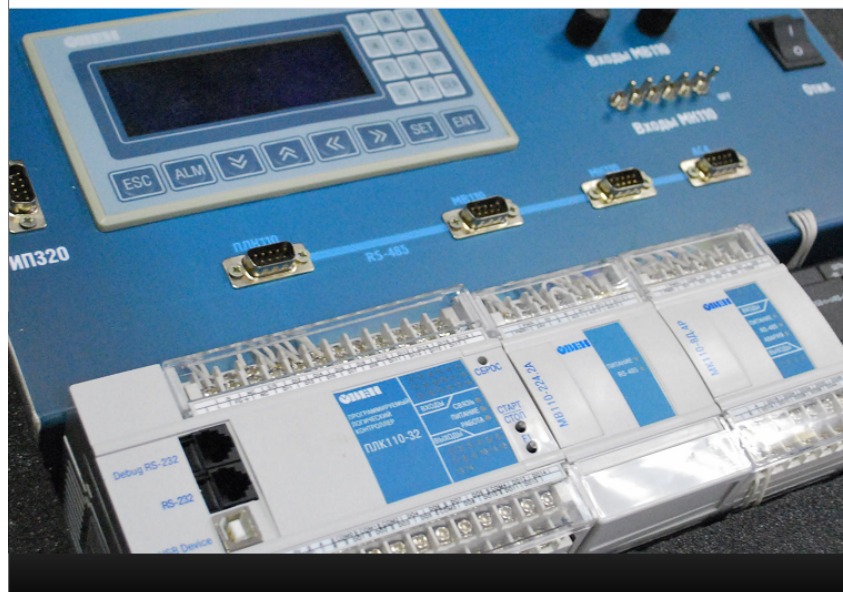


Гайнутдинов Кирилл

Простое и понятное программирование в CoDeSys

кодесис.рф



Часть третья

Простая логика

«Они готовят пушку! Зачем? А, да! Они будут стрелять!!!»

Капитан Смоллетт

м/ф «Остров Сокровищ»

Итак, друзья, Вы получили первые две части книги «Простое и понятное программирование в CoDeSys». В рамках рассылки, организованной на сайте www.kodecuc.pf, я обещал продолжать работать над книгой, точнее над ее третьей частью. Если Вы подписаны на рассылку, Вы наверняка уже посмотрели несколько видеороликов к этой главе. И, я надеюсь, попробовали использовать новый материал в своей работе. Я получил несколько очень приятных писем от читателей. Они указали на те моменты, которые можно сделать лучше. Большое спасибо за эти письма тем, кто откликнулся! Также участники выразили мнение, что формат книги с сопутствующими видео-уроками очень удобен. И просили не концентрироваться только на видео, а продолжать работу над текстовой версией. В результате появилась новая часть книги. Кстати, если Вы тоже хотите получать ссылки на видео уроки, подпишитесь на рассылку на странице <http://kodecuc.pf/wppage/kniga-codesys/>

В третьей части речь пойдет о логических операциях, то есть об операциях «И», «ИЛИ», «НЕ» (рис. 3.0).

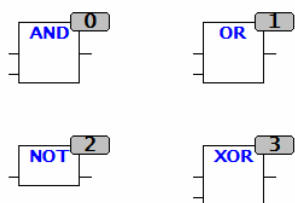


Рис. 3.0

Они составляют основу любых дискретных алгоритмов. На использовании этих логических операций построена работа многих шкафов автоматики и релейных схем. Такие решения до сих пор массово используются на значительном числе производств. Это один из советов читателям, куда нужно применить информацию, полученную на этих страницах. Представьте, огромный шкаф релейной автоматики Вы можете заменить одним или несколькими маленькими ПЛК. При этом перестанете постоянно следить за состоянием вечно ломающихся реле. Кстати, я подготовил дополнительный материал для тех, кто привык работать с релейными схемами (см. рис. 3.1). Разбор принципов работы с языком LD наверняка может быть интересным для таких специалистов. Видео-урок по этой теме придет к Вам в одном из писем нашей рассылки.

