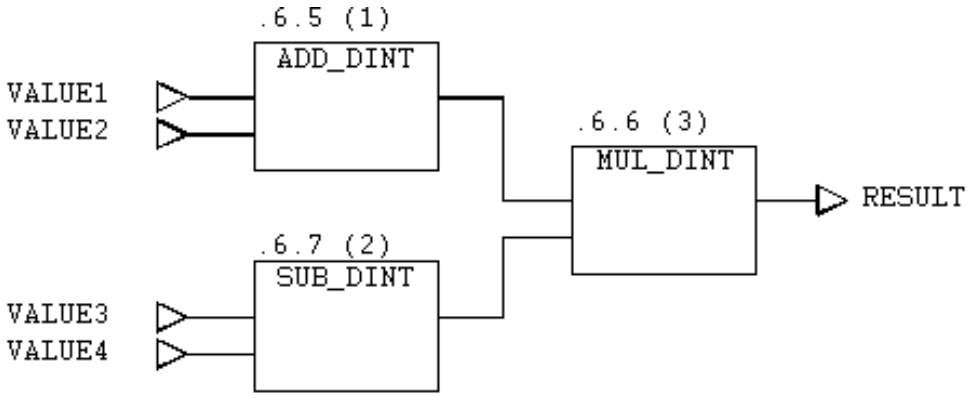
ТЕХНОЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ CONCEPT



Санкт-Петербург

2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ»

Кафедра информационно-измерительных технологий и систем управления

ТЕХНОЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ CONCEPT

М е т о д и ч е с к о е п о с о б и е

Факультет АСУ ТП

Санкт-Петербург

2012

УДК 66.012

Технологическое программирование в среде Concept: методическое пособие. / сост.: Л.А.Русинов, И.В.Рудакова; СПбГТУРП – СПб., 2012. – 31 с.

Пособие содержит методические указания к лабораторным работам по изучению языков технологического программирования микропроцессорных контроллеров, стандартизованные МЭК (стандарт 1131-3). Рассмотрены варианты реализации систем регулирования в среде Concept для программирования контроллеров типа Quantum фирмы Schneider Electric, позволяющей эмулировать контроллер непосредственно на ЭВМ.

Предназначено для студентов, обучающихся по направлениям подготовки бакалавров 220400 «Управление в технических системах» и 220700 «Автоматизация технических процессов и производств», и соответствует рабочей программе дисциплины «Интегрированные системы проектирования и управления». Пособие может быть использовано студентами, обучающимися по направлению подготовки магистров 220700 «Автоматизация технических процессов и производств».

Ил. 18, табл. 4, библиогр. 2 назв.

Рецензент:
профессор кафедры АЭиЭ СПБГТУРП, д-р.техн.наук Кулик В.Д.

Подготовлено и рекомендовано к печати кафедрой информационно-измерительных технологий и систем управления СПбГТУРП (протокол № 3 от 12.11.2012 г.).

Утверждено к изданию методической комиссией факультета АСУТП СПбГТУРП (протокол № 2 от 15.11.2012 г.).

ВВЕДЕНИЕ

В 1993 году международная электротехническая комиссия приняла стандарт IEC 1131 часть 3, который охватывает различные аспекты использования программируемых логических контроллеров (ПЛК), в частности, стандартизацию существующих языков технологического программирования ПЛК. Стандартизация языков технологического программирования позволила значительно упростить процедуру освоения и внедрения контроллеров разных фирм-производителей.

Выбор среды программирования Concept для знакомства студентов с этим этапом проектирования АСУТП определялся наличием эмулятора работы ПЛК, что позволило организовать лабораторную работу полностью на ЭВМ и обеспечить достаточное количество рабочих мест.

1. ОПИСАНИЕ ЯЗЫКОВ ТЕХНОЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ СТАНДАРТА IEC 1131-3

Стандарт IEC 1131-3 описывает синтаксис и семантику пяти языков программирования ПЛК:

- языка инструкций (IL);
- языка релейных диаграмм (LD);
- языка функциональных блоковых диаграмм (FBD);
- языка последовательных функциональных диаграмм (SFC);
- языка структурированного текста (ST).

Из всех языков в силу лучшего понимания и наглядности наибольшее распространение получили два – FBD и LD, которые более подробно рассмотрены далее.

1.1. Язык релейных диаграмм (LD)

Структура LD-сегмента соответствует электрической схеме. С левой стороны сегмента расположена шина питания, принцип действия которой соответствует принципу работы линии питания в электрических схемах. С линией питания соединяются логические элементы языка LD (контакты, катушки, блоки). Линия, реализующая нулевой провод, визуально не отображена, но подразумевается.

Контакт – это элемент языка LD, описывающий входные переменные типа BOOL. Список атрибутов контактов приведен в Приложении А.

Катушка – это элемент языка LD, описывающий выходные переменные типа BOOL. Список атрибутов катушек приведен в Приложении Б. Катушка, по аналогии с реле, активизируется при появлении в горизонтальной связи слева от катушки сигнала логической единицы, т.е. состояние горизонтальной связи назначается связанной с катушкой переменной. Далее изменяется состояние всех контактов, связанных с переменной, назначенной катушке, например, за-

мыкается нормально разомкнутый контакт. Катушки обычно следуют за контактами, блоками элементарных функций или функциональными блоками.

Библиотека среды программирования включает наборы э***лементарных функций и функциональных блоков (EFB)***, логика которых не может быть изменена пользователем. Графически они изображаются прямоугольными рамками с входами, расположенными слева, и выходами, расположенными справа.

Элементарные функции. Пример функции ADD показан на рис. 1.

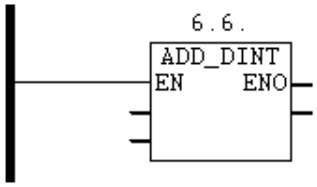


Рис. 1. Функция ADD (сложение)

Название функции (тип) указывается внутри, а порядковый номер – над рамкой. Номер функции назначается при вызове функции, не может быть изменен и имеет следующий формат: ***n.m*****,** где ***n*** – номер сегмента, секции, в которой составляется программа, а ***m*** – номер функции. Функция не имеет внутренней памяти для хранения значений, поэтому входная и выходная переменные одного типа. Совокупность вход/выход EN/ENO используется для установки порядка активизации функции: если значение линии связи EN имеет значение 0, то функция не будет выполняться.

Элементарные функциональные блоки. Пример функционального блока CTU показан на рис. 2.

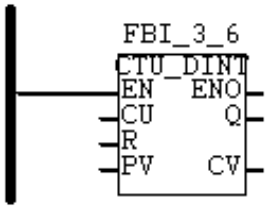


Рис. 2. Функциональный блок CTU

Название функционального блока (тип) указывается внутри рамки. В отличие от функции блок имеет собственную область памяти для хранения промежуточных значений, поэтому входные и выходные переменные могут относиться к разным типам.

Над рамкой дается имя блока – это уникальный идентификатор функционального блока в проекте. Имя генерируется автоматически, но может быть отредактировано в диалоговом окне блока, при этом в формате имени нет разницы между прописными и строчными буквами. Уникальность имени необхо-

дима для однозначности распределения области памяти для двух или более блоков одного типа, задействованных в проекте.

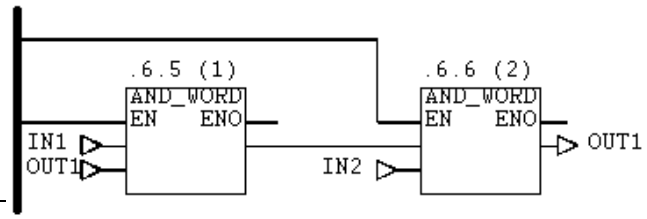
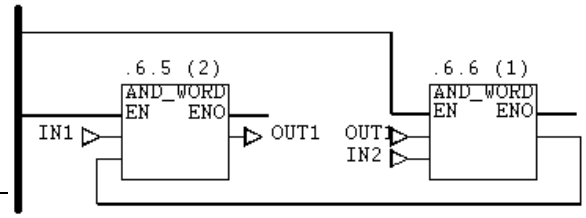
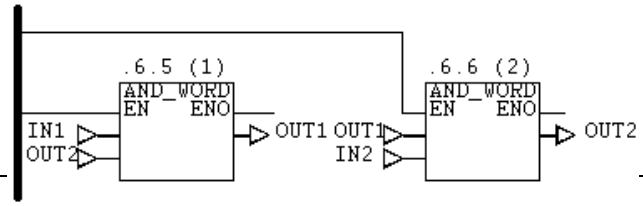
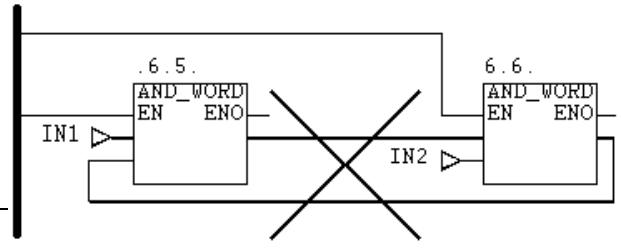
Дополнительно используются ***блоки DFB*** – это специально разработанные, часто повторяющиеся фрагменты реализуемого алгоритма и ***блоки UDEFB*** – это определенные пользователем оригинальные функции и функциональные блоки, которые разрабатываются пользователем на языке C++.

Обработка данных в программе, написанной на языке LD, выполняется последовательно по линиям сверху вниз и слева направо.

Варианты соединения блоков в линии даны в табл. 1.

Способы соединения нескольких блоков на языке LD в одной линии

Таблица 1

Описание соединения	Пример соединения
Правильное выполнение цикла (вариант 1)	
Правильное выполнение цикла (вариант 2)	
Правильное выполнение цикла (вариант 3)	
Неверное выполнение цикла	

Для облегчения написания, отладки, понимания программируемого алгоритма, рекомендуется всю программу разбить на отдельные секции, внутри каждой из которых реализуется отдельный функционально законченный фрагмент. Теоретически каждая секция может иметь любое число блоков.

1.2. Язык функциональных блоковых диаграмм (FBD)

Язык FBD также является графическим и по своей сути похож на язык LD. Однако вместо изображений контактов и катушек здесь используются специальные функциональные блоки.

