ID для PDO и его уникальный идентификатор.

★ Тип передачи
Он представляет собой пусковой режим передачи PDO. Это 8-битное целое значение.

• Событие и таймер события
Передача сообщений вызвана возникновением события конкретного приложения, указанного в профиле устройства, профиле приложения или спецификации производителя, или если указанное время (время события) прошло без возникновения события.
Тип передачи 254 означает событие по спецификации производителя.
Тип передачи 255 означает, что событие определено в профиле устройства, профиле приложения.

• Удаленный запрос
Передача PDO по событию инициируется при приеме RTR инициированного потребителем PDO.
Значение типа передачи 252 или 253.

• Синхронный запуск
Передача сообщений инициируется возникновением объекта SYNC. Условием запуска является номер синхронизации и, необязательно, внутреннее событие.
Циклические (типы передачи 1-240) означают, что передача PDO должна быть связана с объектом SYNC.
Ациклические (тип передачи 0) означает, что сообщение должно быть передано синхронно с объектом SYNC, но не периодически.
Тип передачи 252 означает, что передача PDO должны быть связана с объектом SYNC и RTR.

• Время запрета
Чтобы гарантировать, возможность передачи данных для объектов связи с низкими приоритетами, для PDO может быть присвоено время запрета. Время запрета определяет минимальное время, которое должно пройти между двумя последовательными вызовами PDO.
Значение 0 должно отключить время запрета.

• Таймер событий
Это 8-битное целое число. Значение 0 должно отключить таймер события.
Если указанное время (время события) истекло, передача PDO должна включиться, даже без возникновения определенного события.

2. Функция CANOpen master

Кроме K504, все другие модули CPU могут соединяться с K541, чтобы служить в качестве CANopen master.

2.1 Основные характеристики

• Поддержка CAN2.0A, и согласуется с DS301 V4.2.0.
• Поддержка NMT сервиса (Network Management) и служит в качестве NMT мастер.
• Поддержка нормального ускоренного SDO в качестве клиента, а также предоставление инструкций SDO_READ и SDO_WRITE.
• Поддержка 72 CANopen slaves.
• В более 8 TPDOs и 8 RPDOs для ведомых устройств, и 256 TPDOs и 256 RPDOs для всех.
• Поддержка протокола Heartbeat и протокола Node-guarding.
• Управление ошибкой сети

2.2 Как использовать?

2.2.1 Инструмент конфигурации сети CANOpen

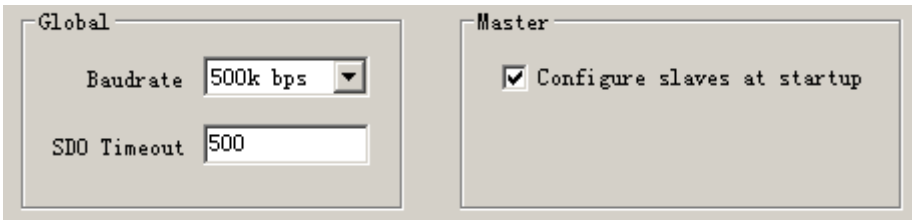
В Kincobuilder откройте окно **[Hardware]**, а затем выберите модуль CPU в верхней части таблицы и перейдите на вкладку **[CANopen]**, теперь вы можете настроить сеть и все устройства.

2.2.2 Управление EDS файлами

В окне **[CANOpen]** → **[Network Settings]** есть 4 кнопки для работы с EDS файлами:
• **[Import EDS ...]**: Нажмите эту кнопку, а затем выберите файл EDS, чтобы импортировать его в KincoBuilder. После импорта EDS файла, соответствующее устройство появится в разделе [Available Devices].
• **[Delete]**: Нажмите и выберите устройство в **[Available Devices]**, затем нажмите кнопку **[Delete]**, это устройство будет удалено из **[Available Devices]** и его EDS файл удалится из Kincobuilder.
• **[Export All EDS]**:
• **[Import All EDS]**:

