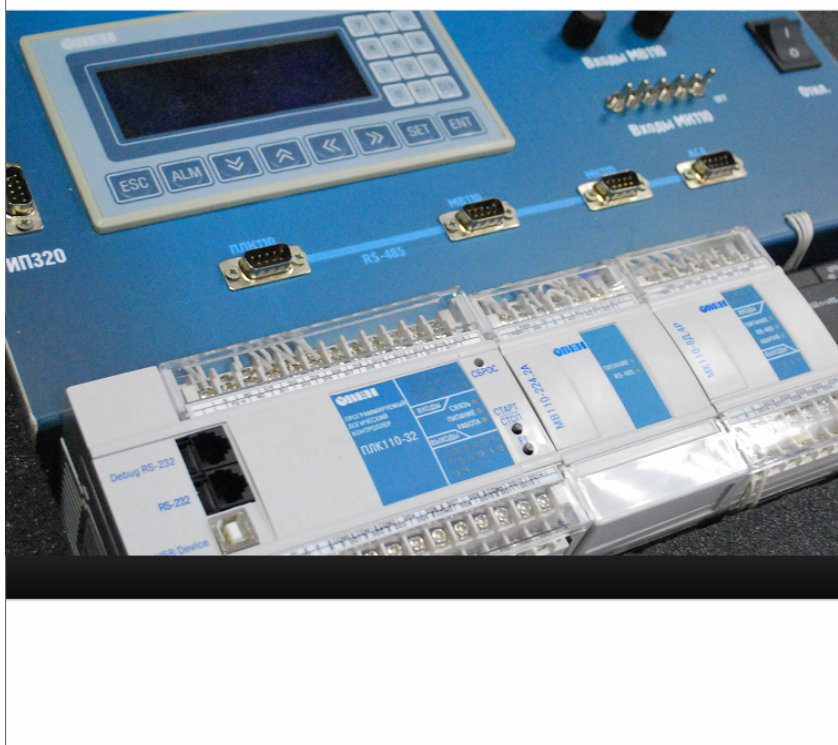


Гайнутдинов Кирилл

Простое и понятное программирование в CoDeSys

кодесис.рф



Часть первая

Знакомство

«Царь, очень приятно, царь...»
Из любимого фильма.

Считается, что серьезная книга, особенно техническая или претендующая на научность, должна начинаться с подробного изложения предметной области. Другими словами, здесь можно было бы долго рассказывать об автоматизации технологических процессов вообще и использовании ОВЕН ПЛК в частности. Затем о классификации программных и аппаратных средств, о том, какие функции, с точки зрения специалистов, должны быть реализованы и в кратчайшее время применены в жизнь... ну и так далее про «корабли, бороздящие просторы больших театров».

Мы не претендуем на научность и важность. Зато претендуем на практичность. Поэтому пропустим долгие рассуждения и сразу начнем знакомиться с предметом обсуждения. То есть станем осваивать систему программирования и ОВЕН ПЛК.

1.1 Установка CoDeSys

CoDeSys для пользователей ОВЕН ПЛК предоставляется бесплатно. При покупке ПЛК в комплекте с контроллером поставляется CD с установочными файлами. Те же файлы можно скачать с официального сайта компании ОВЕН или с ресурса www.кодесис.рф. Будем считать, что читатель справился с задачей поиска дистрибутивов. Теперь можно просто запустить программу установки и просто соглашаться с тем, что она Вам предлагает. При таком подходе все необходимое будет установлено на вашу рабочую машину в папки, заданные по умолчанию. Вместе с бесплатной системой программирования CoDeSys установятся дополнительные программные компоненты в формате демонстрационных версий. Часть из них мы будем рассматривать в дальнейшем. Другие окажутся невостребованными. В конце установки операционная система может выдать Вам следующее сообщение (см. *рис 1.1*):

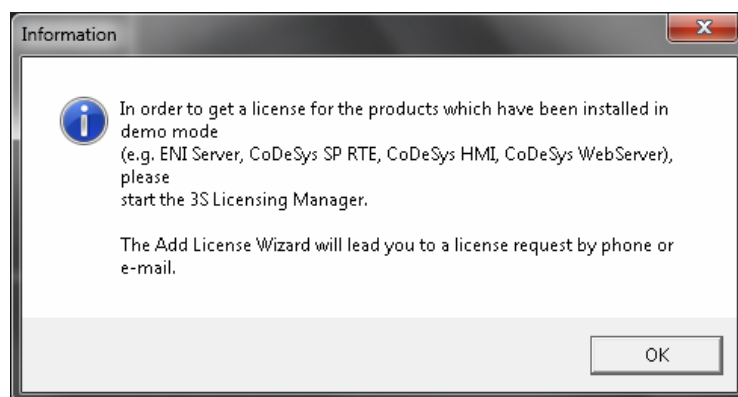



Рис. 1.1


Для читателей, не знакомых с языком Шекспира и Черчилля, можно кратко сказать, что те самые демонстрационные версии требуют покупки лицензий для полнофункционального использования. Однако нас с Вами это никак не касается. Сама система программирования CoDeSys 2.3 в отличие от дополнительных компонентов устанавливается на ваш компьютер без ограничений абсолютно бесплатно. Именно она нам с Вами и нужна. Так что можно смело нажать на кнопку «ОК» и забыть об этом сообщении.

При написании книги автор использовал версию CoDeSys 2.3.9.22. Необходимо заметить, что рассмотренный здесь и далее материал одинаково подходит как для данной версии среды программирования, так для выпущенных позже версий CoDeSys 2.3.

1.2 Создание нового проекта, знакомство с интерфейсом.

После окончания установки на рабочем столе вашего компьютера появится ярлык с

тремя разноцветными шестиугольниками  для запуска CoDeSys. Можно воспользоваться им или выбрать в меню «Пуск» следующий путь «Все программы - 3S Software – CoDeSys 2.3». Если ярлык на вашем рабочем столе не появился, создайте его для удобства запуска системы программирования. После запуска CoDeSys появится серое окно со строчкой меню сверху. Мы создадим новый проект и с его помощью познакомимся с внешним видом системы программирования. Если CoDeSys уже установлен на вашем компьютере, и Вы пробовали свои силы в программировании, то при новом запуске системы она может загрузить предыдущий проект, над которым Вы работали до этого. Вы можете нажать кнопку отмены загрузки или дождаться ее окончания. После этого можно следовать инструкциям, изложенным ниже.

В меню «Файл» необходимо выбрать пункт «Создать» (рис. 1.2), либо немного ниже найти иконку  и воспользоваться ею. В появившемся окне «Настройки целевой платформы» (рис 1.3) пока ничего не меняем, нажимаем кнопку «ОК». В следующем окне (рис. 1.4) система программирования предлагает выбрать нам язык реализации. Можно выбрать язык функциональных блоков CFC согласно рисунку и нажать «ОК».

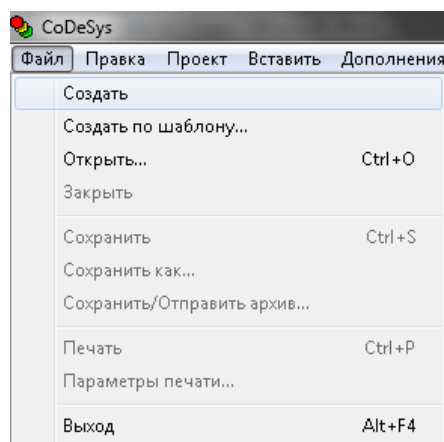


Рис 1.2

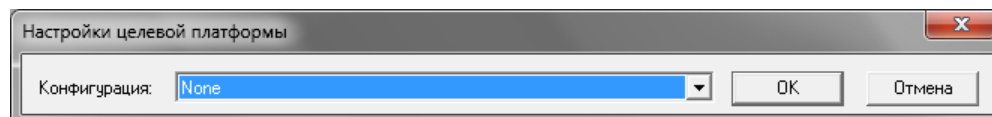


Рис 1.3

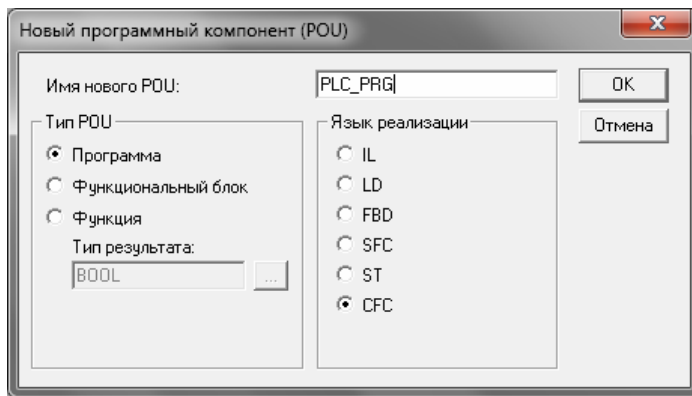


Рис 1.4

После этих операций CoDeSys открывает основную рабочую область (рис. 1.5). **Главное меню** в верхней части содержит пункты «Файл», «Правка» и т.д.