Алгоритм решения в FBD выглядит как функциональная схема электронного устройства. Объекты языка (EFB, UDEFB), описанные выше под общим названием FFB, объединяются через описанные программистом параметры или через графические связи. По аналогии с программированием на языке LD весь алгоритм рекомендуется разбить на отдельные секции.

Пример соединения блоков в языке FBD показан на рис. 3.

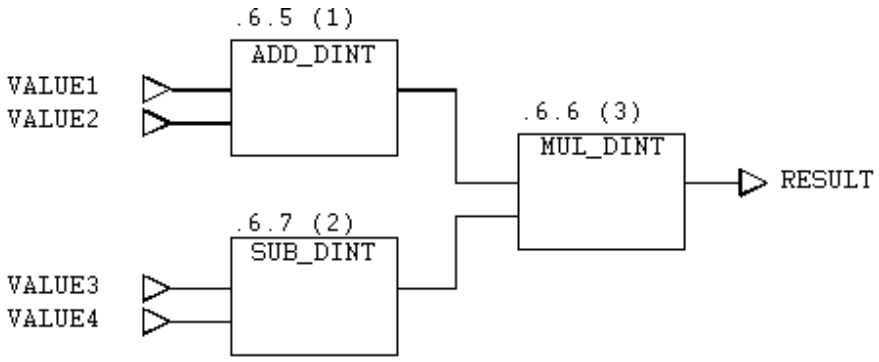


Рис. 3. Фрагмент программы на языке FBD

Обработка данных внутри программы определяется последовательностью связей блоков и выполняется сверху вниз и слева направо. Для пояснений внутри секции может быть помещен и текст. Текстовые объекты не должны перекрываться блоками FFB, хотя могут пересекать связи.

2. РАЗРАБОТКА ПРОЕКТА В СРЕДЕ CONCEPT

Concept – это унифицированная конфигурационная система, предназначенная для разработки программного обеспечения ПЛК, разработанных фирмой Schneider Electric. Среда снабжена средствами эффективного конфигурирования контроллера в соответствии со всеми инструкциями международного стандарта IEC 1131 – 3. Программа разрабатывается как комплекс из нескольких секций, представляющих собой отдельную часть алгоритма управления. Описываемые программистом переменные доступны всему проекту, т.е. являются ***сквозными для всех секций***.

2.1. Методика разработки проекта

В процессе разработки проекта в среде Concept можно выделить семь основных этапов:

1) загрузка среды программирования Concept;

2) сохранение проекта в отдельном каталоге;
3) задание конфигурации контроллера и распределение области памяти модулей периферийной связи;
4) программная реализация алгоритма управления с периодическим сохранением;
5) тестирование проекта на эмуляторе;
6) оптимизация работы реализованной программы по результатам тестирования;
7) документирование проекта.

2.2. Описание 32-битного эмулятора контроллеров Quantum

В лабораторной работе используется 32-битный эмулятор **PLCSIM32**. Он позволяет отлаживать программы, написанные на языках стандарта IEC 1131 (FBD, LD, IL, SFC). Соединение программирующего устройства (компьютера) и эмулятора осуществляется по протоколу TCP/IP. При загрузке симулятора открывается рабочее окно PLCSIM32, вид которого дан на рис. 4.

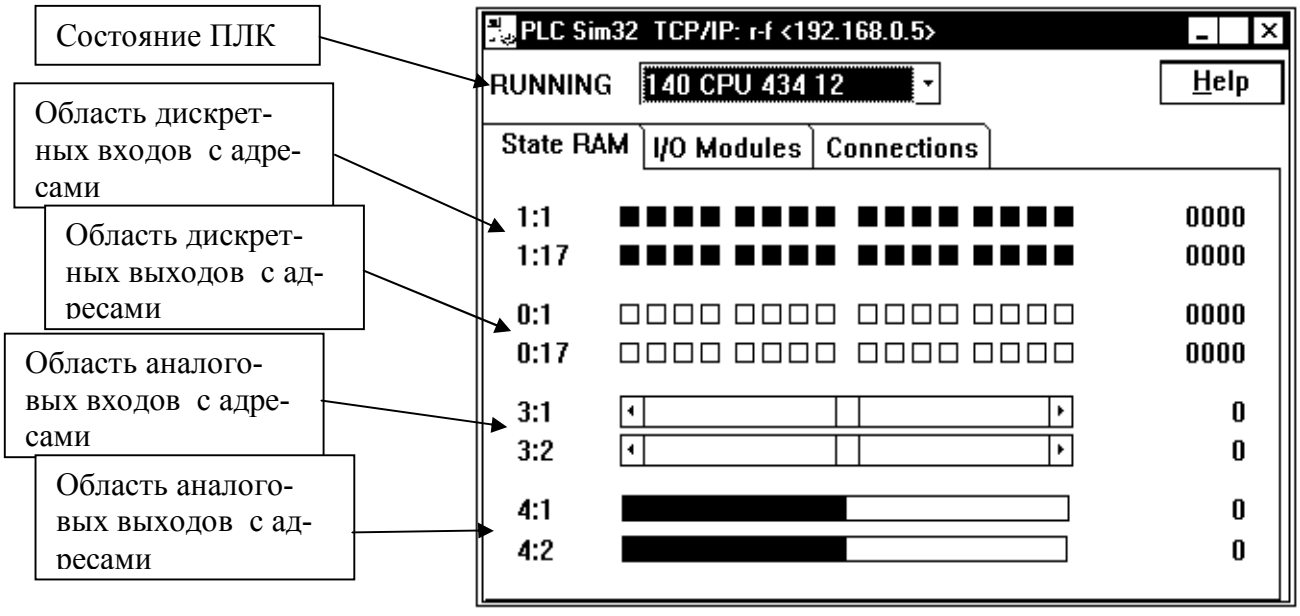


Рис. 4. Рабочее окно 32-битного имитатора

Первое текстовое поле в окне эмулятора показывает состояние моделируемого ПЛК: **DIM** - неопределенное состояние эмулятора), **STOPPED** - эмулятор (моделируемый ПЛК) остановлен, **RUNNING** - эмулятор (моделируемый ПЛК) работает. Рядом с полем состояния расположено окно, в котором указывается тип центрального процессора моделируемого ПЛК.

В рабочем окне есть три раздела:

- **State RAM** – отображение состояния оперативной памяти контроллера;

- **I/O modules** – демонстрация загруженной конфигурации или состояния оперативной памяти выбранного модуля (имитатор не поддерживает объекты удаленного ввода/вывода (RIO) и распределенного ввода/вывода (DIO));
- **Connections** – вывод информации о текущем состоянии соединения эмулятора и компьютера.

В рабочем поле окна **State RAM** построчно отражены области отдельно для дискретных и аналоговых входов и выходов, адреса которых имеют следующий формат:

0xxxxx – для контроля состояния дискретных выходов;
1xxxxx – для контроля и изменения состояния дискретных входов;
3xxxxx – для контроля и изменения значений аналоговых входов;
4xxxxx – для контроля состояния аналоговых выходов;

Здесь символы **xxxxx** представляют собой пятизначный адрес в оперативной памяти, например, адрес 400201 означает ссылку на 16-битный выходной регистр по адресу 00201. На каждый дискретный вход/выход отводится один бит в 16-битных регистрах оперативной памяти; на каждый аналоговый вход/выход отводится целый регистр.

В первом столбце рабочего окна имитатора (рис. 4) приводятся идентификаторы типов входов/выходов (от 0 до 4) и далее через двоеточие показываются номера первых битов регистров, соответствующих этим входам/выходам

В исходном положении в первой строке указан первый регистр, биты которого отражают состояние дискретных входов (переменные типа BOOL) с адресами с 1-го по 16-й, а во второй строке – второй регистр с адресами с 17-го по 33-й. Для изменения адреса достаточно щелкнуть кнопкой «мыши» по первому столбцу соответствующей строки и в открывшемся меню установить начальный адрес того 16-битного регистра, куда попадает нужный пользователю вход (например, если требуется индицировать состояние 52-го входа, то нужно установить в окне начальный адрес – 49).

Так как строки аналоговых входов/выходов (3x и 4x) отражают состояние одной переменной, занимающей весь регистр, то в открывшемся окне просто указывают номер регистра.

В среднем столбце в строках 1x и 0x отражается состояние дискретных входов/выходов. При этом красный квадратик соответствует значению 0, а зеленый – 1. Состояние битов 1x может переключаться последовательными нажатиями левой кнопки «мыши», состояние битов 0x может изменяться вследствие работы контроллера.

В среднем поле строк 3x положение ползунка характеризует значение входной переменной и перемещается левой кнопкой «мыши». В открывшемся при нажатии на номер регистра правой кнопкой «мыши» окне можно установить минимальное (Min) и максимальное (Max) значения диапазона изменения переменных. Для входных переменных (адреса 3x) еще указываются деления шагов перемещения ползуна при изменении значений входных переменных. При этом задаются параметры двух вариантов движения ползунка: при нажатии

кнопок со стрелками на краях полей с ползунком (точное перемещение – Line magnitude) и на соответствующую строку состояния справа или слева от ползунка (грубое перемещение – Page magnitude). Можно сохранить до 8 совокупностей таких настроек, используя кнопки User 1-8.

Строки 4x отражают состояние выходных переменных.

Правые столбцы показывают адреса переменных в памяти контроллера. В строках 0x и 1x отображено по 16 бит, которые интерпретируются как единое слово. Щелчок по столбцу левой кнопкой «мыши» открывает окно Change value, что позволяет изменить содержимое State RAM.

Нажатие правой кнопки «мыши» на правые столбцы строк 3x и 4x приводит к открытию подменю выбора формата представления (шестнадцатеричный Hex, десятичный прямой Dec (от –32768 до +32767) и десятичный однополярный UnsDec (от 0 до 65535).

3. ОПИСАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Целью лабораторных работ является знакомство с методикой разработки и отладки проекта в среде Concept, приобретение студентами навыков программирования ПЛК на технологических языках FBD и LD стандарта IEC1131-3, на примерах моделирования объектов первого порядка, одноконтурной и каскадной систем регулирования, предиктора Смита и широтно-импульсного модулятора. Работы проводятся на ЭВМ с использованием эмулятора PLCSIM32.