2.2.3 Этапы конфигурации сети CANopen

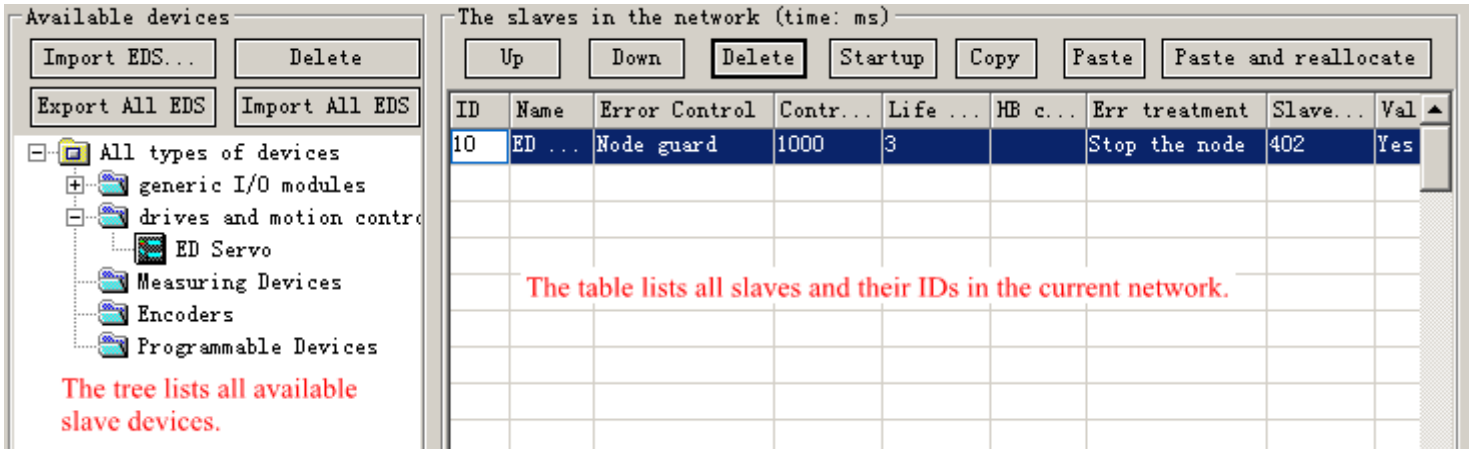
1) Настройка глобальных параметров.
Зайдите на вкладку **[Global Settings]**, как показано на следующем рисунке:



[Baudrate]: Выберите скорость передачи мастера. Обратите внимание, что все устройства в сети должны использовать одну и ту же скорость передачи.
[SDO Timeout]: Установите время для ожидания ответа SDO после отправки запроса SDO мастера. Если мастер не получит ответ SDO до истечения этого времени, мастер должен сообщить об ошибке. Обычно это значение меньше, чем 100 мс.
[Configure slaves at startup]: Мастер управляет NMT состоянием всех ведомых. Кроме того, если этот флажок установлен, то при запуске мастер также передает сообщение для настройки всех ведомых (например, отображение PDO) в соответствии с их конфигураций.

2) Настройка всех ведомых устройств.

Выберите вкладку **[Network Setting]** и продолжите настраивать ведомые узлы и их параметры.

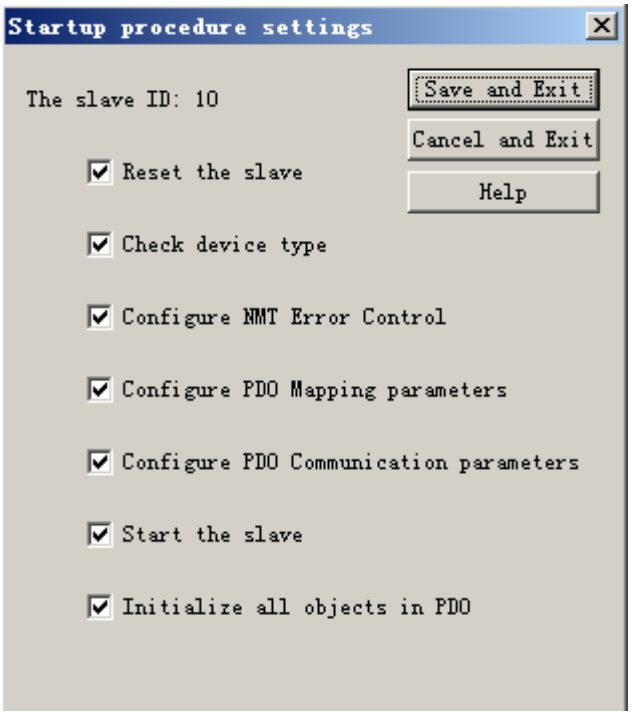


Все функциональные кнопки соответствуют меню правой кнопки мыши. Кликните правой кнопкой мыши на нужном месте, появится соответствующее меню.

a) Добавление ведомого устройства в сеть:
Выберите ведомое устройство и дважды щелкните его в списке слева, после этого устройство добавится в сеть и таблицу справа.