Панель быстрого доступа ниже главного меню позволяет выполнять наиболее часто используемые операции одним нажатием на соответствующую иконку. Состав иконок будет меняться в процессе работы над различными компонентами проекта. Увы, у пользователя нет возможности самостоятельно добавлять или удалять иконки. В дальнейшем мы увидим, что в большинстве случаев все необходимое уже и так вынесено в эту область.

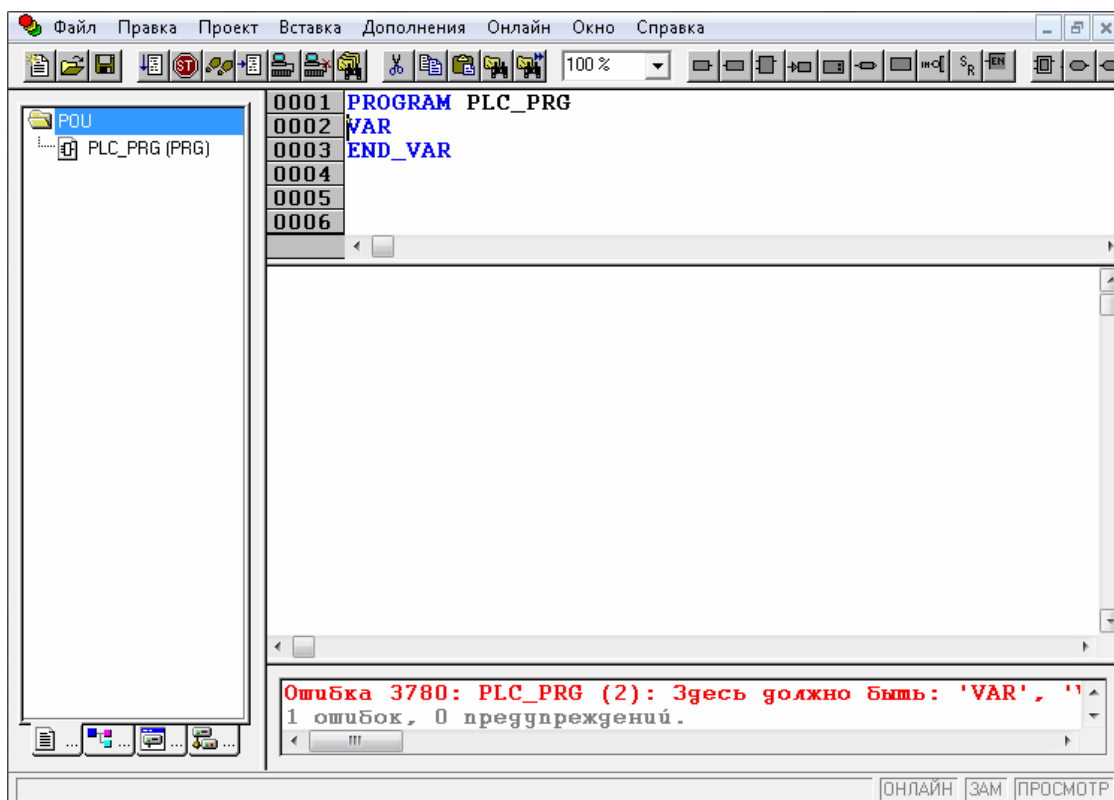




Рис.1.5

Область, занимающая левую сторону экрана – это **Менеджер объектов**. В ее нижней части можно увидеть четыре вкладки.



Рис. 1.6

Переключаясь между ними с помощью мыши, мы можем выбрать различные компоненты (объекты) проекта, посмотреть их содержимое, что-то удалить или добавить. Если выбрать крайнюю левую вкладку  левой кнопкой мыши (далее примем сокращение ЛКМ обозначения нажатия на левую кнопку мыши) и затем дважды нажать ЛКМ на надписи  PLC_PRG (PRG), то в центральной части экрана будет видна **рабочая область** главной программы PLC_PRG. Аналогичным образом, переключаясь между вкладками в менеджере объектов, мы можем вызывать на экран содержимое того или иного компонента для просмотра или редактирования.

Верхняя часть рабочего поля программы называется **областью определения переменных**. Вне зависимости от выбранного языка программирования эта область всегда будет изначально иметь вид, представленный на *рис. 1.7*.



Рис. 1.7

При программировании в CoDeSys все данные, динамически изменяющиеся в процессе работы (измерения на входах, уставки, состояния выходных переменных и т.п.), принято задавать через различные переменные. Об этом будет обстоятельный разговор в одной из следующих глав. А пока полезно запомнить, что именно в области определения задаются те самые переменные, значения которых мы планируем использовать в текущей программе. Вероятно, пока у читателя в голове некоторая путаница в связи со всеми этими переменными, программами, языками и т.д. Обещаем, в самом ближайшем времени мы во всем этом разберемся, и терминология станет предельно ясной.

Ниже области определения находится **область программирования**. Собственно в ней мы и будем в дальнейшем прописывать те или иные алгоритмы либо в текстовом, либо в графическом виде, в зависимости от выбранного языка реализации.

Ниже области программирования располагается **окно сообщений**. Оно появляется не сразу при создании проекта, а в тот момент, когда система проверяет результат нашей работы, например, перед загрузкой алгоритма в ОВЕН ПЛК. Как это сделать, мы рассмотрим немного позже.

В данном окне система приводит данные о проекте, например, размер использованной памяти, количество задействованных переменных и т.п. Вместе с тем, наиболее важной информацией здесь могут являться сообщения о явных ошибках, обнаруженных системой. В окне приводится общее число обнаруженных ошибок, и сообщения о каждой из них. Такие сообщения (см. *рис. 1.8*) выводятся красным шрифтом. При этом система указывает, в каком месте проекта обнаружена ошибка и какая это ошибка. Двойное нажатие ЛКМ на соответствующей строчке с сообщением переносит нас в ту часть проекта, которая, по мнению CoDeSys, содержит ошибку. До тех пор, пока все ошибки не будут исправлены, система не позволит загрузить проект в ПЛК или запустить его в режиме эмуляции.

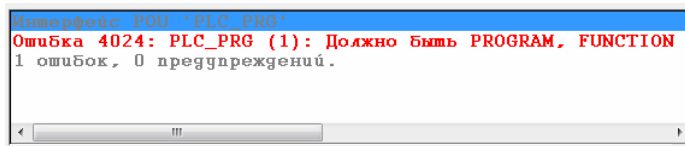


Рис. 1.8

Еще ниже, в правом нижнем углу экрана находится небольшой по размеру, но весьма важный для нас элемент. Это **строка статуса** (рис. 1.9). По ней начинающий пользователь определяет, в каком режиме сейчас находится CoDeSys. Самое очевидное – можем ли мы сейчас редактировать проект, или система находится на связи с ОВЕН ПЛК. Типы режимов, их особенности и индикацию, изменяющую строку статуса мы также будем разбирать по мере продвижения по материалу книги.

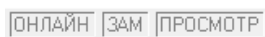



рис. 1.9

Сразу после создания проекта полезно сохранить его под каким-нибудь именем. Для этого можно воспользоваться меню «Файл» и в открывшемся списке команд выбрать ЛКМ пункт «сохранить» (рис. 1.10). С той же целью можно воспользоваться сочетанием клавиш **Ctrl+S** или нажать ЛКМ на иконку . В открывшемся окне «Сохранить как» (рис. 1.11) определяется место на вашем компьютере, куда Вы планируете сохранить файл проекта. Резонно создать отдельную папку для сохранения всех проектов, которые Вы будете реализовывать в процессе общения с этой книгой. Тем более, что к отдельным примерам мы будем обращаться несколько раз в различных главах. Итак, Вы выбрали папку, куда планируете сохраниться. Затем в поле «Имя файла» Вы записываете произвольное имя, которое считаете уместным. Например, prg1. Заметим, что язык здесь не принципиален, Вы можете использовать как кириллические, так и латинские символы. Проверьте, что в поле «Тип файла» установлено «CoDeSys проект (*.pro)» и нажимайте «Сохранить». Автор не настаивает, чтобы Вы называли проекты, переменные и другие элементы также, как это сделано на страницах этой книги. С другой стороны, нам будет удобнее использовать одинаковые обозначения, т.к. в дальнейшем мы будем ссылаться на те или иные примеры (и файлы), созданные ранее в предыдущих главах.

