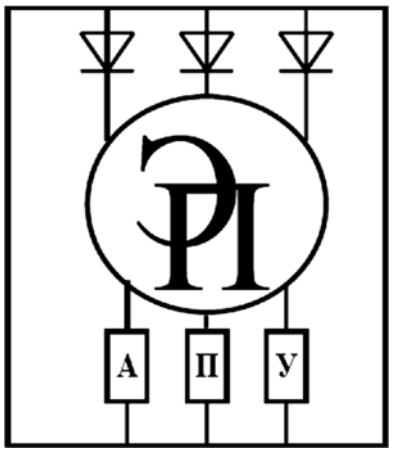
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и АПУ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МЕХАТРОННЫХ СИСТЕМ

Методические рекомендации к лабораторным работам для магистрантов направления подготовки 15.04.06 «Мехатроника и робототехника» очной и заочной форм обучения



Могилев 2023

УДК 004.45:621.3
ББК 32.973.202-018.2
П78

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой ЭПиАПУ «30» ноября 2022 г., протокол № 3

Составитель В. Н. Абабурко

Рецензент В. В. Кутузов

Цель издания методических рекомендаций – регламентация лабораторного практикума по дисциплине «Программное обеспечение мехатронных систем» для магистрантов направления подготовки 15.04.06 «Мехатроника и робототехника».

Методические рекомендации содержат состав курса лабораторных работ по указанной дисциплине. Они включают: цель, описание задания, ход выполнения работ, требования к содержанию отчета и контрольные вопросы по проведению лабораторных занятий.

Учебное издание

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МЕХАТРОННЫХ СИСТЕМ

Ответственный за выпуск	С. М. Фурманов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 26 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Лабораторная работа № 1. Программное обеспечение мехатронных систем для персонального компьютера

Цель работы:

– освоение терминологии в области программного обеспечения персонального компьютера (ПК);

– изучение классификации программного обеспечения (ПО);

– изучение состава и основных характеристик программного обеспечения мехатронных систем (МС) на ПК, используемого для проектирования, программирования, монтажа и наладки МС;

– получение практических навыков работы с общим программным обеспечением ПК для мехатронных и робототехнических систем.

Задание

Определение состава и характеристик программного обеспечения, используемого для проектирования и программирования мехатронных систем, установленного на ПК, используемого для проведения лабораторной работы.

Обучающийся должен выполнить следующие операции:

1) выполнить анализ состава программного обеспечения, установленного на ПК;

2) указать область назначения программного обеспечения с точки зрения мехатронных систем;

3) перечислить типы электронных документов (файлов), с которыми работает рассматриваемое ПО;

4) перечислить периферийные устройства ПК, с которыми может работать ПО.

1.1 Ход выполнения работы

1.1.1 Вход в пользовательский аккаунт на ПК.

Для выполнения работы первоначально следует выполнить вход в аккаунт группы на персональном компьютере, введя имя и пароль пользователя (если они предусмотрены). В случае успешного выполнения входа на экране распахнется интерфейс пользователя операционной системы (ОС).

1.1.2 Создание файла электронного отчета.

Используя доступный текстовый редактор в составе ОС, следует создать в нем новый текстовый документ для последующего формирования отчета о выполненной работе.

1.1.3 Определение параметров операционной системы.

Необходимо определить версию операционной системы, ее разрядность и номер сборки, используя окно консоли ОС и команды `ver`, или команду `winver`

в окне «Выполнить» (Run) для ОС семейства Windows. Также определяются следующие параметры ОС:

1) используемые тип файловой системы внешней памяти;
2) тип интерфейса;
3) наименование папки (каталога) во внешней памяти с расположением файлов ОС;
4) номер лицензии или данные правообладателя;
5) набор рабочих столов, доступных пользователю.

Также следует оценить возможность использования ОС для управления механотронами системами.

1.1.4 Определение состава общего программного обеспечения ПК.

Следует определить установленные пакеты офисного программного обеспечения для электронного документооборота.

1.1.5 Просмотр состава инструментального ПО, используемого для работы с мехатронными системами.

С помощью анализа стартового меню ОС и состава пиктограмм приложений на рабочем столе выделяются следующие группы ПО:

1) интегрированные среды разработчиков (IDE) языков программирования или отдельные компиляторы для настольных систем;
2) интегрированные среды создания приложений для программируемых микроконтроллеров (PLC);
3) системы компьютерной математики (CAE-системы);
4) офисное программное обеспечение;
5) системы автоматизации проектирования (CAD-системы);
6) системы автоматизации проектирования электронных плат (EDA-системы);
7) SCADA-системы;
8) программное обеспечение для обмена данными через компьютерные сети;
9) системы компьютерной анимации и видеоредакторы;
10) прочие графические редакторы.

Для каждого выделенного элемента в группе указывается:

1) полное наименование, номер версии и издатель;
2) папка размещения и объем данных;
3) типы файлов, с которым ассоциируется в системе рассматриваемое ПО;
4) периферийные устройства, которые работают с рассматриваемым ПО.

1.1.6 Оценка безопасности установленного программного обеспечения.

Для оценки безопасности ПО выполняются следующие операции:

1) указывается наличие парольного доступа к ресурсам рассматриваемого ПК;
2) отмечается наличие различного уровня доступа групп пользователей в составе ОС;
3) определяется установленное на ПК антивирусное программное обеспечение;

4) проверяется актуальность антивирусных баз;

5) контролируется наличие и тип установленного программного межсетевого экрана (файрволла);

6) фиксируется наличие и типы средств фильтрации нежелательной электронной рассылки (спама);

7) определяется присутствие в системе блокировщика рекламных модулей.

На основе рассмотренных данных делается вывод об уровне безопасности рассматриваемого ПО.

1.1.7 Выводы о пригодности ПО на рассматриваемом ПК для проектирования, управления или иных работ с мехатронными системами.

На основе выполненного обзора ПО делается вывод о его пригодности к использованию на определённом этапе жизненного цикла мехатронных систем: проектировании, производстве, сертификации, монтаже и наладки, эксплуатации и утилизации.

1.1.8 Защита отчета по лабораторной работе.

Необходимо дать на проверку электронный отчет преподавателю, проводящему лабораторную работу, и ответить на контрольные вопросы. Состав отчета должен соответствовать подразделу 1.2. При необходимости выполняется печать отчета на бумажном носителе.

1.1.9 Окончание работы с системой.

В среде текстового редактора создается электронный документ отчета по выполненной работе. Состав документа отчета должен соответствовать подразделу 1.2. После выхода из среды следует с помощью файлового менеджера удалить ненужные файлы и произвести резервное копирование остальных файлов на мобильное устройство внешней памяти или смартфон.

1.2 Содержание отчета

Отчет по работе № 1 оформляется индивидуально каждым обучающимся на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе. Состав отчета следующий:

- титульный лист;
- текст индивидуального задания;
- параметры операционной системы;
- список программного обеспечения, используемого для проектирования, программирования или анализа мехатронных систем;
- оценка безопасности установленного ПО на рассматриваемом ПК;
- выводы о назначении рассматриваемого ПК для проектирования, анализа и управления мехатронными системами и робототехническими комплексами.

Контрольные вопросы

1 Дать определение понятия «программное обеспечение».

2 По каким критериям выполняется классификация программного обеспечения?

3 Какие виды системного программного обеспечения используются на различных стадиях жизненного цикла мехатронных систем и робототехнических комплексов?

4 Какие виды системного программного обеспечения используются для мехатронных систем и робототехнических комплексов?

5 Дать определение понятия «операционная система компьютера».

6 Какие компоненты входят в состав операционной системы?

7 Каким образом классифицируется программное обеспечение?

8 Дать определение понятия «прикладная программа».

9 Дать определение понятия «электронный документ».

10 Какие функции выполняют CAD-системы?

11 Для каких целей в мехатронных системах и робототехнических комплексах используются EDA-системы?

12 Какой состав имеет офисное программное обеспечение?

13 По каким признакам классифицируются программы компьютерной графики?

14 Какие виды программного обеспечения используются для реализации компьютерной безопасности?

15 Каким образом классифицируются языки программирования?

16 Какое программное обеспечение используется для программирования программируемых логических контроллеров (ПЛК)?

17 Каким образом классифицируются операционные системы?

18 Как можно узнать версию операционной системы персонального компьютера?

19 Дать определение понятия «жизненный цикл программного обеспечения».

20 Дать определение понятия «жизненный цикл мехатронной системы».

2 Лабораторная работа № 2. Тестирование ПО МС

Цель работы:

– освоение терминологии в области тестирования программного обеспечения;

– изучение основ процессов и методов тестирования программного обеспечения мехатронных систем, установленных на ПК;

– получение практических навыков тестирования программного обеспечения с помощью ПК.

Задание

Выполнить тестирование прикладного приложения, представленного преподавателем, проводящим работу. При этом следует выполнить требования подраздела 2.1. Преподаватель представляет портативную версию приложения и по возможности исходный текст файлов приложения.

2.1 Ход выполнения работы

2.1.1 Изучение нормативных документов по тестированию ПО.

Первоначально следует выполнить вход в аккаунт группы на ПК лаборатории. В каталоге группы или компьютерной сети следует изучить терминологию и общий состав следующих нормативных документов: ГОСТ Р 56920–2016, ГОСТ Р 56921–2016 и ГОСТ Р 56922–2016.

2.1.2 Получение индивидуального задания.

