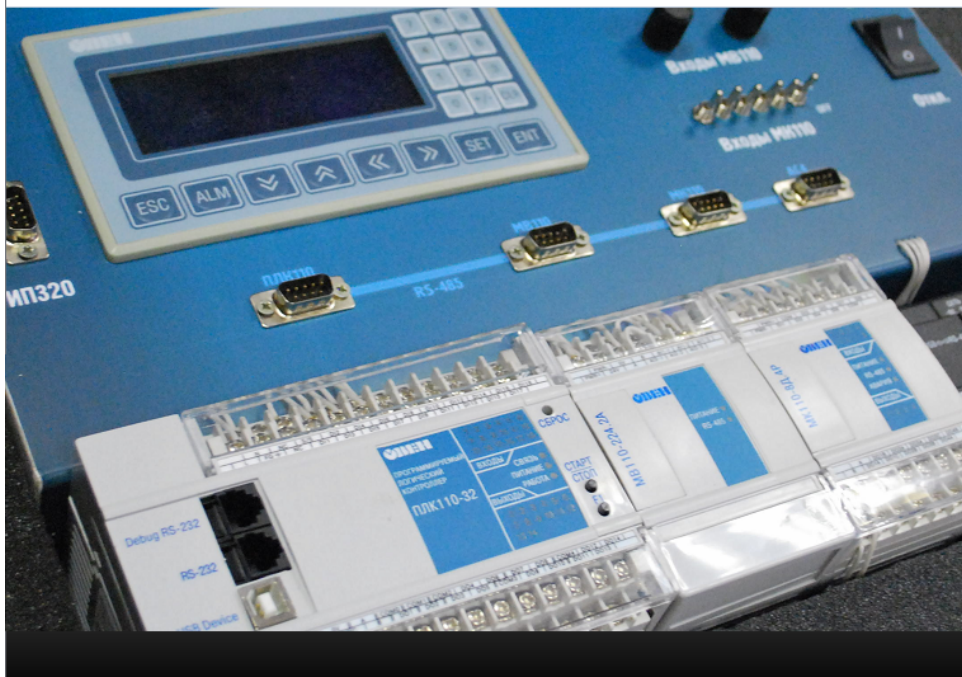


Гайнутдинов Кирилл

Простое и понятное программирование в CoDeSys

кодесис.рф



Часть вторая

ВХОДЫ И ВЫХОДЫ

*«Входит. И выходит. Замечательно выходит!»
Из любимого мультика.*

В предыдущей главе мы рассмотрели пример простой программы, принципы создания проекта, запуска его на исполнение. Для проверки того, как работает наше детище, мы использовали встроенную эмуляцию CoDeSys. Вместе с тем, едва ли этот пример, запущенный на компьютере, был действительно интересен. На экране компьютера изменяются какие-то числа – разве этим сейчас кого-то удивишь? Совсем другое дело, когда, нажимая на кнопки, переключая тумблеры, видишь, как реальное устройство начинает замыкать свои контакты в соответствии с тем, как было задумано.

Автор не один раз подмечал, как у участников семинара по программированию ОВЕН ПЛК в буквальном смысле слова загорались глаза при виде лампочек и кнопок, переключающихся по их программе. Увлеченность процессом, заинтересованность – важнейшие слагаемые успешного, быстрого обучения.

К чему это лирическое отступление? Мы начинаем новую главу, в которой разберемся, каким образом организовать работу с физическими входами и выходами ПЛК. Т.е. как информацию о состоянии объекта управления передать в программу через физические входы контроллера, и как из программы управлять объектом через выходы ПЛК. Это уже гораздо интереснее!

2.1 Target-файл

Прежде всего, необходимо разобраться с тем, как передать системе информацию о технических характеристиках целевой платформы, т.е. контроллера. Если говорить об ОВЕН ПЛК110, точнее о модификации ПЛК110-30.P-L, то необходимо выделить следующие важные для нас характеристики:

- 18 дискретных входов;
- 12 дискретных выходов (э/м реле);
- пользовательская память – 3 Мб;
- 2 интерфейса RS-232;
- 2 интерфейса RS-485;
- интерфейс Ethernet;
- интерфейс для программирования USB-device,
- встроенные часы реального времени.

Очевидно, что реальное количество характеристик значительно больше. Ведь системе программирования необходимо знать тип процессора, под который необходимо создавать программный код, объем доступной для использования оперативной памяти ПЛК и т.д. Словом, информации достаточно много. Прописывать вручную все это было бы затруднительно. Поэтому в CoDeSys принят следующий инструмент: компания производитель контроллеров, в данном случае – ОВЕН, предоставляет так называемые файлы целевой платформы, содержащие в себе всю необходимую системе информацию об используемом ПЛК. Эти самые файлы еще называют для краткости таргет-файлами (от английского target). Открывая коробку с приобретенным ОВЕН ПЛК, вы найдете в ней CD-диск. Он содержит дистрибутив CoDeSys, о нем мы говорили в разделе 1.1. Также на этом диске вы найдете те самые таргет-файлы, или если совсем коротко – таргеты. Наша с Вами задача теперь состоит в том, чтобы установить нужный файл на вашем рабочем компьютере. После этого мы просто выберем его при создании очередного проекта. Все довольно просто и быстро.

Перед установкой полезно закрыть все запущенные на вашем компьютере экземпляры CoDeSys. Затем, запустив CD и оказавшись в разделе «Содержание компакт диска», вы должны нажать кнопку «Программы» (см. рис. 2.1). Далее выбираем кнопку «Установить target-файл», а затем «Установить target-файлы версии 2.10».



Рис. 2.1

После этого на вашем компьютере будет запущена специализированная программа-мастер (см. рис. 2.2). Вам необходимо дважды нажать кнопку «Далее», а затем один раз «Установить». После этого мастер установит файлы целевой платформы для всех существующих на текущий момент модификаций OWEN ПЛК110 и ПЛК160. Их около десятка.