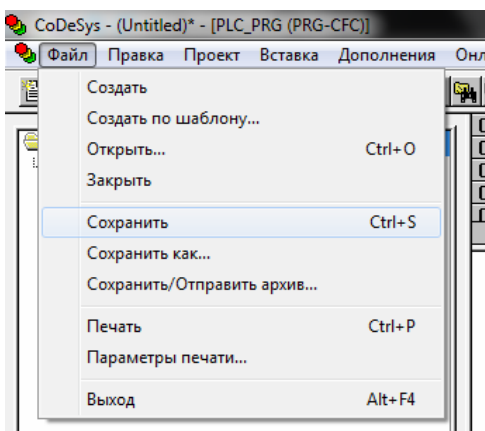


Рис. 1.10

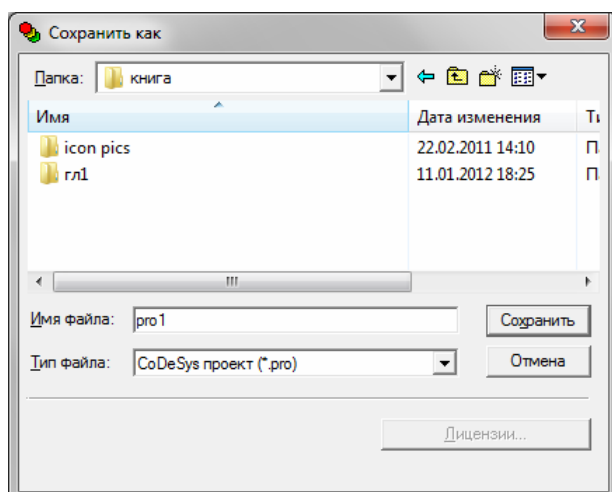


Рис. 1.11

Обратите внимание, что после успешного сохранения в левом верхнем углу окна системы надпись (Untitled)* сменилась на присвоенное Вами имя файла (pro1) с расширением .pro. Теперь можно себя похвалить – начало вашей успешной работы положено!

1.3 Настройка проекта

Для начала полезно сделать несколько настроек системы, которые позволят в дальнейшем сделать работу более удобной. Для этого мы переходим в раздел «Проект» главного меню, а затем ЛКМ выбираем пункт «Опции» (рис. 1.12).

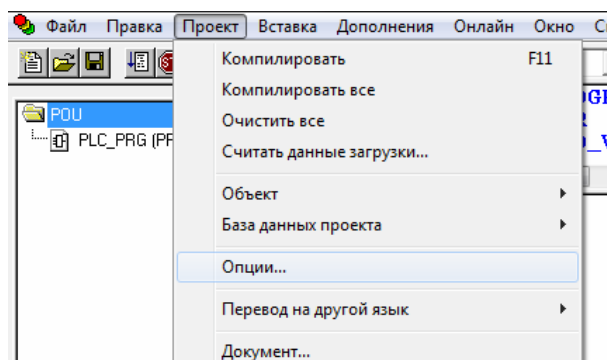


Рис. 1.12

В открывшемся окне «Опции» в левой части выбираем категорию «Сохранение». Вид окна представлен на рис. 1.13.

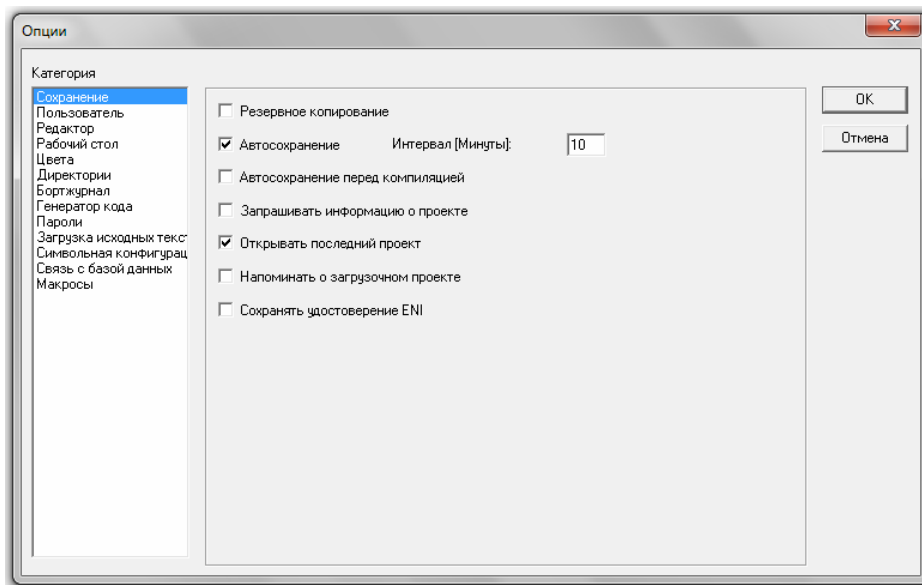


Рис. 1.13

Среди открывшихся опций полезно обратить внимание на пункт «Автосохранение». Наверняка многие сталкивались с ситуацией, когда результат многочасовой напряженной работы пропадал в следствие некорректного завершения работы программы или выключения компьютера. Весьма часто, увлекшись, мы забываем сохранять промежуточные результаты. Чтобы подобное расстройство не постигло Вас в процессе программирования ОВЕН ПЛК, полезно в опциях проекта (см. рис. 1.13) установить галочку в пункте «Автосохранение» и левее указать интервал создания резервных копий. Конечно, это не снимает с нас с Вами необходимость время от времени сохранять проект вручную.

Тем не менее, при возникновении неожиданных неприятностей Вы легко восстановите большую часть работы, загрузив резервную копию. Важный момент здесь состоит в том, что опция автосохранения включается только после того, как Вы присвоили файлу проекта какое-либо имя (см. окончание раздела 1.2). Подчеркнем и постараемся запомнить: **файл проекта полезно сохранить сразу после его создания**. Просто чтобы не забыть. Тогда неприятные случайности не смогут нам помешать.

Здесь же, в категории «Сохранение» (рис. 1.14) можно поставить галочку «Открывать последний проект». В этом случае при возобновлении работы с CoDeSys система автоматически будет загружать последний проект, над которым Вы трудились. Иногда это бывает удобно.

Категория «Редактор» (см. рис. 1.14) позволяет изменить шрифт, который мы будем использовать в процессе программирования, сделать его более читаемым и удобным для восприятия. Можно нажать на кнопку «Шрифт», затем в появившемся стандартном окне выбрать тип шрифта, размер, начертание, удобные для Вас.

В процессе создания непосредственно алгоритма в CoDeSys мы будем использовать латинские символы. Однако при грамотном подходе к процессу программирования всегда полезно оставлять комментарии к тем или иным частям проекта. Естественно, удобнее эти комментарии делать на русском языке. Поэтому при выборе шрифта не забудьте поставить значение «Кириллица» в соответствующем поле. Вообще говоря, комментарии полезно оставлять даже для самого себя. Весьма часто проект откладывается на месяц-два, после чего, открывая его, невольно недоумеваешь: «Что здесь написано? И как это работает?» Поэтому полезно облегчить жизнь себе и тем, кто будет работать с проектом после Вас. Для этого старайтесь объяснить на полях наиболее важные для понимания вещи. Но об этом мы еще будем говорить. А пока..

Пока можно нажать кнопку «ОК» в окне выбора шрифта, а затем перейти в категорию «Рабочий стол».