3.1. Порядок создания, сохранения и конфигурирования проекта

3.1.1. Загрузка системы Concept

Загрузите среду Concept из под ОС Windows. При этом следует отметить, что программы Concept, Concept DFB и 16-битный эмулятор не могут работать одновременно. Поэтому предварительно следует отключить 16-битный эмулятор, для чего необходимо загрузить его файл Simulator 16-bit, перевести его в пассивное состояние (активизировать опцию simulator ON) и выгрузить эмулятор.

Для создания нового проекта необходимо выбрать в главном меню программы File → New project, при этом откроется окно безымянного проекта (Concept untitled).

3.1.2. Сохранение проекта

Если проект нужно сохранить впервые, то следует воспользоваться командой меню File → Save project as. В появившемся окне необходимо указать логический диск, выбрать папку, специально созданную для хранения информации этого проекта, и присвоить имя проекту (имя проекта сохраняется с расширением *.prj). В ходе лабораторной работы сохранение проектов необходимо

осуществлять в рабочий каталог среды Concept: C:\ Concept \ Project \ «№ группы \ «Имя».

Для предотвращения потери данных следует время от времени повторять процедуру сохранения проекта (File → Save). Для внесения изменений и дополнений сохраненный проект в дальнейшем открывается через меню File → Open project.

3.1.3. Конфигурирование контроллера

Правильное задание конфигурации, которое определяется устанавливаемым комплектом модулей ввода-вывода, является необходимым требованием нормальной работы ПЛК.

В рамках данной лабораторной работы, когда нет реального контроллера, принимается следующий **вариант конфигурации**:

- **ПЛК** серии **Quantum,** тип **140 CPU 434 12**;
- модули ввода/вывода (I/O Map):
  - ***дискретного ввода***, тип ***DDI-153-10*** (32 канала);
  - ***дискретного вывода***, тип ***DDO-353-00*** (32 канала);
  - ***аналогового ввода***, тип ***AVI-030-00*** (8 входов);
  - ***аналогового вывода***, тип ***AVO-020-00*** (4 выхода).

Для задания конфигурации ПЛК используется меню Project → Configurator. В открывшемся окне, вид которого дан на рис. 5, необходимо выбрать тип центрального процессора (двойной щелчок в поле Type).

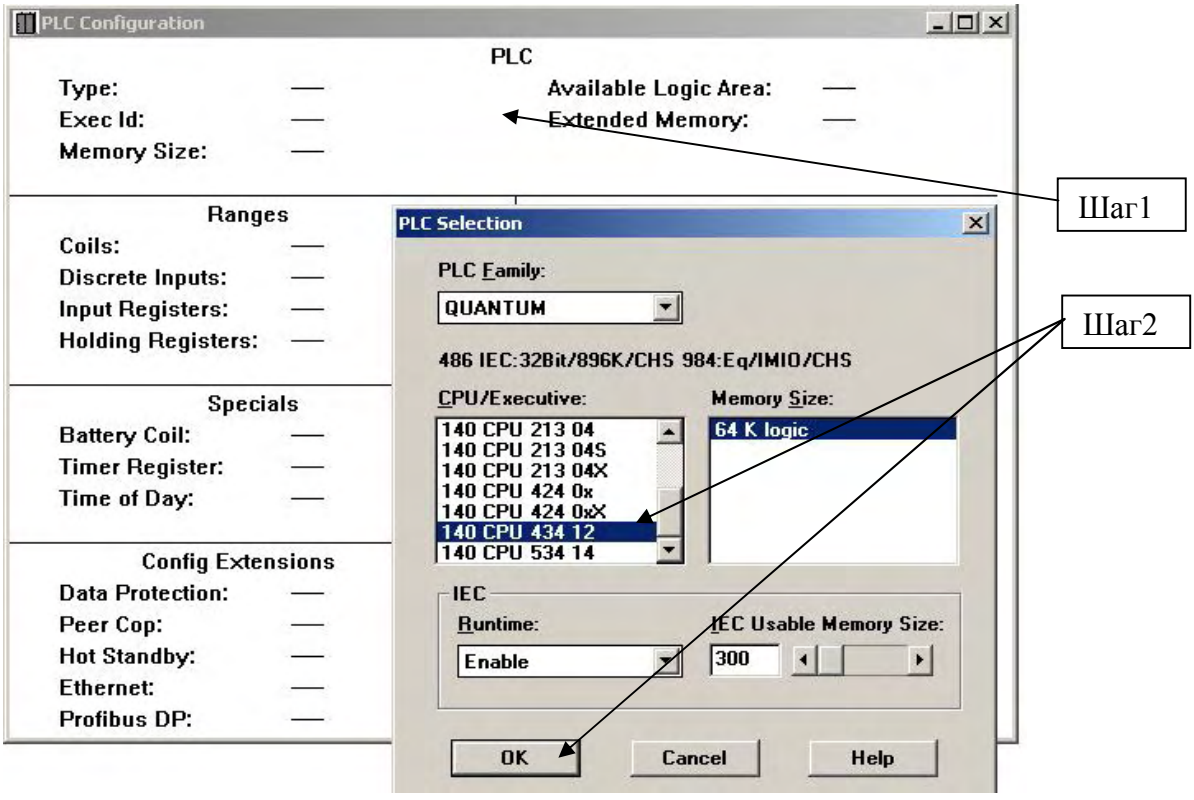


Рис. 5. Окно конфигурации ПЛК

Далее выполняется конфигурация модулей ввода/вывода через меню Configure → I/O map (рис. 6), согласно указанному выше списку модулей.

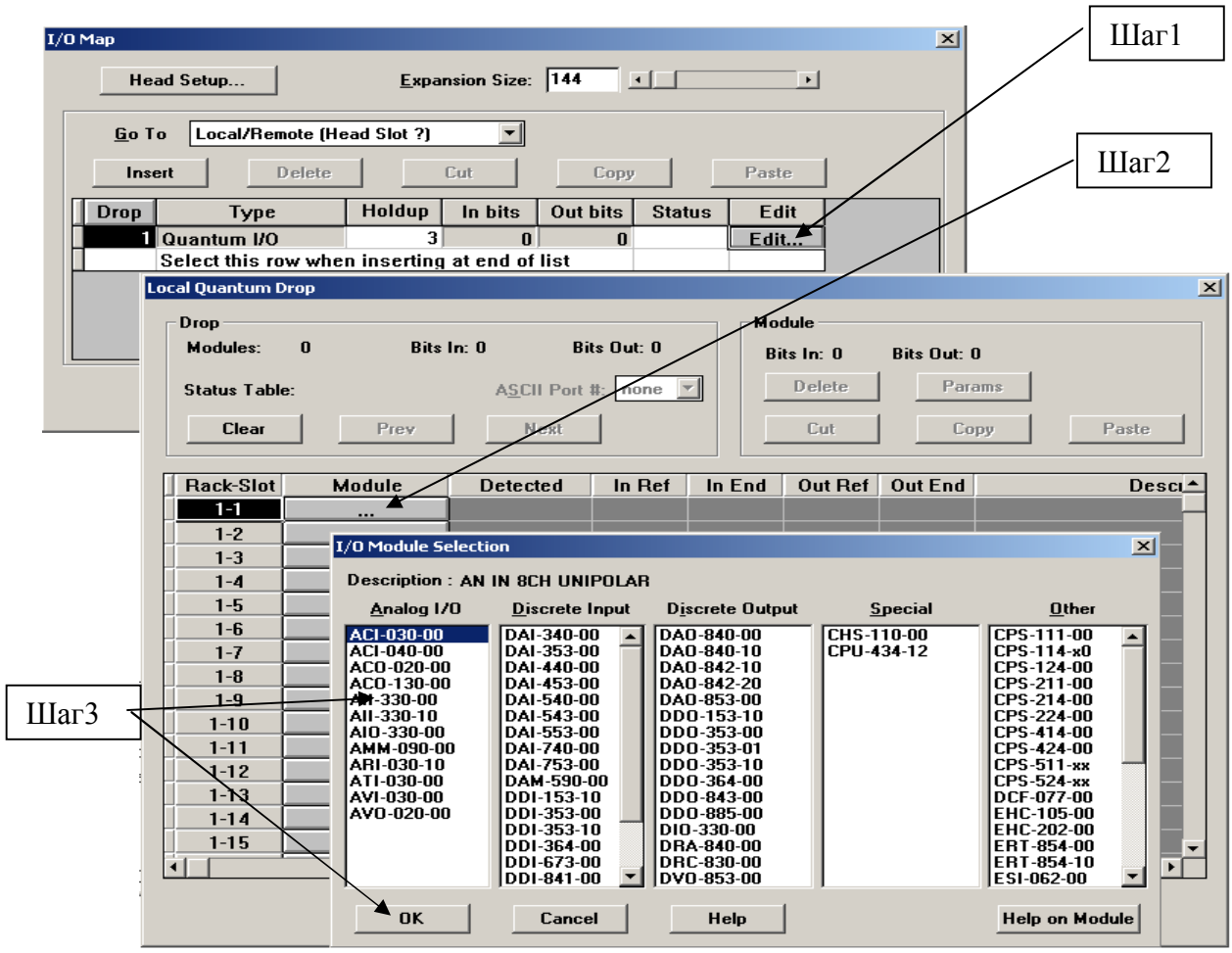


Рис. 6. Порядок заполнения корзины модулями УСО

В окне I/O map можно скомпоновать несколько корзин. Для введения предложенной выше конфигурации необходимо с помощью кнопки Edit вызвать окно Local Quantum Drop, в котором для каждого модуля следует указать начальный адрес области входов/выходов согласно их форматам (идентификатор типа входов/выходов - см. п. 2.2) и нажать клавишу Enter. Так для модуля дискретного ввода ***DDI-153-10*** (32 канала) необходимо заполнить доступное поле **In Ref** , указав первую цифру адреса модуля "1+Enter". Тогда компилятор автоматически укажет конечный адрес области, зарезервированной за этим модулем – 100032 (на 32 канала). Если аналогичных модулей несколько, то нужно указать начальный адрес целиком. После завершения конфигурирования проект необходимо сохранить.

