**Лабораторная работа №5 «Создание многозадачных приложений в среде CoDeSys 2 .3 »**

***Постановка задачи:***

В данной лабораторной работе следует рассмотреть понятие многозадачности при моделировании объекта управления – стиральной машины. Стиральная машина имеет следующие основные этапы работы:

- контроль закрытия дверки;

- ожидание нажатия кнопки «Пуск»;

- выбор режима стирки, отличающихся температурой, временем стирки и количеством используемой воды;

- наполнение бака водой, нагрев, стирка, слив воды.

Добавим в нашу машину несколько режимов стирки, отличающихся температурой, временем стирки и количеством используемой воды.

Для описания режима определим структуру *WashPara*, которая представляет сбой новый тип данных. Такие определения делаются в проекте централизованно на вкладке Data types (типы данных) менеджера проекта. После этого можно объявлять в проекте переменные типа WashPara.

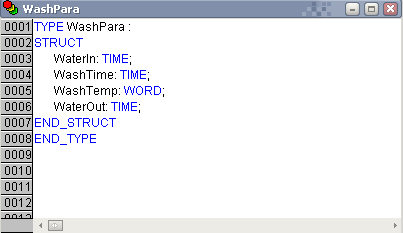


Рисунок 1 – Окно определения структуры *WashPara*

Создадим в проекте новый POU типа программа (PROGRAM), назовем его *Wash*и выберем упрощенный язык SFC. В разделе объявлений *Wash*объявим константу для числа режимов:

**VAR CONSTANT**

maxProgNr :**BYTE**:= 5;

**END\_VAR**

Идея состоит в том, чтобы везде, где потребуется количество режимов, использовать *maxProgNr*, это позволит при необходимости легко менять их число в единственном месте программы. Для хранения режимов объявим массив структур:

MyWashPara: **ARRAY**[0..maxProgNr] **OF**WashPara;

Обращение к элементам структуры происходит через точку после указания элемента массива, например:

MyWashPara[i].WaterOut := T#5s;

Добавим в проект библиотеки (вкладка «Ресурсы» - «Менеджер библиотек»): SysLibFile.lib, SysLibTime.lib, SysTaskInfo.lib, SysLibCallBack.lib (C:\Program Files\Common Files\CAA-Targets\3S\ Lib\_PLCWinNT), AnalyzationNew.lib (C:\Program Files\3S Software\CoDeSys V2.3\Library)

Далее необходимо связать длительность выполнения определенных шагов SFC со значениями элементов массива структур, так чтобы можно было динамически влиять на работу диаграммы, изменяя только значения переменных. Для этого используем такие атрибуты шагов, как минимальное время активности шага. Привязка элементов массива к временам шага показана на рисунке 2. Переходы после выполняемых по времени шагов всегда разрешены, то есть им присвоены константы TRUE.

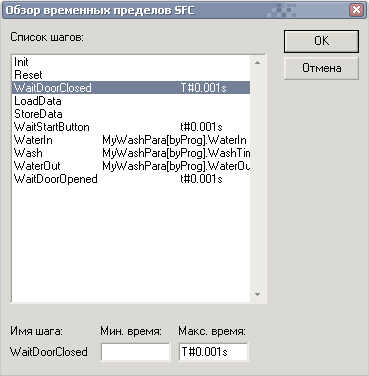


Рисунок 2 – Окно привязки временных пределов для блоков диаграммы SCF

Окно программного объекта Wash представлено на рисунке 3.

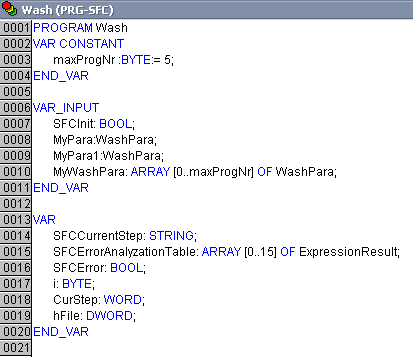
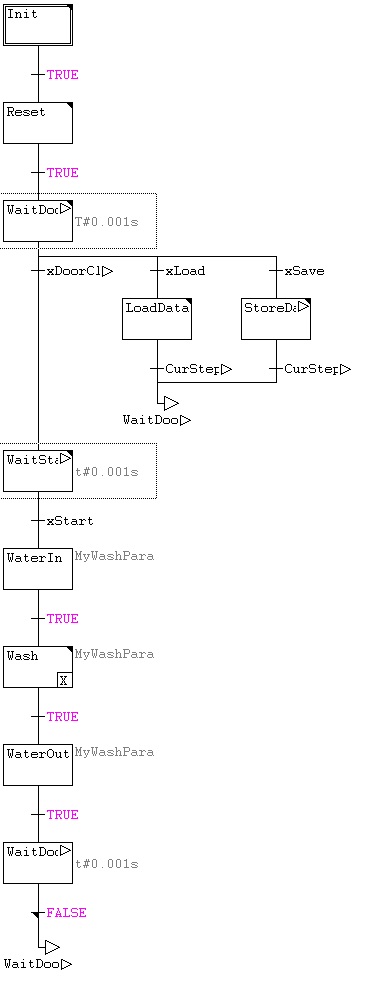


