Лабораторная работа №_

Среда программирования CoDeSys V 2.3.

Цель работы: подкрепление полученных первоначальных навыков работы в среде программирования CoDeSys V 2.3.

О**писание реализуемой задачи:** написать программу для программируемого логического контроллера (ПЛК), обеспечивающую управление устройством по следующему алгоритму: тележка передвигается по замкнутой тракетории в форме прямоугольника, имеющего стороны 100 и 50 м; перемещение осуществляется в направлении по часовой стрелке равномерными шагами длиной 1 м; движение тележки начинается после нажатия кнопки «ПУСК», циклическое прохождение по прямоугольной траектории бесконечно повторяется, если не произойдет его аварийная остановка; за движением тележки наблюдает оператор, от которого требуется непрерывный контроль работы устройства; с промежутком в несколько секунд оператор нажимает кнопку бдительности, тем самым подавая системе сигнал, что он находится во внимании на посту; в случае отсутствия нажатий кнопки длительности в течение 10 секунд система подает сигнал тревоги, который снимается при нажатии кнопки бдительности; если в течение 5 секунд сигнал тревоги не будет снят, система осуществляет аварийный останов тележки; чтобы оператор не мог обмануть систему, искусственно удерживая кнопку бдительности в нажатом состоянии, устройство должно реагировать на чередующееся нажатие и отжатие кнопки.

Реализация задачи:

1. Запуск CoDeSys осуществляется из меню ПУСК:

Пуск → Программы → 3S Software → CoDeSys V 2.3 → CoDeSys V 2.3

2. Пишем программу:
Создаем новый проект при помощи команды **Создать** из меню **Файл**.
Настраиваем целевую платформу, установив на страничке диалогового окна Configuration пункт CoDeSys SP PLC Windows NT Realtime и подтвердив ввод с помощью кнопки OK. Данное действие необходимо чтобы увязать машинно-независимый проект с конкретным ПЛК (в данном случае с программным эмулятором ПЛК под Windows).
Создаем главную программу PLC_PRG POU, в которой будет реализован алгоритм работы кнопки бдительности, включения тревоги и аварийного отключения устройства. Для этого в следующем диалоговом окне определяем тип нового программного компонента (New POU), выбираем конкретный язык реализации (language of the POU) FBD и сохраняем предложенные по умолчанию тип компонента — программа (Type of the POU Program) и имя — Name PLC_PRG.
PLC_PRG — особый программный компонент (POU); в однозадачных проектах он циклически вызывается системой исполнения.
Объявляем переключатель подтверждения (кнопку *бдительности*). В программе реальной кнопке соответствует булевая переменная (данный тип переменных может принимать всего два значения: 0 и 1 (правда и ложь)). Данная переменная изменяет своё значение при нажатии на кнопку ***бдительности***. Для её задания в первой цепи графического FBD редактора выделите строку вопросов **???** и введите название переменной. Пусть это будет **Observer** (наблюдатель). Нажмите 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 определение, нажав OK. Теперь определение переменной Observer должно появиться в окне глобальных переменных проекта (Global_Variables):

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

Определение момента нажатия кнопки *бдительности* (детектор переднего фронта, то есть перехода логического состояния из 0 в 1). Вернитесь в окно редактора PLC_PRG и выделите позицию справа от переменной Observer. Вы должны увидеть маленький пунктирный прмоугольник. Щелкните по нему правой клавишей мыши. В контекстном меню задайте команду Элемент. По умолчанию вставляется элемент AND. Воспользуйтесь 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. Данный элемент является детектором переднего фронта: он формирует логическую единицу на выходе при переходе из 0 в единицу на входе данного элемента. То есть R_TRIG выдаёт сигнал в момент нажатия на кнопку. Для функционального блока необходимо задать имя. Щёлкните мышкой над изображением триггера и введите имя, например, Trig1. В диалоге определения переменных должен быть указан класс Class VAR (локальные переменные), имя (Name Trig1) и тип (Type R_TRIG). После завершения ввода нажмите OK. Внимание: в большинстве случаев тип объектов и переменной средой CoDeSys верно устанавливается в режиме по умолчанию. Однако, в ряде случаев корректно установить тип можно только в ручную. Обращайте внимание при задании типов для переменных и объектов.