3.2. Лабораторная работа 1
Моделирования объекта первого порядка с запаздыванием

3.2.1. Цель работы

Целью работы является создание модели объекта, представленного апериодическим звеном первого порядка с запаздыванием. Для получения переходной характеристики на вход модели объекта следует подать возмущающее воздействие в виде ступенчатой единичной функции. Структурная схема разрабатываемой модели приведена на рис. 7.

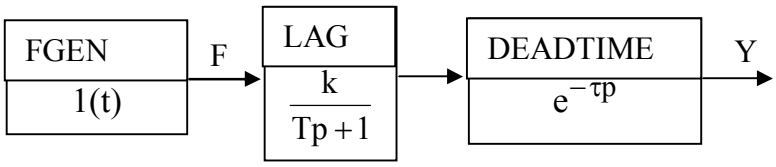


Рис. 7. Структурная схема моделируемого объекта управления, где F – возмущающее воздействие, а Y – результат моделирования

Значения параметров объекта выбираются из табл. 2 согласно варианту, указанному преподавателем. Выполнить разработку на языке технологического программирования FBD

Параметры объекта и регулятора для лабораторных работ 1 и 2

Таблица 2

Вариант	**k**	**T, с**	**τ, с**	**$k_{ПИ}$**	**$T_{ПИ}$, с**
1	2	20	4	1,1	19,5
2	3	25	3	0,5	20
3	4	30	2	0,5	16,5
4	5	35	3	0,5	20
5	2	35	3	0,8	19
6	3	30	2	0,6	20
7	4	25	4	0,5	20
8	5	20	3	0,6	21

3.2.2. Создание новой секции в навигаторе проекта

Программирование ведется в разделах проекта – секциях. Каждый проект может содержать от одной до 16000 секций (section). В отдельно взятой секции может использоваться только один язык технологического программирования, при этом весь проект может состоять из секций, написанных на разных языках.

Для создания секции выбирается команда меню File → New section. В появившемся диалоговом окне необходимо указать язык программирования для этой секции и имя секции (Section name). Причем имя секции должно обязательно включать буквенную комбинацию.

Для перехода между секциями, создания секций или их удаления используется навигатор проекта (Project → Project Browser), вид которого дан на рис. 8.

В навигаторе указывается порядок выполнения отдельных секций в программном цикле и показывается язык написания секций и их названия. Для проекта, изображенного на рис. 8, секции будут обрабатываться в следующей последовательности: Object_1 → Object_2 → Object_3.

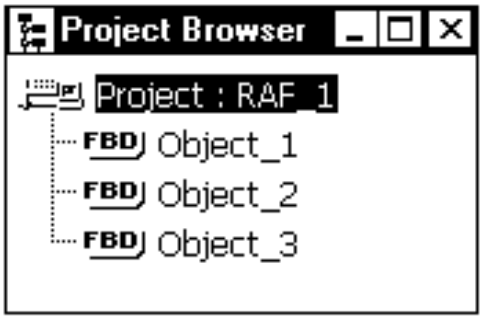


Рис.8. Навигатор проекта

3.2.3. Программирование

В приведенной на рис. 7 структуре модели объекта выделены 3 функциональных блока: блок формирования ступенчатого возмущающего воздействия, блок апериодического звена, блок чистого запаздывания. Однако, чтобы увидеть одновременно тренды выходных переменных всех блоков в структуре рис. 7 на одном графике, потребуется их собрать на одном функциональном блоке. Это объясняется тем, что Concept позволяет вывести на один график тренды только переменных, связанных с одним блоком. Функциональным блоком, позволяющим собрать интересующие переменные на своих входах, является блок мультиплексора **MUX_INT**.

Библиотека готовых функций и функциональных блоков среды Concept содержит все указанные блоки в готовом виде. Для того чтобы ими воспользоваться, необходимо выбрать команды меню Objects → FFB selection или нажать на кнопку  . В меню Select FFB (рис.) в окне, обозначенном FFB, набрать название интересующего блока (или хотя бы первую букву названия), найти в появившемся алфавитном списке нужный блок, например FGEN, и установить его в желаемую точку рабочего поля секции. Установка производится нажатием левой кнопки «мыши». После установки блока не забудьте нажать кнопку отмены  , иначе можно установить еще несколько копий.

Рис. 9. Окно библиотеки функций и функциональных блоков

Описание используемых в этой работе блоков дано в Приложении В.Для каждого блока необходимо задать несколько параметров, а также декларировать входные и выходные переменные. Для этого необходимо описать их в редакторе переменных, вид которого дан на рис. 10.

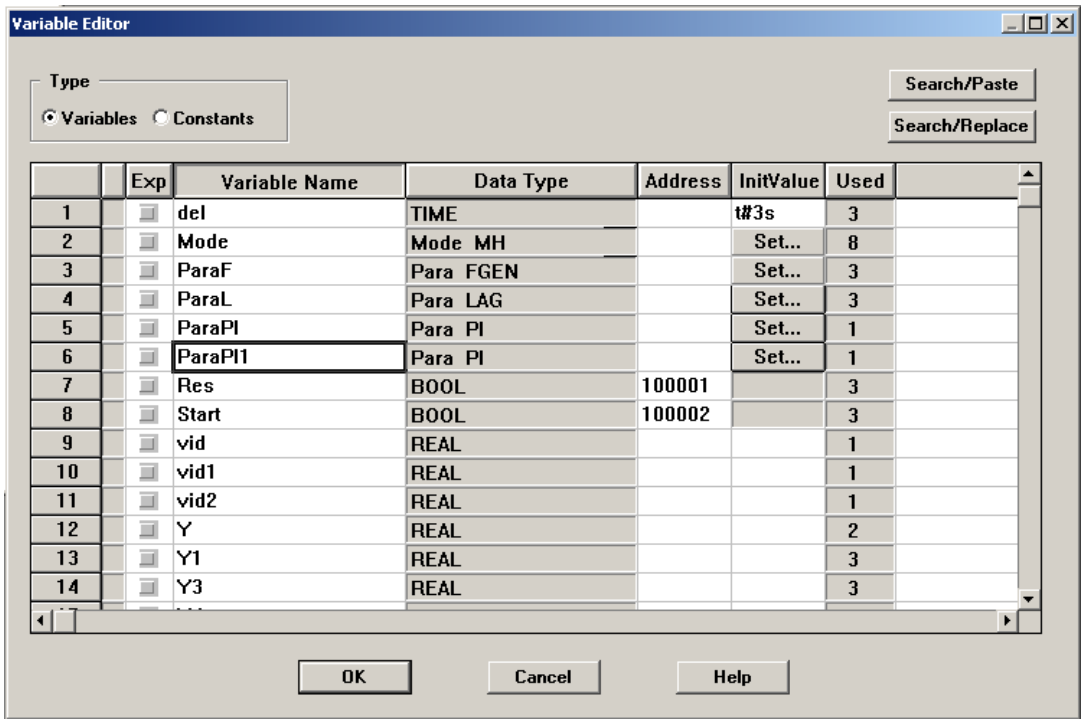


Рис. 10. Редактор описания переменных

Окно можно вызвать, воспользовавшись командами меню Project → Variable declarations или кнопкой основного меню  . Каждой переменной назна-

чается имя, тип, и, если это необходимо, адрес, начальное значение и комментарий. Переменные типа «Para_...», «Mode» являются комплексными и содержат несколько обычных переменных, список которых можно просмотреть, а при необходимости редактировать по ссылке «Set…» (рис. 10). Например, в списке параметров блока FGEN для выбора ступенчатой функции необходимо указать ее номер (Func_no = 1) и амплитуду (Amplitude = 1), остальные параметры можно оставить без изменений.

Переменные, которые обязательно нужно обозначить и декларировать, набраны в Приложении В в описаниях функциональных блоков полужирным шрифтом.

Связь между блоками можно устанавливать графически (кнопка ) или, что лучше, так как не загружает поле секции линиями связи, через обозначение подлежащих связи точек блоков одинаковыми, предварительно задекларированными переменными.

Назначение переменных входам/выходам блока осуществляется посредством двойного клика по определяемой связи. В открывшемся окне, показанном на рис. 11, можно выбрать переменную из списка (если она была предварительно задекларирована), задать числовое значение без описания переменной или указать конкретный адрес переменной.

После завершения программирования секции можно выполнить проверку компилятором (выявить синтаксические ошибки). Компилятору можно назначить для проверки весь проект или отдельную секцию.

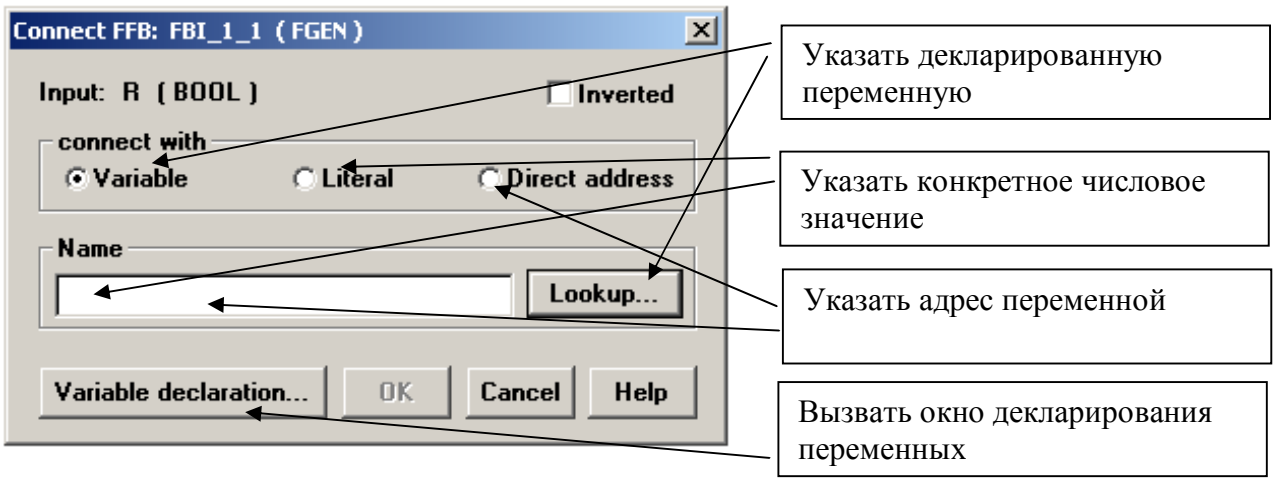


Рис. 11. Окно назначения переменных

Для этого существуют команды меню Project → Analyze section и Project → Analyze program. **Все ошибки**, выявленные в процессе программирования, должны быть обязательно исправлены, так как это необходимое требование для загрузки готовой программы в ПЛК. При этом следует отметить, что предупреждения, появившиеся в процессе проверки, ошибками не являются, но просматривать их все же рекомендуется.