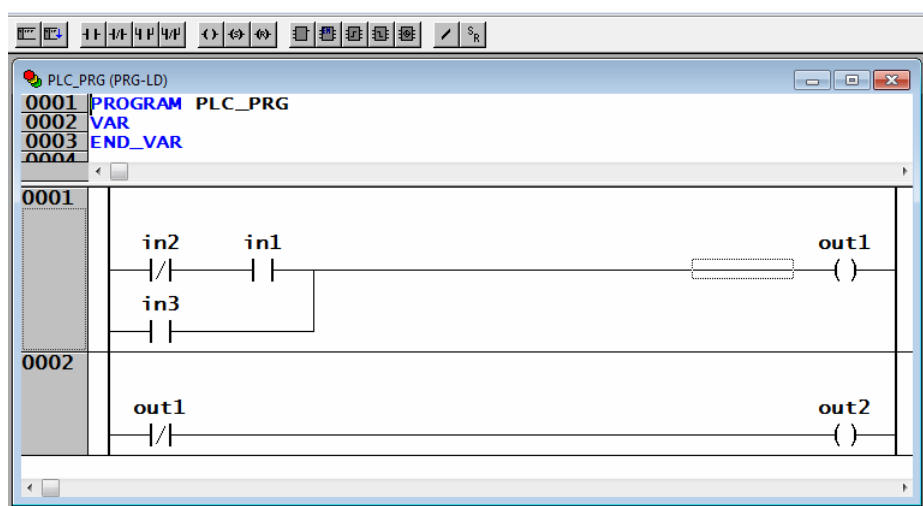


Рис. 3.1

Хорошо, давайте приступим к изучению материала данной части книги.

Напомню, с чего мы начинаем всякий раз работу с CoDeSys. Когда Вы загрузили систему программирования, Вы создаете новый проект через меню «Файл». Там необходимо выбирать пункт «Создать». В появившемся окне «Настройки целевой платформы» мы выбираем тот тип ПЛК, с которым будем работать. Это мы подробно разбирали во второй части (см. рис. 2.5 и рис. 2.6). Если что-то позабылось, пожалуйста, откройте нужный файл и освежите информацию в памяти. В своем случае я выбираю ПЛК150-И-М. У вас может быть любой другой контроллер ОВЕН ПЛК1xx, например ПЛК110 или 160. В данном случае это не так важно, нам подойдет любой из них. Мы продолжаем работать с языком функциональных блоков CFC. Именно этот язык выбираем при создании главной программы PLC_PRG (см. рис. 2.7).

После этого переходим на вкладку ресурсы, выбираем пункт «Конфигурация ПЛК». Ранее я уже упоминал, что работа ПЛК происходит циклически. В программе мы описываем действия для одного цикла, одного повторения. А дальше, после загрузки проекта в ПЛК, контроллер постоянно повторяет подготовленный нами алгоритм. Именно в этом разница между программированием приложений для персонального компьютера, и работой с задачами АСУ ТП. Итак, нам необходимо задать контроллеру время цикла его работы (время цикла ПЛК). То есть определить, с какой частотой необходимо повторять обработку программы. Для этого в окне «Конфигурации ПЛК» выделяем верхний пункт «PLC 150 I». У Вас, в зависимости от вашей модели ПЛК, могут быть другие обозначения. Затем переходим на вкладку «Параметры модуля», расположенную в правой верхней части экрана (рис. 3.2).

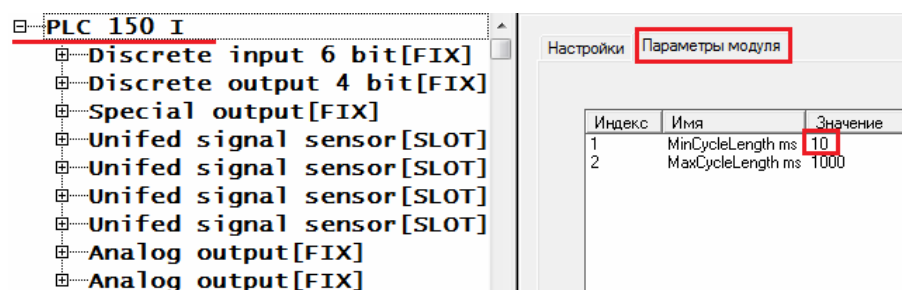


Рис. 3.2

По умолчанию, время цикла ПЛК1xx составляет 1 мс. То есть каждую миллисекунду ПЛК заново запускает обработку нашего алгоритма. И должен успеть ее завершить. Для тех простых программ, которыми мы занимаемся в рамках обучения, такая скорость обработки вполне

допустима. С другой стороны, в более серьезных задачах полезно задать некоторый запас производительности ПЛК. То есть указать время цикла, скажем, 10 мс. Для 80% задач, решаемых ПЛК1хх, этого будет достаточно. Возьмите себе за правило сразу же после создания проекта задавать время цикла 10 мс. Для этого в первой строчке, в поле «Значение» вместо 1 мс поставьте 10 мс (см. рис. 3.2). Это время цикла, за которое контроллер будет обрабатывать нашу программу.