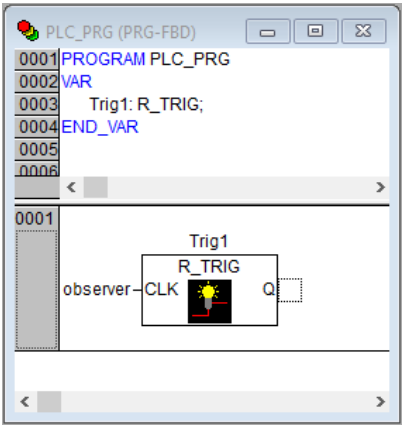

Рис. 1.1 Определение момента нажатия кнопки.

Определение момента отжатия кнопки бдительности (детектор заднего фронта, то есть перехода логического состояния из 1 в 0) и объединение двух детекторов с помощью блока ИЛИ. Блок ИЛИ осуществляет логическое суммирование. Если на входе блока ИЛИ присутствует хотя бы одна 1, то на выходе появится 1. о на выходе блока ИЛИ будет только при условии, что на всех входах находятся 0. В нашей программе блок ИЛИ нужен для объединения детекторов переднего и заднего фронта (нам не важно нажал ли оператор кнопку бдительности или отпустил; важен только факт манипуляции с кнопкой, который свидетельствует об активности оператора). Чтобы добавить блок ИЛИ выделите выход функционального блока Trig1 и вставьте (как было описано выше) элемент AND и переименуйте его в OR (логическое ИЛИ). Выделите свободный вход OR и вставьте перед ним блок F_TRIG (детектор заднего фронта) под именем Trig2. На вход Trig2 с помощью ассистента ввода (клавиша F2) подайте (категория Global Variables) переменную Observer.

Ввод задержки времени на включение сигнала тревоги. Вставьте после OR функциональный блок TOF (таймер с задержкой выключения; данный тип таймера выдает 1 на выходе при подаче 1 на вход, после изменения сигнала на входе с 1 на 0 на выходе сохраняется 1 в течение времени, которое

определяется выдержкой таймера, после чего на выходе устанавливается 0). Задайте для таймера имя Timer1. Замените три знака вопроса на входе PT константой T#10s, что соответствует 10 секундам.

Вывод команды включения тревоги. Выделите вход Q таймера Timer1 и в контекстном меню (вызывается с помощью правой кнопки мыши) дайте команду Assign (присвоить). Замените вопросы на имя переменной Warning. В диалоге определения задайте класс Class VAR_GLOBAL и тип BOOL. Если программу оставить без изменений, то в нормальном режиме работы устройства (когда оператор не реже одного раза в 10 секунд осуществляет манипуляции с кнопкой бдительности) сигнал тревоги будет сохраняться постоянно, а при прекращении активности оператора сигнал тревоги напротив пропадет через 10 секунд. То есть программа будет работать с точностью до наоборот. Для нормализации работы программы необходимо сигнал на выходе таймера перед его вводом в переменную Warning инвертировать: выделите позицию в середине линии, соединяющей выход таймера и переменную Warning; задайте команду Negate в контекстном меню; появившийся маленький кружек означает инверсию значеня логического сигнала.