Следует получить у преподавателя архив с настольным приложением для решения определенной математической задачи, которое требуется протестировать в ходе работы. Кроме самого приложения, в архиве могут содержаться исходные коды к программе или иная дополнительная информация по реализованному приложению.

2.1.3 Составление общего плана тестирования и анализ требований.

Общий план тестирования заданной программы включают ее проверку на соответствие определенным группам требований.

Требования должны соответствовать следующим условиям.

2.1.3.1 Функциональные требования к программе:

а) точность производимых математических вычислений;

б) поддержка определенного семейства операционных систем.

2.1.3.2 Нефункциональные требования к программе:

а) удобство использования (применение традиционных элементов интерфейса, возможность изменения параметров отображаемых текстовых данных и т. п.);

б) надежность – отсутствие зависаний и несанкционированных прекращений работы ПО;

в) возможность подключения расширений.

2.1.3.3 Ограничения на применение программы:

а) минимальный размер экрана или разрешение компьютера;

б) минимальный объем внешней памяти;

в) минимальный размер оперативной памяти.

2.1.3.4 Требования к интерфейсу пользователя включают:

а) наличие действующей системы подсказок и информационной поддержки в виде отдельного раздела «О программе»;

б) требования к наличию определенных системных библиотек прикладного программного интерфейса (API): версия Microsoft Frameworks или иного SDK.

2.1.3.5 Требования к данным:

а) корректный ввод исходных данных с текстового файла;

б) корректное сохранение результатов расчета в текстовый файл;

в) защита от неверного формата задания исходных данных (ввод символьных данных вместо численных, наличие незаполненных полей);

г) контроль корректности ввода исходных данных (правильность задания диапазона решения, наличие действительного решения для нелинейного уравнения на заданном интервале).

2.1.4 Уточнение стратегии тестирования.

В зависимости от наличия исходного кода тестируемой программы выбирается схема тестирования:

1) схема «белого ящика» при наличии полного текста исходного кода программы. В этом случае возможно выполнить только статическое тестирование, если тестировщик хорошо знает язык программирования, на котором выполнено приложение;

2) схема «черного ящика» при отсутствии текста исходного кода приложения. При этом возможен только вариант динамического тестирования;

3) схема серого ящика при наличии только части кода. В зависимости от полноты представления исходного кода и знаний, тестировщиком используемого языка программирования возможно использование и статического и динамического тестирования.

2.1.5 Разработка тест-кейсов.

На основе принятой модели тестирования разрабатываются тест-кейсы для анализа каждого элемента требований подпунктов из п. 2.1.3. Для тестирования требований к данным (см. п. 2.1.3.5) следует разрабатывать тест-кейсы для случая негативного тестирования, для остальных групп требований – тесты позитивного тестирования. Допускается выбрать по согласованию с преподавателям неполный набор требований из п. 2.1.3 для последующего тестирования. Тест-кейсы оформляются в виде таблицы, фрагмент которой показан ниже (таблица 2.1).

Таблица 2.1 – Пример упрощенного описания тест-кейса

Идентификатор	Тестируемое требование	Предварительное действие	Описание шагов, необходимых для проверки	Ожидаемый результат по каждому шагу
1	2.1.3.1, а	–	1 Запуск приложения	1 Раскрытие главного окна
			2 Установка в полях значений: a = 0, b = 10, e = 0,01 и выбор пункта «Принять данные»	2 Появление окна сообщения «Данные приняты»
			3 Выбор пункта меню «Решение»	3 Появление окна сообщения «Время решения …»
			4 Выбор пункта меню «Результаты»	4 Отображение формы результатов
			5 Проверка значения поля X	5 Ответ должен находиться в диапазоне от 0,99 до 1,01

2.1.6 Выполнение тест-кейсов.

Следует выполнить разработанные кейс-тесты для методов тестирования, определенных в п. 2.1.4. При выполнении лабораторной работы используется ручное системное тестирование. Результаты успешности тестов заносятся в журнал тестирования, который оформлен в виде таблицы 2.2. В последней колонке таблицы указывается следующий результат выполнения теста:

1) *не выполнен* (*not tested*) – тест-кейс готов к выполнению, но по некоторой причине пока не был выполнен;

2) *пропущен* (*skipped*) – выполнение тест-кейса отменяется по соображениям нехватки времени или изменения логики тестирования;

3) *провален* (*failed*) – в процессе выполнения был обнаружен дефект, заключающийся в том, что ожидаемый результат по как минимум одному шагу тест-кейса не совпадает с фактическим результатом. Если в процессе выполнения тест-кейса был обнаружен дефект, не связанный с шагами тест-кейса и их ожидаемыми результатами, тогда тест считается пройденным успешно и по обнаруженному дефекту создаётся отчёт о дефекте;

4) *пройден успешно* (*passed*) – в процессе выполнения тест-кейса не было обнаружено дефектов, связанных с расхождением ожидаемых и фактических результатов его шагов;

5) *заблокирован* (*blocked*) – по некоторой причине выполнение тест-кейса невозможно (как правило, такой причиной является наличие дефекта, не позволяющего реализовать пользовательский сценарий);

6) *закрыт* (*closed*) – исключительный случай, подчеркивающий завершение всех действий с тестом на рассматриваемой итерации тестирования;

7) *требует доработки* (*not ready*) – в тест-кейсе обнаружена ошибка или изменились требования, по которым он был написан, или наступила ситуация, не позволяющая считать его пригодным для выполнения и перевода в иные состояния.

Таблица 2.2 – Пример заполнения результатов тестирования

Идентификатор текст-кейса	Обнаруженный дефект	Результат успешности
1	–	Пройден успешно

2.1.7 Фиксация выявленных дефектов ПО.

Дефектом при тестировании является расхождение фактического и ожидаемого результата тест-кейса. Различают следующие типы дефектов:

1) ошибка – действие оператора, приведшее к некорректному результату;

2) дефект – недостаток в приложении, способный вызвать сбой или отказ;

3) сбой – самоустраняющийся или однократный отказ, легко устраняемый незначительным вмешательством оператора;

4) отказ – прекращение работоспособности приложения.

Выявленный при тестировании дефект и его тип указывается во второй колонке таблицы 2.2.

2.1.8 Анализ результата тестирования.

По окончании выполнения всех заданных тестов следует определить процент успешно выполненных кейс-тестов относительно их общего числа. Также указывается процент выполненных кейс-тестов относительно заданных, если выявлено обстоятельство, препятствующее дальнейшему выполнению плана тестирования.

2.1.9 Документирование результатов тестирования.

После проведения тестов составляется отчет о результатах тестирования, в котором отражается следующая информация:

1) число исходных кейс-тестов;
2) число успешно пройденных кейс-тестов;
3) число выявленных дефектов при тестировании;
4) использованные методы тестирования;
5) заключение о пригодности приложения для практического применения.

2.1.10 Формирование отчета по работе.

В текстовом редакторе создается электронный документ отчета по выполненной работе, состав которого описан в подразделе 2.2. Требуется предоставить на проверку электронный отчет преподавателю, проводящему лабораторную работу, и ответить на контрольные вопросы.

2.1.11 Завершения работы с ПК.

После выхода из текстового редактора следует с помощью файлового менеджера удалить ненужные файлы, созданные при выполнении работы, произвести резервное копирование нужных файлов на мобильное устройство внешней памяти или смартфон. В конце работы с ПК выполняется выход из аккаунта.

2.2 Содержание отчета

Отчет по работе № 2 оформляется индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе в следующем составе:

- титульный лист;
- наименование тестируемого приложения и описание его назначения;
- наличие исходного кода и иной дополнительной информации;
- таблица со списком разработанных тест-кейсов;
- журнал результатов тестирования;
- список выявленных дефектов тестируемого ПО;
- итоговый анализ и отчет по результатам тестирования приложения.

Контрольные вопросы

1 Дать определение понятия «верификация».
2 Дать определение понятия «валидация».
3 Каким образом выполняется статическое тестирование?

4 Каким образом выполняется динамическое тестирование?
5 Дать определение понятия «стрессовое тестирование».
6 Дать определение понятия «тестирование доступности».
7 Дать определение понятия «тестирование копирования и восстановления».
8 Дать определение понятия «тестирование потенциальных возможностей».
9 Дать определение понятия «тестирование износостойкости».
10 Дать определение понятия «исследовательское тестирование».
11 Дать определение понятия «тестирование устанавливаемости».
12 Дать определение понятия «нагрузочное тестирование».
13 Дать определение понятия «тестирование сопровождаемости».
14 Дать определение понятия «тестирование переносимости».
15 Дать определение понятия «тестирование надежности».
16 Дать определение понятия «тестирование защищенности».
17 Что определяет стратегия тестирования?
18 Дать определение понятия «тестирование методом черного ящика».
19 Дать определение понятия «тестирование методом белого ящика».
20 Какие виды дефектов фиксируются при тестировании ПО?
Дополнительно могут задаваться контрольные вопросы к работе № 1.

3 Лабораторная работа № 3. Интегрированные среды разработчика ПО МС

Цель работы:

– изучение интерфейса и основных функций интегрированной среды разработчика (IDE) CODESYS;

– освоение терминологии в области создания ПО для ПЛК МС;

– получение практических навыков работы в интегрированной среде CODESYS с языком программирования CFC.

Задание

Обучающийся должен выполнить следующее:

1) в среде разработчика создать проект под заданное преподавателем устройство управления МС и заданный язык программирования МЭК 61131-3;
2) назначить имена для входов выходов устройства;
3) реализовать согласно варианту логические выражения из таблицы 3.1;
4) скомпилировать созданный проект;
5) создать загрузочное приложение для последующего переноса на мобильный носитель данных.

