1 Запуск CoDeSys

CoDeSys запускается точно также как большинство Windows приложений:
Пуск -> Программы -> 3S Software -> CoDeSys V2.3 -> CoDeSys V2.3

2 Пишем первую программу

- **Задача:**

Контроль оператором движения некоторого механизма. Оператор должен периодически подтверждать правильность функционирования механизма. В противном случае, необходимо выдать предупреждение, а затем остановить работу.

Рабочий орган нашей машины совершает циклическое движение по периметру прямоугольника.

- **Создаем новый проект**

Это просто. Создайте новый проект командой **File -> New**.

- **Настройка целевой платформы (Target Settings)**

Проект является машинно-независимым, его можно опробовать в режиме эмуляции. Но давайте выберем для определенности конкретный контроллер. На страничке диалогового окна '**Configuration**' установите CoDeSys SP for Windows NT Realtime и подтвердите ввод – **Ok**.

- **Главная программа PLC_PRG POU**

Следующее диалоговое окно определяет тип первого программного компонента (**New POU**). Выберете язык реализации (**language of the POU**) **FBD** и сохраните предложенные по умолчанию тип компонента – программа (**Type of the POU Program**) и имя – **Name PLC_PRG**.

PLC_PRG это особый программный компонент (POU). В однозадачных проектах он циклически вызывается системой исполнения.

- **Объявляем Переключатель подтверждения**

Давайте начнем с переключателя подтверждения. Это переменная, которая будет изменять значение при подтверждении корректности работы механизма оператором.

В первой цепи графического FBD редактора выделите строку вопросов **???** и введите наименование нашей первой переменной. Пусть это будет **Observer** (наблюдатель). Теперь нажмите на клавиатуре стрелку вправо. В появившемся диалоге определения переменной сохраните наименование (**Name Observer**) и логический тип (**Type BOOL**). Измените класс переменной (**Class**) на глобальный (**VAR_GLOBAL**). Подтвердите определебние – **OK**. Теперь определение переменной **Observer** должно появиться в окне глобальных переменных проекта (**Global_Variables**):

```
VAR_GLOBAL
        Observer: BOOL;
END_VAR
```

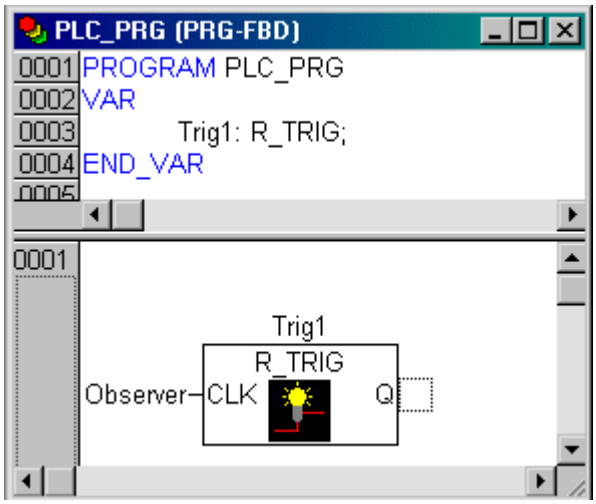
- **Детектор переднего фронта**

Оператор должен подтверждать работу именно переключением клавиши, а не просто спать с постоянно нажатой клавишей подтверждения. Чтобы разделить эти ситуации необходимо определить моменты нажатия и отпускания, т.е. переходы значения логической переменной их нуля (FALSE) в единицу (TRUE) и наоборот.

Вернитесь в окно редактора PLC_PRG и выделите позицию справа от переменной **Observer**. Вы должны увидеть маленький пунктирный прямоугольник. Щелкните по нему правой клавишей мыши. В контекстном меню ввода задайте команду **Box**.

По умолчанию, вставляется элемент **AND**. Воспользуемся ассистентом ввода: нажмите клавишу **F2**. В диалоговом окне (слева) выберете категорию: стандартные функциональные блоки (**Standard Function Blocks**). Из триггеров (trigger) стандартной библиотеки (standard.lib) выберете **R_TRIG**. R_TRIG формирует логическую единицу по переднему фронту на входе.

Необходимо задать имя для нового экземпляра функционального блока R_TRIG. Щелкните мышкой над изображением триггера и введите имя **Trig1**. В диалоге определения переменных должен быть указан класс **Class VAR** (локальные переменные), имя (**Name**) **Trig1** и тип (**Type R_TRIG**). Нажмите **OK**.