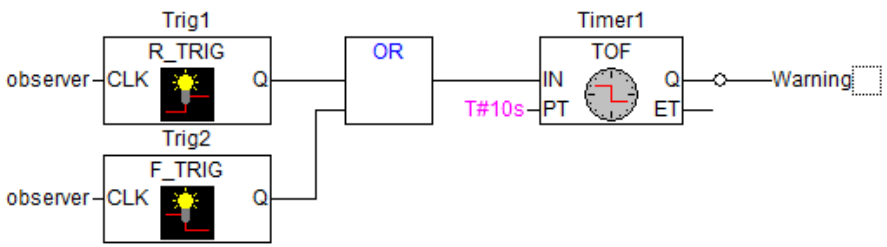

Рис. 1.2. Вывод команды включения тревоги

Формирование сигнала остановки устройства. В случае бездействия оператора в течение 5 секунд после включения сигнала тревоги устройство должно быть остановлено. Для ввода программного кода, обеспечивающего остановку, создадим новую цепь с помощью команды из контекстного меню Insert → Network (after). Вставьте из стандартной библиотеки в новую цепь элемент (Box) типа TON (таймер с задержкой включения; данный тип таймера выдает на выходе 0 при подаче на вход 0, при появлении на входе 1 на выходе сохраняется 0 в течение установленной выдержки времени, после чего на выходе появится 1, которая сохраняется до тех пор, пока на входе не будет установлен 0). Назовите таймер Timer2. Подайте переменную Warning на вход IN и константу T#5s на вход PT (задание выдержки в 5 секунд). Выход функционального блока Timer 2 присвойте (с помощью функции Assign) новой глобальной (Class VAR_GLOBAL) логической переменной Stop.

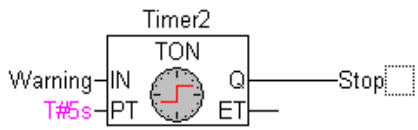

Рис. 1.3. Вывод команды включения тревоги

Создаем POU (программный модуль) управления перемещением тележки. В левой части окна CoDeSys расположен организатор объектов POUs (в нем присутствует созданный нами PLC_PRG). В ставьте при помощи команды 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 ему переход Trans0, заканчивающийся возвратом к Init.

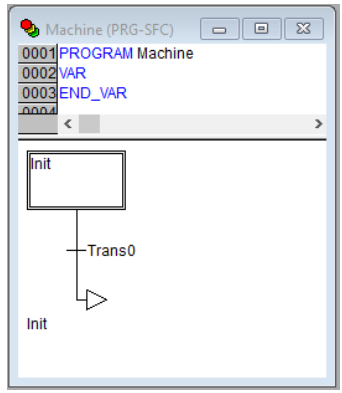

Рис. 1.4. POU (программный модуль)

Определяем алгоритм движения тележки. Каждому направлению движения должен соответствовать определенный этап программы (шаг программы). Чтобы создать алгоритм, определющий движение тележки, необходимо задать 4 этапа программы, каждый из которых задает определенное направление перемещения устройства и счетчик числа пройденных циклов. Выделите переход Trans0 так, чтобы он оказался окружен пунктирной рамкой. В контекстном меню дайте команду встави 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 называться Go_Right. Под ним Go_Down, Go_Left, Go_Up и Count. Соответственно, эти шаги отвечают за перемещение вправо, вниз, влево, вверх и счетчик циклов.

Программируем первый шаг программы. 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. Так как данный программный шаг отвечает за движение тележки влево, в нем должна

увеличиваться координата по оси X. Назовем переменную, определяющую позицию по оси X X_pos, тогда программа должна выглядить 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. Этот признак того, что действие 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. Преход должен содержать условие, разрешающее перемещение на следующий шаг программы. Переход после шага Init назовите Start и определите новую логическую переменную (Class VAR_GLOBAL, тип Type BOOL). Данная переменная соответсвует пусковой кнопке устройства, при ее единичном значении начинается цикл работы устройства. Следующий переход должен содержать условие X_pos=100, что на практике соответсвует остановке движения по оси X и началу движения по оси Y после того, как пройдены 100 м по оси X.
Условие третьего шага: Y_pos=50,
условие четвертого шага: X_pos=0,
условие пятого шага: Y_pos=0,
условие шестого шага: TRUE (переход на следующий шаг програмы разрешен сразу же после однокаратного выполнения предыдущего программного шага).
Останов устройства. Вернитесь в PLC_PRG POU и добавьте третью цепь. Вместо вопросов вставьте переменную Stop и