3.2.4. Загрузка проекта в 32-битный эмулятор

Перед началом работы с эмулятором его необходимо загрузить. Для соединения с ним используется команда меню Online → Connect . В открывшемся окне, вид которого приведен на рис. 12, необходимо указать тип протокола обмена (IEC Simulator (32-bit)), IP-адрес ПК (LocalHost) и что загружается при соединении с эмулятором контроллера (обычно это Change Configuration).При выполнении соединения появляется окно эмулятора (см. рис. 4).

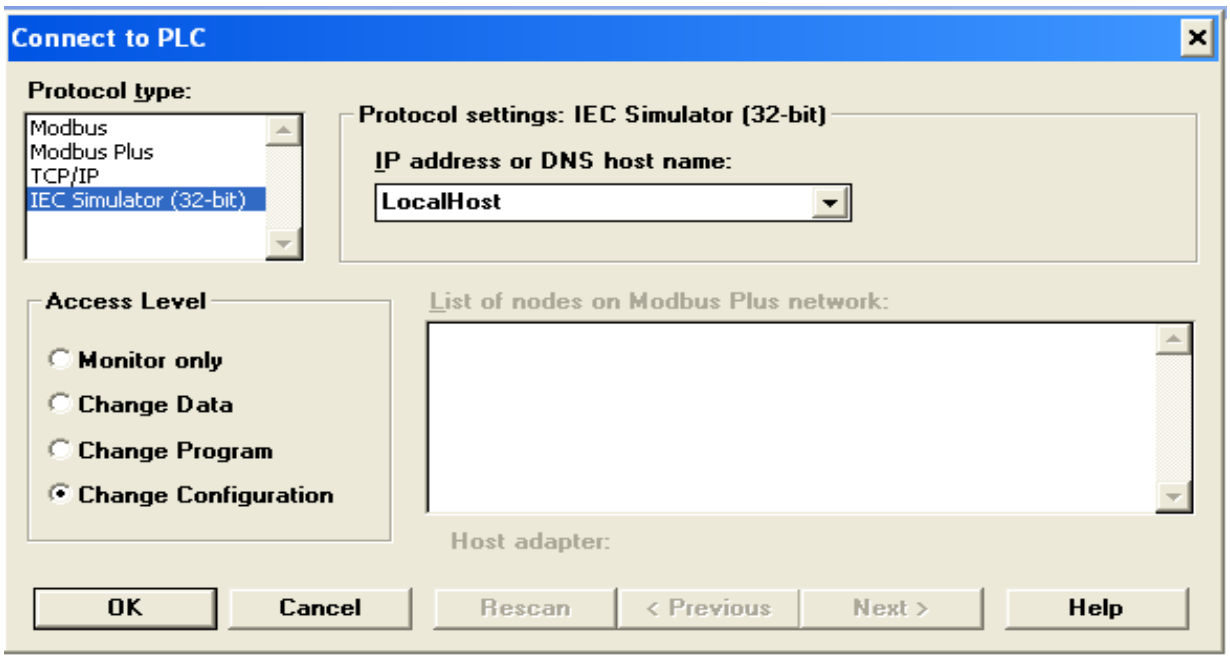


Рис. 12. Окно соединения с 32-битным эмулятором контроллера

Для загрузки проекта в эмулятор используется команда меню Online → Download.

3.2.5. Тестирование работы проекта

Для контроля и отладки загруженной в эмулятор ПЛК программы используется функция анимации, позволяющая интерактивно наблюдать за состоянием переменных, шагов, перемещений и т.д. Функция анимации запускается для каждой секции программы отдельно.

После того как проект загружен в эмулятор, необходимо в навигаторе проекта открыть окно анализируемой секции, далее – выделить все элементы программы (функции, функциональные блоки, переменные, линии связи и т.д.) с помощью команды меню Edit → Select all или сочетания клавиш Ctrl_A, после этого запустить анимацию командой меню Online → Animate Selected или кнопкой  . Желательно сразу подать возмущение, инициализировав блок FGEN включением на панели эмулятора квадратика, соответствующего адресу, данному команде Start.

Изменение состояния переменных типа BOOL будет отражено цветом (красный цвет – сигнал соответствует 0, а зеленый – соответствует 1). Все переменные других типов данных будут отражаться в виде числовых значений, динамически изменяющихся.

Для того чтобы просмотреть изменение переменных во времени в виде трендов, необходимо двойным щелчком левой кнопки «мыши» выделить интересующий функциональный блок (в работе это мультиплексор **MUX_INT**, на входы которого поданы интересующие переменные). В открывшемся окне нажать кнопку Advanced. В окне Advanced Monitor нужно выбрать желаемые для просмотра переменные («мышкой», удерживая кнопку Ctrl, выделить эти переменные) и нажать кнопку Graphics, что приведет к открытию окна Concept Graphic Tool (рис. 13).

В подменю Scaling установите удобный масштаб (например, поставив для всех переменных max=5), биполярный вид графика и сетку. Для включения в график еще одной или нескольких переменных используйте команды DDE → Connection.

Если при тестировании программы средствами, доступными в меню Online, или программистом будут выявлены ошибки, их необходимо исправить, после чего вновь загрузить проект в ПЛК командой Online → Download changes. После нескольких операций внесения изменений в программу и повторной ее загрузки в ПЛК, рекомендуется провести операцию оптимизации той области памяти, в которой хранятся тексты программ.

Порядок создания проекта в целом и отдельных секций приведен в Приложении Г.

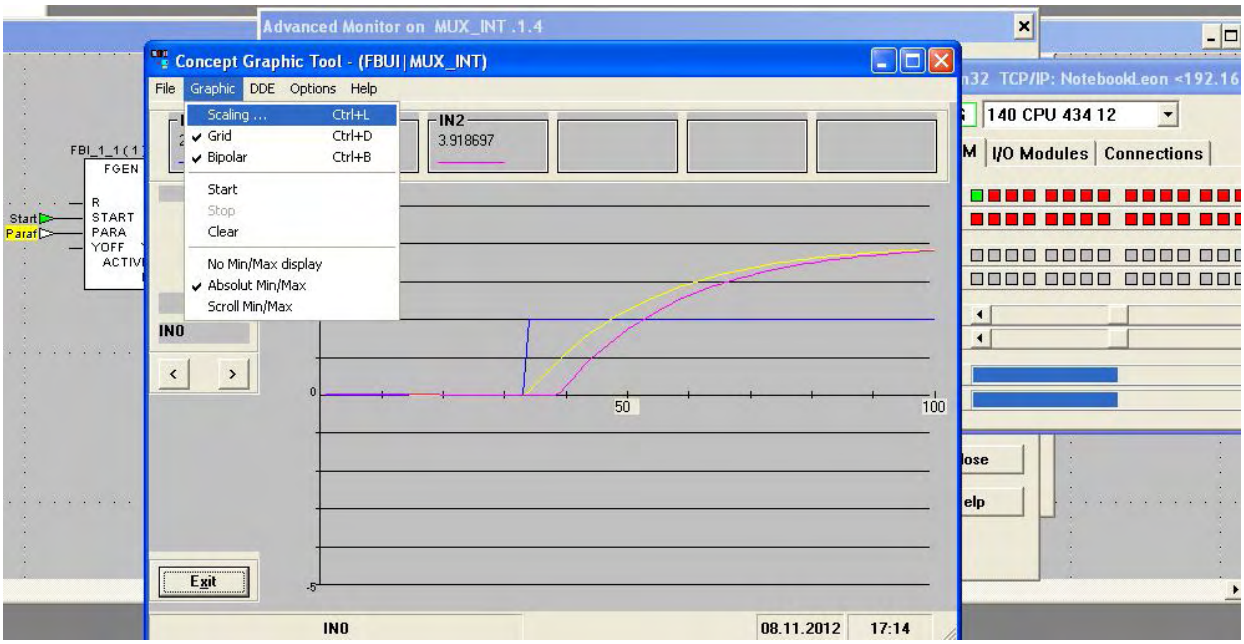


Рис. 13. Окно анимации Concept

3.3. Лабораторная работа 2
Дополнение проекта моделью одноконтурной АСР

Цель работы – создать модель одноконтурной АСР с объектом первого порядка с запаздыванием и ПИ-регулятором. Структурная схема АСР приведена на рис. 14. Значения параметров объекта и настроек регулятора выбираются из табл. 2 согласно варианту, указанному преподавателем.

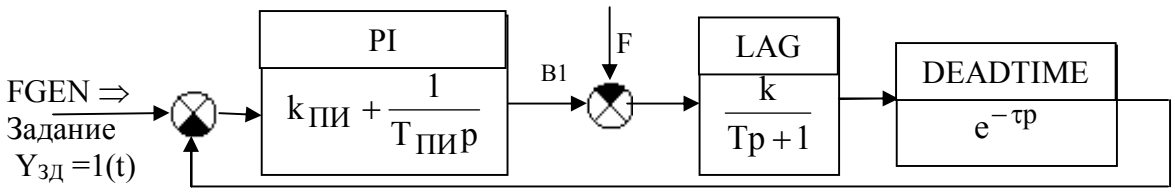


Рис. 14. Структурная схема одноконтурной АССР: $Y_{ЗД}$ – заданное значение; F – возмущение

Работу можно выполнить в той же секции проекта, что и предыдущую, дополнив структуру лабораторной работы 1 тремя блоками: PI-регулятором, вычитателем B1 (блок SUB_REAL) и еще одним блоком FGEN для задания возмущения в канал управления. Первый блок вычитания – текущего значения регулируемой переменной из заданного – реализован внутри блока регулятора PI.