- **Детектор заднего фронта**

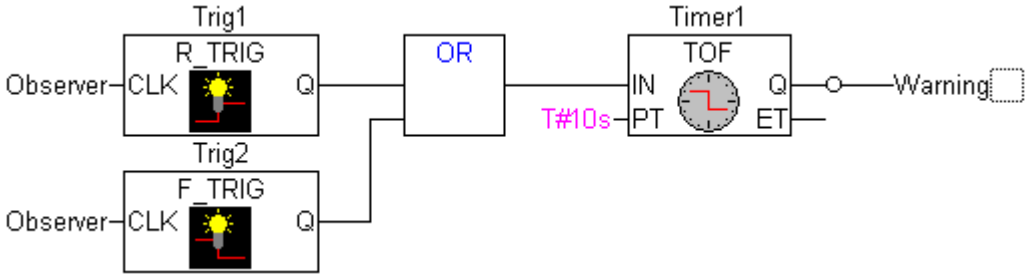
Выделите вход функционального блока **Trig**1 и вставьте (как было описано выше) элемент **AND** и переименуйте его в **OR** (логическое ИЛИ). Выделите свободный вход **OR** функционального и вставьте перед ним экземпляр функционального блока **F_TRIG** под именем **Trig2**. На вход **F_TRIG**, с помощью ассистента ввода (F2) подайте (категория **Global Variables**) переменную **Observer**.

- **Контроль времени, первая часть**

Вставьте после **OR** экземпляр функционального блока **TOF** (таймер с задержкой выключения) под именем **Timer1**. Замените три знака вопроса на входе **PT** константой **T#10s**. Она соответствует 10 секундам. Это время можно менять, в процессе отладки.

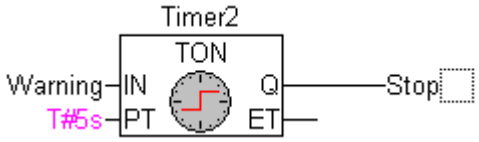
- **Выход Предупреждение**

Выделите выход Q таймера Timer1 и в контекстном меню (правая клавиша мыши) дайте команду Assign (присвоить). Замените вопросы на имя переменной **Warning**. В диалоге определения задайте класс **Class VAR_GLOBAL** и тип BOOL.
Теперь выделите позицию в середине линии соединяющей выход таймера и переменную **Warning**. Задайте команду **Negate** в контекстном меню. Маленький кружек означает инверсию значения логического сигнала.



- **Формируем Стоп Сигнал по второму интервалу времени**

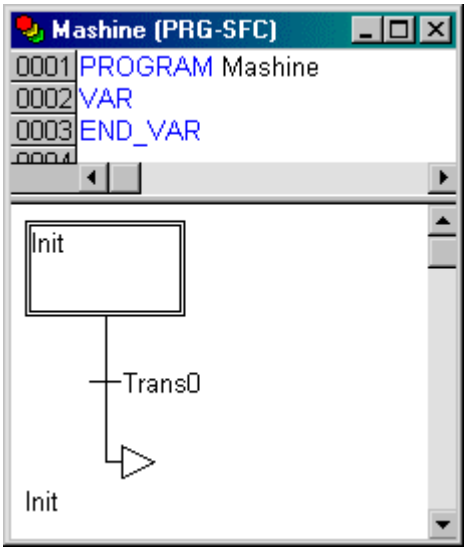
Создайте новую цепь командой меню **Insert->Network (after)**. Вставьте из стандартной библиотеки в новую цепь элемент (**Box**) типа **TON** (таймер с задержкой включения) под именем **Timer2**. Подайте переменную **Warning** на вход **IN** (используйте ассистент ввода <F2>) и константу **T#5s** на вход **PT**. Выход экземпляра функционального блока Timer2 присвойте (опять **Assign**) новой глобальной (Class VAR_GLOBAL) логической переменной **Stop**.



- **Вставляем POU управления механизмом**

В левой части окна CoDeSys расположен организатор объектов POUs (в нем присутствует PLC_PRG). Вставьте командой **Add object** в контекстном меню новый программный компонент с именем **Machine**, типом **Type program** и определите для него язык SFC (**Language SFC**).

По умолчанию, создается пустая диаграмма, содержащая начальный шаг "Init" и соответствующий переход "Trans0" заканчивающийся возвратом к Init.



** Мы будем далее использовать упрощенный SFC, без МЭК действий. Если справа от Init вы увидите прямоугольник с действием (Action), снимите в контекстном меню флаг **Use IEC-Steps** и переопределите заново POU Machine.*

- **Определяем последовательность работы механизма**

Каждой фазе работы должен соответствовать определенный этап (шаг). Выделите переход (Trans0) так, чтобы он оказался окружен пунктирной рамкой. В контекстном меню дайте команду вставки шага и перехода под выделенным: **Step-Transition (after)**. Аналогично повторите вставку еще 4 раза. Включая Init, должно получиться 6 шагов с переходами.