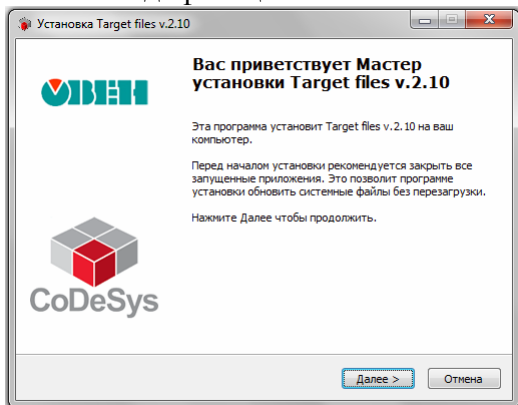


Рис. 2.2

Далеко не всегда пользователю необходимы все таргет-файлы. Реально используются обычно 2-3 модификации. Поэтому мы рассмотрим еще один способ установки таргет-файлов на ваш компьютер. Теперь вы сможете выбрать только те модификации, которые вам действительно необходимы. Для этого необходимо открыть содержимое вашего CD. В папке *targets\Версия 2.10* вы обнаружите несколько архивов с англоязычными наименованиями ПЛК. К примеру, рассмотренная выше модификация ПЛК110-30.P-L здесь будет иметь название PLC110.30_1 (рис. 2.3).

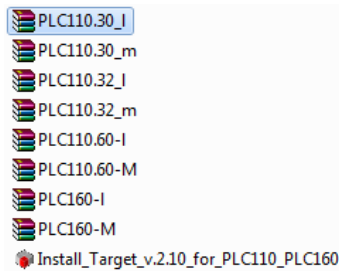


Рис. 2.3

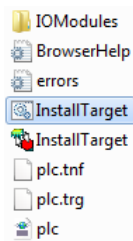


Рис. 2.4

Наша задача - разархивировать содержимое этого файла PLC110.30_1. В полученной при этом папке мы запускаем файл InstallTarget.bat. (рис. 2.4). После этого на экране на короткое время появится окно загрузки. Затем процедура установки будет завершена, необходимый файл будет установлен в соответствующие директории. Будьте внимательны к расширению *.bat. В папке также содержится файл InstallTarget.exe (рис. 2.5). Работа с ним подробно описана в руководстве по эксплуатации ОВЕН ПЛК. Поэтому здесь мы на этом останавливаться не будем.

2.2 Добавление таргет-файла в проект CoDeSys

После того, как файл целевой платформы установлен на компьютере, его можно добавить в проект. Таким образом, мы однозначно определяем, для какой модификации ПЛК будет создаваться в дальнейшем тот или иной алгоритм. Для начала надо создать новый проект. Для этого, напомним, в меню «Файл» необходимо выбрать пункт «Создать».

Самое первое окно, которое появляется при создании нового проекта, это окно «Настройки целевой платформы» – именно то, что нам сейчас нужно (рис. 2.5).

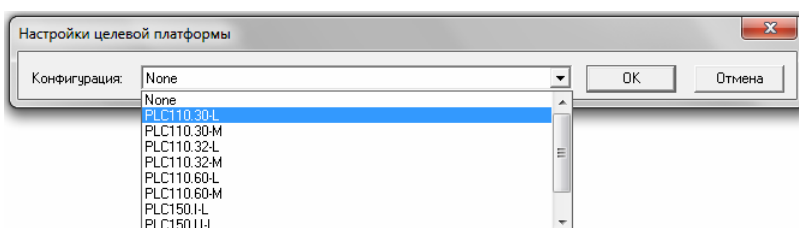


Рис. 2.5

Нажимаем ЛКМ на надписи «попе» и в раскрывающемся списке видим все те таргет-файлы, которые установили в предыдущем подразделе. Среди них выбираем PLC110.30-L. Именно с этой модификацией ПЛК мы будем работать в дальнейшем. Окно примет немного другой вид, откроется доступ к дополнительным настройкам. Однако нам пока все эти настройки не нужны, по умолчанию там все необходимое уже сделано. Нажимаем «ОК» (рис. 2.6).

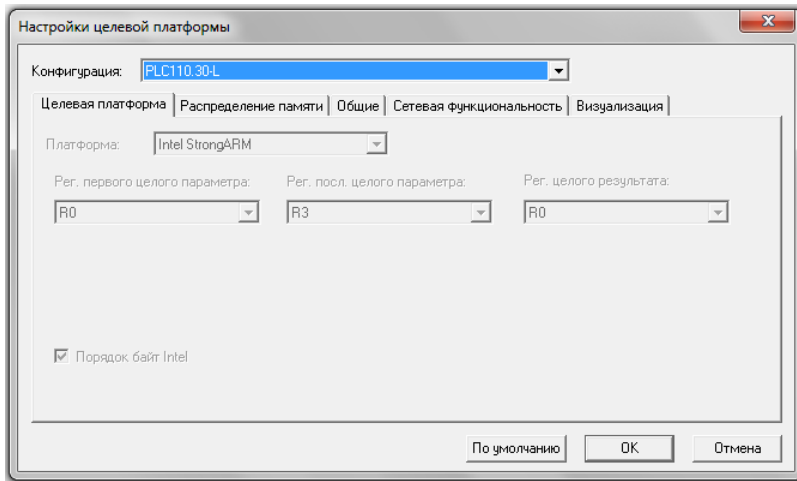


Рис. 2.6

В следующем окне (рис. 2.7) система программирования предлагает выбрать нам язык реализации. Можно выбрать язык функциональных блоков CFC согласно рисунку и нажать «ОК». Затем проект полезно сохранить (📁).

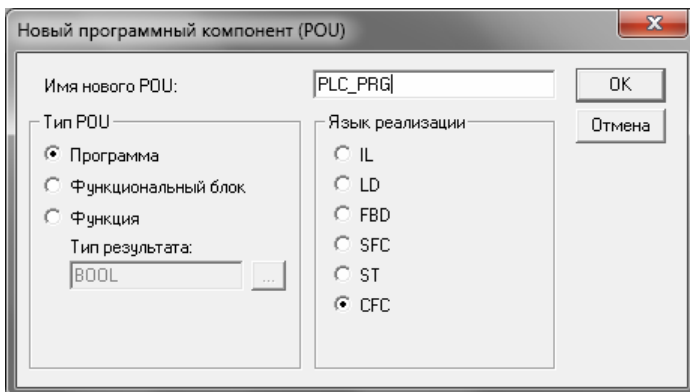



Рис. 2.7

2.3 Объявление переменных для входов и выходов

Практически все данные, которые используются при описании алгоритма, в CoDeSys задаются в виде переменных. Переменные имеют имя и тип данных, т.е. тип значения, которое хранится в этой переменной. Значения, которые ПЛК получает со входов и передает на выходы, также используются в программе в виде переменных. Наша задача – определить, какая переменная будет хранить значение того или иного выхода или входа. Определение переменных, связанных со входами и выходами (в дальнейшем мы будем называть их глобальными),

производится в ресурсе «Конфигурация ПЛК». Для этого в менеджере объектов мы переходим на крайнюю правую вкладку «Ресурсы» (рис. 2.8). Затем выбираем нужный пункт, дважды щелкая ЛКМ на надписи  Конфигурация ПЛК.