Для описания объекта используются те же функциональные блоки, что и для предыдущей лабораторной работы (LAG, DEADTIME). Для этих блоков можно использовать уже созданные задекларированные переменные, в том числе и переменные PARA и MODE, так как параметры объекта сохраняются.

Формирование задающего и возмущающего воздействий можно реализовать двояко:

- в виде ступенчатой функции с помощью блоков FGEN (для старта блоков необходимо использовать разные переменные с разными адресами);
- путем назначения переменным задания $Y_{ЗД}$ и возмущения F (в данном случае обе - типа REAL) внешнего адреса из области 300001-399999. В последнем случае задание и возмущение изменяются непосредственно из окна эмулятора перемещением ползунков в строках 3х, как описано в п. 2.2;
- комбинированно: для возмущения по заданию оставить FGEN, а для возмущения в канале управления - задание из области 3х.

Для просмотра трендов выходных переменных разных блоков, по аналогии с первым заданием, нужно использовать блок MUX_INT. На график вывести оба возмущения (Yзд и F), выходы регулятора и объекта (без запаздывания - с блока LAG, с запаздыванием – с блока DEADTIME).

3.4. Лабораторная работа 3. Моделирование каскадной АСР

Цель работы – запрограммировать каскадную АСР с возмущением по заданию (рис. 15). Принять ПИД-регулятор для основного канала и ПИ-регулятор – для вспомогательного. Получить переходные характеристики 1-го и 2-го объектов, а также основного и вспомогательного регуляторов.

Разработку выполнить в отдельной секции на языке технологического программирования FBD.

Для описания объектов по основному и вспомогательному каналам управления использовать апериодические звеня первого порядка с запаздыванием:

- **параметры 1-го объекта** (основной канал): $k_1 = 1$, $T_1 = 10$с, $\tau_1 = 4$с;
- **параметры 2-го объекта** (вспомогательный канал): $k_2 = 1$, $T_2 = 4$с, $\tau_2 = 0{,}4$с;
- **настройки ПИД-регулятора**: $k_{P1} = 1{,}25$, $T_{И1} = 11{,}8$с, $T_{Д1} = 1{,}4$с;
- **настройки ПИ-регулятора**: $k_{P2} = 1$, $T_{И2} = 3{,}6$с.

Скомпилировать проект и выполнить его тестирование с помощью эмулятора PLCSIM32.

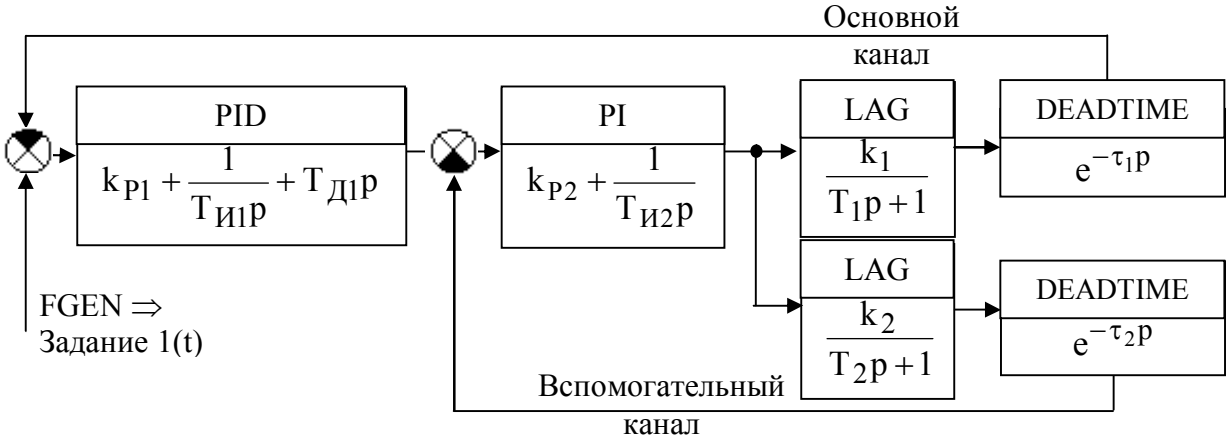


Рис. 15. Структурная схема каскадной АСР

Краткая схема действий для разработки проекта дана в Приложении Г.

Для моделирования объекта управления, ПИ-регулятора и формирования задания используются те же функциональные блоки, что и в п. 3.4 (FGEN, LAG, DEADTIME, PI).

Для реализации ПИД-закона регулирования также можно воспользоваться готовым блоком регулятора PID, описание которого приведено в Приложении В. Задающее воздействие ввести в виде ступенчатой функции.

Для вывода трендов применить мультиплексор MUX_INT. На график вывести сигнал с блока FGEN, выходы регуляторов и обоих объектов.

3.5. Лабораторная работа 4. Моделирование предиктора Смита

Цель работы – создать и сконфигурировать проект, как это кратко описано в Приложении Г, с моделью АСР с предиктором Смита, в котором использовать ПИ-закон регулирования.

Структурная схема АСР приведена на рис. 16. Параметры объекта и настройки регулятора даны в табл. 3.

Разработку выполнить в отдельной секции на языке технологического программирования LD.

Скомпилировать проект и выполнить его тестирование с помощью эмулятора PLCSIM32, как показано в п. 3.2.5.

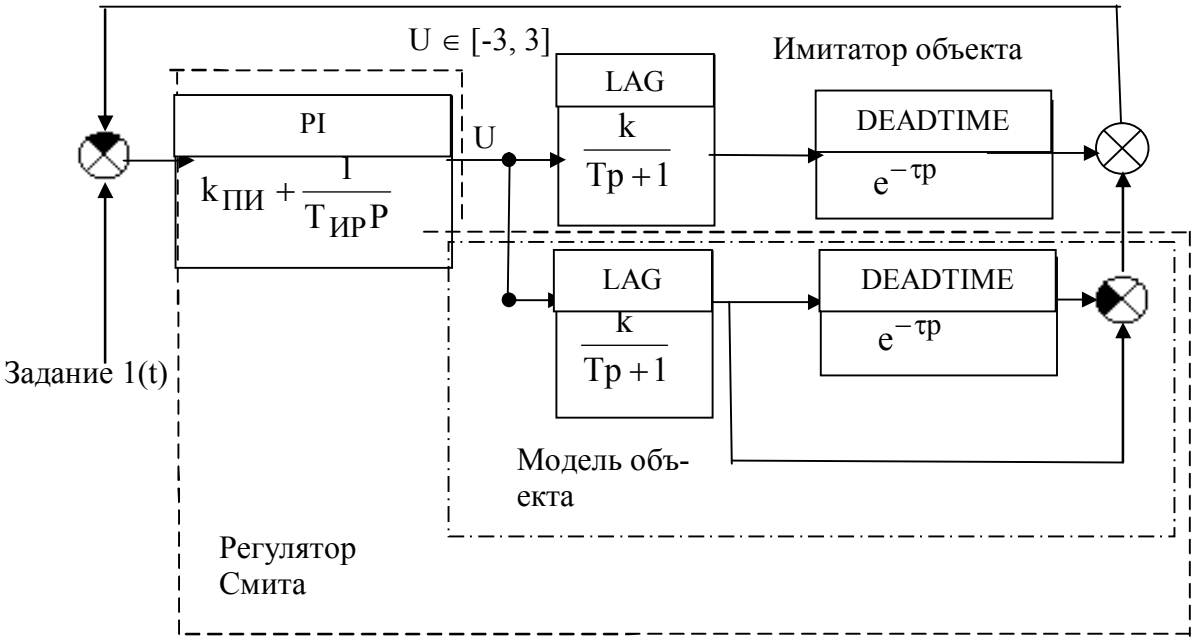


Рис. 16. Структурная схема АСР с регулятором Смита

Параметры объекта и регулятора для АСР с регулятором Смита

Таблица 3

Объект			Регулятор	
k	T, с	τ,с	$k_{ПИ}$	$T_{ИП}$, с
1	5	5	1	0,36

Для программирования использовать готовые блоки из библиотек среды Concept. Блок сложения ADD_REAL работает аналогично блоку вычитания, описанному ранее.

Получить переходные характеристики объекта, регулятора и модели при подаче возмущения по заданию. На график вывести три сигнала: выходы FGEN, PI-регулятора и объекта. Убедиться, что АСР устойчиво работает при

хорошей идентификации модели объекта, т.е. при совпадении параметров модели и объекта. Для этого по заданию преподавателя разбалансировать параметры объекта и модели.

3.6. Лабораторная работа 5
Модель АСР с использованием широтно-импульсной модуляции сигнала на выходе регулятора

Цель работы – создать секцию моделирования одноконтурной АСР с ПИД-регулятором и с использованием широтно-импульсного модулирования (ШИМ) сигнала. Для АСР в качестве объекта принять апериодическое звено первого порядка с запаздыванием. Параметры объекта и настройки регулятора даны в табл. 4. Структурная схема АСР приведена на рис. 17.

Разработку выполнить в одной секции на языке технологического программирования LD или FBD по заданию преподавателя.