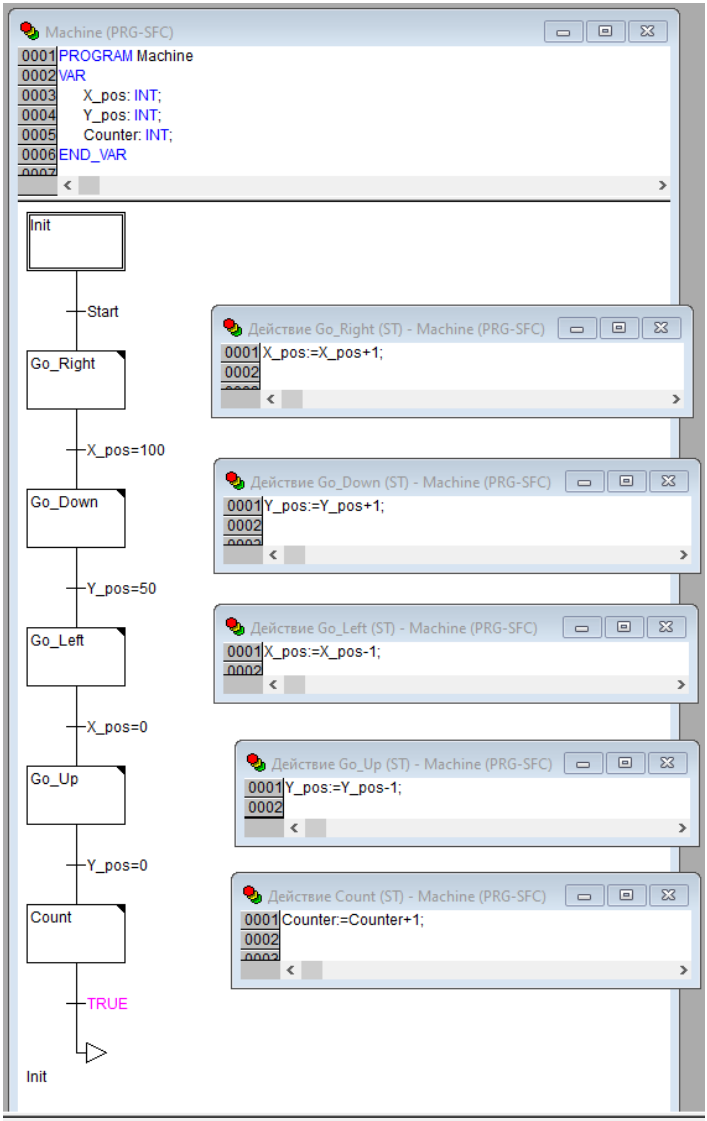

Рис. 1.5. Программирование переходов

затем из контекстного меню вставьте оператор Return. Данный оператор прерывает работу программы PLC_PRG POU при единичном значении Stop.

Вызов POU управления тележкой. Чтобы включить в работу программы алгоритм перемещения тележки, его необходимо ввести в основной модуль. Для этого добавьте еще одну цепь, выделете ее и вставьте элемент Box из контекстного меню. Как обычно это будет AND. Нажмите F2 и в ассистенте ввода задайте POU управления механизмом из категории пользовательских программ (User defined programs category).

Компиляция проекта. Откомпилируйте проект целиком командой меню Project → Rebuild all, либо клавишей F11. Если мы все сделали верно, то в нижней части окна появится сообщение: ,,0 errors,, В противном случае необходимо найти и устранить допущенные ошибки. Для упрощения поиска ошибок необходимо прочитать и проанализировать сообщения об ошибках в развернутом виде (представлено на английском языке).

3. Визуализация:

Визуализация помогает видеть процесс работы устройства на экране компьютера и управлять им с помощью клавиатуры и мыши. Визуализация имеет значительную практическую ценность, так как в ряде случае процесс доработки и отладки программы для ПЛК с использованием виртуальных компьютерных инструментов намного легче подобной работы, проводимой на реальном контроллере.

Создаем визуализацию. Третье страничка организатора объектов CoDeSys называется визуализация (Visualizitaion). Перейдите на страничку визуализации. В контекстном меню введите команду добавления объекта Add object. Присвойте объекту имя Observation. В конце работы окно визуализации будет выглядить следующим образом:

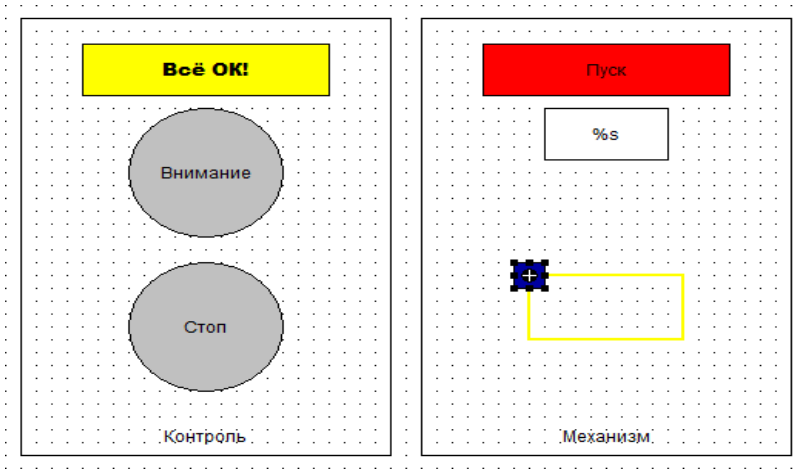

Рис. 1.6 Визуализация

Рисуем элемент визуализации. Давайте начнем с реализации кнопки бдительности (на рисунке 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, после чего отпустите клавишу.

Настройка первого элемента визуализации. Диалоговое окно настройки элемента вызывается двойным щелчком мыши на его изображении. Зайдите 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 результате выполненных

действий достигается следующее: виртуальная кнопка связывается с переменной Observer и начинает выполнять роль кнопки бдительности; при нажатии на виртуальную кнопку она переходит в возбужденное состояние и цвет ее меняется со светло-голубого на голубой; значение переменной Observer при отпущенной кнопке соответствует 0, а при нажатой 1.

Развиваем визуализацию. Нарисуйте окружность 'Внимание'. В настройках Text Category, Contents задайте текст 'Внимание'. В Colors Category, Color задайте закраску объекта: Inside (нормальное состояние) серым цветом и Alarm color (возбужденное состояние) красным цветом. В Variable Category, Color change задайте связанную с элементом переменную (Warning), при изменении которой будет меняться цвет объекта. Данная окружность будет выполнять роль индикатора включения сигнала тревоги.

Далее создадим индикатор аварийного останова устройства 'Стоп'. Для этого скопируйте созданную окружность командой Edit → Copy и вставьте ее командой Edit → Paste. Измените настройки индикатора 'Стоп': в соответствующем поле задайте текст 'Стоп', а связанную с элементом переменную замените на Stop.
Нарисуйте прямоугольник для клавиши Пуск со следующими настройками: 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. Данная кнопка используется для запуска устройства в работу. После нажатия на данную кнопку запускается программный код, отвечающий за движение тележки.

Для отображения счетчика циклов движения тележки по замкнутой траектории нарисуем прямоугольник со следующими настройками: Text Category, Contents текст Счетчик: %s (%s в данном случае показывает, что на этом месте будет

отображаться текущее значение связанной переменной); Variable Category, Textdisplay (выв.текст) переменная Counter из модуля Machine (Machine.Counter).

Нарисуйте прямоугольник, движение которого имитирует движение тележки. Для этого используйте следующие настройки: Absolute movement (Положение) Category, X-Offset (настройка отвечает за перемещение по оси X) переменная Machine.X_pos; Absolute movement Category, Y-Offset (настройка отвечает за перемещение по оси Y) переменная Machine.Y_pos; Colors Category, Color закраска Inside голубым цветом.

Для разделения областей управления устройством и перемещения тележки можно создать две декоративные рамки. Задайте в них надписи с выравниванием по низу (Vertical aligment bottom). Используя контекстное меню, поместите декоративные прямоугольники на задний план (Send to back).

4. Запуск целевой системы:

Запустите систему исполнения (CoDeSys SP RTE), после чего в панели задач появится соответствующая иконка. Щелкните по ней правой клавишей и дайте команду на старт системы (Start System).

5. Запуск проекта

Соединение с контроллером устанавливается командой Online → Login из среды программирования CoDeSys. Команда Online → Run запускает проект. Перейдите в окно визуализации и проверьте работу нашего устройства. Для запуска проекта в режиме эмуляции установите флажок в меню Online → Simulation. Далее переходите в режим online и запускайте проект, как описано выше.