Рисунок 4 – Окно программного объекта Wash

В результате получим диаграмму последовательных переходов:

Далее определим действия шагов:

Шаг Init:

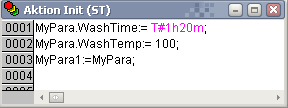


Рисунок 5 – Действия шага Init

Шаг Reset:

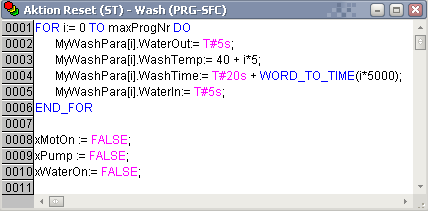


Рисунок 6 – Действия шага Reset

Шаг WaitDoorClosed:

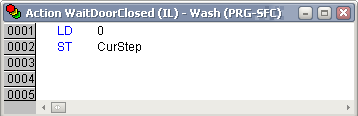


Рисунок 7 – Действия шага WaitDoorClosed

Шаг LoadData:

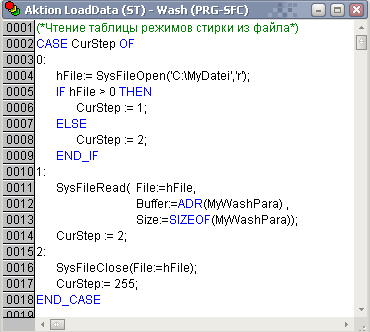


Рисунок 8 – Действия шага LoadData

Шаг StoreData:

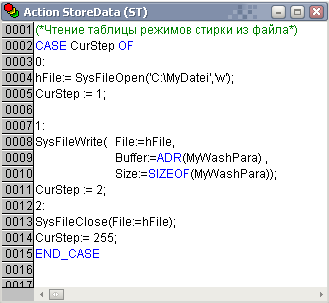


Рисунок 9 – Действия шага StoreData

Шаг WaitStartButton:

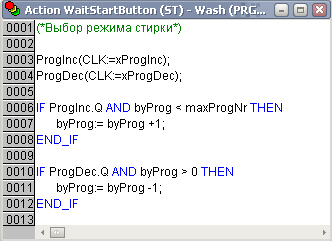


Рисунок 10 – Действия шага WaitStartButton

Шаг Wash:

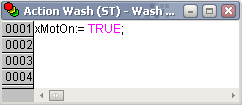


Рисунок 11 – Действия шага Wash

Условие перехода на шаг WaitDoorClosed:

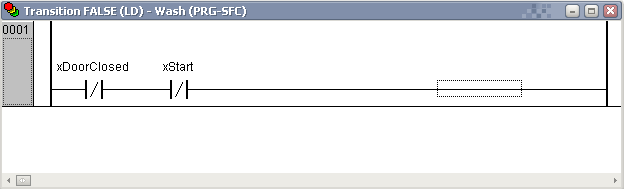
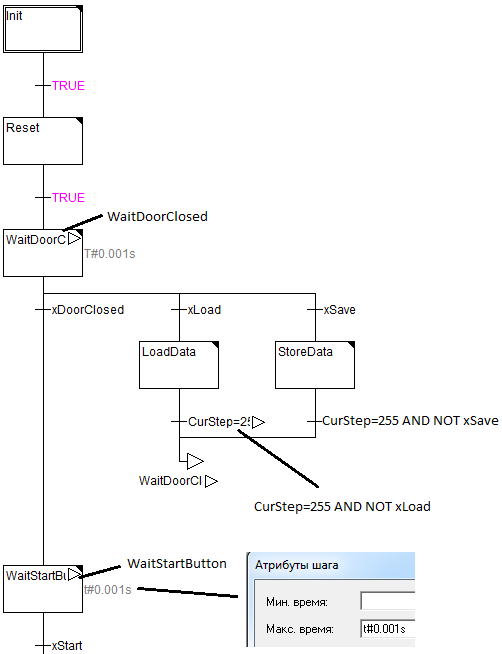
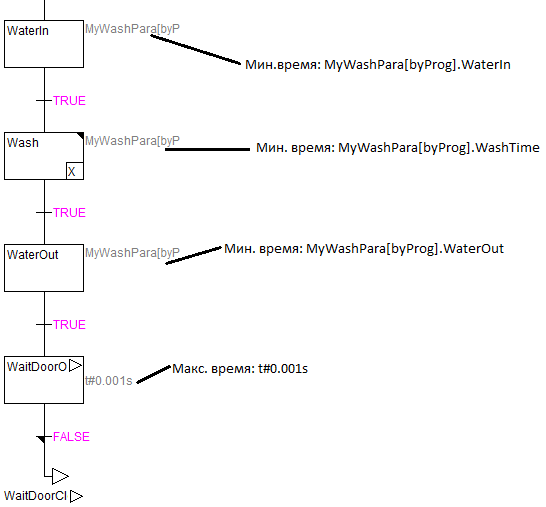


Рисунок 12 – Условие перехода на языке LD

В результате диаграмма последовательных переходов будет иметь вид:





В главном окне проекта PLC\_PRG выполним программную реализацию вызова программы стирки, сброса программы стирки и определение состояния программы по активности шагов (рисунок 12).

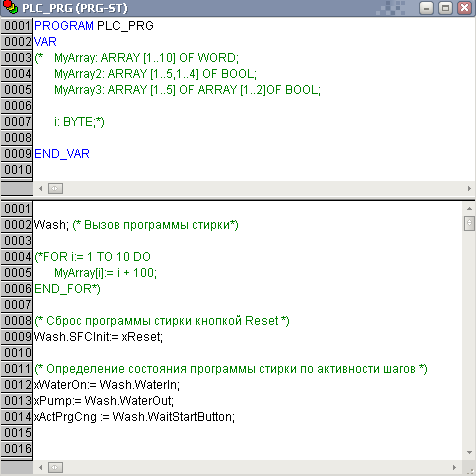


Рисунок 13 – Окно объекта PLC\_PRG на языке ST

Добавим в проект новый объект WaterlnOut для эмуляции наполнения и слива воды в окне визуализации. Программный код эмулятора воды представлен на рисунке 14.

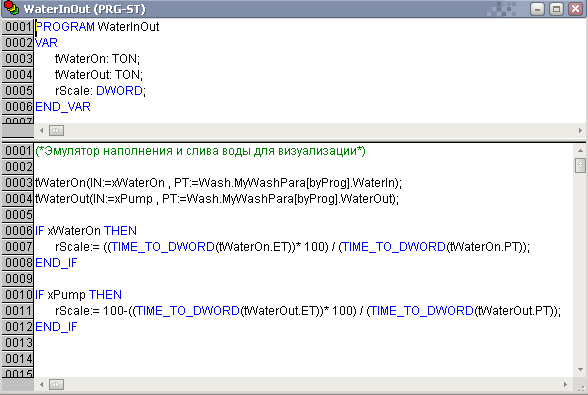


Рисунок 14 – Окно объекта WaterlnOut на языке ST

Окно глобальных переменных проекта представлено на рисунке 14.

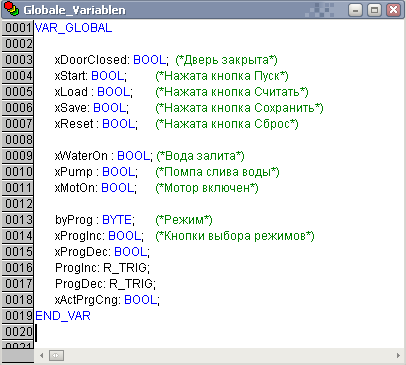


Рисунок 15 – Окно глобальных переменных проекта

Визуализация

Для наглядной демонстрации работы стиральной машины удобно использовать встроенную графическую визуализацию CoDeSys. Ее возможный вариант показан на рисунке 16.