Параметры объекта и настройки ПИД-регулятора для АСР с ШИМ сигнала на выходе регулятора

Таблица 4

Объект			Регулятор		
k	T	τ	$k_Р$	$T_И$	$T_Д$
1	15с	5с	1,5	10с	0,5с

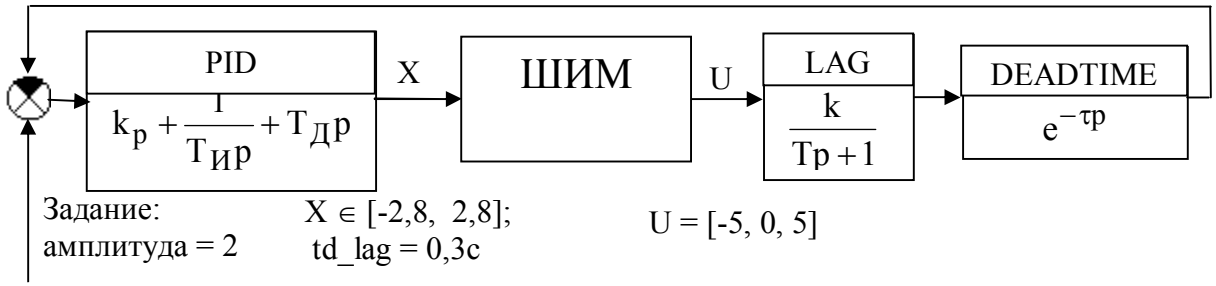


Рис. 17. Структурная схема АСР с ШИМ сигнала на выходе регулятора

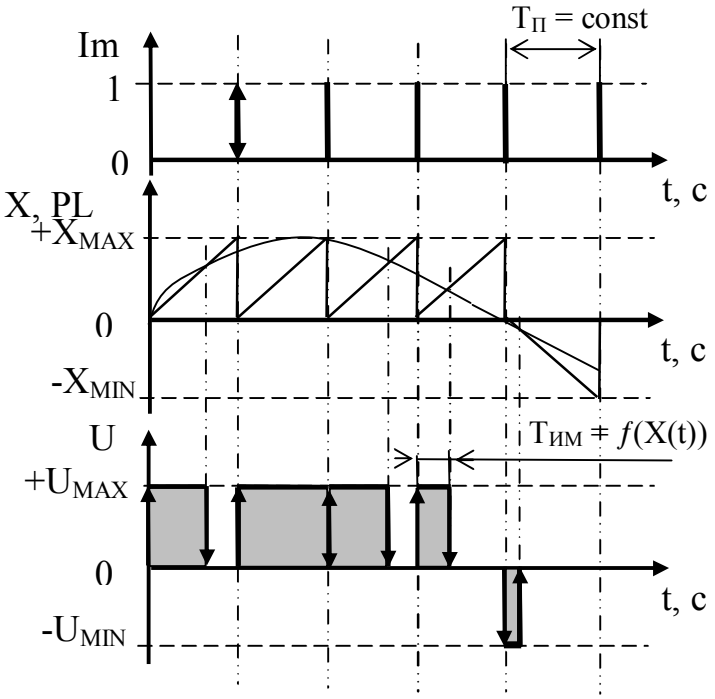
ШИМ сигнала предполагает квантование по времени непрерывного сигнала с регулятора X в дискретный U, повторяющиеся с постоянным интервалом, как показано на рис. 18.

Из рисунка следует, что для формирования импульса ШИМ необходимо преобразовать выходной сигнал регулятора во временной интервал. Это выполняется сравнением выходного сигнала регулятора с пилообразным сигналом: импульс длится, пока выполняется неравенство Uрег≥Uпилы.

В лабораторной работе необходимо получить переходные характеристики объекта, регулятора и ШИМ-сигнала при ступенчатом возмущении по каналу задания. Скомпилировать проект и выполнить тестирование его с помощью

симулятора PLCSIM32. Вывести на график сигналы с FGEN, PID-регулятора, выхода объекта, а также пилообразный сигнал и выход ШИМа (переменная U – рис. 17)

Для программирования использовать готовые блоки из библиотек среды Concept, описание которых дано в Приложении В.



Формирование тактовых импульсов Im с периодом $T_П$, определяющим квантование для модулятора

Формирование пилообразного сигнала PL с периодом, равным $T_П$, и амплитудой - $|X_{MAX}|$. Сравнение выхода регулятора X с пилой с учетом знака X.

Формирование сигнала ШИМ U. Подъем импульса в зависимости от знака X в момент появления тактового импульса периода ($0 \rightarrow U_{MAX}$ или $0 \rightarrow -U_{MIN}$). Отпускание в момент равенства сигнал пилы и выхода регулятора X.

Рис. 18. Пример работы ШИМ сигнал:
$T_{ИМ}$ – выходной сигнал ШИМ, $T_Ц$ - время цикла

Последовательность разработки модулятора может быть следующей:

1) Сформировать пилообразный сигнал **PL** с постоянным периодом $\mathbf{T_П}$, равным 3 секундам (период работы ШИМ) и амплитудой, сопоставимой с верхней границей диапазона изменения выходной величины регулятора $\mathbf{X} \in$ [-2,8, +2,8]. Пилу можно сформировать с помощью блока FGEN (параметры блока *Para_FGEN*: *Func_no* = 3; *Amplitude* = 3; *Halfperiod* = 3c; *Unipolar* = 1).
2) Определить знак выходного сигнала регулятора **X** и сохранить эту информацию в виде состояния переменной типа BOOL. Функцию сравнения можно найти в библиотеках Concept: GT_REAL – функция, которая присваивает выходу Out = 1, если In1>In2>…>InN.
3) Найти абсолютное значение выходной величины регулятора |**X**| (функция ABS_REAL, Out = |In|).
4) Сравнить текущее значение пилообразного сигнала **PL** и абсолютное значение выхода регулятора |**X**|. Результат сравнения сохранить, как переменную типа BOOL (функция GE_REAL, которая присваивает выходу Out = 1, если In1≥In2≥…≥InN).

5) Установить два блока SEL для выбора полярности импульса на выходе ШИМ. Блоки выполняют функции аналоговых мультиплексоров. В зависимости от входного сигнала G типа BOOL формируется выход: если G=0, то Out = In0, а если G=l, то Out = In1. В свойствах блока (*двойной щелчок по графическому обозначению блока)* установить «галочку» EN/ENO, вследствие чего на входе блока появится еще одна переменная EN - разрешение работы блока.

6) Задать на входах блоков SEL следующие пары значений: SEL1 - 0 и 5; SEL2 - 0 и -5.

7) На вход EN привязать переменную, хранящую знак выходного сигнала регулятора, причем для SEL2 эта переменная должна быть инвертирована. Для этого можно использовать кнопку инверсии и, нажав на нее, установить знак инверсии (кружок) на линии подключения соответствующего параметра.

8) На вход G привязать переменную, содержащую информацию о сравнении абсолютной величины выхода регулятора |**X**| и пилы **PL.**

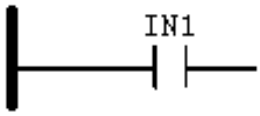
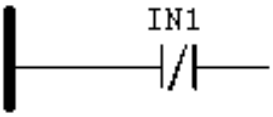
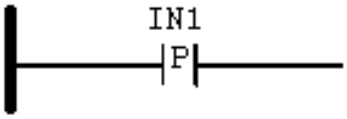
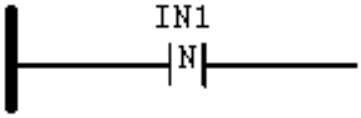
9) Выходы обоих блоков SEL обозначить одной и той же переменной, которая является входом объекта U (она же является выходом блока ШИМ). После загрузки программы в контроллер будет выдаваться предупреждение об использовании двумя блоками одного адреса, в качестве выходной переменной. Предупреждение можно игнорировать, так как по логике программы будет всегда активен только один блок

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного программирования / под ред. В.П.Дьяконова. – М.: Солон-Пресс, 2004. – 256 с.
2. Рош О.Л. Библия по техническому обеспечению. – Минск: Динамо, 1992. – 416 с.

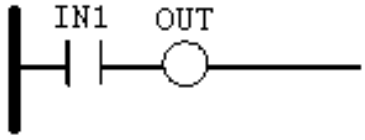
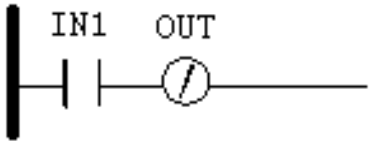
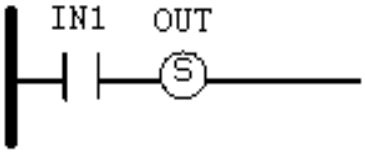
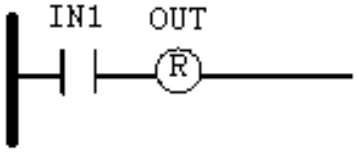
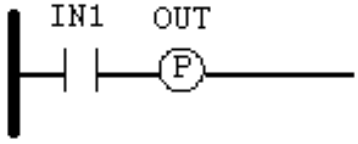
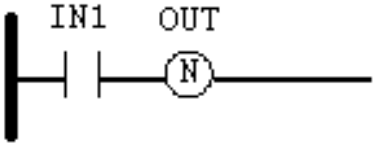
ПРИЛОЖЕНИЕ А
(справочное)

Атрибуты контактов как элементов языка LD

Наименование	Изображение	Примечание
Нормально открытый		Сигнал с левой от контакта горизонтальной связи копируется на правую сторону, если связанная с контактом переменная хранит значение логической 1 (контакт замкнут). В противном случае на выходе контакта 0 (контакт разомкнут).
Нормально закрытый		Сигнал с левой от контакта горизонтальной связи копируется на правую сторону, если связанная с контактом переменная хранит значение логического нуля (контакт замкнут). В противном случае на выходе контакта 1 (контакт разомкнут).
Контакт включения по положительному фронту		Контакт, реагирующий на переходы 0→1. Если в течение одного программного цикла связанная с контактом переменная меняет свое значение с 0 на 1, то сигнал на выходе контакта принимает значение 1.
Контакт включения по отрицательному фронту		Контакт, реагирующий на переходы 1→0. Если в течение одного программного цикла связанная с контактом переменная меняет свое значение с 1 на 0, то сигнал на выходе контакта принимает значение 1.

ПРИЛОЖЕНИЕ Б
(справочное)

Атрибуты катушек как элементов языка LD

Наименование	Изображение	Примечание
Установка катушки за контактом		Если контакт IN1 замыкается, то переменной OUT, связанной с катушкой, присваивается значение 1.
Катушка типа Negated		Если контакт IN1 разомкнут, то переменной OUT, связанной с катушкой, присваивается значение 1; при замыкании контакта OUT=0.
Катушка типа Set		При замыкании контакта IN1 переменной OUT присваивается значение 1, которое сохраняется и далее при размыкании контакта. Значение переменной OUT может быть сброшено из 1 в 0 катушкой Reset.
Катушка типа Reset		При замыкании контакта IN1 переменной OUT присваивается значение 0, которое сохраняется и далее при размыкании контакта. Значение переменной OUT может быть установлено в 1 катушкой Set.
Катушка обнаружения переходов $0 \rightarrow 1$		Значение переменной OUT будет переведено в 1, если в течение одного программного цикла сработает контакт IN1, т.е. состояние левосторонней связи перебросится из 0 в 1.
Катушка обнаружения переходов $1 \rightarrow 0$		Значение переменной OUT будет переведено в 1, если в течение одного программного цикла разомкнется контакт IN1, т.е., состояние левосторонней связи перебросится из 1 в 0.