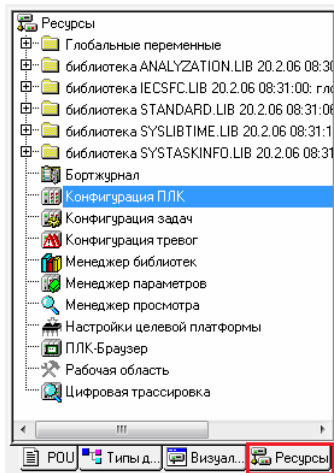




Рис. 2.8

Слева от менеджера объектов откроется окно конфигурации. Нажимаем ЛКМ на  рядом с надписью PLC110_30, таким образом раскрываем конфигурацию контроллера (рис. 2.9). Здесь представлены собственные входы и выходы ПЛК. Последовательно нажимая ЛКМ на  рядом с соответствующей группой, Вы можете видеть всю конфигурацию, включающую в себя скоростные и обычные дискретные входы (рис.2.10), скоростные и обычные дискретные выходы (рис.2.11).

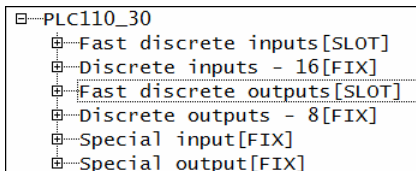


Рис. 2.9

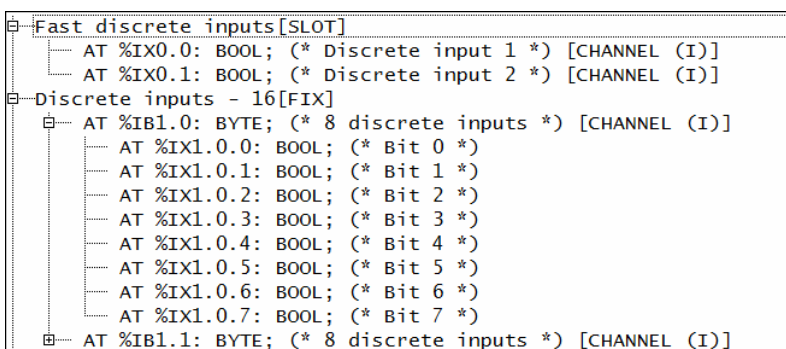


Рис. 2.10

```

Fast discrete outputs[SLOT]
├── AT %QX2.0: BOOL; (* Fast discrete output 1 *) [CHANNEL (Q)]
├── AT %QX2.1: BOOL; (* Fast discrete output 2 *) [CHANNEL (Q)]
├── AT %QX2.2: BOOL; (* Fast discrete output 3 *) [CHANNEL (Q)]
├── AT %QX2.3: BOOL; (* Fast discrete output 4 *) [CHANNEL (Q)]
└── Discrete outputs - 8[FIX]
    ├── AT %QB3.0: BYTE; (* 8 discrete outputs *) [CHANNEL (Q)]
    ├── AT %QX3.0.0: BOOL; (* Bit 0 *)
    ├── AT %QX3.0.1: BOOL; (* Bit 1 *)
    ├── AT %QX3.0.2: BOOL; (* Bit 2 *)
    ├── AT %QX3.0.3: BOOL; (* Bit 3 *)
    ├── AT %QX3.0.4: BOOL; (* Bit 4 *)
    ├── AT %QX3.0.5: BOOL; (* Bit 5 *)
    ├── AT %QX3.0.6: BOOL; (* Bit 6 *)
    └── AT %QX3.0.7: BOOL; (* Bit 7 *)

```

Рис. 2.11

Необходимо подчеркнуть, что при выборе модификации ПЛК с релейными выходами (буква Р в обозначении соответствующей модификации, например, ПЛК110-30.220.P-L) скоростные выходы не предусмотрены. Однако структура конфигурации области ввода-вывода (мы ранее назвали ее конфигурацией ПЛК) остается аналогичной моделям, в которых быстрые выходы присутствуют. Другими словами, релейные выходы в пункте *Fast discrete outputs* работают с той же скоростью, что и выходы в группе *Discrete outputs*. Наверняка, Вы быстро с этим разберетесь.

Для того, чтобы использовать в алгоритме значение на том или ином входе или выходе, необходимо напротив соответствующей строчки в конфигурации ПЛК задать имя переменной. В дальнейшем именно это имя мы будем использовать при написании программы. Для задания имени необходимо дважды нажать ЛКМ на надписи АТ в соответствующей строчке, и в открывшемся небольшом окошке напечатать имя переменной. Имя должно быть записано латинскими буквами и цифрами в одно слово. На *рис. 2.12* мы подобным образом указываем имя переменной **in1**, связывая ее с первым скоростным входом ПЛК. После задания имени необходимо нажать Enter. Теперь в зависимости от наличия или отсутствия сигнала на первом входе ПЛК в переменной **in1** будет появляться значение «Истина» (**TRUE**) или «Ложь» (**FALSE**).

```

Fast discrete inputs[SLOT]
├── in1 | X0.0: BOOL; (* Discrete input 1 *) [CHANNEL (I)]
└── AT %IX0.1: BOOL; (* Discrete input 2 *) [CHANNEL (I)]

```

Рис. 2.12

Аналогичным образом необходимо задать еще три входа, один скоростной и два обычных, присвоив им имена **in2**, **in3** и **in4** соответственно (*рис. 2.13*). Обычные входы ПЛК110-30 разбиты на две группы по восемь входов. Вы раскрываете сначала блок *Discrete inputs*, а затем одну из групп *8 discrete inputs*. Уже внутри группы напротив нужных строк прописываете имена переменных **in3** и **in4**. Тем же способом задаются имена переменных для выходов ПЛК. Вам необходимо раскрыть блок *Fast discrete outputs*, нажав ЛКМ на символе \boxtimes . Затем, дважды нажав ЛКМ на буквах АТ, для трех первых выходов указать имена переменных **out1**, **out2** и **out3**. Сделанные изменения полезно сохранить. После этого конфигурация ПЛК должна принять вид, изображенный на *рис. 2.13*.