Для того чтобы разобраться с работой дискретной логики, нам понадобится несколько входов и выходов ПЛК. Для этого Вы раскрываете элементы конфигурации ПЛК и задаете шесть дискретных входов. Пусть это будут переменные **x1**, **x2**, **x3** и т.д.

Напомню, для того, чтобы определить имя переменной для того или иного входа или выхода, Вы два раза щелкаете ЛКМ на надписи «АТ» в нужной строчке. Затем вводите имя переменной (английскими буквами и цифрами без пробелов), нажимаете «Enter» на вашей клавиатуре. Таким образом, у нас получилось 6 дискретных входов. По аналогии задаем четыре дискретных выхода. Пусть это будут переменные **y1**, **y2**, **y3** и **y4** (рис. 3.3).

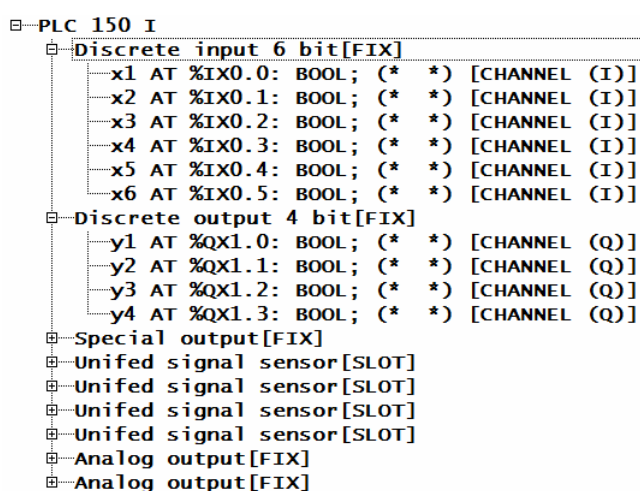


Рис. 3.3

Полагаю, что всё это должно быть достаточно понятно после прочтения предыдущих глав.

- Вы определились с target-файлом,
- создали главную программу и выбрали язык программирования,
- задали время работы цикла ПЛК,
- определились с необходимыми входами и выходами,
- **сохранили** проект на вашем компьютере.

Вернемся к нашей главной программе. Напомню, она находится на вкладке «POU». Переходим на нее и два раза щелкаем ЛКМ на программу «PLC_PRG» (рис. 3.4).

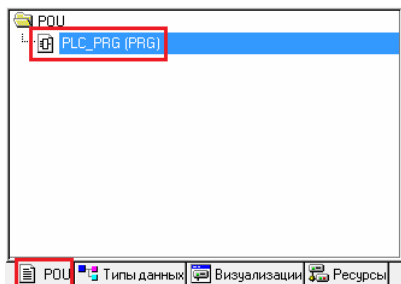


Рис. 3.4

Для того чтобы задать в вашем алгоритме ту или иную логическую операцию, поступаем следующим образом. В верхней части экрана, на панели быстрого доступа есть кнопка «Элемент» (рис. 3.5). Мы уже пользовались ей в первой части книги (см. рис. 1.24 и рис. 3.5). Нажимаем на нее, переносим курсор на рабочую область и нажимаем левой кнопкой мыши еще раз, устанавливая элемент в нужном месте рабочей области. По умолчанию блок, который мы добавили, будет выполнять операцию логического «И». Она задается ключевым словом «AND» (рис. 3.6). Аналогичный блок можно вызвать через контекстное меню, нажав в рабочей области ПКМ и выбрав пункт «Элемент».