Щелкая мышью по именам переходов и шагов, вы заметете, что они выделяются цветом. Таким способом вы можете определить новые наименования.

Первый после **Init** шаг должен назваться **Go_Right**. Под ним **Go_Down**, **Go_Left**, **Go_Up** и **Count**.

- **Программируем первый шаг**

Щелкните дважды на шаге **Go_Right**. CoDeSys начнет определение действия шага и попросит выбрать язык его реализации (**Language**). Выберете **ST** (structured text) и перейдите в автоматически открытое окно текстового редактора. В этом шаге рабочий орган нашего механизма должен перемещаться по оси X вправо. Программа должна выглядеть так:

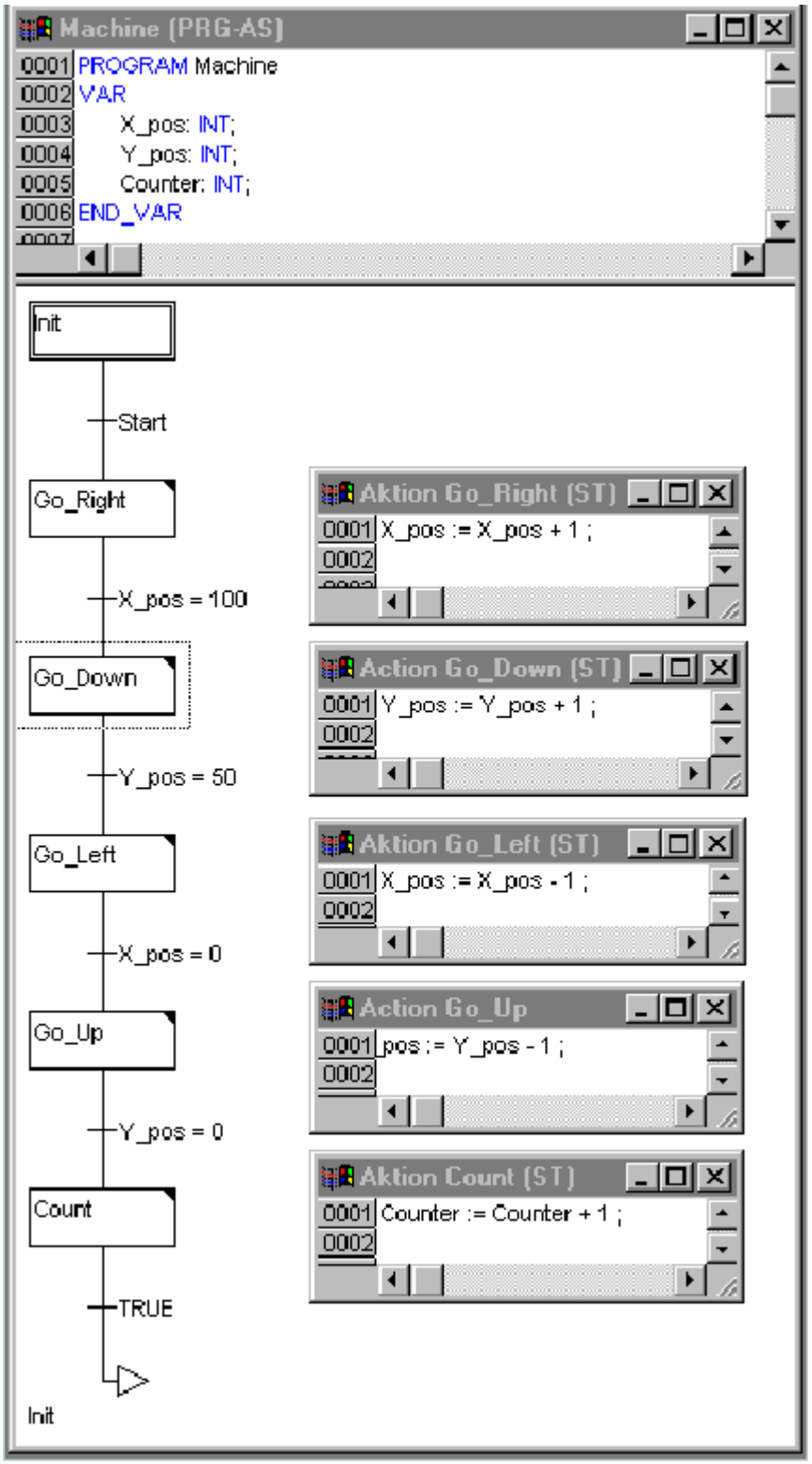
```
X_pos := X_pos + 1 ;
```

Завершите ввод клавишей Return, и определите переменную **X_pos** типа **INT** (целое).
Теперь верхний уголок шага должен быть закрашен. Это признак того что действие этого шага определено.