Таблица 3.1 – Варианты заданий к лабораторной работе № 3

Вариант	Логическое выражение выходных сигналов
1	Y1=!(X1&&!(X2))||(!(X3)&&X4) Y2=(!(X1)||!(X2))&&(!(X3)||X4)) Y3=(X1&&X2)||(!(X3)||X4) Y4=(!(X1)&&X2)&&(X3&&!(X4))
2	Y1=(!(X1)||X2)||(!(X3)&&X4) Y2=(X1||!(X2))&&!(X3&&X4) Y3=!(X1&&X2)&&((X3)||!(X4)) Y4=(!(X1)&&X2)||(!(X3)||!(X4))
3	Y1=(X1&&X2))&&!(X3||!(X4)) Y2=(X1||!(X2))||(!(X3)&&!(X4)) Y3=!(X1&||X2)&&(!(X3)||X4) Y4=(!(X1)&&!(X2))||!(X3&&X4))
4	Y1=(X1||X2))&&!(X3&&!(X4)) Y2=!(!(X1)&&!(X2))||!(X3&&X4)) Y3=!(X1&&!(X2))||(!(X3)||!(X4)) Y4=(!(X1)||X2)&&(!(X3)&&X4)
5	Y1=!(!(X1)||!(X2))&&!(X3&&!(X4)) Y2=(X1&&!(X2))||!(!(X3)||X4)) Y3=(!(X1)||X2)&&(!(X3)||!(X4)) Y4=!(X1&&X2)||(X3&&X4)
6	Y1=(X1||X2))&&!(!(X3||!(X4)) Y2=(!(X1)&&X2)||(X3&&!(X4)) Y3=!(X1&&!(X2))||!(!(X3)&&X4) Y4=!(!(X1)&&!(X2))&&(!(X3)||X4)
7	Y1=(X1||!(X2))&&!(X3&&X4) Y2=!(X1&&X2)||(!(X3)&&!(X4)) Y3=(!(X1)||X2)||(!(X3)||X4) Y4=!(!(X1)&&!(X2))&&(X3||!(X4))
8	Y1=(!(X1)&&!(X2))&&!(X3||!(X4)) Y2=!(X1||!(X2))||!(X3&&X4) Y3=!(!(X1)&&X2)&&(!(X3)||!(X4)) Y4=!(X1||X2)||(!(X3)&&X4)
9	Y1=(X1||!(X2))&&!(!(X3)||X4) Y2=!(!(X1)&&!(X2))||(X3&&!(X4)) Y3=!(!(X1)||X2)&&(!(X3)&&!(X4)) Y4=(X1&&X2)||!(X3||X4)
10	Y1=!(!(X1)||X2)&&!(!(X3)&&!(X4)) Y2=!(!(X1)&&!(X2))||(X3&&X4) Y3=(X1||X2)&&(!(X3)&&!(X4)) Y4=(X1&&!(X2))||(X3||!(X4))

В таблице 3.1 даны обозначения согласно синтаксиса языка С:

– X1, …, X4 – входные дискретные сигналы на входе программируемого логического контроллера (ПЛК);

– Y1, …, Y4 – выходные дискретные сигналы ПЛК;

– && – операция логического умножения (AND);

– || – операция логического сложения (OR);

– ! – операция логического отрицания (NOT).

3.1 Ход выполнения работы

3.1.1 Получение варианта индивидуального задания.

Магистрант получает вариант индивидуального задания у преподавателя, проводящего работу.

3.1.2 Вход в пользовательский аккаунт на ПК.

Следует выполнить вход в аккаунт группы на ПК, введя имя и пароль пользователя (если они предусмотрены). В случае успешного выполнения входа на экране распахнется интерфейс ОС.

3.1.3 Запуск IDE системы программирования.

Следует с помощью ярлыка на рабочем столе или стартового меню запустить установленную среду программирования ПЛК. В случае успешного запуска среды на экране распахнется стартовое окно IDE.

3.1.4 Создание нового проекта.

Для создания нового проекта используется комбинация клавиш *Ctrl + N* или пункт меню *Файл/Новый проект*. В правой части раскрывшегося диалогового окна следует указать тип проекта – *Стандартный проект*, а в соответствующих полях уникальное имя проекта и месторасположение – собственный рабочий каталог (папку) на ПК.

Во втором диалоговом окне указывается в поле *Устройство* (*Device*) тип устройства ПЛК или сенсорной панели оператора (HMI) по согласованию с преподавателем, проводящим работу. В качестве языка программирования в поле *PLC_PRG* из выпадающего списка выбирается CFC или FBD, или иной язык по согласованию с преподавателем.

3.1.5 Назначение имен переменных для входов и выходов ПЛК.

В ветке *Устройства* дерева проекта следует выбрать пункт *PLC_PRG*. В средней части окна IDE распахнется одноименная закладка с пустой заготовкой программы. Следует в области описания переменных VAR задать по четыре переменных типа BOOL, представляющие дискретные входы и выходы с именами, соответствующими таблице 3.1. Пример дан на рисунке 3.1.

PLC_PRG

```
1  PROGRAM PLC_PRG
2  VAR
3      X1, X2, X3, X4: BOOL;   // переменные входов
4      Y1, Y2, Y3, Y4 : BOOL;   // переменные выходов
5
6  END_VAR
```

Рисунок 3.1 – Пример объявления переменных в IDE Codesys 3.5

3.1.6 Программирование ввода исходных данных.

Для ввода исходных данных в окне редактора на языке CFC используется *Панель инструментов*, расположенная в правой части окна IDE. Мышкой с панели переносятся в рабочую область редактора четыре элемента *Ввод*, внутри которых подписываются имена переменных входных сигналов, которые были объявлены в п. 3.1.5. При этом можно нажать символ многоточия, который распахнет диалоговое окно *Ассистент ввода*, из списка которого можно выбрать нужные имена переменных.

3.1.7 Программирование выходных данных.

С *Панели инструментов* мышкой переносятся в область редактора четыре блока *Вывод*. Внутри элементов записываются имена переменных дискретных выходов системы. При этом также можно использовать *Ассистент ввода* для указания имен переменных.

3.1.8 Программирование заданных логических выражений.

На основе исходных данных таблицы 3.1 реализуется составление программы, описывающей заданные выражения на основе блоков *Элемент* из *Панели элементов*. Выполнив клик по символу из трех знаков вопроса внутри блока, вызывается диалоговое окно *Ассистент ввода*. В левой части окна диалога выбирается пункт *Ключевые слова*, что вызовет в правой части отображение списка возможных слов. Для реализации логических функций используются следующие ключевые слова:

– `AND` – логическое умножение;
– `OR` – операция логического сложения;
– `NOT` – операция логического отрицания.

Логические выражения для схемы программы (рисунок 3.2):

```
Y1=!(X1&&X2);
Y2=!(X1)||(X1&&X2);
Y3=!(X3)&&(X3||X4);
Y4=!(X3||X4).
```

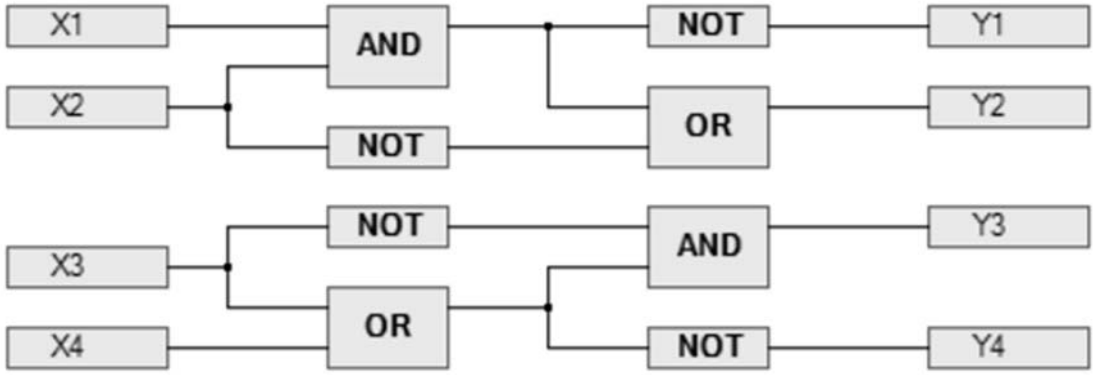


Рисунок 3.2 – Пример реализации схемы логических выражений

3.1.9 Соотнесение выходов и входов ПЛК с переменными.

Для привязки созданных переменных к входам и выходам ПЛК следует перейти в окно дерева проекта и выбрать наименование устройства, использованное при создании проекта. Далее кликнуть мышкой по ветке *LeftSide*, чтобы распахнуть в рабочей области одноименную закладку. В рабочей области следует выбрать закладку *Соотнесение входов/выходов*. Затем в рабочей области окна в поле *Переменная* выполняется клик мышкой и вызывается кнопка с многоточием, нажатие которой вызывает окно со списком переменных. Кликом по имени переменных выполняется назначение для выбранного входа или выхода переменной программы. Пример показан на рисунке 3.3.