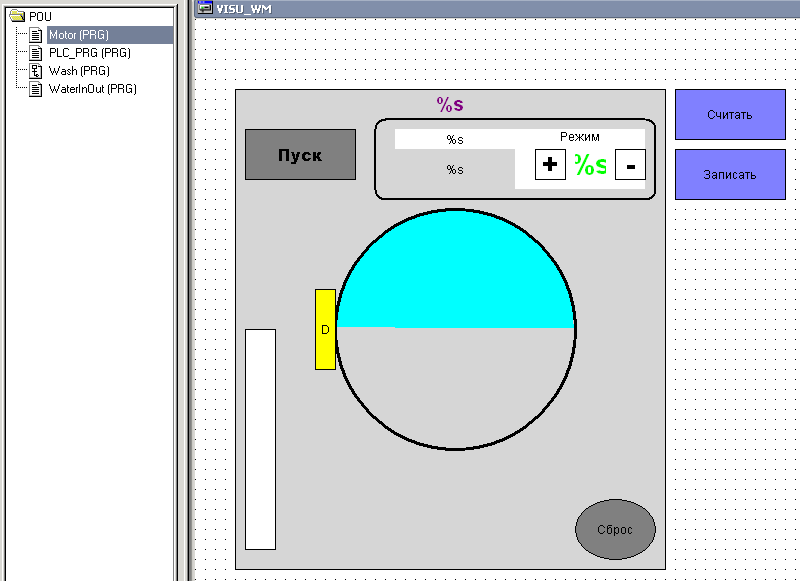


Рисунок 16 – Визуализация проекта

Рассмотрим привязку графических элементов к переменным проекта:

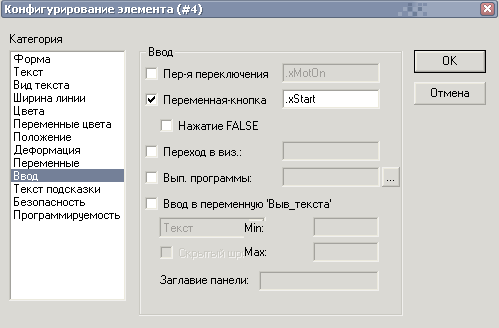


Рисунок 17 – Конфигурирование кнопки «Пуск»

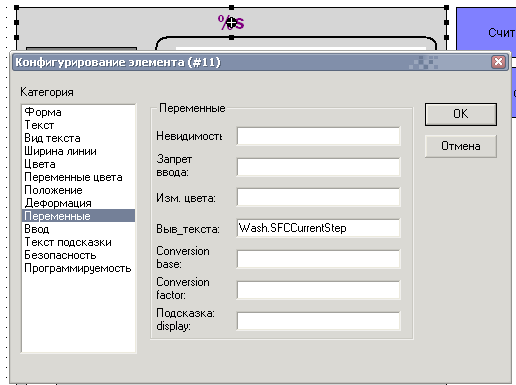


Рисунок 18 – Конфигурирование строки вывода текста

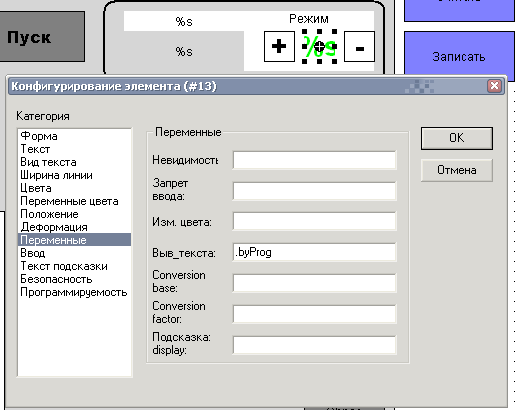


Рисунок 19 – Привязка к переменной byProg

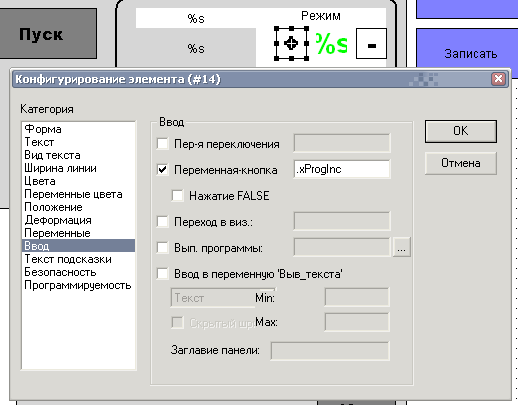


Рисунок 20 – Привязка объекта изменения режима стирки к переменной кнопке xProgInc