Рис. 3.5

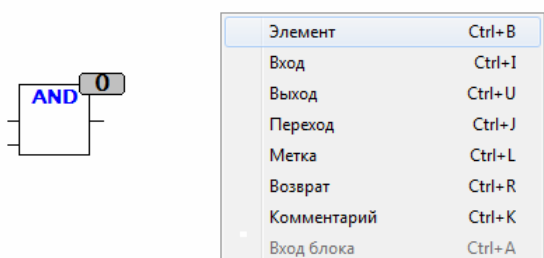


Рис. 3.6

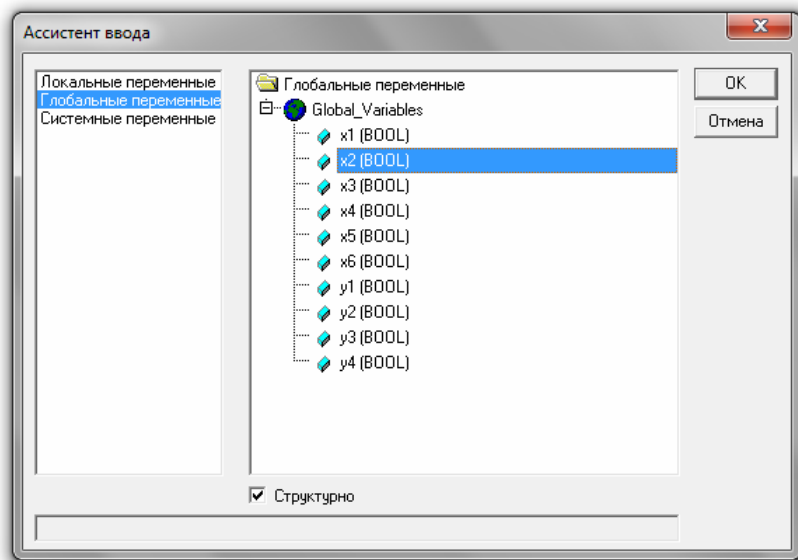



Рис. 3.7

На вход операции «AND» подаются переменные типа «BOOL». Как раз такими переменными являются наши дискретные входы. В нашей программе мы вызываем значение переменной, используя кнопку «Вход» (рис. 3.7). Можно это сделать через кнопку «F2» на вашей клавиатуре. В появившемся окне выбираем список глобальных переменных, и в нем находим нужные нам переменные **x1** и **x2** (рис. 3.7). Затем проводим линии связи.

На выходе операции логического «И» (блок AND) будет появляться сигнал TRUE или FALSE («ИСТИНА» или «ЛОЖЬ»). То есть значение типа BOOL. Это значение мы можем присвоить переменной того же типа, например, выходу **y1**. Для этого добавляем выход () на рабочую область. Вместо знаков вопроса прописываем имя переменной **y1**. Затем проводим линию связи между выходом операции AND и переменной **y1**. Результат Вы видите на *рис. 3.8*. Думаю, у Вас получилось то же самое.

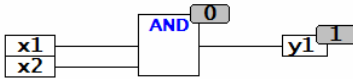


Рис. 3.8

Во время первых опытов работы с CoDeSys у Вас могут возникать некоторые сложности с изменением положения тех или иных элементов программы в рабочей области. Давайте разберем подробнее этот момент. Если Вам необходимо передвинуть один или несколько элементов программы в другое место, Вы выделяете нужную часть алгоритма, удерживая ЛКМ (см. *рис.3.9*). Постарайтесь сделать так, чтобы все элементы целиком попали в выделенную область. В результате, после отпускания ЛКМ один или несколько элементов окажутся выделенными пунктиром (*рис.3.10*)

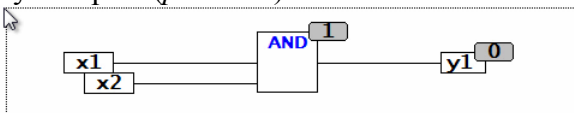


Рис. 3.9

Затем Вы нажимаете ЛКМ на выделенной области так, как это сделано на *рис.3.10*. Например, внутри блока. И удерживая ЛКМ, перетаскиваете сразу все выделенные объекты в нужную часть рабочей области. В видео уроке это показано более наглядно, если у Вас не все получается, пересмотрите его внимательно. Потренируйтесь, попробуйте несколько раз перетащить разные элементы вашего алгоритма. В итоге все будет отлично получаться!

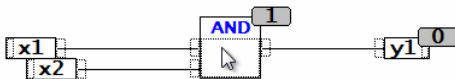


Рис. 3.10

Если Вам необходимо подобным образом поменять положение только одной переменной, тогда надо учесть следующее (см. *рис.3.11*): входы перетаскиваются за левую часть элемента, как это показано на примере **x3**. Выходы Вы переносите, «цепляясь» ЛКМ за правую часть элемента, например, как показано на *рис.3.11* для переменной **y3**. Надеюсь, с этим мы внесли полную ясность. Потренируйтесь!

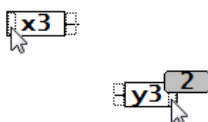


Рис. 3.11

Давайте посмотрим, какие еще логические операции можно добавить в наш проект. Если Вам необходимо поработать с операцией «ИЛИ» вместо «И», то сделать это можно следующим образом. Вы добавляете новый элемент, выделяете заголовок ЛКМ и печатаете новое название «OR», затем нажимаете «Enter» (*рис. 3.12*).