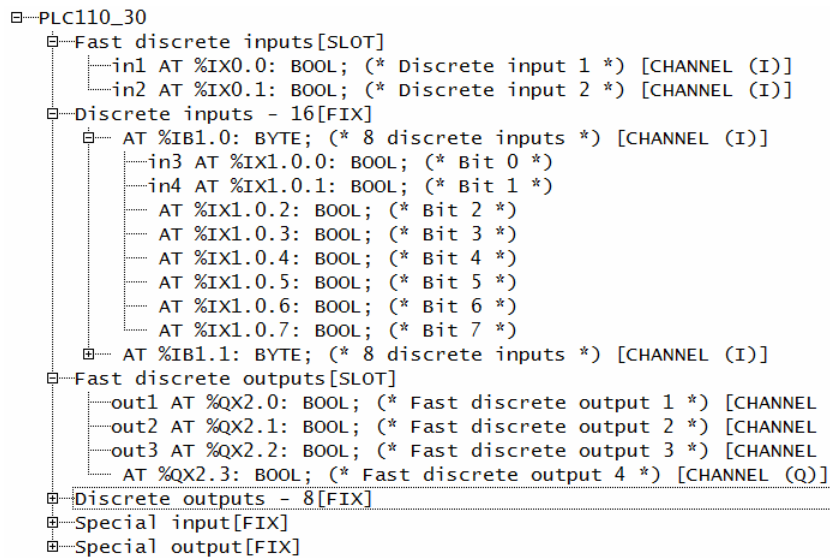



Рис. 2.13

2.4 Операции присваивания дискретных значений

Задав имена переменных для входов и выходов, определив эти переменные в проекте (в дальнейшем мы будем называть такие переменные глобальными), мы получаем возможность использовать входы и выходы ПЛК в какой-нибудь простой задаче. Например, задавать состояние выхода в зависимости от сигнала на входе.

Переходим к редактированию главной программы PLC_PRG. Для этого в менеджере объектов нажимаем ЛКМ на вкладке POU. Затем находим надпись  PLC_PRG (PRG) и дважды нажимаем на ней ЛКМ (рис. 2.14).

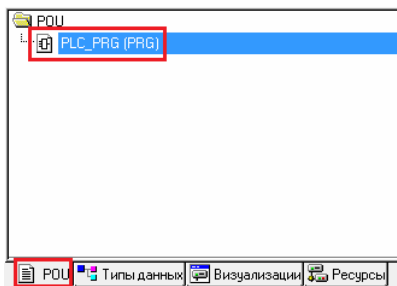




Рис. 2.14

Используя кнопки «вход» () и «выход» () или пункты контекстного меню, мы переносим на рабочую область заготовку алгоритма (рис. 2.15). Подробно о том, как это сделать, мы говорили в предыдущей главе, можно вернуться на несколько страниц назад и вспомнить. В нашем простом примере первый вход ПЛК будет управлять состоянием первого выхода. Второй вход будет воздействовать на второй выход. Что касается третьего выхода, то его состояние будет определяться сигналами на третьем и четвертом входах.

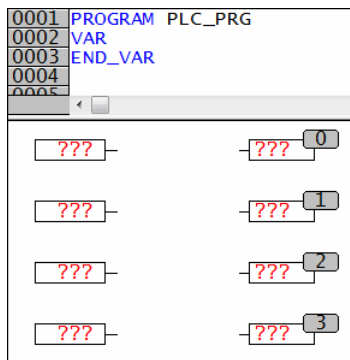


Рис. 2.15

Напомним, что при написании алгоритма вместо красных знаков вопроса нам необходимо подставить имена переменных. К примеру, в левом верхнем элементе мы будем вызывать значение на первом входе ПЛК, то есть значение переменной **in1**. Необходимо выделить содержимое элемента, нажав внутри него ЛКМ (L^{???}). Затем с клавиатуры вы задаёте имя переменной.

В случае если Вы хорошо помните названия своих переменных, объявленных в проекте, написать соответствующее имя будет не трудно. Бывают ситуации, когда имена переменных сложны и их много, есть шанс ошибиться в написании. Или бывает просто неудобно использовать клавиатуру, навыки работы с ней у всех нас на очень разном уровне. Словом, в случае, если вам неудобно печатать с клавиатуры, можно воспользоваться следующей очень удобной функцией CoDeSys. Вы выделяете содержимое элемента (знаки вопроса в нашем случае), а затем на клавиатуре нажимаете кнопку F2. На экране появляется окно «Ассистент ввода». Как только та или иная переменная объявлена в проекте, например глобальная переменная в конфигурации ПЛК (см. рис. 2.13) или локальная переменная в области определения (см. рис. 1.16), ее можно найти в ассистенте ввода. Нам необходима переменная **in1**, связанная с первым дискретным входом контроллера. В окне ассистента ввода, в левой части мы выбираем категорию «Глобальные переменные». В открывшемся справа списке выделяем ЛКМ нужную переменную и нажимаем «ОК» (рис. 2.16). После этого окно ассистента закрывается, а в нужном Вам месте появляется необходимая переменная. Остается только нажать Enter на клавиатуре. Согласитесь, это удобно и просто! Потренируйтесь вызывать окно «Ассистента ввода» кнопкой F2. Задайте имена переменных для четырех верхних элементов в соответствии с рис. 2.17 и проведите соответствующие линии связи.