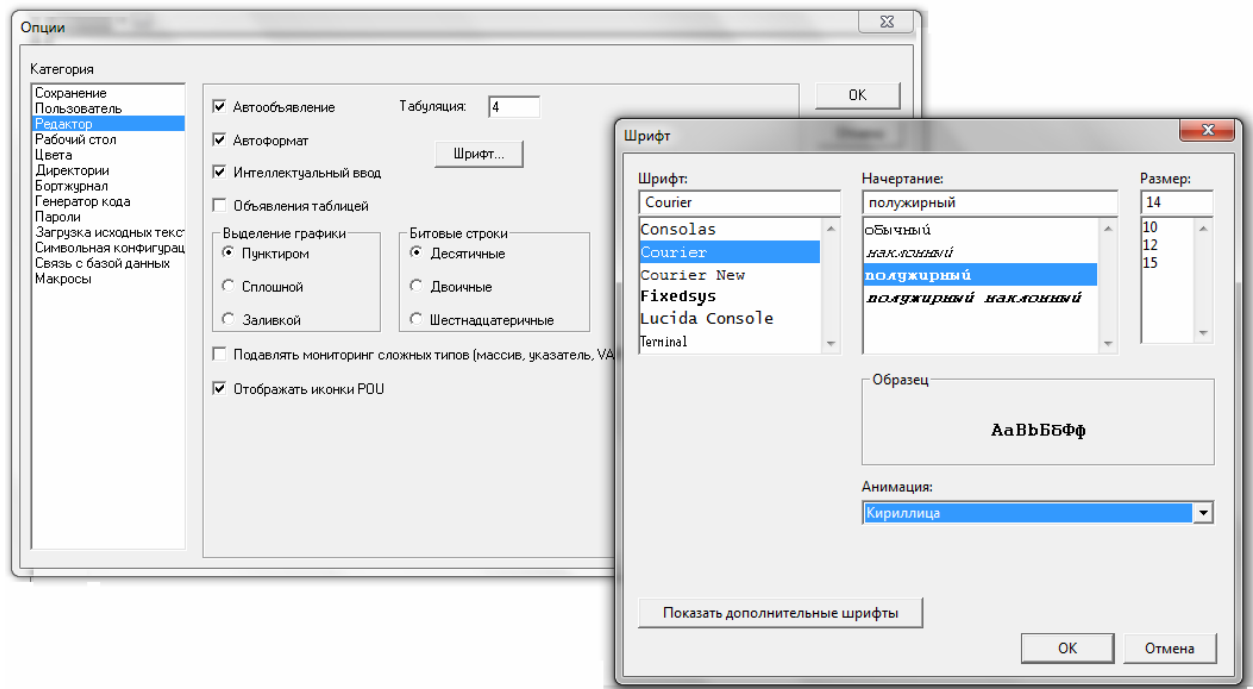


Рис. 1.14

Версия системы программирования CoDeSys 2.3.9.22, равно как и версии, появившиеся позже (имеющие более высокий порядковый номер) обладают важной особенностью. Интерфейс среды, меню и окна практически на 100% переведены на русский язык, что значительно упрощает знакомство и взаимодействие с системой. Вместе с тем сохранена возможность использовать английский, немецкий интерфейс и некоторые другие. Если вдруг после установки и запуска CoDeSys Вы видите на экране, к примеру, англоязычную версию, перейти к более привычному языку Вы можете как раз в опциях проекта на вкладке «Рабочий стол» (рис. 1.15).

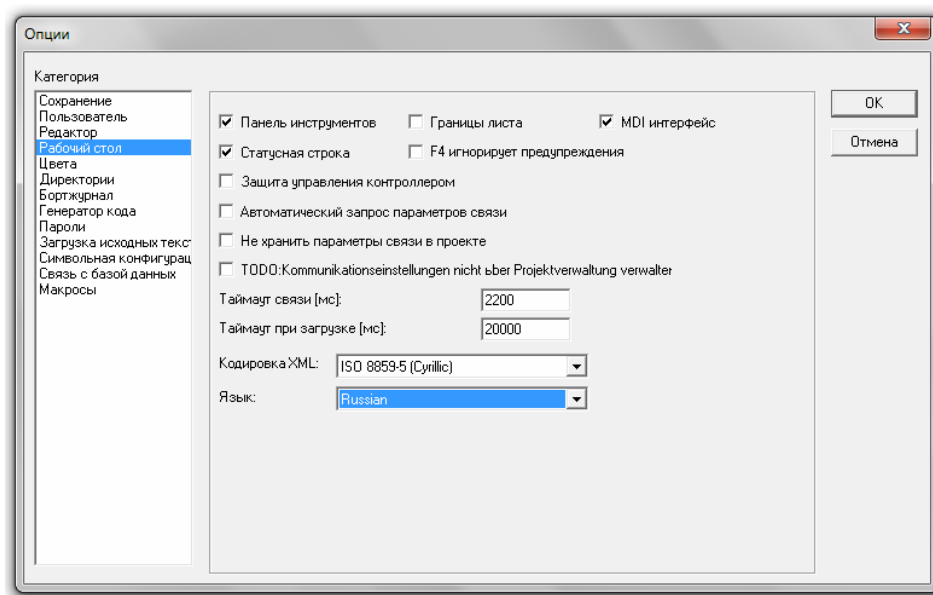




Рис. 1.15

Для смены языка интерфейса Вы выбираете нужный пункт в самом нижнем поле «Язык». Например, «Russian». После этого система автоматически переводит все свои элементы на выбранный язык. Вместе с тем переводятся файлы справки, весьма объемной и очень толково написанной. К справке мы очень скоро начнем обращаться, а пока можно выбрать язык интерфейса или оставить как есть, если Вас все устраивает. Затем нажать кнопку «Ок», приняв все произведенные в окне «Опции» изменения. После этого полезно сохранить проект, например, нажав сочетание клавиш **Ctrl+S**.

1.4 Простейшая задачка

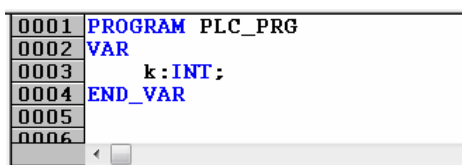
Мы познакомились с интерфейсом системы программирования, провели настройку среды для удобства работы с ней. Самое время собственно поработать, т.е. решить маленькую задачку. А заодно разобраться с режимом редактирования и режимом исполнения.

Необходимо реализовать на языке функциональных блоков CFC выражение $k=k+1$; здесь k – целочисленная переменная, к которой по условию нам необходимо прибавлять единицу.

В том же самом проекте `pro1.pro`, который мы рассматривали в предыдущих параграфах, нам необходимо открыть рабочую область главной программы `PLC_PRG`. Напомним, для этого необходимо ЛКМ выбрать в Менеджере объектов (это область в левой части экрана) крайнюю левую вкладку  и затем дважды нажать ЛКМ на надписи  `PLC_PRG (PRG)`. На экране должна появиться рабочая область, пока пустая, а над ней область определения переменных.

Начинать реализацию любой задачи полезно с определения тех переменных, которые мы планируем задействовать. Напомним, большинство данных, используемых в алгоритме, обычно существуют в проекте в качестве значений определенных переменных. На практике это означает, что если мы хотим брать значение из переменной k или записывать в нее полученный результат – переменная k должна в проекте существовать. Или другими словами, должна быть объявлена. Под объявлением понимается, что мы объясняем системе, что под именем k хотим понимать некоторое значение определенного типа. В нашем случае, по условию k – целое число, поэтому выбирается целочисленный тип переменной. (Подробнее об объявлении переменных и их типах мы поговорим в одной из следующих частей). Довольно много не совсем понятных слов, а реализация очень проста. Посмотрите на *рис. 1.16*. Нам необходимо ЛКМ поставить курсор в области определения после ключевого слова `VAR`, нажать кнопку `Enter` на клавиатуре, освобождая таким образом свободную строчку. Затем мы просто набираем строчку: `k:INT;`

Обратите внимание на необходимость правильно напечатать все указанные знаки препинания. Важно также, чтобы сделанная надпись располагалась между ключевыми словами `VAR` и `END_VAR`. Вероятнее всего, Вы всё сделаете правильно. Мы верим, что эту книгу читают люди способные, внимательные и понятливые. Дорогой читатель, Вы ведь именно такой, правда?



```
0001 PROGRAM PLC_PRG
0002 VAR
0003   k:INT;
0004 END_VAR
0005
0006
```

Рис. 1.16

После нажатия клавиши Enter в конце строки слово INT автоматически окрасится в синий цвет. Сделанная нами надпись однозначно сообщает CoDeSys, что мы хотим использовать имя k, и объясняет, что в этой переменной необходимо хранить целые значения (значения типа INT).