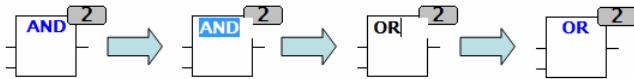


Рис. 3.12

Получили блок «ИЛИ». Принцип работы с ним абсолютно тот же самый. Мы вызываем в алгоритме значения дискретных входов ПЛК, на этот раз **x3** и **x4**. Затем подводим линии связи к нашему блоку OR. Результат работы этого блока мы будем передавать на второй дискретный выход контроллера, т.е. в переменную **y2**. На рис.3.13 изображено то, что должно было у Вас получиться. Не так уж сложно, правда?

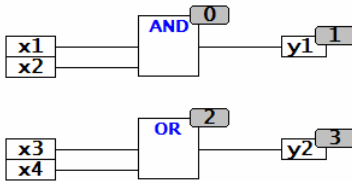



Рис. 3.13

Теперь поработаем с операцией «логическое НЕ» - блок NOT. Она, по сути, является той же самой инверсией, которая разбиралась в предыдущей главе . Сейчас мы разберем просто другой способ, как ее можно задать. Нажимаете кнопку «Элемент» или через ПКМ вызываете контекстное меню и выбираете пункт «Элемент». Таким образом создается новый блок и вместо надписи AND в нем задаете ключевое слово NOT. На выход операции NOT подключаете переменную **y3** (см. рис. 3.14).

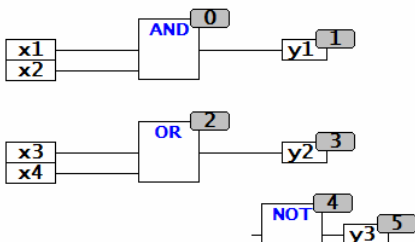


Рис. 3.14

Важный момент! С выхода одного и того же блока или из одной и той же переменной линию связи Вы можете вести на один или несколько возможных приемников этого самого сигнала. Поясню: из операции OR мы можем протянуть вторую линию связи на вход операции NOT. Для этого нажимаем ЛКМ на выход блока OR, т.е. на самый конец выходящей из него линии связи (рис. 3.15). Удерживая ЛКМ, перетаскиваем вторую линию связи на вход блока NOT (рис. 3.15), затем отпускаем ЛКМ. Полученный результат изображен на рис. 3.17. Таким же образом можно добавить и третью, и четвертую, и еще большее количество связей. Тут главное - не переусердствовать, во всем необходима мера. Конечно, более наглядно это видно в видео-уроке. Напомню, получить его можно, подписавшись на рассылку на сайте <http://kodecuc.pф/wppage/free-kurs/>.

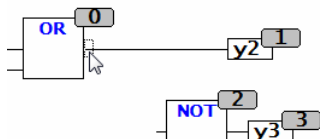


Рис. 3.15

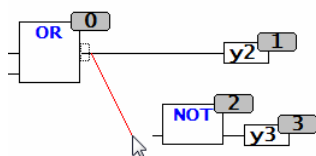


Рис. 3.16

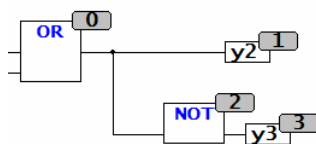


Рис. 3.17

Тем же способом вы можете значение одной и той же переменной передавать на несколько блоков. Пример приведен на *рис. 3.18*, *рис. 3.19* и *рис. 3.20*. Для того чтобы удалить ненужную линию связи, Вы выделяете один из ее концов (см. *рис. 3.18*). Затем нажимаете на клавиатуре кнопку Del (Delete).

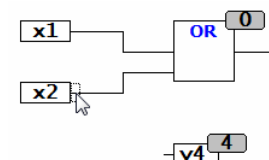


Рис. 3.18

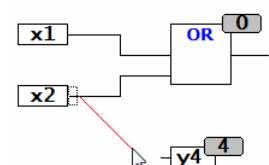


Рис. 3.19

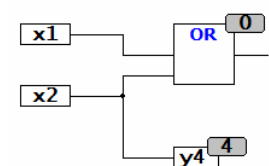


Рис. 3.20

Вернемся к нашему алгоритму (см. *рис. 3.21*). Теперь, когда на выходе операции OR будет значение FALSE, у нас с Вами на выходе **y3** будет появляться сигнал TRUE, в то время как в переменной **y2** он будет пропадать. И наоборот, когда на выходе OR появляется сигнал TRUE, он записывается в **y2**. В блоке NOT этот сигнал будет преобразовываться в FALSE и поступать в переменную **y3**. А в результате релейные выходы нашего ПЛК все время будут в разных состояниях.