ПРИЛОЖЕНИЕ В
(обязательное)

Описание функциональных блоков, необходимых для выполнения лабораторных работ

FGEN *(CONT_CLT⇒CLC_PRO)* – **генератор функций**

Параметр	Тип	Комментарий
R	BOOL	1 ⇒ сброс
Start	BOOL	1 ⇒ старт генератора функций
Para	Para_FGEN	Параметры функции: **Func_no** ⇒ INT – выбор типа функций ⇒ 1 (Step) **Amplitude** ⇒ REAL – амплитуда функции ⇒ 1 Halfperiod ⇒ TIME – продолжительность полупериода ⇒ по умолчанию T_off ⇒ TIME – постоянная времени простоя ⇒ по умолчанию T_rise ⇒ TIME – постоянная времени нарастания ⇒ по умолчанию T_acc ⇒ TIME – время сглаживания ⇒ по умолчанию unipolar ⇒ BOOL – 1 ⇒ униполярный сигнал, 0 ⇒ биполярный сигнал ⇒ н по умолчанию
Yoff	REAL	Смещение выхода
Y	REAL	Выход генератора
Active	BOOL	1 ⇒ генератор функций активен
N	INT	Число периодов со старта (счетчик циклов)

LAG (*CONT_CLT⇒CLC_PRO*) – **апериодическое звено первого порядка**

Параметр	Тип	Комментарий
X	REAL	Входная величина
MODE	Mode_MH	Тип рабочего режима: **Man** ⇒ BOOL (ручной режим) **Halt** ⇒ BOOL (режим останова) **0, 0** – автоматический режим 1, 0(1) – ручной режим (YMAN – задается вручную и передается на Y) 0, 1 – режим остановки, на Y подается последнее расчетное значение
Para	Para_LAG	Параметры объекта: **Gain** ⇒ REAL – коэффициент усиления **Lag** ⇒ TIME – постоянная времени
YMAN	REAL	Выходное значение, задаваемое вручную
Y	REAL	Выход объекта

DEADTIME (*CONT_CLT⇒CLC_PRO*) – **задержка по времени** (звено чистого запаздывания). Этот блок имеет 128-элементный буфер данных (хранит 128 последних значений X)

Параметр	Тип	Комментарий
X	REAL	Входная величина
MODE	Mode_MH	Тип рабочего режима: **Man** ⇒ BOOL (ручной режим) **Halt** ⇒ BOOL (режим останова) **0, 0** – автоматический режим 1, 0(1) – ручной режим (YMAN – задается вручную и передается на Y, READY=1) 0, 1 – режим остановки, на Y подается последнее расчетное значение, но буфер заполняется
T_DELAY	TIME	Время запаздывания (формат N секунд ⇒ **t#Ns**)
YMAN	REAL	Выходное значение, задаваемое вручную
Y	REAL	Выход объекта
READY	BOOL	1(0) – внутренний буфер заполнен (не заполнен)

PI *(CONT_CLT⇒CLC_PRO)* – **ПИ-закон регулирования**.

Параметр	Тип	Комментарий
SP	REAL	Задание (уставка)
PV	REAL	Регулируемая величина
MODE	Mode_MH	Тип рабочего режима: **Man** ⇒ BOOL (ручной режим) **Halt** ⇒ BOOL (режим останова) **0, 0** – автоматический режим 1, 0(1) – ручной режим (YMAN – задается вручную и передается на Y) 0, 1 – режим остановки, на Y подается последнее расчетное значение
Para	Para_PI	Параметры ПИ-закона: **Gain** ⇒ REAL – коэффициент усиления **Ti** ⇒ TIME – постоянная времени **Ymax** ⇒ REAL – максимальное значение регулирующего воздействия **Ymin** ⇒ REAL – минимальное значение регулирующего воздействия Ymax > Ymax
YMAN	REAL	Регулирующее воздействие, вводимое вручную
Y	REAL	Регулирующее воздействие
ERR	REAL	Ошибка регулирования ERR = SP-PV
STATUS	Stat_MAXMIN	Состояние регулирующего воздействия: Qmax – BOOL ⇒ 1, если Y ≥ Ymax Qmin – BOOL ⇒ 1, если Y ≤ Ymax

MUX_INT (*IEC⇒Selection*) – **мультиплексор**

В зависимости от значения K на выход блока подается один из входов:

К = 0 ⇒ на выход подается сигнал IN0;
К = 1 ⇒ на выход подается сигнал IN1;
К = n ⇒ на выход подается сигнал INn.

n∈[1, 32] Дополнительные входы показываются при растяжении графического изображения блока вниз. IN ⇒ BOOL, INT, REAL, TIME, WORD и т.д.

PID *(CONT_CLT⇒CLC_PRO)* – **ПИД-закон регулирования**

Параметр	Тип	Комментарий
SP	REAL	Задание (уставка)
PV	REAL	Регулируемая величина
MODE	Mode_PID	Тип рабочего режима: **Man** ⇒ BOOL (ручной режим) **Halt** ⇒ BOOL (режим останова) **0, 0** – автоматический режим 1, 0(1) – ручной режим (YMAN – задается вручную) 0, 1 – режим остановки, на Y подается последнее значение **en_p** ⇒ BOOL – установить 1 при активизации П-составляющей **en_i** ⇒ BOOL – установить 1 при активизации И-составляющей **en_d** ⇒ BOOL – установить 1 при активизации Д-составляющей d_on_pv ⇒ BOOL ⇒ Ставится 1, если Д-составляющая берется от PV или 0, если Д-составляющая берется от ERR.
PARA	Para_PID	Параметры ПИД-закона: **Gain** ⇒ REAL – коэффициент усиления **Ti** ⇒ TIME – постоянная времени интегрирования **Td** ⇒ TIME – постоянная времени дифференцирования Td_lag ⇒ TIME – постоянная времени фильтра **Ymax** ⇒ REAL – максимальное регулирующее воздействие **Ymin** ⇒ REAL – минимальное регулирующее воздействие
YMAN	REAL	Регулирующее воздействие, устанавливаемое вручную
Y	REAL	Регулирующее воздействие
ERR	REAL	Ошибка регулирования ERR = SP-PV
STATUS	Stat_MAXMIN	Состояние регулирующего воздействия: Qmax – BOOL ⇒ 1, если Y ≥ Ymax Qmin – BOOL ⇒ 1, если Y ≤ Ymax

SUB_REAL *(IEC⇒Arithmetic)* – **функция вычитания** переменных типа REAL. В зависимости от типа обрабатываемых переменных осуществляется конкретизация функции SUB (REAL). Поддерживает сигнал EN/ENO.

ADD_REAL *(IEC⇒Arithmetic)* – **функция сложения** переменных типа REAL. В зависимости от типа обрабатываемых переменных осуществляется конкретизация функции SUB (REAL). Поддерживает сигнал EN/ENO.

GE_REAL *(CONT_CLT⇒CLC_PRO)* - **функция сравнения** переменных типа REAL. На выходе Out появляется единица (Out = 1), если In1≥In2≥…≥InN).

GT_REAL *(CONT_CLT⇒CLC_PRO)* - **функция сравнения** переменных типа REAL. На выходе Out появляется единица (Out = 1), если In1>In2>…>InN.

ABS_REAL *(IEC⇒Numerikal)* - **функция** возвращает абсолютную величину значения входной переменной (Out = |In|).

SEL *(IEC⇒Selection)*- **мультиплексор.** В зависимости от значения сигнала на входе G на выход блока подается один из входов:

если G=0, то на выход подается сигнал со входа In0 (Out = In0);

если G=l, то на выход подается сигнал со входа In1 (Out = I n1).

Поддерживает сигнал EN/ENO/ если TN=1, то работа блока разрешается; если EN=0, то работа блока запрещается. Переменная, поступающая на вход EN, может быть инвертирована.

ПРИЛОЖЕНИЕ Г
(обязательное)

Основные этапы создания, разработки и отладки проекта в среде Concept

1. Создание нового проекта – File → New project.
2. Сохранение проекта с присвоения ему имени – File → Save project as.
3. Описание аппаратной конфигурации:
   - открыть окно PLC Configuration;
   - по команде Configure → PLC Type выбрать семейство ПЛК (PLC Family) и тип центрального процессора (CPU/Executive);
   - по команде Configure → I/O map сформировать корзину периферийных модулей.
4. Создание новой секции – File → New section (указать ее имя и выбрать язык программирования).
5. Для использования готовых библиотек функций и функциональных блоков – команда Objects.
6. Для описания (декларирования) переменных, используемых в программе - Project → Variable declaration.
7. Сохранение программы – File → Save.
8. Загрузка 32-битного эмулятора – Online → Connect.
9. Загрузка созданной программы в эмулятор ПЛК – Online → Download.
10. Для анимации работы выбранного фрагмента программы – выделить все элементы фрагмента (Edit → Select all) и включить анимацию (Online → Animate Selected).
11. Для получения графиков переходных процессов выбрать двойным щелчком «мыши» требуемый функциональный блок и в открывшемся окне нажать кнопку Advanced, далее в окне Advanced Monitor указать требуемую переменную или группу переменных (Ctrl + курсор «мыши») и нажать кнопку Graphics .

Примечания:

1. Этапы конфигурации, программирования и сохранения программы можно выполнять в произвольном порядке.

2. Чтобы избежать потери данных, сохранение программы следует время от времени повторять в процессе сеансов конфигурации или программирования.

Содержание

Леон Абрамович Русинов
Ирина Викторовна Рудакова

ТЕХНОЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ CONCEPT

Методическое пособие

Редактор и корректор В.А.Басова
Техн. редактор Титова Л.Я. Темплан 2012 г., поз. 82

Подп. к печати 06.12.12 г. Формат 60x84/16. Бумага тип. № 1.
Печать офсетная. Усл. печ. л. 2,0; уч.-изд. л. 2,0. Тираж 50 экз. Изд.№. 82.
Цена «С». Заказ

Ризограф Санкт-Петербургского государственного технологического университета растительных полимеров, 198095, СПб., ул. Ивана Черных, 4.