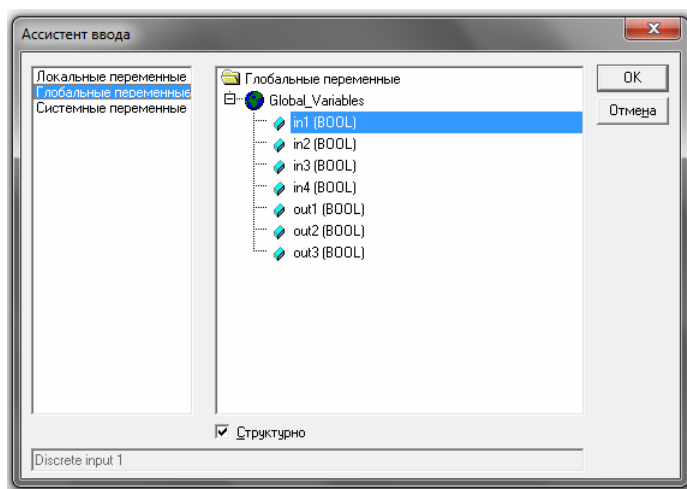


Рис. 2.16

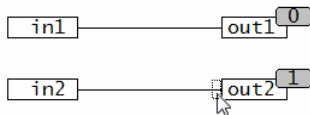



Рис. 2.17

Таким образом, при наличии сигнала на входе, то есть в переменных **in1** или **in2**, происходит срабатывание определенного выхода **out1** или **out2**. И наоборот – в отсутствии сигнала на входе выход остается разомкнутым. Давайте внесем разнообразие в этот алгоритм, добавив во вторую цепочку инверсию. То есть зададим включение второго выхода тогда, когда нет сигнала на входе. Для этого необходимо выделить ЛКМ край линии связи рядом с переменной **out2** (см. рис. 2.17). После этого сверху, на панели быстрого доступа мы нажимаем кнопку «Инверсия» . По-другому, можно нажать ПКМ на выделенном конце линии связи и в контекстном меню выбрать пункт «Инверсия» (рис. 2.18).

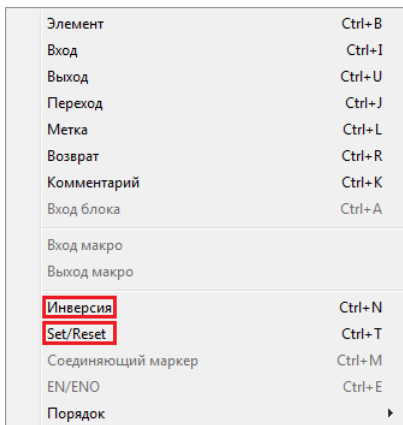



Рис. 2.18

В результате рядом с переменной **out2** появится небольшой круг (рис. 2.19). Он указывает на инвертирование (логическое отрицание) значения, приходящего по линии связи от переменной **in1**. Если приходит значение TRUE, то в выходную переменную записывается FALSE и наоборот. Последовательное нажатие на кнопку  позволяет добавлять и убирать инверсию на линии связи.

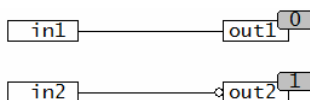


Рис. 2.19

Двигаемся дальше. Нам необходимо добавить еще две цепочки. В зависимости от состояния переменных **in3** и **in4** мы будем включать или выключать выходную переменную **out3**. Дополните ваш алгоритм вызовом этих переменных так, как это сделано на рис. 2.20.

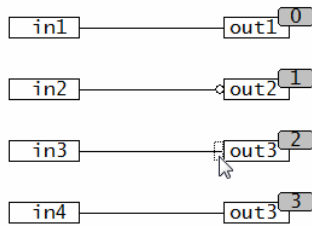









Рис. 2.20

Весьма часто в технологическом программировании, то есть при создании алгоритмов работы оборудования возникает следующая задача. Вам необходимо разделить условия включения какого-то исполнительного механизма и условия его отключения. К примеру, при управлении вентилятором условием его запуска может быть высокая температура в помещении. А условием остановки того же вентилятора будет окончание отсчета заданного времени. Для того, чтобы реализовать такие отдельные условия, в CoDeSys реализованы операции установки и сброса значения дискретной переменной. Наши входы и выходы ПЛК как раз объявлены, как дискретные. Таким образом, мы можем задать включение выхода **out3**, если есть сигнал на входе **in3**. Выключение **out3** возможно тогда, когда есть сигнал на входе **in4**. Для реализации этого алгоритма мы выделяем линию связи рядом с переменной **out3** (см. рис. 2.20). Затем находим на панели быстрого доступа кнопку  и нажимаем ее ЛКМ. Рядом с нашей выходной переменной появляется символ  (см. рис. 2.21). Теперь к переменной **out3** в данной цепочке будет применена операция установки. Если Вы еще раз нажмете на кнопку , то появится символ , то есть будет настроена операция сброса. При следующем нажатии  система вернет цепочку в состояние обычного присваивания, и дополнительные символы  и  будут автоматически убраны. Потренируйтесь переключать операции установки и сброса. Вместо использования кнопки на панели быстрого доступа Вы можете найти пункт Set/Reset в контекстном меню (см. рис. 2.18). Это меню, напомним, вызывается нажатием ПКМ, сделать это необходимо на выделенном кусочке линии связи. В итоге, у Вас должно получиться четыре цепочки, изображенные на рис. 2.21.

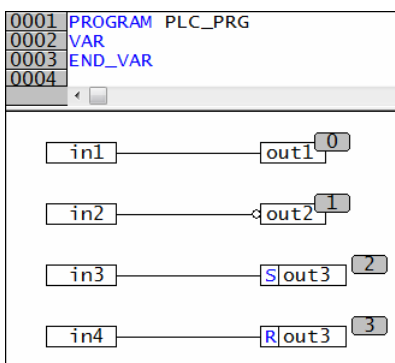


Рис. 2.21

Наш алгоритм готов. Его полезно сохранить, надеемся, что Вы помните об этом. Кроме того, Вам необходимо расставить порядок обработки (см. рис. 1.30). Затем произвести компиляцию вашего проекта, например, нажав кнопку F11, или сверившись с рис. 1.31. Если Вам не удалось избежать ошибок, исправьте их, пожалуйста. Затем, необходимо снова скомпилировать проект.

При работе с данным алгоритмом Вы могли случайно задать лишние локальные переменные. Убедитесь, что в области объявления переменных, между ключевыми словами VAR и

END_VAR у вас нет никаких записей (см. *рис. 2.22*). Если они все же есть, удалите их вручную.