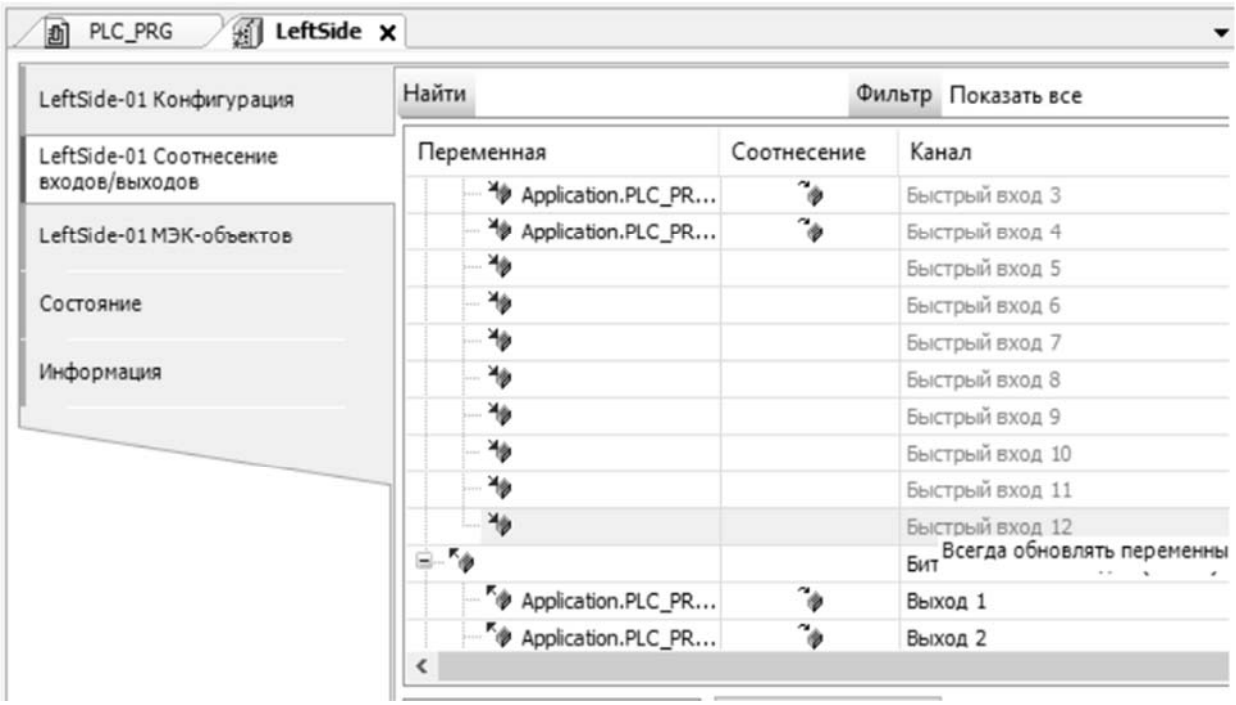


Рисунок 3.3 – Пример сопоставления переменных входам и выходам ПЛК

3.1.10 Компиляция созданного проекта.

Компиляция созданной программы выполняется выбором пункта главного меню IDE *Компиляция/Генерировать код* (или клавишей F11). В случае успешного завершения процесса компиляции в нижнем окне панели компиляции будет показано 0 ошибок. При наличии ошибок их необходимо устранить и повторно скомпилировать проект.

3.1.11 Загрузка и отладка проекта.

После успешной компиляции следует выбрать пункт *Онлайн/Эмуляция* и виртуально загрузить проект в ПЛК. Для этого выбирается пункт *Онлайн/Логин* (клавиши Atl+F8), далее в диалоговом окне выбрать верхний пункт меню.

Выполнение программы производится с помощью ресурсов раздела *Отладка* главного меню среды программирования. Запуск проекта на выполнение выполняется выбором пункта *Старт*, останов – пунктом *Стоп*. После запуска проекта устанавливаются значения переменных входов и фиксируются

значения на выходах. По результатам вычислений следует составить таблицу истинности запрограммированного устройства. После окончания отладки выполняется пункт *Онлайн/Отключение*.

3.1.12 Создание загрузочного файла проекта.

Создание загрузочного файла разработанной программы для возможного последующего программирования ПЛК или иного устройства выполняется с помощью раздела главного меню *Онлайн/Создать загрузочное приложение*.

Следует зафиксировать имя и тип созданного файла, а также его размер.

3.1.13 Формирование отчета.

В текстовом редакторе ПК создается электронный документ отчета, содержание которого соответствует подразделу 3.2.

3.1.14 Завершения работы со средой программирования.

Перед завершением работы с IDE необходимо сохранить проект, используя раздел главного меню *Файл/Сохранить проект* (*Ctrl + S*).

Завершается работа с IDE закрытием главного окна системы или клавишами *Alt + F4*. После чего удаляются ненужные файлы и производится резервное копирования созданных m-файлов и отчета.

3.2 Содержание отчета

Отчет по работе № 3 оформляется индивидуально на листах формата А4 на бумажном или электронном носителе и имеет следующий состав:

- титульный лист;
- текст индивидуального задания;
- исходные данные задания;
- копия экрана с назначенными именами входов и выходов ПЛК;
- копия экрана с программной реализацией заданных выражений;
- таблица истинности запрограммированного устройства;
- данные о загрузочном файле созданного проекта.

Контрольные вопросы

1 Какие компоненты включает в себя интегрированная среда разработчика для системы программирования?

2 Дать определение понятия «проект».

3 Дать определение понятия «программный модуль».

4 Дать определение понятия «функция».

5 Дать определение понятия «функциональный блок».

6 Дать определение понятия «переменная».

7 Какую структуру имеет интерфейс интегрированной среды программирования?

8 Каким образом в IDE выполняется создание нового файла с исходным кодом?

9 Каким образом в среде программирования выполняется пошаговая отладка?

10 Каким образом в IDE выполняется отладка разработанной программы?

11 Каким образом в IDE выполняется компиляция созданной программы?

12 Какие инструменты входят в состав IDE?

13 Какие данные хранятся в файлах целевой платформы (таргет-файлах)?

14 Каким образом подключить таргет-файл к проекту в составе IDE?

4 Лабораторная работа № 4. Программирование интерфейса ПО

Цель работы:

– изучение методов построения и программирования элементов визуального интерфейса в среде Codesys при проектировании ПО МС;

– изучение программных ресурсов, используемых для реализации визуальных элементов интерфейса в среде Codesys;

– получение практических навыков программирования визуального интерфейса оператора, использующего среду Codesys.

Задание

Обучающийся для выполнения работы использует номер варианта и данные программы, полученные при выполнении работы № 3 на основе таблицы 3.1.

Следует выполнить следующее:

1) визуализировать дискретные входы как кнопки с контактами, срабатывающие при нажатии или перемещении (NO);

2) визуализировать дискретные выходы системы в виде светосигнальных индикаторов;

3) добавить полосу прокрутки и прямоугольную область для ручного задания и отображения действительного значения, в указанном преподавателем диапазоне, с заданной дискретностью;

4) для четных вариантов задания доработать схему CFC-программы, чтобы отображать целочисленное значение, которое в двоичной форме формируют дискретные сигналы выходов, а для четных вариантов заданий – значение, формируемое дискретными входами.

4.1 Ход выполнения работы

4.1.1 Вход в пользовательский аккаунт на ПК.

Для начала работы с ПК следует выполнить вход в аккаунт группы на ПК, введя имя и пароль пользователя (если они предусмотрены).

4.1.2 Запуск интегрированной среды и загрузка проекта системы.

На ПК производится запуск среды программирования. В главном окне среды выбирается пункт *Файл/Открыть проект*, загружается созданный ранее при выполнении работы № 3 проект. Следует сразу сохранить проект под новым

именем в своем рабочем каталоге (папке), используя пункт меню *Файл/Сохранить проект как*.

4.1.3 Добавление в проект объекта визуализации.

В левом окне на дереве проекта выполняется клик правой кнопки мышки по ветке *Application* и в распахнувшемся выпадающем меню выбрать пункт *Добавление объекта/Визуализация* и указать наименование объекта визуализации, указав его имя в соответствующем диалоговом окне. После этого в дереве появляется менеджер визуализации, а в рабочем окне закладка с указанным именем.

Выполняется клик по ветке дерева *Менеджер визуализации*, что распахнет в рабочей области окно одноименной закладки. Следует поставить мышкой маркер в позиции *Использовать строки Unicode*.

Далее следует выбрать в дереве проекта вкладку Web-визуализация, что раскроет одноименную закладку в рабочей области IDE. Следует установить маркер в группе *Опции масштабирования* в позиции *Изотропная*.

4.1.4 Задание параметров окна визуализации.

При необходимости настройки размеров окна визуализации вызывается окно свойств кликом правой кнопки мыши в дереве проекта по созданному объекту визуализации. В нем на закладке *Визуализация* в полях *Ширина* и *Высота* устанавливаются требуемые размеры.

Далее на вкладке с именем визуализации выполняется задание цвета фона. Для этого кликом правой кнопки мыши в области окна вызывается выпадающее меню, в котором выбирается пункт *Фон*. В раскрывшемся диалоговом окне устанавливается маркер в позицию Фон и указывается код цвета или выбирается нужный цвет из палитры при нажатии на кнопку с многоточием. Возможно также задать в этом окне и фоновое изображение из внешнего файла.