- **Программируем следующие шаги**

Повторите описанную последовательность для всех оставшихся шагов. Переменные **Y_pos** и **Counter** должны быть типа **INT**.

Шаг **Go_Down** программа **Y_pos := Y_pos + 1 ;**
Шаг **Go_Left** программа **X_pos := X_pos - 1 ;**
Шаг **Go_Up** программа **Y_pos := Y_pos - 1 ;**
Шаг **Count** программа **Counter := Counter + 1 ;**



- **Определяем переходы**

Переход должен содержат условие, разрешающее переключение на следующий шаг. Переход после шага Init назовите **Start** и определите новую логическую переменную (**Class VAR_GLOBAL** тип **Type BOOL**). При единичном значении этой переменной начинается цикл работы механизма.

Следующий переход должен содержать условие **X_Pos = 100**, так при значении позиции X включается следующая фаза движения.

Условие третьего шага **Y_pos = 50**,
четвертого **X_pos = 0**,
пятого **Y_pos = 0** и
шестого **TRUE** (переход разрешен сразу же, после однократного выполнения)

- **Останов механизма**

Вернитесь к **PLC_PRG** POU и добавьте третью цепь.

Вместо вопросов вставьте переменную **Stop**, и затем из контекстного меню вставьте оператор **Return**. Return прерывает работу программы PLC_PRG POU при единичном значении Stop.

- **Вызов POU управления механизмом**

Добавьте еще одну цепь, выделите ее и вставьте элемент **Box** из контекстного меню. Как обычно это будет "AND". Нажмите <F2> и в ассистенте ввода задайте POU управления механизмом в категории пользовательских программ (**User defined Programs category**).

- **Компиляция проекта**

Откомпилируйте проект целиком командой меню **Project->Rebuild all**, либо клавишей **<F11>**.

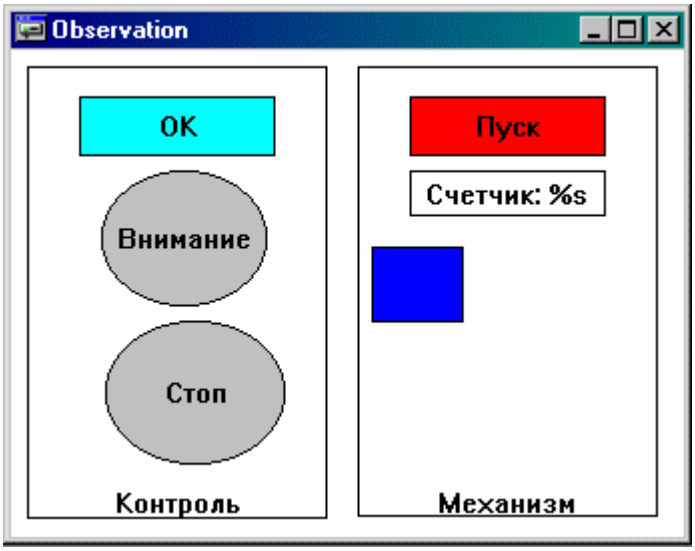
Если вы все сделали верно, то в нижней части окна должно появиться сообщение: **„0 errors„**. В противном случае необходимо исправить допущенные ошибки. В это помогут развернутые сообщения об ошибках.

3 Визуализация

- **Создаем визуализацию**

Третья страничка организатора объектов CoDeSys называется визуализация (Visualization). Перейдите на страничку визуализации. В контекстном меню введите команду добавления объекта **Add object**. Присвойте новому объекту имя **Observation**.

В конце работы, окно визуализации будет выглядеть так:



.

- **Рисуем элемент визуализации**

Давайте начнем с переключателя подтверждения (на рис. прямоугольник с текстом OK).

На панели инструментов выберете прямоугольник (**Rectangle**). В окне редактора визуализации нажмите левую клавишу мыши и растяните прямоугольник до нужной высоты и ширины, отпустите клавишу.

- **Настройка первого элемента визуализации**

Диалоговое окно настройки элемента вызывается двойным щелчком мыши на его изображении. Задайте в окошке содержимое (**Contents**) категории текст (**Text Category**) слово **OK**.