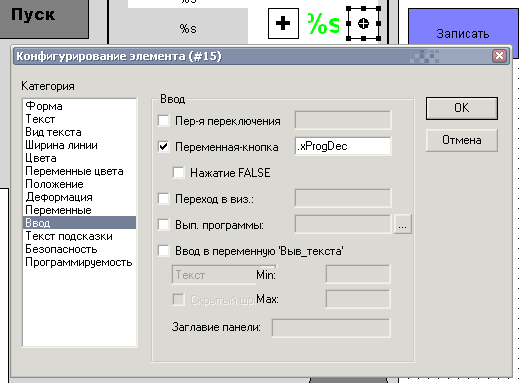


Рисунок 21 – Привязка объекта изменения режима стирки к переменной кнопке

xProgDec

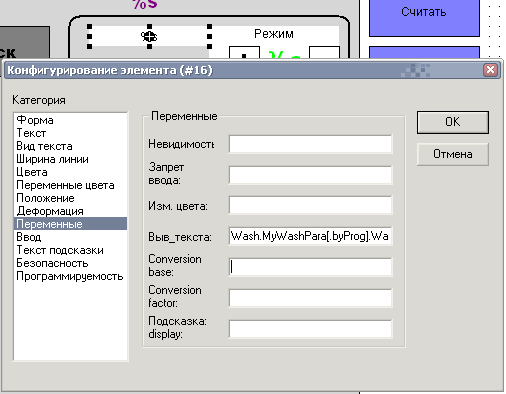


Рисунок 22 – Привязка к переменной Wash.MyWashPara[.byProg].WashTemp

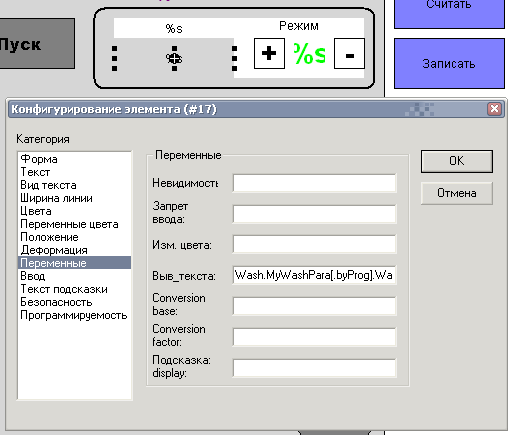


Рисунок 23 – Привязка к переменной Wash.MyWashPara[.byProg].WashTime

Эмулятор наполнения воды: добавьте на форму прямоугольник, и задайте ему размер height=0, перейдите на вкладку «Деформация», и задайте свойство «Верх» как указано на рисунке 24.

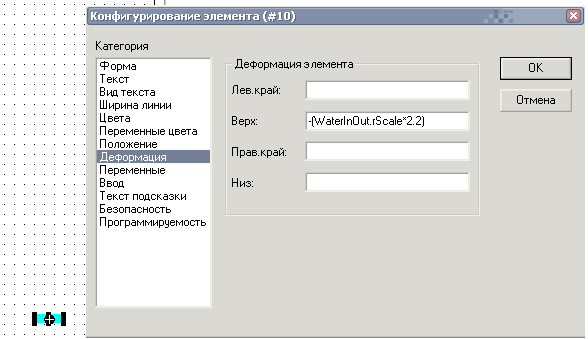


Рисунок 24 – Конфигурирование эмулятора наполнения воды

Для графического объекта – ручки необходимо в разделе «Переменные» к изменению цвета привязать переменную xDoorClosed, а в разделе «Ввод» для переменной переключения также задать переменную xDoorClosed.

Для кнопки «Сброс» в разделе «Ввод» для переменной кнопки задать переменную xReset.

Для кнопки «Считать» необходимо в разделе «Переменные» к изменению цвета привязать переменную xLoad, а в разделе «Ввод» для переменной кнопки также задать переменную xLoad.

Для кнопки «Записать» необходимо в разделе «Переменные» к изменению цвета привязать переменную xSave, а в разделе «Ввод» для переменной кнопки также задать переменную xSave.