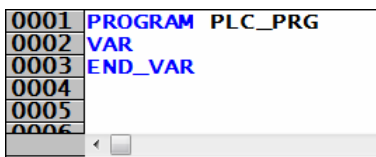


Рис. 2.22

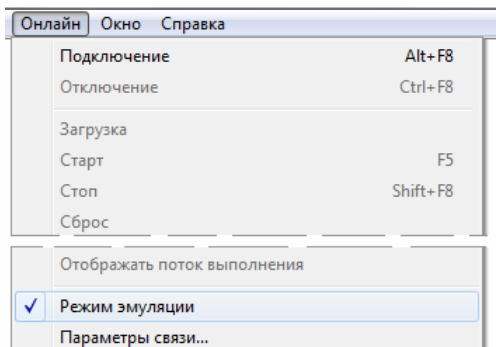


Рис. 2.23

Когда ошибки исправлены, необходимо включить режим эмуляции. Вам необходимо зайти в меню «Онлайн» и поставить галочку в пункте «Режим эмуляции» (см. *рис. 2.23*). В дальнейшем мы эту галочку уберем. А пока запускаем проект на исполнение («Онлайн» - «Подключение» или **Alt+F8**). Помним о необходимости запустить выполнение нашей программы, для этого нажимаем кнопку **F5**.

Внешний вид вашей программы должен быть похожим на то, что изображено на *рис. 2.24*. В CoDeSys принята следующая цветовая кодировка: черным цветом подсвечиваются те переменные, значение которых False («ложь»). Синий цвет используется тогда, когда в переменную записано значение True. Теми же цветами отображаются линии связи при отсутствии или прохождении через них сигнала. К примеру, в результате инверсии на выходе **out2** у нас появилось значение «истина».

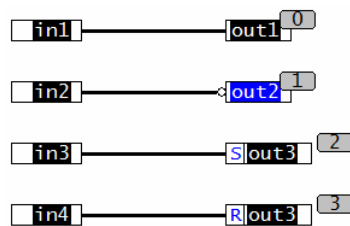


Рис. 2.24

Для того, чтобы изменить значения входных переменных и повлиять таким образом на состояние выходов мы используем уже знакомый способ. Т.к. в эмуляции работать с физическими кнопками и тумблерами мы не можем, то задаем значения непосредственно в окне CoDeSys. Необходимо дважды нажать ЛКМ на нужной переменной, чтобы поменять ее значение (см. *рис. 2.25*) и продолжать нажимать до тех пор, пока не появится то, что Вам нужно. Для записи значений в программу мы используем уже знакомую Вам комбинацию клавиш **Ctrl+F7** (см. также *рис. 1.44*).

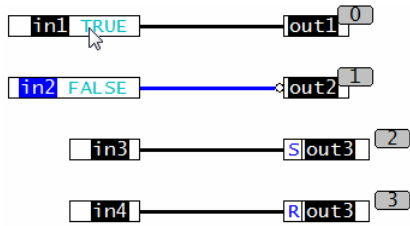


Рис. 2.25

Немного подробнее рассмотрим работу операций установки и сброса, это две наших нижних цепочки. Когда в переменных **in3** и **in4** значение False, переменная **out3** будет оставаться в том состоянии, в котором она находилась до выполнения этих операций. Т.е. она сохраняет свое значение (рис. 2.26 и рис. 2.27).

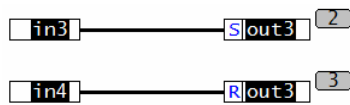


Рис. 2.26

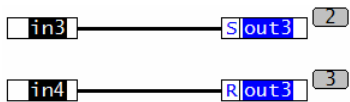


Рис. 2.27

Когда на входе **in3** появляется значение True, происходит включение выхода **out3** (рис. 2.28). В свою очередь когда на входе **in4** появляется значение True, происходит отключение выхода **out3** (рис. 2.29).

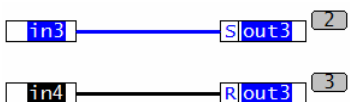


Рис. 2.28

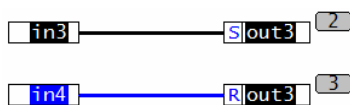


Рис. 2.29

Когда же на обоих входа присутствует значение «истина», необходимо оценить очередность выполнения операций. Здесь действует следующий принцип: последняя операция, произведенная с переменной, задает ее состояние в момент окончания цикла ПЛК. Мы уже говорили, что контроллер (и эмуляции тоже) работают циклично. И созданный нами алгоритм обрабатывается последовательно в соответствии с номерами операций. Порядок выполнения мы с Вами задавали перед запуском проекта на исполнение. В нашем проекте очередность выполнения цепочек идет сверху вниз. Таким образом, операция сброса у нас производится после операции установки. И значит, что последняя операция – это сброс. Поэтому на рис. 2.30 переменная **out3** имеет значение False и окрашена в черный цвет. Когда этот алгоритм будет загружен в ПЛК, контакты выходного реле контроллера, связанные с переменной **out3**, в

подобной ситуации будут разомкнуты. Надеемся, что эта особенность теперь Вам понятна. Проверьте самостоятельно, как работают операции установки и сброса.

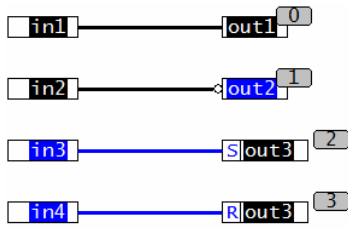


Рис. 2.30

По сути дела, последние две цепочки нашего алгоритма представляют собой реализацию обыкновенного RS-триггера, т.е. триггера с приоритетом сброса. Если мы поменяем местами порядок обработки последних цепочек (см. рис. 2.31) и поставим сброс перед установкой, то получим другое значение на выходе. Здесь мы реализовали SR-триггер, имеющий приоритет по сигналу установки. И при наличии значения «истина» одновременно на входах **in3** и **in4** выходная переменная останется включенной.

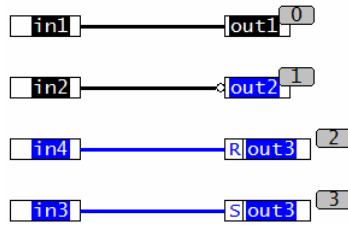


Рис. 2.31

2.5 Настройка связи с ПЛК и загрузка проекта в контроллер

Теперь давайте загрузим тот же самый алгоритм непосредственно в ПЛК и посмотрим, как он будет работать непосредственно на целевой платформе. Для начала Вам необходимо выйти из режима исполнения в рамках эмуляции работы проекта (**Ctrl+F8** или «Онлайн» - «Отключение»). Затем необходимо снять галочку напротив пункта «Режим эмуляции» в меню «Онлайн» (см. рис. 2.23). Теперь CoDeSys при последующих запусках проекта на исполнение будет стремиться подключиться к нашему ПЛК. Для того, чтобы это подключение прошло успешно, необходимо задать канал связи системы программирования и контроллера.