Возможно, объяснение этой простой операции показалось Вам затянутым и излишне подробным. Однако хотелось бы, чтобы Вы сразу воспринимали работу с CoDeSys, как нечто простое, элементарное и понятное. Именно с этой целью автор пускается в долгие объяснения, разбирая материал практически по буквам. В дальнейшем, большой объем информации будет постепенно накапливаться и оседать в вашей памяти. И важно, чтобы эта информация была для Вас проста и максимально понятна. Кроме того, давно известно, что для того, чтобы учащийся что-то запомнил, необходимо повторить это минимум три раза. Ведь мы хотим научиться, правда?

Хорошо, заканчиваем лирическое отступление и возвращаемся к нашей задаче. Если обратить внимание на панель быстрого доступа (она расположена под главным меню), то в правой ее части есть несколько иконок (рис. 1.17). Они характерны для программирования на языке функциональных блоков CFC (напомним, именно такой язык мы выбрали при создании проекта – см. рис. 1.4).



Рис. 1.17

Первыми тремя иконками нам предстоит воспользоваться. Познакомимся с ними:






- «вход» - позволяет получить текущее значение переменной, чтобы использовать его в алгоритме.



- «выход» - позволяет записать некое полученное значение в переменную.



- «элемент» - позволяет добавить нужный оператор на рабочую область.

Начнем со «входа». Нажмем на иконке  ЛКМ, передвинем курсор на рабочую область и еще раз нажмем ЛКМ в нужном месте, чтобы добавить вызов значения в нашу программу. В рабочей области появляется изображение . Аналогичную операцию можно проделать, если в рабочей области нажать правой кнопкой мыши (далее ПКМ) и в появившемся контекстном меню (см. рис. 1.18) выбрать пункт «Вход». Затем появившийся значок  устанавливаем в нужное место рабочей области с помощью ЛКМ.

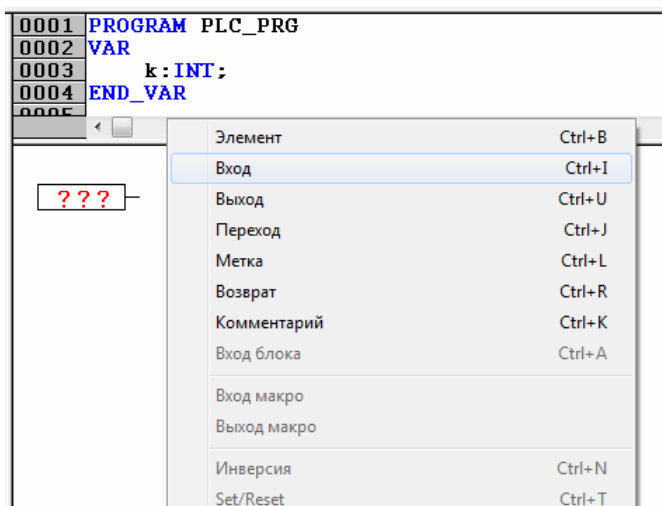


Рис.1.18.

Попробуйте разными способами добавить два «входа» на рабочую область. Если элемент необходимо передвинуть, он выделяется в рабочей области. Для этого зажимается ЛКМ и появившаяся на экране пунктирная рамка растягивается, захватывая те элементы, которые необходимо выделить (рис.1.19). Затем ЛКМ необходимо зажать выделенную область в левой части нужного элемента (рис.1.20), перетащить его на нужное место и отпустить ЛКМ. Выделенный элемент можно удалить с рабочей области, например, нажав кнопку Del на клавиатуре. Для того, чтобы сбросить выделение, достаточно нажать ЛКМ в любом свободном месте рабочей области.

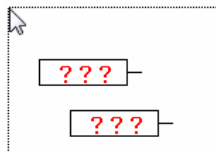


Рис.1.19

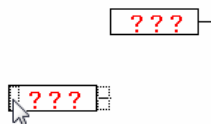



Рис.1.20

В двух появившихся в рабочей области изображениях присутствуют знаки вопроса красного цвета. Таким образом, система подсказывает нам, что вместо вопросительных знаков нам необходимо указать значение или ту переменную, из которой такое значение нужно взять. В нашем задании необходимо использовать значение целой переменной *k*, а также использовать значение 1. Мы нажимаем ЛКМ на знаках вопроса в любом из добавленных элементов. Надпись выделяется ???, ее можно стереть и написать имя переменной *k*, а затем нажать Enter. Аналогично выделяя текст внутри второго «входа» мы записываем туда значение 1 (см. рис.1.21).



Рис.1.21

Теперь можем добавить на рабочую область «Выход», то есть операцию записи значения в переменную. Для этого, по аналогии со «Входом», нажимаем ЛКМ на иконке  и устанавливаем «Выход» в нужное место рабочей области, либо используем контекстное меню (см. рис.1.18) и в нем выбираем пункт «Выход». Перетаскивание этого изображения производится также аналогично «Входу», однако область, за которую надо «ухватиться» курсором, расположена у «Выхода» справа (рис.1.22).

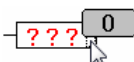


Рис.1.22

Пока не обращайте внимания на цифру 0 рядом с «Выходом». С ней мы разберемся обязательно немного позже. Пока же вспомним о том, что по условию задачи нам

необходимо результат работы программы записать обратно в переменную k. Поэтому вместо знаков вопроса в «Выход» записываем имя переменной и нажимаем Enter. Полученный результат виден на *рис. 1.23*.

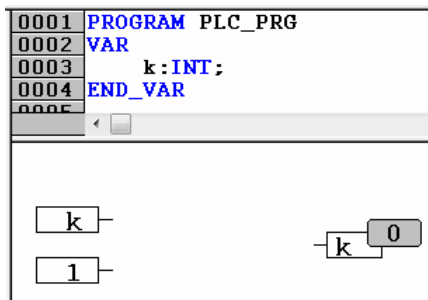



Рис.1.23

Нам осталось реализовать заключительную часть нашей простой программы – операцию сложения, результат которой необходимо передать на «Выход», то есть записать в переменную k. Для этого мы используем «Элемент», нажав ЛКМ на иконке . Одноименный пункт есть в контекстном меню (см. *рис. 1.18*), которое можно вызвать, нажав на свободном месте рабочей области ПКМ. «Элемент» мы размещаем между входами и выходом, как показано на *рис. 1.24*.

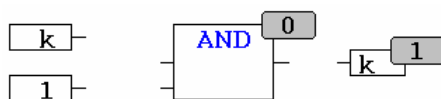


Рис.1.24

«Элемент», который часто называется блоком, это простейшая операция. Слева к нему подводятся исходные данные, справа можно получить результат и по линиям связи передать в другую часть алгоритма, например, как исходные данные для другого блока. Заголовок элемента определяет, какую операцию этот блок будет выполнять. Практически все алгоритмы на языке CFC выглядят, как набор блоков, входов и выходов. Конечно, алгоритмы могут содержать и другие элементы. Однако без трех, разобранных выше, обычно обойтись не удастся. Они составляют основу программирования на языке функциональных блоков CFC.

Операция сложения, которой нам необходимо воспользоваться, имеет в CoDeSys заголовок ADD. По умолчанию в «элемент» подставляется другой заголовок – AND – логическое «И». Нажав ЛКМ на заголовке блока, мы можем подправить его, напечатав нужное нам ADD. Затем необходимо задать для операции сложения исходные данные. Другими словами, подвести значение переменной k и единицу, используя линии связи. Для этого мы нажимаем ЛКМ на «хвостике» входа k (*рис. 1.25*). Удерживая ЛКМ, аккуратно перетаскиваем линию связи к «хвостик» слева от операции ADD (*рис. 1.26, 1.27*) и потом отпускаем ЛКМ. Теперь переменную k и блок ADD соединяет линия связи (*рис. 1.28*). В первые разы у Вас может не всё получаться, поэтому постарайтесь точнее доводить курсор до нужного «хвоста». И тогда все получится!

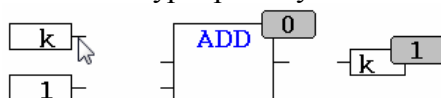


Рис. 1.25

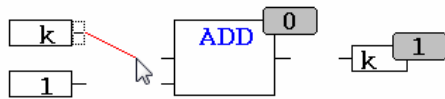


Рис. 1.26



Рис. 1.27



Рис. 1.28

Таким же образом соединяем второй вход и выход с соответствующими «хвостами» ADD (рис. 1.29). С разрешения Читателя в дальнейшем мы будем называть эти «хвосты» входами и выходами блока, подразумевая, что на входы подаются исходные данные, а с выхода мы можем забрать результат операции.