Следует выполнить текстовую надпись вверху экрана визуализации с наименованием варианта задания и данных исполнителя. Для этого в панели инструментов визуализации из группы *Стандартные элементы управления* устанавливается элемент *Метка*. При выборе мышкой установленного элемента в правой части среды раскрывается окно его следующих основных свойств:

– *Имя элемента* – символьный идентификатор визуального элемента;
– *Тип элемента* – наименование визуального элемента;
– *ID текста* – цифровой идентификатор текста элемента;
– *Позиция* – координаты расположения и размеры визуального элемента;
– *Текст* – текстовая строка содержания метки;
– *Шрифт* – наименование, размер и прочие атрибуты шрифта текста;
– *Цвет шрифта* – цвет символов текстовой надписи;
– *Горизонтальное выравнивание* – способ выравнивания текста по горизонтали: по центру, справа, слева;
– *Вертикальное выравнивание* – способ выравнивания текста по вертикали: по центру, по верху, по низу.

4.1.5 Добавление элементов визуализации дискретных входов.

На панели инструментов визуализации из группы *Стандартные элементы управления* устанавливается элемент *Группа*, объединяющий в выделенную прямоугольную область набор визуальных элементов. В свойствах элемента *Текст* выполняется надпись «Входы ПЛК», а в свойствах текста указывается приемлемый размер и цвет шрифта надписи.

В область установленной группы мышкой помещаются из раздела *Lamps/Switches/Bitmaps* четыре элемента по согласованию с преподавателем: *Нажимной выключатель, Переключатель питания, клавишный выключатель* или *Вращающий выключатель*. После установки элементов для каждого выключателя в свойстве *Переменная* с помощью кнопки с многоточием в диалоговом окне указываются соответствующие имена переменных дискретных входов. В свойстве *Поведение элемента* указывается значение *Переключатель*, чтобы выполнить фиксацию положения выключателя. При использовании нескольких однотипных выключателей в свойстве *Фон* задается уникальный цвет элементов отображения.

4.1.6 Добавление элементов визуализации дискретных выходов.

Из группы панели инструментов визуализации *Стандартные элементы управления* в окно визуализации помещается элемент *Группа*, в свойствах которой выполняется надпись «Выходы ПЛК», а также нужный размер и цвет шрифта надписи.

Далее в область группы *Индикатор* в свойстве *Переменная* с помощью кнопки в диалоговом окне указываются соответствующие имена переменных дискретных входов. В свойстве *Поведение элемента* указывается значение *Переключатель*, чтобы выполнить фиксацию положения выключателя. Для каждого индикатора в свойстве *Фон* задается уникальный цвет отображения сигнала.

4.1.7 Добавление переменных целочисленного кода и действительного сигнала задания.

В закладке PLC_PRG рабочей области в области переменных добавляются две переменные: целочисленная – для отображение кода типа `INT`, а для ручного задания – типа `REAL`. Пример объявления дан на рисунке 4.1.

PLC_PRG | LeftSide | Visualization | Менеджер визуализации

```
1  PROGRAM PLC_PRG
2  VAR
3      X1, X2, X3, X4: BOOL;     // переменные входов
4      Y1, Y2, Y3, Y4 : BOOL;    // переменные выходов
5      Xkod  : BYTE;             // целочисленный код входа
6      Uz  : REAL;               // вещественный сигнал задания
7  END_VAR
```

Рисунок 4.1 – Пример добавления переменных в проект

4.1.9 Дополнение схемы программы.

Для преобразования дискретных сигналов в целочисленный код необходимо добавить из *Панели элементов* программирования четыре блока *Элемент*, внутри которых вставляется оператор преобразования логического типа в целочисленный TO_BYTE. Входы вставленных блоков подключаются к дискретным сигналам, а выходы – к блокам *Элемент*, с функцией перемножения (ключевое слов MUL). На вторые входы множителей подаются входные элементы со значениями 1, 2, 4 и 8 для первого, второго, третьего и четвёртого сигналов. Выходы элементов перемножения подаются на входы элемента суммирования (с ключевым словом ADD). Выходной сигнал с элемента суммирования подается на вход блока Выход, внутри которого назначается целочисленная переменная. Пример получения кода входных сигналов показан на рисунке 4.2.

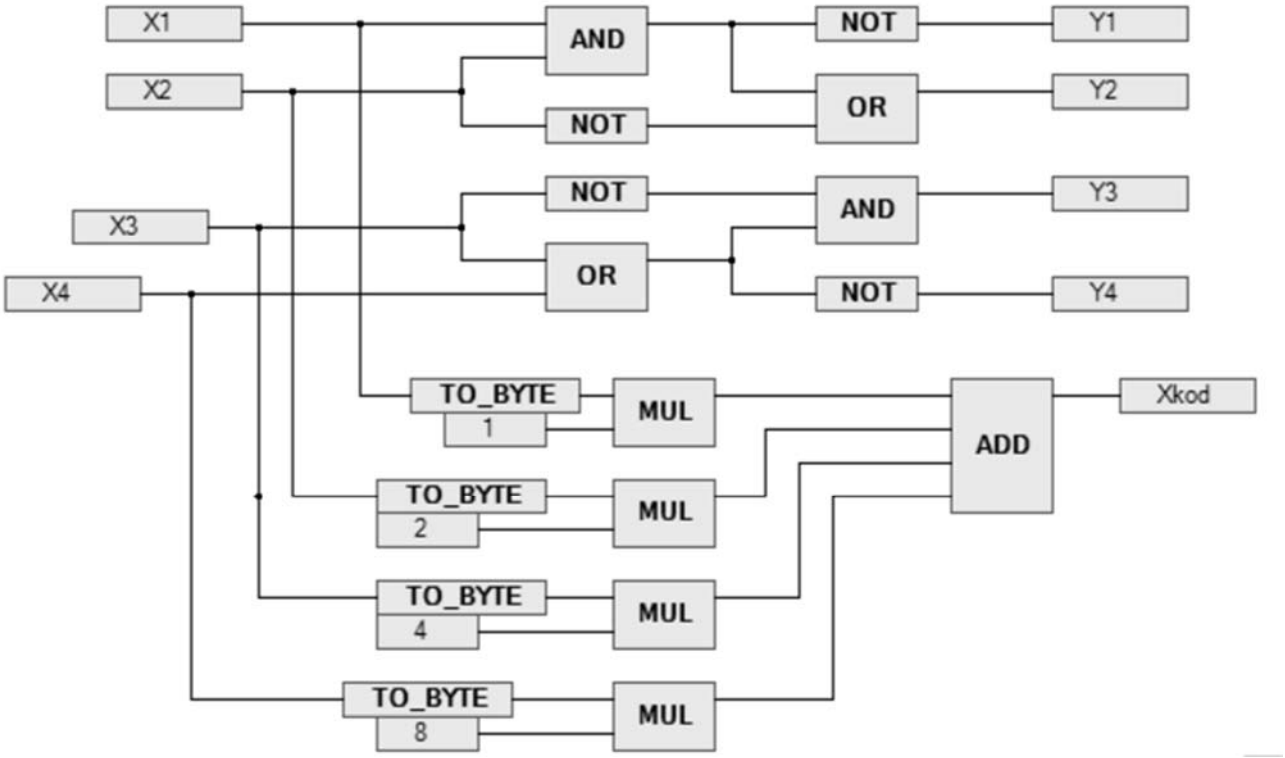


Рисунок 4.2 – Пример реализации расчета цифрового кода

4.1.10 Добавление элементов отображения цифрового кода.

Для отображения целочисленного кода, формируемого дискретными сигналами, в окне визуализации размещается из базовой группы панели инструментов визуализации блок *Прямоугольник*. В его свойстве *Текстовая переменная* с помощью кнопки с многоточием из диалогового окна выбирается объявленная целочисленная переменная.

Для отображения текстовых данных внутри свойства *Текст* указывается текст комментария и формат вывода целочисленных данных: %d.

4.1.11 Визуализация ручного задания действительного сигнала.

Для отображения вещественного сигнала задания на экран визуализации из базовой группы панели элементов помещается элемент *Прямоугольник* или *Скругленный прямоугольник*. В свойстве *Текстовая переменная* из диалогового окна выбирается объявленная вещественная переменная. Для отображения текстовых данных внутри свойства *Текст* указывается текст комментария и формат вывода вещественных данных с точностью до десятичной доли: `%3.1f`.

Для ручного задания значения действительной переменной на экран визуализации из группы *Стандартные элементы управления* помещается блок *Полоса прокрутки*. В свойствах *Значение* и *Текстовая переменная* указывается с помощью диалогового окна объявленное в п. 4.1.8 наименование действительной переменной. В свойствах *Минимальное значение* и *Конечный индекс* указываются границы диапазона изменения сигнала, а в свойстве *Дискретность перемещения* – шаг изменения.

Пример окна визуализации с различными видами переключателей дан на рисунке 4.3.