В комплекте поставки ПЛК есть кабель для программирования через порт RS-232, то есть через обычный СОМ-порт компьютера. Сейчас мы воспользуемся именно этим вариантом связи. О настройке соединения по другим интерфейсам Вы всегда сможете прочитать в руководстве по эксплуатации, которое тоже прилагается к комплекту поставки ОВЕН ПЛК. Для начала нам необходимо подключить физически наш контроллер к компьютеру. Нужный порт, обозначенный как Debug RS-232, находится на верхней крышке прибора (рис. 2.32).

Важно! Подключение кабеля к СОМ-порту компьютера Вы должны проводить только при отключенном питании ПЛК! Будьте внимательны!



Рис. 2.32

Для настройки связи в CoDeSys мы находим в меню «Онлайн» пункт «Параметры связи» и заходим в него (рис.2.33). В появившемся окне «Communication Parameters» (рис.2.34), в левой части располагается список настроенных для данного проекта каналов связи. В центральном окне видны настройки выбранного канала. Вероятно, у Вас там сейчас нет нужных нам настроек, поэтому мы будем создавать новый канал. Для этого в правой части окна «Communication Parameters» нажимаем кнопку «New». Открывается дополнительное окно (рис. 2.35).

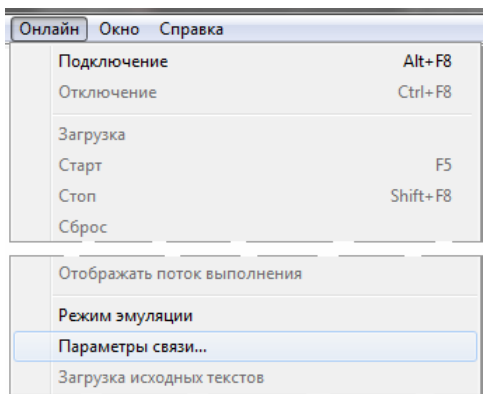


Рис. 2.33

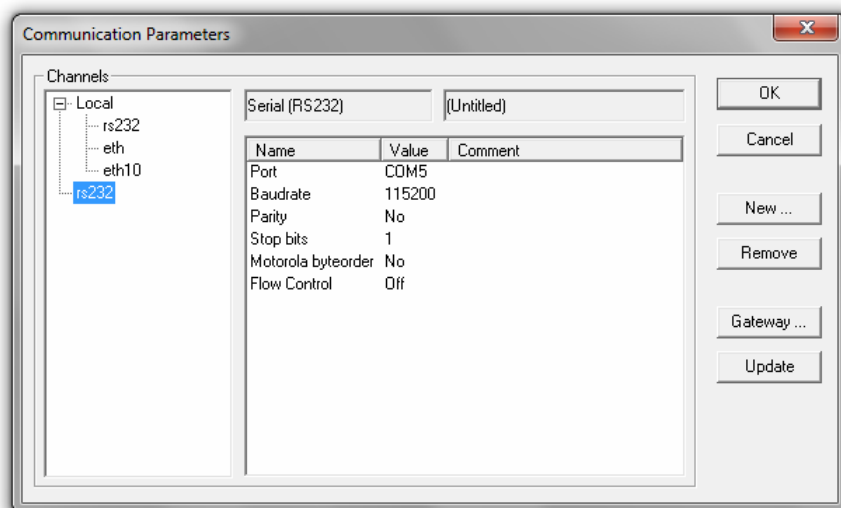


Рис.2.34

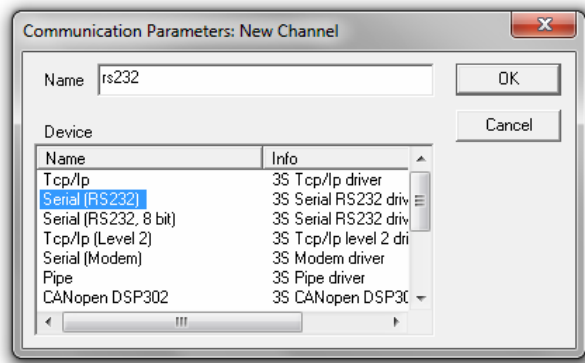


Рис.2.35

В этом окне, сверху в поле «Name» мы прописываем произвольное название нового канала, например, rs232. Теперь крайне важно из списка внизу выбрать правильный драйвер для работы. Нужный нам пункт называется Serial (RS232), он второй сверху. Выбираем именно его с помощью ЛКМ, как показано на рис.2.35. Затем нажимаем кнопку «Ок» и возвращаемся к работе с окном «Communication Parameters». Сейчас в левой его части появился наш вновь созданный канал rs232. Настроим его свойства в центральной части окна. Нам необходимо изменить значения для первой строки, здесь Вы должны указать номер того COM-порта, к которому подключили ПЛК, используя кабель из комплекта поставки. К примеру, у Вас используется COM5. Вы дважды нажимаете ЛКМ на текущей настройке COM1, затем с помощью кнопок «вверх» и «вниз» на клавиатуре вашего компьютера выбираете COM5. Если у Вас будет другой номер используемого порта, выставьте здесь именно ваши настройки. Можно также выбрать порт, последовательно нажимая ЛКМ на соответствующем поле. Во второй строке «Baudrate» Вам необходимо выбрать скорость соединения. Вне зависимости от модели контроллера мы всегда выбираем скорость 115200 б/с. Также, как и в верхней строке, Вы два раза нажимаете на текущей скорости 9600 и с помощью стрелок на вашей клавиатуре доводите скорость до нужного значения и нажимаете Enter. Все остальные настройки мы оставляем без изменений и нажимаем «Ок» для подтверждения.