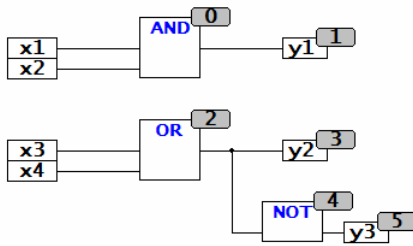


Рис. 3.21

Последний логический оператор в рамках этой главы - это «Исключающее ИЛИ». Он довольно часто используемый, например, для реализации проходных выключателей. «Исключающее ИЛИ» задается с помощью ключевого слова XOR.

На вход этой операции будем подавать значения переменных типа BOOL, например, переменные **x5** и **x6**. Для того чтобы использовать повторно какие-то готовые части уже написанного алгоритма, мы можем воспользоваться стандартными операциями «Копировать» и «Вставить». При добавлении блока XOR используем их. Для начала, выделим цепь с оператором OR, как показано на рис.3.22.

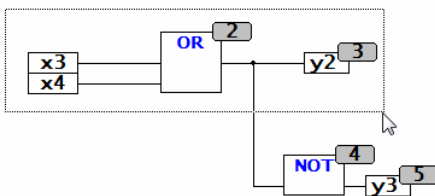



Рис. 3.22

Затем сверху, на панели быстрого доступа нажимаем кнопку «Копировать»  (рис.3.23). Можно также использовать хорошо знакомое многим сочетание клавиш **Ctrl+C**.

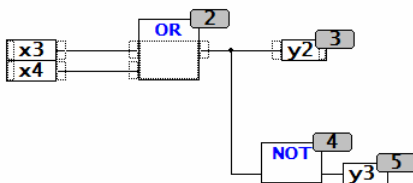



Рис. 3.23

Затем мы вставляем сделанную копию, для этого нажимаем кнопку «Вставить»  (рис.3.24). На экране появляется копия выбранной части алгоритма. Нажав и удерживая ЛКМ на выделенных частях появившейся копии, мы перетаскиваем ее на новое место в рабочей области (рис.3.25).

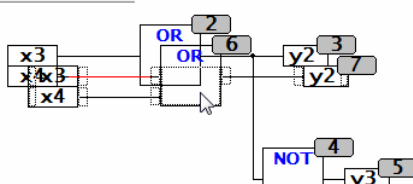


Рис. 3.24

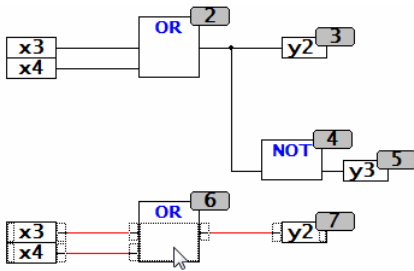


Рис. 3.25

Теперь нам необходимо поменять в заголовке элемента оператор OR на оператор XOR. В качестве входных данных мы будем использовать переменные **x5** и **x6**. Для замены имени переменной Вы просто нажимаете на соответствующий блок входа ЛКМ. Затем с клавиатуры вводите новое имя. Результат работы блока XOR будем передавать в переменную **y4**. Финальный вид нашего алгоритма представлен на рис.3.26.

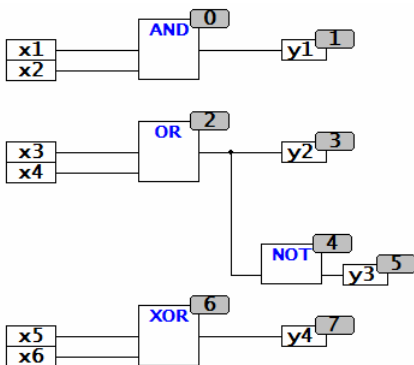


Рис.3.26

Суть операции XOR в следующем: когда значение переменных **x5** и **x6** будут неодинаковыми, например, один – FALSE, а другой - TRUE, в переменной **y4** будет сигнал TRUE. То есть выход **y4** будет включен. Когда же **x5** равно **x6** (либо обе переменные в состоянии «ЛОЖЬ», либо обе в состоянии «ИСТИНА»), тогда на выходе «ИСКЛЮЧАЮЩЕГО ИЛИ» будет значение «ЛОЖЬ».

Сейчас мы с Вами все это проверим, запустив программу непосредственно на ПЛК. Прежде чем это делать, еще несколько вещей, которые полезно себе записать и запомнить.