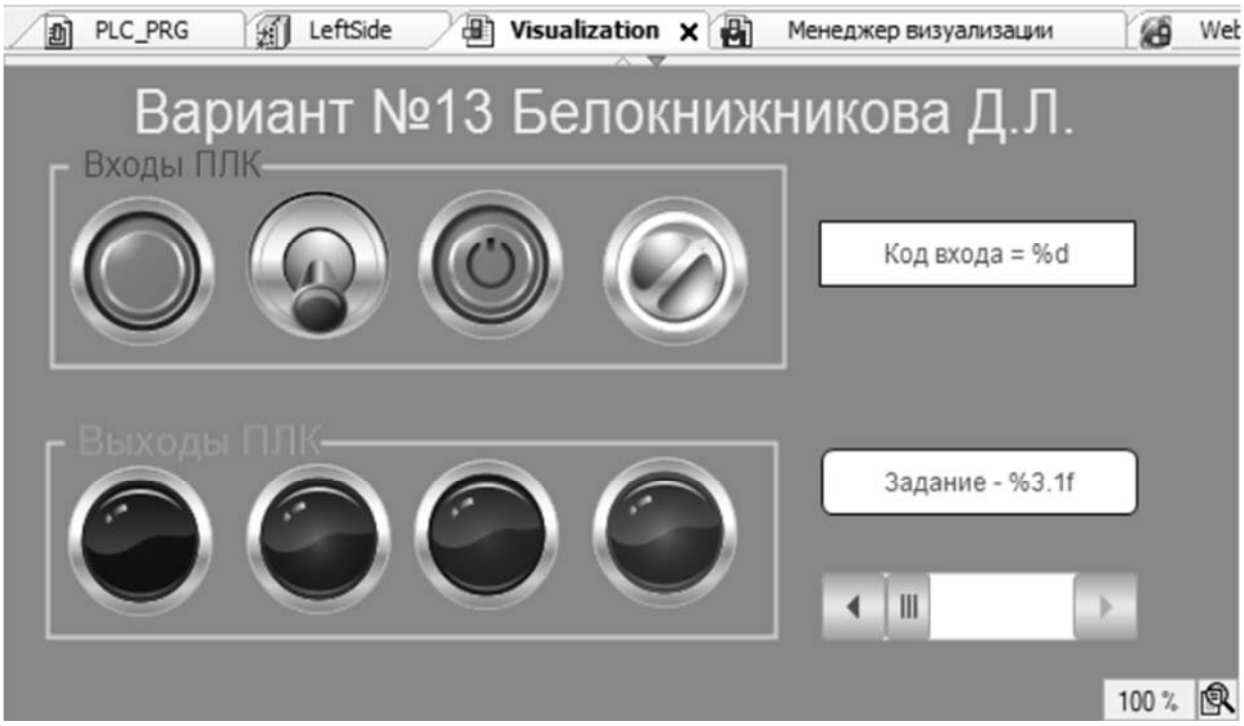


Рисунок 4.3 – Пример выполнения экрана визуализации

4.1.12 Компиляция и отладка программы.

Компиляция программы выполняется выбором пункта меню *Компиляция/Генерировать код* или клавишей F11. При отсутствии синтаксических ошибок следует включить эмуляцию установкой маркера на пункте *Онлайн/Эмуляция* и загрузить программу в эмулятор ПЛК кликом по пункту *Онлайн/Логин* (Alt + F8). В диалоговом окне следует выбрать пункт *Логин с загрузкой*, а далее начать отладку программы с помощью пункта *Отладка/Старт* (F5). Следует продемонстрировать преподавателю

работоспособность выполненных визуальных блоков. Остановка отладки выполняется пунктом *Отладка/Стоп* (Shift + F8). При необходимости перекомпиляции приложения следует выгрузить его, используя пункт *Онлайн/Отключение* (Ctrl + F8).

4.1.13 Формирование отчета.

В среде текстового редактора создается электронный документ отчета. Содержание отчета оформляется согласно требованиям из подраздела 4.2.

4.1.14 Завершение работы со средой программирования.

Перед завершением работы следует сохранить проект, используя пункт ю *Файл/Сохранить проект* (Ctrl+S). Завершается работа со средой закрытием главного окна системы. После чего удаляются ненужные файлы и производится резервное копирование созданных файлов и отчета на мобильный носитель данных.

4.2 Содержание отчета

Отчет по работе № 4 оформляется индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист;
- текст индивидуального задания;
- копия экрана рабочей области с переменными и схемой расчета;
- копия экрана визуализации;
- копия экрана визуализации в момент отладки.

Контрольные вопросы

1 Дать определение понятия «интерфейс пользователя».

2 Какие виды группы элементов интерфейса пользователя различают при проектировании программного обеспечения в среде Codesys?

3 Какие визуальные элементы используются в интерфейсе пользователя?

4 Каким образом выполняется настройка параметров элементов визуализации в среде Codesys?

5 Какие основные свойства элементов визуализации настраиваются в формируемом проекте?

6 Каким образом объявляются целочисленные переменные в проекте Codesys?

7 Как выполняется преобразование логического типа данных в целочисленный в среде языка программирования CFC?

8 Как формируется цифровой код на основе отдельных дискретных сигналов, представляющих двоичные разряды числа?

9 Для каких целей используется Web-визуализация при создании программного обеспечения ПЛК?

10 Каким образом выполняется формирование ручного задания вещественного сигнала при визуализации в среде Codesys?

11 Как выводятся на экран визуализации численные значения переменных?

12 Каким образом выполняются настройки экрана визуализации?

Дополнительно могут задаваться контрольные вопросы к работе № 3.

5 Лабораторная работа № 5. Программирование структур данных ПО

Цель работы:

– изучение способов описания структурированных типов данных и операций над ними с помощью языков программирования CFC и ST в среде CODESYS;

– получение практических навыков программирования обработки и визуализации данных массивов и структур в среде программирования CODESYS для ПЛК мехатронных систем;

– получение практических навыков создания и применения функций в среде CODESYS;

– получение практических навыков работы со счетчиками импульсов в среде программирования языка CFC.

Задание

Необходимо выполнить следующее:

1) реализовать подсчет импульсов с дискретного входа до заданного преподавателем значения инкрементальным счетчиком с последующим обнулением его выхода;

2) полученное значение счетчика используется для расчета значения вещественной независимой переменной от заданного начального значения с заданным интервалом путем прибавления к начальному значению значения произведения переменной выхода счетчика на величину шага;

3) создать функцию с выходом типа REAL, выполняющую интерполяцию на основе полинома Лагранжа входной вещественной переменной, используя данные из таблицы 5.1;

4) выполнить расчет зависимой вещественной переменной на основе созданной функции для полученных в пункте 2 значений независимой переменной;

5) создать в разделе DUP приложения структуру для описания двухмерной декартовой координаты, которая имеет два поля данных независимой и зависимой переменной;

6) создать функцию, имеющую две входные переменные типа REAL, формирующую из них возвращающую значение типа созданной структуры;

7) в программе сохранить каждое значение выхода счетчика и полученные значения независимой и зависимой переменной в одномерный массив с элементами типа созданной структуры;

8) выполнить визуализацию для созданной программы на основе прямоугольной кнопки, формирующей входные импульсы, и демонстрации

номера выхода счетчика в прямоугольнике, а полученный массив в виде таблицы и декартового графика.

Таблица 5.1 – Варианты индивидуальных заданий к лабораторной работе № 5

Вариант	Табличная функция исходных данных								
1	X	0	1	2	3	4	5	6	7
	Y	1	2,5	3	5	5,5	7	8	9
2	X	0	1	2	3	4	5	6	8
	Y	–1	–0,5	1	5	6,5	9	11,5	15
3	X	0	1	2	3	4	5	7	9
	Y	0,5	1	2,5	5	8,5	9,5	15	17,5
4	X	0	2	4	6	8	10	12	14
	Y	1	4	9	13	15	21	23	29
5	X	0	3	6	9	12	15	18	21
	Y	–2	8	16	25	35	45	52	62
6	X	0	2	4	6	10	16	20	25
	Y	–22	–14	–2	6	15	25	35	50
7	X	0	1	2	3	4	5	6	7
	Y	2	4	5	6	6,5	7	7,5	8
8	X	0	1	2	3	4	5	6	7
	Y	1	2,5	7	14,5	25	38,5	55	74,5
9	X	0	1	2	3	4	5	6	7
	Y	–1	0	3	8	15	24	35	48
10	X	0	1	2	3	4	5	6	7
	Y	–2	0	0,828	1,464	2	2,472	2,9	3,292

5.1 Ход выполнения работы

5.1.1 Загрузка среды программирования CODESYS.

Следует выполнить вход в аккаунт на ПК и произвести запуск среды системы CODESYS. Следует создать новый проект аналогично п. 3.1.4.

5.1.2 Описание основных переменных программы.

На вкладке PLC_PRG в разделе описания переменных VAR следует задать следующие переменные: две переменные типа REAL, одну типа BOOL, одну типа WORD и две инициализированные переменные типа REAL с данными шага и начального значения. Также устанавливается переменная счетчика типа CTU.

Для дальнейшей работы с несколькими вызовами численного значения максимального выхода счетчика следует в конце описания блока переменных описать именованную константу. Для этого формируется отдельный раздел описания VAR CONSTANT. Внутри раздела помещается константа с указанием

имени, типа и ее значения. Пример описания переменных и константы показан на рисунке 5.1.

```
1   PROGRAM PLC_PRG
2   VAR
3       Imp : BOOL;      // переменная дискретного импульса
4       InX : REAL;      // независимая переменная
5       OutY : REAL;     // зависимая перемнная
6       Num : WORD;      // Номер на выходе счетчика
7       dX : REAL :=1;   // Шаг изменения независимой переменной
8       Xn : REAL:=0.5;   // Начальное значениенезависимой переменной
9       CTU1: CTU;        // переменная инкрементного счетчика
10  END_VAR
11  VAR CONSTANT  // описание именованных  констант
12      N: WORD :=10;  // Максимальное значение выхода счетчика
13  END_VAR
```

Рисунок 5.1 – Пример объявления основных переменных и константы

5.1.3 Программирование счетчика и расчета независимой переменной.

В рабочую область программы помещаются входные переменные: логическая переменная импульса, инкрементный счетчик CTU, выходная целочисленная переменная и две вещественных переменных начала и шага изменения. Также размещаются элементы сложения ADD и перемножения MUL, которые формируют выходную независимую переменную. Вторая независимая переменная подключается к выходу счетчика. Выход счетчика Q подается на его вход RESET. Пример схемы программы дан на рисунке 5.2.