Теперь нам осталось только запустить проект на исполнение (**Alt+F8**). Если все было сделано правильно, то связь будет установлена. В противном случае через несколько секунд CoDeSys выдаст сообщение об ошибке (см. рис. 2.36). Вам необходимо еще раз проверить, все ли Вы правильно настроили в соответствии с рекомендациями предыдущего параграфа или руководства по эксплуатации. Не забудьте включить питание вашего ПЛК и подождать 10-15 секунд, пока он перезагрузится :)

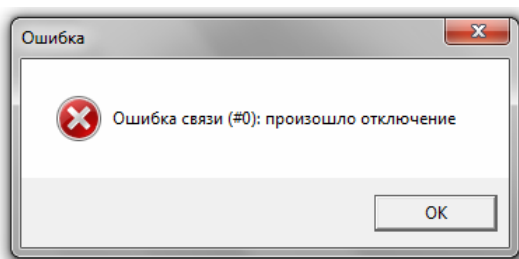


Рис.2.36

Итак, если Вы все сделали правильно, система выдаст Вам запрос на загрузку новой программы. Если в контроллере на текущий момент программ нет, то появился сообщение с рис. 2.37. Если Вы уже пробовали свои силы с данным ПЛК, может появиться сообщение с рис.

2.38. В любом случае, на любое из этих предложений Вы отвечаете согласием, то есть нажимаете кнопку «Да».

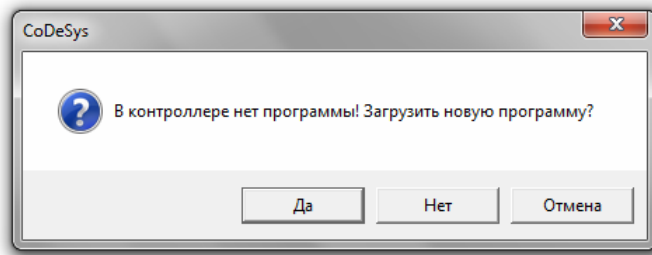


Рис.2.37

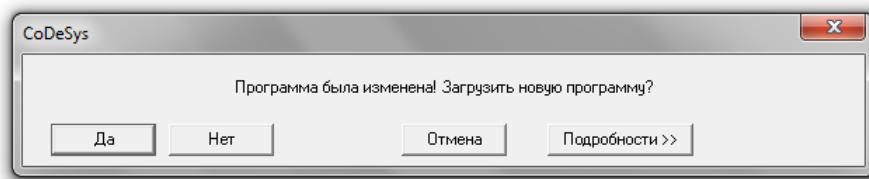


Рис.2.38

Ваш проект загружается в контроллер. По окончании этой короткой загрузки может появиться системное сообщение. Если это произойдет, в открывшемся окне вы просто нажимаете «Ок». И затем вспоминаете про кнопку **F5**, которая запускает вашу программу в работу. Внешний вид программы не отличается от того, что мы наблюдали в режиме эмуляции. А вот задание значений для наших переменных Вы теперь должны производить непосредственно с физических входов ПЛК. Для этого полезно заранее подключить к нему кнопки или переключатели, имитирующие сигналы. Логика работы алгоритма у нас осталась та же, мы подробно рассматривали ее несколько страниц назад. Ваша задача сводится к тому, чтобы получить удовольствие от достигнутого Вами результата. Отслеживать состояние физических входов и выходов ПЛК в процессе выполнения алгоритма Вы можете как из окна редактирования программы (см. *рис 2.24III*), так и из окна конфигурации ПЛК. Внешний вид конфигурации в режиме исполнения представлен на *рис. 2.39*.

```

-PLC110_30
  -Fast discrete inputs[SL0T]
    -in1 AT %IX0.0: BOOL; (* Discrete input 1 *) [CHANNEL (I)]
    -in2 AT %IX0.1: BOOL; (* Discrete input 2 *) [CHANNEL (I)]
  -Discrete inputs - 16[FIX]
    -AT %IB1.0: BYTE; (* 8 discrete inputs *) [CHANNEL (I)] = 1
      -in3 AT %IX1.0.0: BOOL; (* Bit 0 *)
      -in4 AT %IX1.0.1: BOOL; (* Bit 1 *)
      -AT %IX1.0.2: BOOL; (* Bit 2 *)
      -AT %IX1.0.3: BOOL; (* Bit 3 *)
      -AT %IX1.0.4: BOOL; (* Bit 4 *)
      -AT %IX1.0.5: BOOL; (* Bit 5 *)
      -AT %IX1.0.6: BOOL; (* Bit 6 *)
      -AT %IX1.0.7: BOOL; (* Bit 7 *)
    -AT %IB1.1: BYTE; (* 8 discrete inputs *) [CHANNEL (I)] = 0
  -Fast discrete outputs[SL0T]
    -out1 AT %QX2.0: BOOL; (* Fast discrete output 1 *) [CHANNEL (Q)]
    -out2 AT %QX2.1: BOOL; (* Fast discrete output 2 *) [CHANNEL (Q)]
    -out3 AT %QX2.2: BOOL; (* Fast discrete output 3 *) [CHANNEL (Q)]
    -AT %QX2.3: BOOL; (* Fast discrete output 4 *) [CHANNEL (Q)]
  -Discrete outputs - 8[FIX]
  -Special input[FIX]
  -Special output[FIX]

```

Рис. 2.39

В заключении данной главы хочется отметить одну важную мысль. Может показаться, что мы ничего особенного не сделали. Но это впечатление обманчиво. На протяжении двух глав Вы познакомились и, надеемся, освоили большой, действительно большой пласт информации. Быть может, это было самое сложное в освоении CoDeSys. И Вы отлично потрудились, сделав этот первый важный шаг!

Электронная книга, с которой вы сейчас работаете, пока еще в стадии написания. При появлении новых глав она обязательно будет обновляться. Если этот документ попал к Вам в руки от знакомых, или Вы случайно наткнулись на него в Интернете, возможно, Вы имеете не самую полную версию на текущий момент.

Получить актуальную версию и подписаться на рассылку, посвященную работе с CoDeSys и ОВЕН ПЛК, Вы можете на сайте www.кодесис.рф. В рамках данной рассылки мы планируем распространять обновления этого бесплатного издания. Кроме того, в ближайших планах проведение онлайн-обучения программированию в CoDeSys. Мы знаем, что часто специалисту трудно осваивать новое оборудование на семинарах с отрывом от производства. Для того, чтобы помочь решить эту проблему, мы разрабатываем курс для удаленного обучения. Подробности вы найдете на сайте проекта www.кодесис.рф.

С уважением, Гайнутдинов Кирилл

Проект «Кодесис.рф»