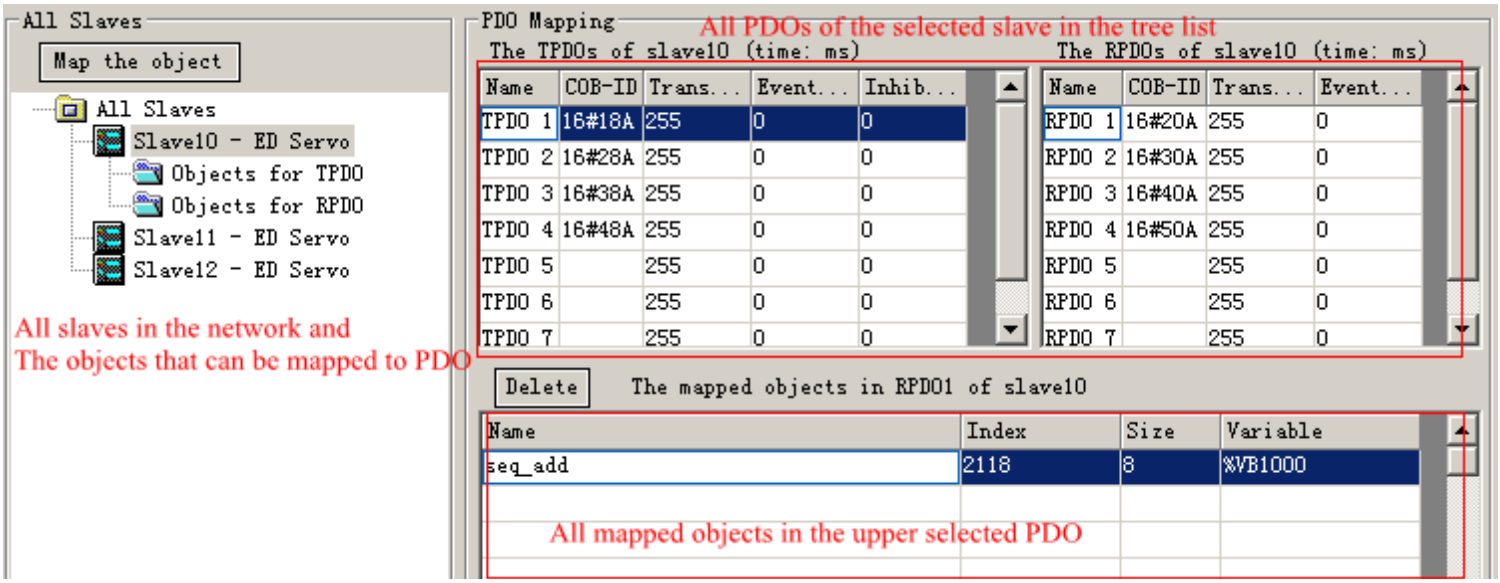
b) Настройка параметра ведомого устройства (например, ID, управления ошибкой):
В таблице справа в столбце **[ID]** перечислены идентификаторы каждого ведомого. В первой строке указан ID ведомого №10.
После добавления ведомого устройства, будут отображаться параметры конфигурация по умолчанию. Ведомые устройства добавляются сверху внизу по списку по умолчанию. Пользователь может выделить одним нажатием строку, чтобы выбрать ведомое устройство, а затем одним нажатием кнопки **[UP]** или **[Down]**, изменить его номер станции, или удалить, нажав **[Delete]**.
[Error Control] используется для выбора метода управления NMT ошибками (NMT Node Guarding или Heartbeat). Если ведомое устройство поддерживает эти два метода, то настоятельно рекомендуется использовать Heartbeat.
[Control Cycle] является временем цикла узла охраны или Heartbeat. Рекомендуется установить это время более, чем 2000ms.
[HB consumer time] это время heartbeat потребителей. Heartbeat Consumer охраняет прием Heartbeat в пределах времени Heartbeat Consumer. Если heartbeat не получен в течение этого времени, появится Heartbeat Error. Рекомендуется установить это время больше, чем 3000ms.
[Error treatment] используется для выбора метода устранения ошибки (в том числе "None", "Stop the node" и "Stop network"), когда мастер обнаруживает ошибку ведомого устройства. Ошибки ведомых устройств, которые могут быть обнаружены включают в себя SDO Time-out, Node Guarding time-out, Heartbeat time-out, Emergency objects и др.

c) Настройка процедуры запуска ведомого устройства:
Нажмите и выберите ведомое устройство в таблице, а затем нажмите кнопку **[Startup]**, теперь вы можете настроить процедуру запуска этого устройства.