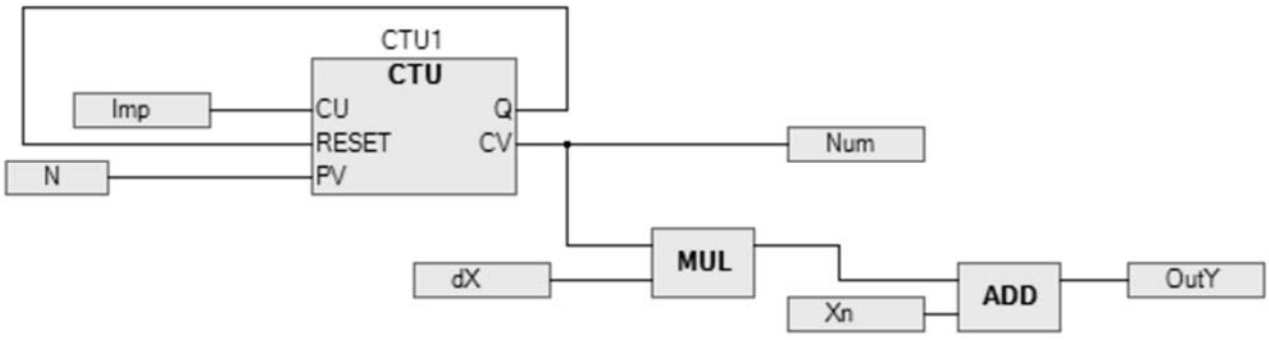


Рисунок 5.2 – Пример реализации счета импульсов и расчета независимой переменной

5.1.4 Описание функции с интерполяцией.

Для вставки в проект функции выполняется добавление объекта POU. После этого указывается тип объекта – функция, наименование функции и указывается тип возвращаемого результата REAL. При этом выбирается из выпадающего

списка язык реализации ST. После этого раскрывается вкладка с именем функции.

В разделе описания переменных функции в секции VAR_INPUT указывается одна входная переменная типа REAL. В разделе VAR объявляются локальные переменные. Данные варианта из таблицы 5.1 оформляются в виде двух одномерных массивов типа REAL. Для каждого из массивов указываются значения элементов с помощью оператора присваивания. Число элементов массивов описываются целочисленной константой. Также добавляются три переменных типа REAL и два счетчика типа INT. Пример дан на рисунке 5.3.

INTERP | PLC_PRG

```
FUNCTION INTERP : REAL  // имя и тип результата функции
VAR_INPUT    // секция входных переменных функции
    X:REAL;  // искомое значение
END_VAR      // конец описания входных переменных
VAR          // раздел описания локальных переменных
    Y:REAL:=0;  // результат расчета
    I, J : INT; // счетчики циклов
    L1,L2:REAL; // числитель и знаменатель дроби полинома
    Xi : ARRAY[1..10] OF REAL:= [0,2,4,6,8,10,12,14,16,18]; // массив X
    Yi : ARRAY[1..10] OF REAL:= [1,5,17,37,65,101,145,197,257,325]; // массив Y
END_VAR      // конец описсания локальных переменных
VAR CONSTANT   // описание констант
    N:INT:=10; // число элементов массива
END_VAR        // конец описания именованных констант
```

Рисунок 5.3 – Пример состава переменных функции интерполяции

В операторной части выполняется программирование на языке ST алгоритма интерполяции полиномом Лагранжа, пример которой показан на рисунке 5.4.

```
FOR I:=1 TO N DO // цикл перебора точек таблицы
    L1:=1;  // произведение составляющих числителей
    L2:=1;  // произведение составляющих знаменателя
    FOR J:=1 TO N DO // цикл расчета параметров полинома
        IF I<>J THEN L1:=L1*(x-Xi[J]); // числитель
            L2:=L2*(Xi[I]-Xi[J]);      // знаменатель
        END_IF // конец оператора if
    END_FOR // конец внутреннего цикла
    Y:=Y+Yi[i]*L1/L2;
END_FOR // конец основного цикла
INTERP:=Y;  // возврат результата
```

Рисунок 5.4 – Пример интерполяции полиномом Лагранжа

5.1.5 Описание типа структуры и функции для ее задания.

Описание структуры выполняется путем добавления объекта DUT. В окне диалога указывается: маркер в позиции «структура» и имя структуры. В распахнувшейся вкладке с именем структуры выполняется описание ее состава: два поля типа REAL. Пример состава структуры дан на рисунке 5.5.

INTERP | PLC_PRG | Pos2D

```
1  TYPE Pos2D :  // наименование структуры
2  STRUCT        // указание типа-структуры
3      x:REAL; // координата X
4      y: REAL; // координата Y
5  END_STRUCT    // конец описания состава структуры
6  END_TYPE      // конец описания типа
```

Рисунок 5.5 – Пример описания структуры с данными двухмерной точки

Для внесения данных в структуру создается отдельная функция как объект POU с двумя входными переменными типа REAL и реализацией на языке ST. Функция возвращает тип описанной структуры. Состав переменных и операторов функции дан на рисунке 5.6.

INTERP | PLC_PRG | Pos2D | REAL_TO_POS2D

```
1  FUNCTION REAL_TO_POS2D : Pos2D
2  VAR_INPUT   // описание входных переменных
3      XR : REAL;     // координата X
4      YR : REAL;     // координата Y
5  END_VAR     // конец описания входных переменных
6  VAR         // локальные переменные
7  END_VAR
8
```

```
1  REAL_TO_POS2D.x:=XR; // помещение данных в поле X
2  REAL_TO_POS2D.y:=YR; // помещение данных в поле Y
```

Рисунок 5.6 – Пример состава функции добавления данных в структуру

5.1.6 Программирование схемы расчета зависимой переменной.

В список переменных на закладке PLC_PRG добавляется строка одномерного массива элементов типа созданной структуры. Пример показан на рисунке 5.7.

В рабочей области программы PLC_PRG добавляется схема расчета выходной функции с использованием элемента созданной функции интерполяции. Результаты расчета независимой и зависимой переменной помещаются

в массив точек решения с помощью второй созданной функции. Пример схемы программы дан на рисунке 5.8.

```
10      DRec : ARRAY[1..N] OF Pos2D; // массив полученных точек функции
11  END_VAR
12  VAR CONSTANT  // описание именованных  констант
```

Рисунок 5.7 – Пример добавления массива структур сохранения результатов

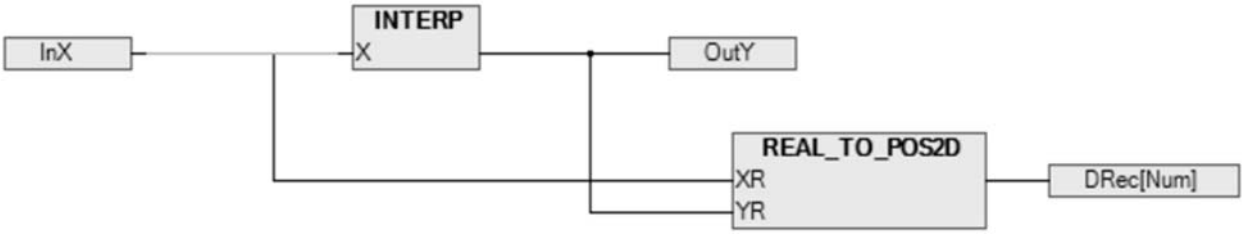


Рисунок 5.8 – Пример расчета функции

5.1.7 Визуализация расчетов.

В проект добавляется и настраивается объект визуализации (рисунок 5.9) аналогично пп. 4.1.3 и 4.1.4. Для создания импульса на входе системы используется элемент визуализации *Кнопка* (*Стандартные элементы управления*). Значение на выходе счетчика отображается в элементе *Прямоугольник* (*Базовый*). Отображение таблицы расчетов функции формируется элементом *Таблица* (*Стандартные элементы управления*), которая связывается с массивом структур. Также устанавливается элемент двухмерного графика *Декартовый график XY* (*Специальные элементы управления*), поля данных *X* и *Y* связываются с соответствующими составляющими массива структур.

5.1.8 Компиляция, загрузка и отладка программы.

Выполняется компиляция проекта клавишей F11. При отсутствии ошибок включается режим эмуляции и загружается программа клавишами Alt + F8. Старт проекта производится клавишей F5. Следует продемонстрировать преподавателю расчет функции на основе клика мышкой по кнопке. Остановка отладки клавишами Shift+F8. Выгрузка проекта выполняется нажатием Ctrl + F8.

5.1.9 Формирование отчета и завершения работы.

В текстовом редакторе ПК создается электронный документ отчета, содержание которого соответствует подразделу 5.2. Перед завершением работы следует сохранить проект и выполнить его резервное копирование вместе с электронным документом отчета на мобильный носитель.