Теперь перейдите в категорию переменных (**Variables Category**), щелкните мышью в поле изменение цвета (**Change Color**) и воспользуйтесь ассистентом ввода **<F2>**. Вставьте переменную **.Observer** из списка глобальных переменных. Далее перейдите в категорию цвета (**Colors**). Задайте цвет закраски элемента (**Inside**), например, светло-голубой. Для «возбужденного» состояния необходимо определить другой цвет (**Alarm color**), например, голубой. В категории ввода (**Input Category**) необходимо еще раз ввести переменную Observer и поставить флажок **Toggle variable**. Закройте диалог настройки.

В итоге, прямоугольник будет отображаться светло-голубым при значении переменной Observer равном FALSE и голубым, при значении TRUE. Ее значение будет изменяться при каждом «нажатии» нашей клавиши.

- **Развиваем визуализацию**

Нарисуйте окружность **Внимание**. В настройках, **Text Category**, **Contents** задайте текст *Внимание*.

Colors Category, **Color** закраска **Inside** серым цветом, **Alarm color** красным цветом.

Скопируйте созданную окружность командой **Edit -> Copy** и вставьте ее один раз командой **Edit -> Paste**.

Поправьте настройки новой окружности **Стоп**:

- **Text Category, Contents** текст *Стоп*.
- **Variable Category, Color change** переменная **.Stop**

Нарисуйте прямоугольник для клавиши **Пуск**, имеющей следующие настройки:

- **Text Category, Contents** текст **Пуск**
- **Variable Category, Color change** переменная **.Start**
- **Input Category**, флажок **Toggle variable** включен, переменная **.Start**
- **Colors Category, Color** закраска **Inside** красным, и **Alarm color** зеленым.

Нарисуйте прямоугольник для счетчика со следующими настройками:

- **Text Category, Contents** текст **Счетчик: %s** (%s заместитель для отображения значения переменной)
- **Variable Category, Textdisplay** переменная **Machine.Counter**

Нарисуйте небольшой прямоугольник, обозначающий рабочий инструмент механизма, со следующими настройками:

- **Absolute movement Category, X-Offset** переменная **Machine.X_pos**
- **Absolute movement Category, Y-Offset** переменная **Machine.Y_pos**
- **Colors Category, Color** закраска **Inside** голубым цветом.

Если хотите, нарисуйте две декоративных рамки для разделения областей контроля и механизма. Задайте в них соответствующие надписи с выравниванием по низу (**Vertical** alignment **bottom**). Используя контекстное меню, поместите декоративные прямоугольники на задний план (**Send to back)**.

** Последующие пункты 4,5 есть смысл разбирать, только если вы имеете контроллер с CoDeSys. Мы рассмотрим подключение на примере CoDeSys SP RTE. В ином случае вы можете проверить работу примера в режиме эмуляции. Для этого переходите к пункту 6.*

4 Запуск целевой системы

Запустите систему исполнения (обратите внимание, что **CoDeSys SP RTE** работает только в Windows NT 4.0, Windows 2000 или Windows XP). Теперь в панели задач вы увидите иконку CoDeSys SP RTE. Щелкните по ней правой клавишей и дайте команду на старт системы (**Start System**).

5 Настройка канала и соединение

Если вы в первый раз подключаете контроллер к CoDeSys необходимо выполнить определенные настройки.

В меню **Online** откройте диалог **Communication parameters**. Нажмите клавишу **New** для настройки нового соединения. Желательно присвоить ему некоторое осмысленное имя.

В простейшем случае CoDeSys SP RTE работает на той же машине (компьютере) что и среда программирования CoDeSys. Это означает, что мы можем применить способ соединения посредством разделяемой памяти (**Shared memory (Kernel)**). Если контроллер расположен на другой машине сети, мы должны изменить параметр 'localhost' на имя машины или задать соответствующий IP адрес.

Настройка подтверждается клавишей **OK**.

6 Запуск проекта

Соединение с контроллером устанавливается командой **Online -> Login** из среды программирования CoDeSys. Если используется удаленное соединение, CoDeSys попросит вас подтвердить загрузку (download) кода проекта.

Команда запускает **Online -> Run** проект. Перейдите в окно визуализации и проверьте работу нашего механизма.

Для запуска проекта в режиме эмуляции установите флажок в меню **Online -> Simulation**. Далее переходите в режим *online* и запускайте проект, как описано выше.

7 Что дальше?

Теперь вы владеете базовыми приемами работы в CoDeSys и вполне можете попробовать реализовать собственные задачи. Необходимую информацию вы найдете в «*Руководстве пользователя по программированию ПЛК в CoDeSys»* (UserManual_V23_RU.pdf) и оперативной справочной системе CoDeSys.

Желаем вам успеха в работе!