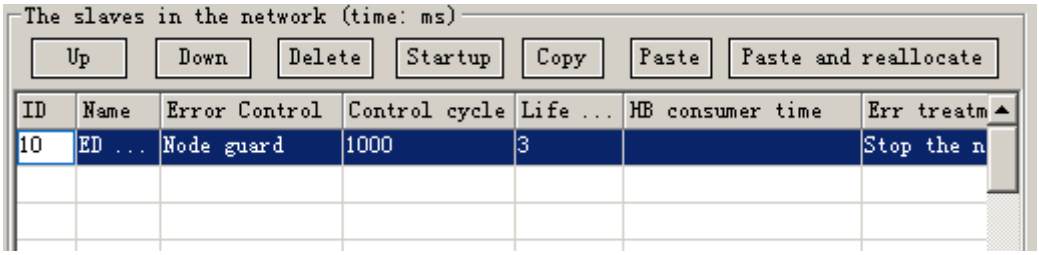
[Reset the slave]: Отправка сообщения "Reset Note" прежде, чем ведущая станция отправит команду конфигурации ведомой станции.
[Check device type]: Чтение и проверка данных устройства, прежде чем ведущая станция отправит команду конфигурации ведомой станции.
[Configure NMT Error Control]: Настройка типа узла управления и соответствующих параметров для ведомой станции.
[Configure PDO Mapping parameters]: Настройка параметров отображения PDO ведомой станции.
[Configure PDO Communication parameters]: Настройка параметров связи PDO ведомой станции.
[Start the slave]: Отправка сообщения "Start Node" ведомой станции.
[Initialize all objects in PDO]: Установка ведущей станцией всех PDOs = 0 ведомой станции, после запуска ведомой станции.

d) Настройка отображения PDO ведомых устройств:
Войдите на вкладку **[Mapping parameters]** и настройте отображения ПДО ведомых устройств.



С левом столбце **[All Slaves]** перечислены все ведомые устройства в сети и все объекты каждого ведомого, которые могут быть отображены в PDO. **[Objects for TPDO]** могут быть отображены только в TPDO, а **[Objects for RPDO]** могут быть отображены только в RPDO. Этапы настройки PDO выглядят следующим образом:
1) Выберите ведомое устройство в списке **[All_Slaves]**, все PDO этого устройства появятся таблице с права.
2) Выберите PDO в таблице, затем вы сможете изменить его параметры связи, такие как Event timer, Inhibit time и др.
COB-IDs для TPDO1-4 и RPDO1-4 не могут быть изменены, потому что они используют COB-ID по умолчанию в заранее определенной конфигурации подключения DS301. Вы можете свободно назначить COB-ID для TPDO5-8 и RPDO5-8.
Кроме того, двойным щелчком мыши на любом объекте из левого списка, объект будет добавлен в текущий PDO. В то же время, KincoBuilder автоматически назначает адрес в области V для объекта, например: VW1006. Затем пользователи могут управлять объектом с помощью VW1006 в программе.
3) Повторите описанные выше шаги, пока не настроите все PDOs текущей ведомой станции.

e) Копирование и вставка ведомой станции:
На вкладке **[Network Setting]** есть кнопки **[Copy]**, **[Paste]** и **[Paste and reallocate]**.



[Copy]: Выберите одну ведомую станцию, которая уже настроена, а затем нажмите **[Copy]** для копирования всей информации выбранной ведомой станции (все параметры PDO отображения / связи). Если PDOs не настроены для выбранного ведомой станции, то он сообщит о неудачном копировании.
[Paste]: После успешного копирования ведомой станции, выберите одну пустую строку и нажмите кнопку **[Paste]**, чтобы вставить скопированную информацию о конфигурации, после этого будет создана новая станция.
Примечание: отображение адреса PDOs нового ведомого устройства будет таким же, пользователи должны изменить его самостоятельно.
[Paste and reallocate]: эта команда работает аналогично **[Paste]**. Но, с помощью **[Paste and reallocate]** PDOs адреса будут перемещены автоматически, пользователям не придётся изменять его вручную.

2.3 Индикатор ERR для K541

Как правило, в чипе CAN контроллера полностью реализован протокол CAN2.0. Чип CAN контроллера может автоматически обнаруживать bit error, CRC error, ACK error и др., и устанавливает соответствующие регистры ошибок для доступа внешнего MCU.
K541 считывает регистры ошибок чипа контроллера CAN. После обнаружения сообщения об ошибке, индикатор ERR будет гореть. Если значение ошибки = 0, индикатор ERR будет выключен. Поэтому, когда ERR горит, это указывает на плохую связь и ошибки. После того, как ERR загорится, пользователь может проверить следующие аспекты:
1) Проверьте правильность подключения шины и виртуальное соединение.
2) Проверьте скоростью передачи данных всех узлов.
3) В сети CAN шины, на клеммах 1-ого и последнего узла должны быть установлены резисторы 120ohm.
4)Имеется ли сильный источник помех вблизи сети.
5) Лучше установить "Inhibit time" для частого чтения / записи PDOs (положение, скорость и др.).