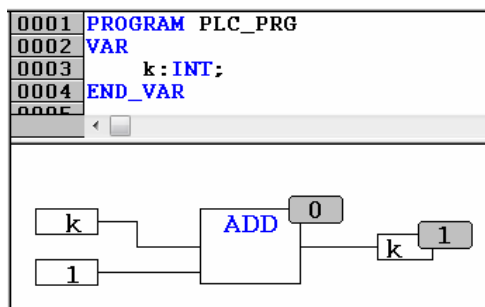



Рис. 1.29

На рис. 1.29 наша программа приобрела практически законченный вид. Похоже, что это было не слишком сложно, правда?!

Теперь пара слов про нумерацию, которая появилась при добавлении на рабочую область выхода и блока ADD. В данном случае число **0** или **1** в правом верхнем углу блока (операции) определяет порядок, в котором система будет обрабатывать действия нашей программы. Исходя из нумерации на рис. 1.29 сначала система сложит значение переменной k и единицу, и только затем полученный результат будет снова помещен в переменную. Такая последовательность вполне логична.

При написании алгоритма CoDeSys автоматически расставляет порядок обработки действий в направлении из левого верхнего в правый нижний угол. Другими словами, если Вы добавляете операцию выше или правее, то в программе это действие будет выполняться раньше. Однако когда мы начинаем вносить исправления, копировать элементы и куски программы, перетаскивать их на новое место, порядок может нарушаться. Причем, чем больше ваш алгоритм, тем сложнее отследить эти несоответствия. А они могут вызвать неработоспособность, казалось бы, правильно написанной программы. Действительно, если сначала мы используем результат

сложения ADD, а только потом его формируем, то в переменную k все время будут записываться неправильные значения. Таким образом, необходимо следить за тем, чтобы правильный порядок был соблюден. Особенно это важно перед тем, как Вы собираетесь запускать алгоритм для проверки, а мы займемся этим очень скоро. К счастью, восстановить нормальный порядок в CoDeSys не сложно. Для этого мы нажимаем ПКМ на свободном месте рабочей области и в появившемся контекстном меню выбираем пункт «Порядок», а затем подпункт «В соответствии с потоком данных» (рис. 1.30). После этого система автоматически восстановит порядок, взяв за начало левый верхний элемент и от него двигаясь по линиям связи до конца программы. В данном случае нам необходимо только позаботиться о том, чтобы начальные операции располагались в рабочей области выше и правее. После расстановки порядка, как, впрочем, и после других важных изменений, полезно сохранить проект .

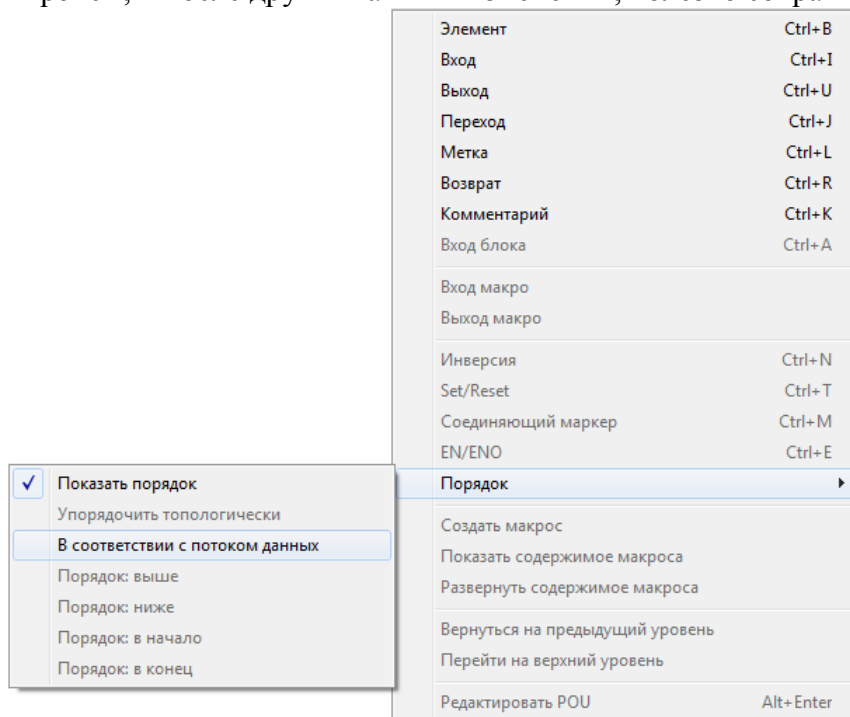


Рис. 1.30

Отметим еще раз один важный момент, важный для языка CFC. После внесения исправлений, пусть даже минимальных, перед тем, как запускать проект на проверку, **необходимо расставить порядок обработки операций**, используя нужный пункт контекстного меню (рис. 1.30).

Для тех, кто хочет расставлять порядок для каждого действия самостоятельно, заметим, что такая возможность имеется (см. пункты контекстного меню ниже надписи «В соответствии с потоком данных»). Гораздо правильнее, и что более важно, гораздо проще и удобнее сразу размещать блоки в нужном месте рабочей области и потом каждый раз автоматически приводить все в порядок. Представляется весьма затруднительным после каждой правки в середине программы вручную править нумерацию. Хотя, «на вкус и цвет у каждого свой фломастер».

Конечно, в других языках программирования, например в ST (Структурированный текст), нет необходимости в расстановке порядка. Между тем язык CFC выбран нами не случайно. Многолетняя практика преподавания показывает, что при знакомстве с программированием именно функциональные блоки являются наиболее понятным и наглядным способом создавать первые программы. А большинство читателей, как нам кажется, именно знакомятся с программированием ОВЕН ПЛК в процессе чтения. И

хотелось бы, чтобы Вы учились и работали с понятными и удобными инструментами. Именно простота и понятность материала чаще всего является причиной успешного усвоения и обучения. Для автора важно, чтобы время, потраченное на работу с этим пособием, прошло для Вас полезно и результативно.

По мере накопления опыта, Вы сможете выбирать тот способ (язык) создания программы, который Вам будет удобнее. Пока же мы будем продолжать делать новые шаги по изучению CoDeSys. Вы уже сделали самый первый и самый важный.

Поздравьте себя!

1.5 Запуск проекта на исполнение

Наш простой пример реализован. Вполне естественно полюбопытствовать, будет ли он работать правильно. Проверять сделанную работу вообще полезно (сейчас автор мысленно обращается ко всем программистам). Для начала системе CoDeSys необходимо оценить его на наличие ошибок, критичных для выполнения алгоритма (помните, немного раньше мы о них уже говорили, когда разбирали окно сообщений на *рис. 1.8*). Кроме того, системе необходимо скомпилировать проект или, говоря проще, из нашего графического представления алгоритма получить машинный код, который в дальнейшем будет загружаться непосредственно в память ОВЕН ПЛК. Очевидно, что нули и единицы контроллеру гораздо понятнее наших прямоугольников. Для выполнения проверки и компиляции необходимо в главном меню CoDeSys выбрать уже знакомый Вам раздел «Проект», и далее найти в списке пункт «Компилировать все» (*рис. 1.31*).

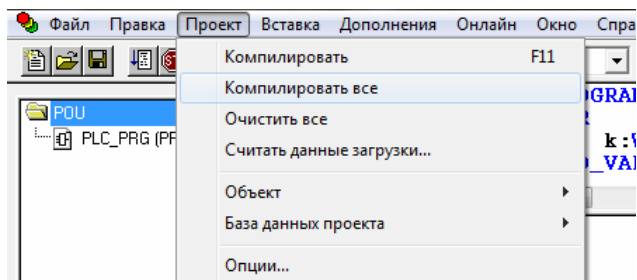


Рис. 1.31

После этого система произведет необходимый анализ и под рабочей областью в окне сообщений Вы увидите статистику проекта. Цифры могут несколько отличаться от приведенных на *рис. 1.32*, это нормально. Для нас с Вами важно, чтобы в самой нижней строчке было указано «0 ошибок». В этом случае можно запускать проект на исполнение.