***Запустите проект в режиме эмуляции и проверьте его работу.***

Разработанная визуализация получилась статичной, поэтому реализуем отображение вращения барабана. Это можно сделать при помощи закрашенного сегмента. Напишем отдельную программу *Motor*, ее задача плавно циклически изменять переменную *wAngle*от 0 до 360 градусов, которую мы свяжем с углом сектора соответствующего элемента визуализации. Для этого добавим в проект новый объект «Motor» и на языке ST реализуем движение:

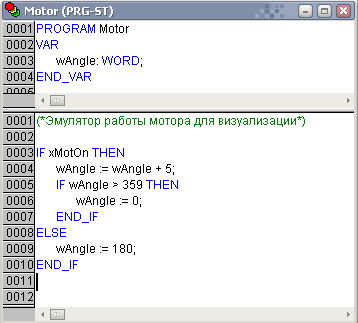
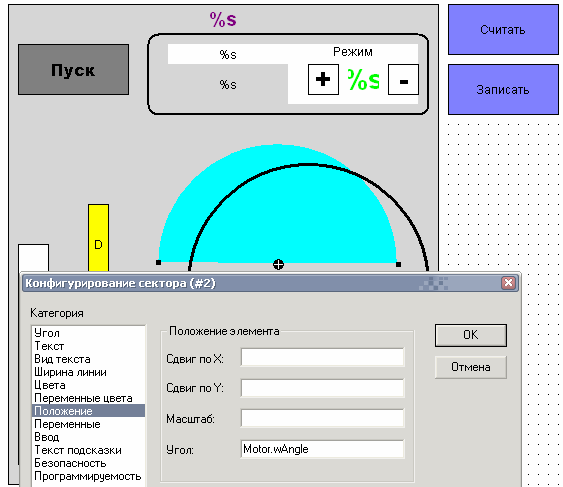


Рисунок 25 – Программный код объекта «Motor»

Проведем привязку сектора барабана к программе:

  
Рисунок 26 – Привязка к программе движения

Как видно, здесь нет таймеров, обеспечивающих достаточно медленную для зрительного восприятия скорость вращения барабана. Мы можем поступить проще, создадим для целей моделирования циклическую задачу с достаточно медленным циклом работы, который легко регулировать. После отладки эту задачу можно будет просто отключить, не изменяя ничего в основных программах. По умолчанию CoDeSys создает новый проект с единственной программой PLC\_PRG, которая неявно включена в единственную циклическую задачу. Чтобы сделать многозадачный проект, необходимо использовать конфигуратор задач (Task configuration), расположенный на вкладке «Ресурсы» менеджера проектов. Щелкните правой кнопкой мыши по элементу Task configuration и в появившемся контекстном меню выберете команду Append task. Сначала создадим главную задачу, она будет назваться *Main*, ее тип – циклический (cyclic) и интервал работы T#10ms. Аналогичным способом откройте контекстное меню для нашей новой задачи и добавьте в нее (Append program call) программу PLC\_PRG();. Теперь создайте еще одну циклическую задачу *MotorSim*с интервалом T#100ms, в нее вставьте вызов программы Motor();. Теперь конфигурация задач должна выглядеть так, как показано на рисунке 27.

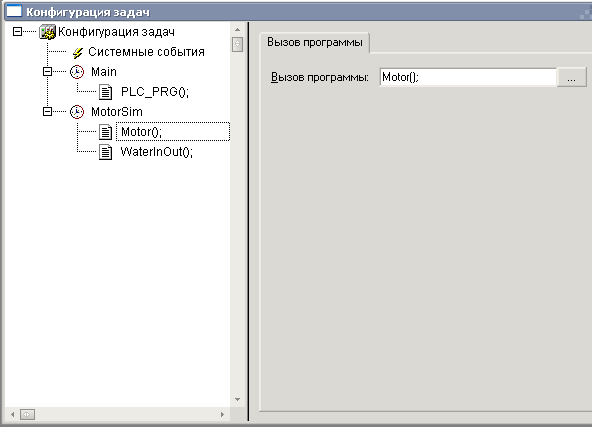


Рисунок 27 – Окно конфигурации задач

Для запуска проекта в режиме эмуляции установите флажок в меню **Online -> Simulation.** Далее переходите в режим *online* и запустите проект.