Перед загрузкой проекта на исполнение:

- Расставляем порядок обработки: нажимаем ПКМ на свободную область, выбираем «Порядок», а затем «В соответствии с потоком данных». Нумерация элементов, то есть порядок их обработки, станет более правильной. Подробнее это рассмотрено в первой части книги.
- Сохраняем проект (**Ctrl+s**).
- Запускаем проект на проверку, т.е. компилируем. Для этого в меню «Проект» выбираем пункт «Компилировать все», либо нажимаем клавишу **F11**. Внизу, в окне сообщений, мы должны увидеть, что ошибок у нас «0». То есть можно этот простой алгоритм запускать на исполнение. Если ошибки все-таки есть, вам необходимо их исправить и снова запустить компиляцию.

Здесь я специально не привожу иллюстрации, надеюсь, что материал предыдущих частей Вы изучили внимательно, и подсказки вам не нужны. В любом случае, всегда можно подглядеть в ранее изученные материалы. Я также надеюсь, что по материалам предыдущих глав Вы смогли настроить у себя один из каналов связи: либо по Ethernet, либо через COM-порт. Сейчас самое время этим каналом связи воспользоваться.

Мы заходим в меню «Онлайн», выбираем пункт «Подключение». Связываемся с нашим ПЛК, загружаем новую программу. Видим, что у нас внизу загорается надпись «ОНЛАЙН», это означает, что есть связь с контроллером. На самом ПЛК в этот момент должен загореться на светодиоде «СВЯЗЬ». Проверьте себя. Для того чтобы эта программа начала работать, нам необходимо снять ее с паузы. Для этого, напоминаю, в меню «Онлайн» выбираем пункт «Старт», либо нажимаем клавишу **F5**. Внизу появится активная надпись «ЗАПУЩЕНО» (рис.3.27).



Рис. 3.27

Сразу же начинает действовать блок NOT. На вход у него приходит значение «ЛОЖЬ», на выходе мы видим сигнал «ИСТИНА», который передается на соответствующий дискретный выход **y3** (см. рис. 3.28).

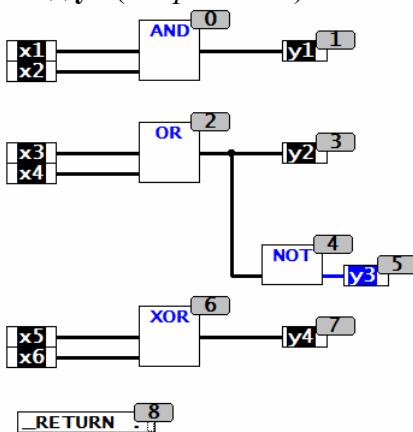


Рис. 3.28

Ну и теперь ваша задача заключается в том, чтобы просто проверить то, что Вы запрограммировали. То есть Вы подаете сигнал на переменную **x1**, на переменную **x2**, и у Вас появляется «ИСТИНА» в переменной **y1** (см. рис.3.31). То есть срабатывает первый выход. Если один из входных сигналов отключить, то в **y1** будет «ЛОЖЬ» (FALSE) (см. рис.3.29).

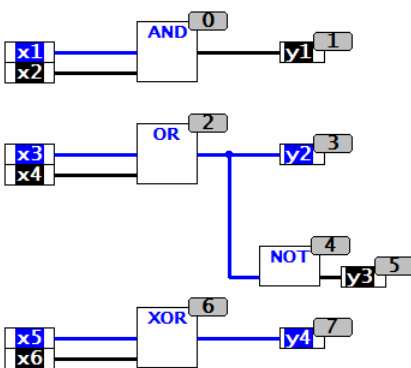


Рис. 3.29

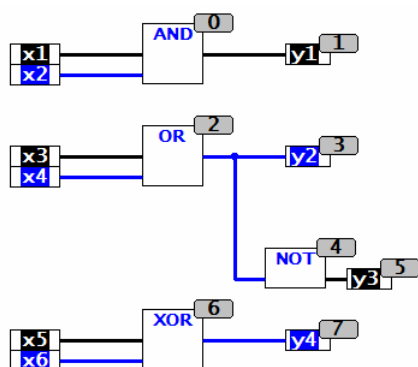


Рис. 3.30

В другом случае, когда у Вас появляется сигнал на третьем или четвертом входе, загорается выход **y2**, а выход **y3** гаснет, поскольку там стоит инверсия (см. рис.3.30). Если же появляется сигнал на четвертом входе, ситуация не меняется. Если пропадает сигнал на третьем входе, она опять не меняется. Когда оба входа, и **x3** и **x4** в выключенном состоянии, тогда сигнал на выходе операции «ИЛИ» пропадает (см. рис.3.28).