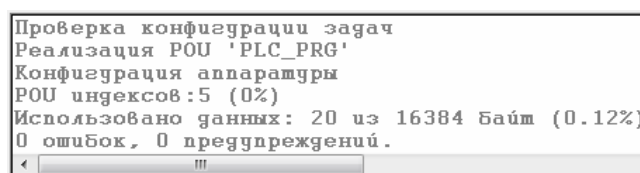
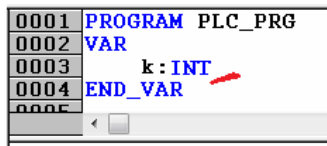


Рис. 1.32

Если система все же обнаружила ошибки, Вы можете просмотреть их список в окне сообщений. Можно дважды нажать ЛКМ на подсвеченные красным строчки и оценить, какие элементы в рабочем окне или в области определения окажутся выделенными. Например, ошибка на *рис. 1.33* говорит о том, что в области определения в конце

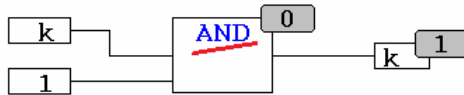
объявления переменной k не поставлена обязательная точка с запятой. Правильный вариант посмотрите на *рис. 1.16*.



Ошибка 4024: PLC_PRG (4): Должно быть ';' или ':=' перед 'END_VAR'

Рис. 1.33

В другом примере на *рис. 1.34* программа содержит вызов неправильной операции AND. В нашем примере мы должны использовать блок ADD для реализации сложения. Поэтому в заголовок блока необходимо внести исправления. После этого полезно еще раз скомпилировать проект, например, используя клавишу **F11**. Подчас система одно ошибочное действие описывает несколькими сообщениями. Поэтому пусть Вас не удивляет, что после одного исправления пропадают несколько красных строк.



Ошибка 4345: PLC_PRG (0): Несоответствие операнда 1 в 'AND': невозможно преобразовать 'INT' в 'ANY_BIT'

Рис. 1.34

Чаще всего ошибки возникают из-за невнимательности, причем у абсолютного большинства пользователей, не исключая автора этих строк. Внимательность важна в любом деле. А нам с Вами полезно еще и научиться быстро находить и исправлять свои ошибки. Поэтому далее мы будем разбирать те из них, которые возникают чаще всего, так чтобы Вам было легко ориентироваться. Если все ошибки исправлены или если они не были допущены, после компиляции в окне сообщений мы имеем надпись:

0 ошибок, 0 предупреждений.

Рис. 1.35

Теперь можно запускать проект. Для этого в меню «Онлайн» мы выбираем пункт «Подключение» (*рис. 1.36*). Также можно использовать сочетание клавиш **Alt+F8** или иконку

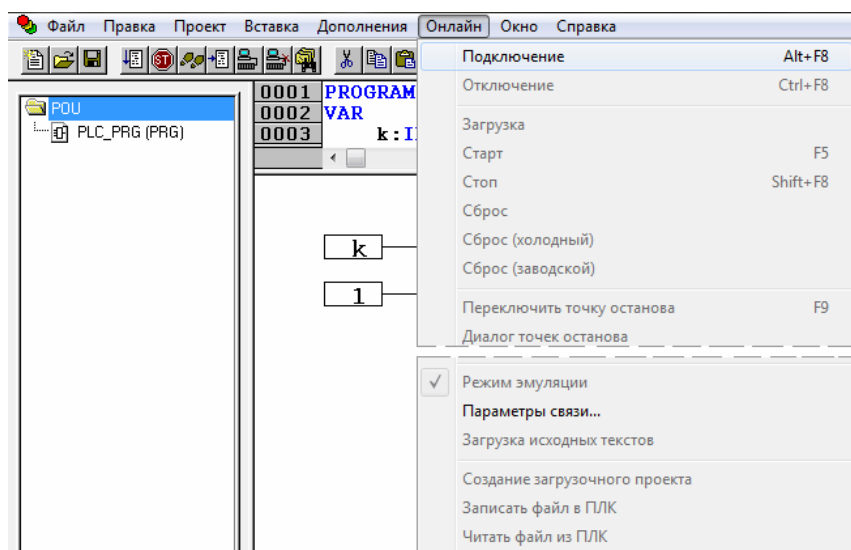


Рис. 1.36

Если появится сообщение с предупреждением (рис. 1.37), то необходимо нажать «ОК» и вернуться к исправлению ошибок.

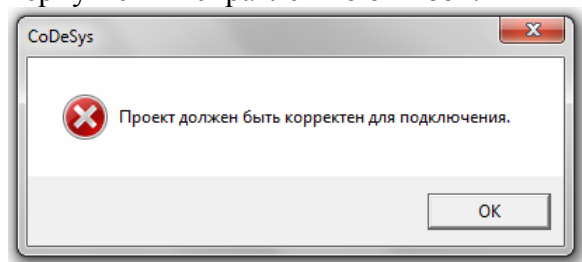


Рис. 1.37

Если все в порядке, то при запуске проекта на исполнение внешний вид системы немного меняется (см. рис. 1.38). В области определения мы видим переменную и ее текущее значение. Также значение переменной и результат работы блока видны на рабочей области. С этой точки зрения работа с языком CFC является очень удобной и наглядной, т.к. всегда понятно какие данные в какой части программы формируются и куда передаются. Кроме того, в программу добавляется операция RETURN. Система делает это автоматически для организации так называемого цикла работы ПЛК, о нем поговорим немного позже. Пока отнеситесь к появлению операции RETURN спокойно, она нам никак не мешает.

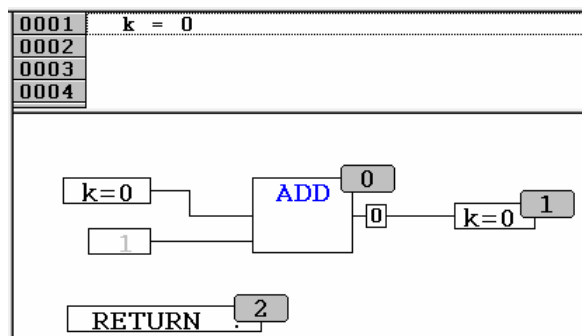


Рис. 1.38


До текущего момента система программирования находилась в **режиме редактирования**. Мы изменяли программу, определяли переменные. При этом в строке состояния, в правом нижнем углу экрана можно видеть надпись «Онлайн». Пока мы редактируем проект, эта надпись имеет неактивный, серый цвет, что является подсказкой. После выбора пункта «Подключение» в меню «Онлайн» система перешла в **режим исполнения**. При этом в строке статуса надпись «ОНЛАЙН» стала активной (см. *рис. 1.39*).




Рис. 1.39

В **режиме онлайн** (другое название режима исполнения) у нас уже нет возможности вносить изменения по ходу процесса. Зато здесь мы можем проверять наш алгоритм на соответствие заявленным требованиям, задавать различные значения переменных, останавливать и запускать процесс в нужные нам моменты.

У читателя может возникнуть вопрос, каким образом будет работать алгоритм, если к компьютеру не подключен контроллер. В данном случае мы имеем дело с одной из наиболее полезных и удобных особенностей CoDeSys - возможностью эмулировать работу алгоритма. Это происходит непосредственно на компьютере, без загрузки проекта в ОВЕН ПЛК. Это удобно, например, когда Вы еще не приняли окончательное решение о покупке конкретной модели контроллеров из линейки ОВЕН. Или если Вы хотите сначала попробовать свои силы в программировании, познакомиться со средой программирования, а уже потом приобретать то или иное необходимое Вам изделие. Скажем сразу, что эмуляция позволяет в достаточной мере изучить сами принципы программирования. Однако без учета особенностей той или иной модификации ПЛК, без использования физических входов и выходов, подключения модулей расширения или операторских панелей построить реально работающую систему затруднительно. В этой книге информация представлена таким образом, что Вы можете при обучении обойтись без ПЛК. Вместе с тем, дополнительная краткая информацию о работе непосредственно с контроллером включена в некоторые разделы. Мы верим, что ваш интерес к применению ОВЕН ПЛК будет расти по мере прочтения книги и изучения CoDeSys.

О том, что система запустила проект в эмуляции можно судить по той же строке статуса (см. *рис. 1.39*). Если надпись «ЭМУЛ.» активна, значит программа работает на вашем компьютере, а не загружена в контроллер. Пока мы разбирали первый пример, мы нигде не указывали системе данные о том ПЛК, который хотим использовать. Поэтому CoDeSys по умолчанию загрузил проект в эмуляции. В дальнейшем при работе с реальным устройством Вы сможете переключаться между эмуляцией и целевой платформой (выполнением алгоритма в ПЛК). Для этого перед выполнением операции «Подключение»  необходимо в меню «Онлайн» поставить или убрать галочку напротив пункта «Режим эмуляции» (см. *рис. 1.36*). Пока же мы продолжим работать в эмуляции.

Сразу после запуска проекта на исполнение система ставит его на паузу. Делается это с тем, чтобы пользователь был готов к тому моменту, когда алгоритм начнет работу. О постановке на паузу можно судить по строке статуса. Если надпись «Запущено» не активна, как, например, на *рис. 1.39*, система ожидает от нас команды на запуск. При этом Вы можете видеть, что значения переменной и выхода операции ADD неизменны и

равны 0. Для запуска работы программы необходимо зайти в меню «Онлайн» и выбрать пункт «Старт» (рис. 1.40). Однако в работе гораздо удобнее использовать клавишу **F5** или иконку . После этого надпись «Запущено» в строке статуса станет активной. А в рабочей области побегут цифры в переменной k.

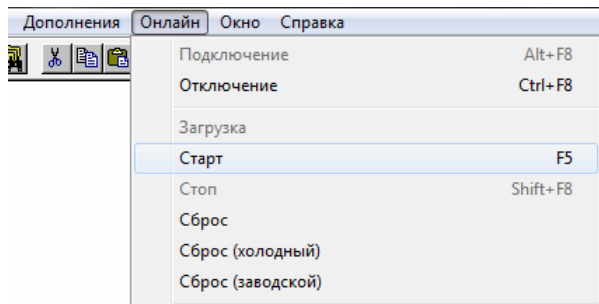



Рис. 1.40

ПЛК, равно как и его эмуляция в системе программирования, работают циклически. Это означает, что контроллер повторяет все описанные нами инструкции через определенные промежутки времени. В эмуляции это время равно приблизительно 50 мс. В ОВЕН ПЛК в зависимости от модификации и сложности алгоритма время цикла может составлять до 1 мс. Таким образом, когда мы описали операцию сложения и предложили системе записывать результат в переменную k, мы организовали постоянный рост значения этой переменной. Каждые 50 мс система забирает из k значение, прибавляет к нему единицу и снова записывает в k. На экране это выглядит, как постоянное увеличение числа. Этот процесс можно приостановить, если поставить выполнение алгоритма на паузу. Для этого можно выбрать в меню «Онлайн» пункт «Стоп» (рис. 1.41) или использовать сочетание клавиш **Shift+F8** или иконку .

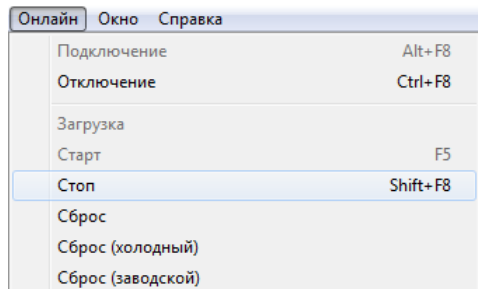


Рис. 1.41

После нажатия «СТОП» система останавливает расчет и оставляет на экране последние полученные результаты, то есть результаты последнего отработанного до конца цикла. Таким образом у нас появляется возможность отследить те изменения, которые произошли при работе алгоритма, оценить правильность его работы. Последующий запуск производится, как и в начале работы, через меню «Онлайн» и строчку «Пуск» или с помощью кнопки **F5**.

Подчас, в процессе работы может потребоваться вручную изменить те или иные значения. В нашей задаче, к примеру, можно обнулить переменную k. Для этого необходимо в области определения или на рабочей области дважды нажать ЛКМ на

имени переменной и в появившемся окне (см. *рис. 1.42*) в нижнем поле ввести новое значение, а затем нажать «ОК».

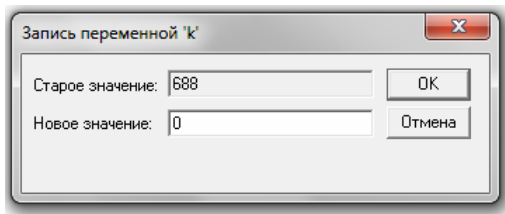


Рис. 1.42

После этого в рабочей области, справа от переменной голубым цветом будет отображаться новое значение (*рис. 1.43*). Для ввода нового значения вместо текущего необходимо выбрать в меню «Онлайн» пункт «Записать значения» или нажать сочетание клавиш **Ctrl+F7** (*рис. 1.44*). После этого значение переменной k обнулится.

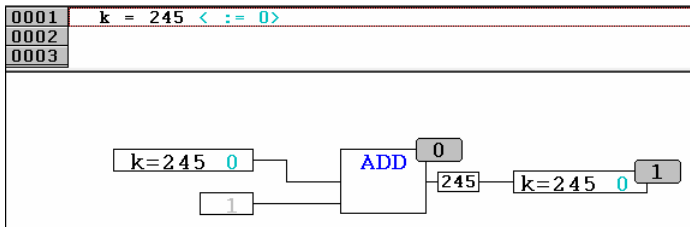


Рис. 1.43

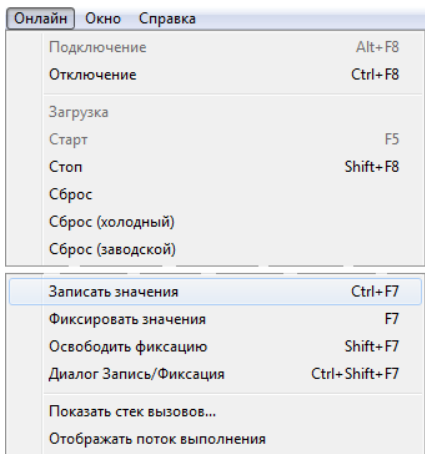


Рис. 1.44

Для того чтобы одновременно поменять несколько переменных, Вы сначала подготавливаете нужные значения. Эти значения голубым цветом отображаются в вашей рабочей области. А уже затем нажимаете **Ctrl+F7**, и выбранные переменные одновременно изменяются в программе.

К слову, если в процессе работы с режимом исполнения Вам необходимо обнулить значения всех переменных, т.е. сбросить их в начальное состояние, то для этого можно использовать пункт «Сброс» меню «Онлайн» (см. *рис. 1.44*). После выполнения этой операции программа автоматически встает на паузу. Об этом можно судить по

неактивной надписи «Запущено» в строке статуса (см. *рис. 1.39*). Далее, нажимая кнопку F5, мы вновь запускаем работу алгоритма с самого начала. Повторим еще раз одну важную мысль: если после запуска вашей программы на исполнение она не работает, то есть значения переменных не меняются, необходимо обратиться к строке статуса. Проверьте, **запущена ли программа**, и если она на паузе, **воспользуйтесь кнопкой F5**. В начале знакомства с CoDeSys такая глупая ошибка может появляться часто. Немного внимательности, и Вы легко с ней справитесь! Равно как и с освоением системы программирования!

После того, как мы написали, запустили и проверили наш первый проект, необходимо вернуться в режим редактирования. Это нужно, если Вы хотите внести дополнения или изменения в алгоритм. В процессе работы программы в эмуляции или на ПЛК сделать такие изменения не удастся. Поэтому для возврата к редактированию проекта мы выбираем в меню «Онлайн» пункт «Отключение».

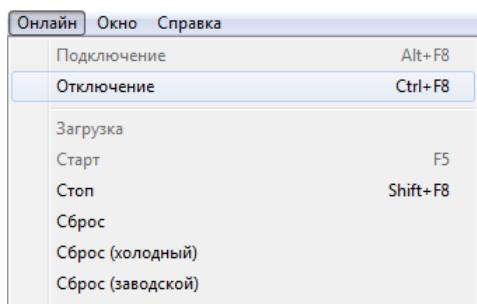



Рис. 1.45

Кроме того, для операции отключения можно использовать иконку  или сочетание клавиш Ctrl+F8. Окно вернется к прежнему виду, с которым мы привыкли работать при написании нашего первого примера. Теперь полезно сохранить проект и поблагодарить себя за хорошую работу. Начало положено!

Хотите знать больше?

Заходите на сайт www.кодесис.рф

Мы приготовили для вас много интересного!