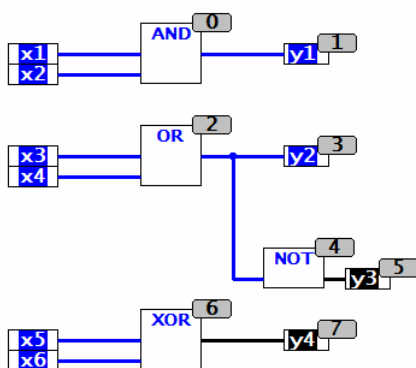


Рис. 3.31

Проверьте также работу блока XOR в соответствии с сигналами, представленными на рис.3.28 – 3.31. Попробуйте поработать с этой программкой, может быть, Вам захочется добавить сюда какие-то новые блоки, создать более сложную логическую схему.

Когда этот материал будет в достаточной степени освоен, через меню «Онлайн» Вы выбираете «Отключение», то есть разываете связь CoDeSys и контроллера. Сама программа в нем, заметьте, остается. Если сейчас на ПЛК Вы поработаете со входами, то выходы исходя из вашей программы будут переключаться.

В заключение этой части хочу показать одну полезную «фишку». Достаточно часто бывают ситуации, когда для операции AND или для операции OR Вам будет необходимо больше, чем два входных значения. Задать дополнительные входы можно следующим образом: Вы выделяете ЛКМ блок, в котором Вам необходимо добавить вход. Затем нажимаете на нем ПКМ, выбираете пункт «Вход блока». Появляется еще один вход (рис.3.32). Такого же эффекта можно добиться, выделив блок и нажав сочетание клавиш **Ctrl+A**. Если какой-то вход Вам не нужен, то выделяете его ЛКМ и нажимаете **Delete** на вашей клавиатуре (рис.3.33). Думаю, что с этим всё должно быть понятно.

Помните, что время от времени полезно сохранять свой проект!

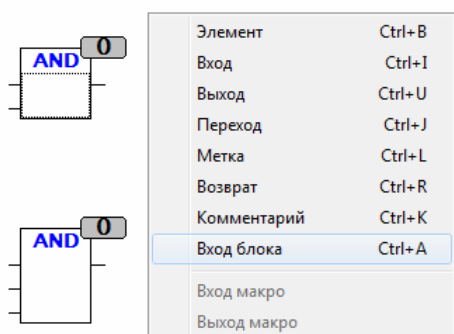


Рис. 3.32

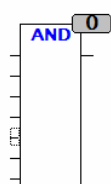


Рис. 3.33

Поздравьте себя! Вы прошли **самую сложную часть обучения** CoDeSys, поверьте мне! Вся прочая информация требует только вашего внимания и некоторого времени. Дальше Вы все сможете освоить самостоятельно. Заходите на форум на сайте компании ОВЕН (<http://www.owen.ru/forum>). Делитесь своими мыслями, задавайте вопросы, помогайте тем, кто пока еще не умеет того, чему смогли обучиться Вы.

Давайте придумаем с Вами вместе домашнее задание. Вы немного освоились с CoDeSys, Вы научились работать со входами и выходами ПЛК, настраивать связь с контроллером. Вы уже можете создавать пусть простые, но весьма разветвленные релейные алгоритмы, используя материал этой части. Придумайте себе два или три применения для вашей работы, где эти знания можно применить. Попробуйте создать свои собственные алгоритмы на основе той информации, которую почерпнули из этой книги. И напишите мне о них. Наверняка, несколько таких проектов будет интересно оформить в формате видео-урока. И я смогу их выложить для ознакомления всем участникам курса. Так, например, мы поступили в рамках нашей рассылки с примером работы простого электрического котла. Все интересное будет появляться на сайте www.kodecuc.pf.

Что делать, если есть желание продолжить работу в формате Интернет-курса? Вместе мы можем быстро и легко освоить работу с аналоговыми сигналами, библиотеками, временем и т.п. На большом числе простых примеров научиться и сразу начать применять новые знания.

Скоро будет запускаться **новый курс!**

Информация о нем обязательно придем всем, кто подписался на рассылку. Смотрите за новостями на сайте www.kodecuc.pf/internet-kurs/

Удачи вам, коллеги!

С уважением, Гайнутдинов Кирилл

p.s.

Хочу еще раз обратиться к вам с просьбой.

Если среди ваших знакомых есть люди, которым может пригодиться умение работать с CoDeSys, посоветуйте им подписаться на наш курс! Специалисты по КИП, электромонтажу, АСУ ТП, программисты, студенты – всем этим людям будет полезен CoDeSys и ОВЕН ПЛК. Спасибо Вам!