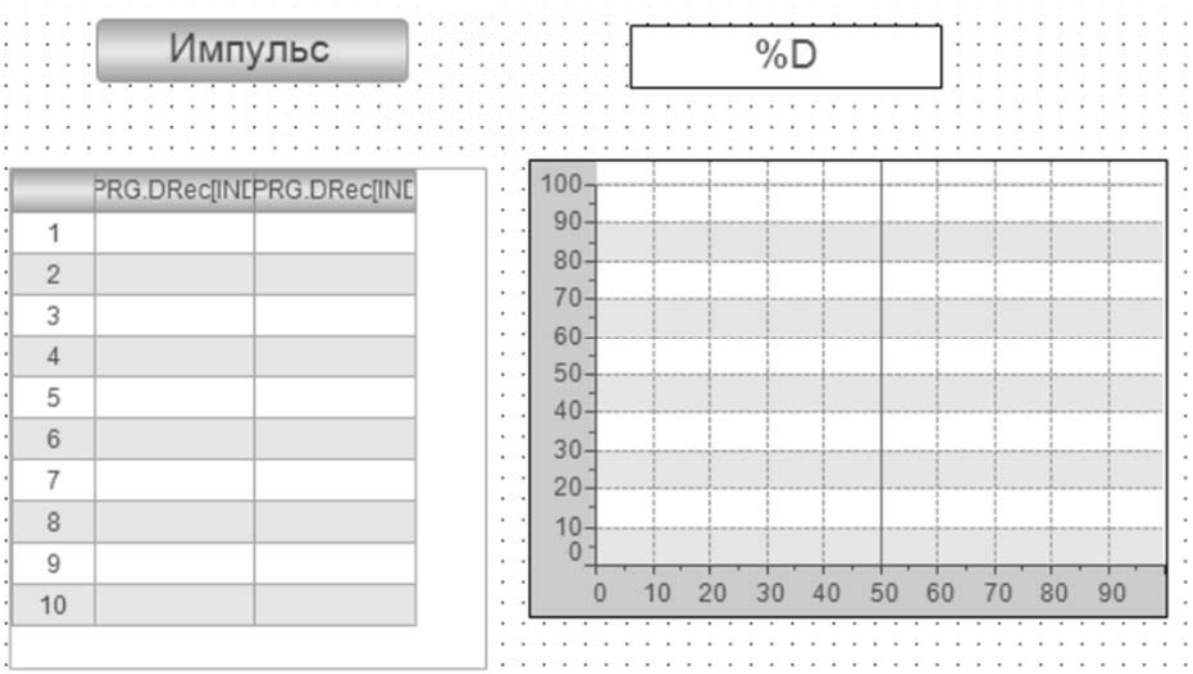


Рисунок 5.9 – Пример экрана визуализации программы

5.2 Содержание отчета

Отчет по работе № 5 оформляется индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист;
- текст индивидуального задания;
- исходные данные задания работы;
- список переменных PLC_PRG;
- схема основной программы PLC_PRG;
- текст функции интерполяции;
- текст описания структуры;
- текст функции помещения данных в элемент структуры.

Контрольные вопросы

1 Дать определение понятия «массив».
2 Дать определение понятия «структура».
3 Дать определение понятия «POU».
4 Дать определение понятия «DUP».
5 Каким образом в проект добавляется функция?
6 Каким образом в проект добавляется тип структуры?
7 Какой формат имеет описание переменной одномерного массива?
8 Какой формат имеет описание типа структуры?
9 Какой формат имеет описание переменных функции?
10 Каким образом выполняется подключение инкрементального счетчика?
11 Каким образом осуществляется визуализация результатов расчетов

функции в виде таблицы?

12 Каким образом осуществляется визуализация расчетов функции в виде двухмерного графика в прямоугольной системе координат?

13 Каким образом выполняется формирование импульсов при визуализации проекта программы?

14 Каким образом на языке ST выполняется программирование цикла?

15 Каким образом на языке ST выполняется программирование операции условного перехода?

Дополнительно могут задаваться контрольные вопросы к работам № 3 и 4.

6 Лабораторная работа № 6. Программирование внешнего обмена данными

Цель работы:

– изучение методов обмена данными между элементами системы управления МС;

– получение практических навыков программирования обмена данными по последовательному интерфейсу RS-232/485 в среде CODESYS для ПЛК.

Задание

Обучающийся для выполнения работы выполняет следующее:

1) создает проект с несколькими устройствами (число и типы которых согласует с преподавателем, проводящим работу);

2) выполняет настройку конфигурации интерфейсов созданных устройств для обмена данными;

3) программирует по заданию преподавателя обмен данными между созданными устройствами;

4) выполняет визуализацию обмена данными между устройствами.

6.1 Ход выполнения работы

6.1.1 Вход в аккаунт и загрузка среды CODESYS.

Сначала выполняется вход в аккаунт на ПК и запускается среда программирования CODESYS.

6.1.2 Создание проекта CODESYS.

В среде CODESYS следует создать новый проект аналогично п. 3.1.4. Изначально используемое в проекте устройство назначается ведущим (Master). Далее в состав проекта в дереве следует добавить новые ведомые (Slave) устройства. Число и типы устройств задаются преподавателем. Каждое из созданных устройств отображается в виде отдельной ветки на дереве проекта.

6.1.3 Конфигурация интерфейса обмена данными между устройствами.

В зависимости от моделей, используемых в проекте устройств, и задания преподавателя определяется тип промышленного интерфейса и версия протокола обмена.

6.1.4 Настройка ведущего устройства и мастера сети.

На ветке ведущего устройства дерева проекта выбирается вложенная группа *Drive*. На ней подключается устройство последовательного порта с протоколом Modbus (например, Modbus_COM). При необходимости корректируется номер порта и скорость обмена данными на вкладке *Конфигурация последовательного порта*. Возможно использование программных устройств обмена данными по сети Ethernet. Для этого добавляется соответствующим образом устройство Ethernet. Обязательно указывается режим работы для ведущего устройства – *Master*.

После добавления режима Master (RTU, TCP) для случая использования Ethernet, выполняется его настройка с помощью диалогового окна специального мастера на вкладке *Конфигурация ModbusTCP Master*. В позиции *Таймаут ответа* указывается время ожидания ответа в миллисекундах от ведущего устройства. Если за указанное время Slave-устройство не ответило, тогда начинается опрос следующего ведомого устройства. Настройка для случая использования COM-порта аналогична, только используется параметр *время между фреймами* – интервал между ответом (или окончанием ожидания) и следующим запросом.

Следует в окне переменных выполнить описание заданного количества и типа переменных для реализации заданного алгоритма обмена данными.

6.1.5 Настройка ведомых устройств.

На ветках подчиненных устройств аналогично п. 6.1.4 добавляются устройство последовательного порта или Ethernet, протоколы обмена которых совпадают с ведущим устройством. При необходимости корректируется номер порта и скорость обмена данными. Для всех ведомых устройств устанавливается режим работы *Slave*. Для каждого из ведомых устройств можно указать свое время ожидания на вкладке *Конфигурация Slave*.

При использовании Ethernet-порта задаются следующие настройки:

– *IP Адрес Slave* – IP-адрес устройства, которое запрашивает мастер сети;

– *Unit-ID* – сетевой адрес устройства, который может принимать значения от 1 до 247 и используется для идентификации ведомого устройства в сети;

– *Таймаут ответа* устанавливается для каждого Slave-модуля и имеет приоритет по отношению к соответствующему параметру Master;

– *Порт* – номер порта для Slave-модуля.

Подчиненное устройство, подключенное к мастеру COM-порта, имеет аналогичные настройки.

6.1.6 Добавление групп адресов каналов.

Настройка каналов выполняется независимо от типа выбранного интерфейса. Она состоит в задании одного или нескольких регистров, а также указанию режима работы канала: чтение или запись с последующей привязкой к переменным конфигурации.

Добавление каналов выполняется на вкладке *Канал Modbus Slave* соответствующего ведомого устройства нажатием кнопки *Добавить канал*. Что распахнет диалоговое окно со следующими параметрами:

– *Имя* – наименование канала;

– *Тип доступа* – чтение регистра хранения (код 03), чтение входных регистров (код 04), запись в один регистр (код 06), запись в несколько регистров (код 16), запись/чтение регистров (код 23);

– *Триггер* – тип запроса: CYCLIC – запрос выполняется периодически, RISING_EDGE – запрос выполняется по переднему фронту логической переменной;

– *Время цикла* – интервал между опросами в миллисекундах, указывается если триггер имеет значение CYCLIC: (должен быть равен или кратным времени цикла приложения);

– *Комментарий* – краткое описание передаваемых данных;

– *Сдвиг* – номер регистра для чтения (значение от 0 до 65535);

– *Длина* – количество регистров (слов), которые будут прочитаны;

– *Обработка ошибок* – обработка данных в случае возникновения ошибок соединения: *Set to ZERO* устанавливает все значения в 0, *Keep last Value* сохраняет предыдущее значение.

6.1.7 Привязка каналов к переменным.

После задания каналов и режимов работы выполняется привязка читаемых или задаваемых регистров к переменным проекта на вкладке *Соотнесение входов/выходов*. Для этого выбирается канал после двукратного клика по полю *Переменная*. После этого станет доступным диалоговое окно *Ассистент ввода*, в котором связывается выбранный регистр с переменной проекта.

У ведомых устройств имеется вкладка *Modbus Slave Init*, на которой выполняется предварительная инициализация данных. Методика добавления регистров для инициализации аналогична методике добавления каналов п. 6.1.6. При этом имеется дополнительный пункт – значение инициализации.

6.1.8 Создание программы управления обменом данных.

В окне PLC_PRG записывается программа, реализующая заданный алгоритм обмена данными. Это выполняется аналогично пп. 3.1.6–3.1.8, 4.1.9, 5.1.3–5.1.6. При необходимости вводятся дополнительные переменные и функциональные блоки.

6.1.9 Визуализация программы обмена данными.

Визуализация обмена данными выполняется аналогично пп. 4.1.4–4.1.6 и 5.1.7. Для отображения состояния дискретных сигналов рекомендуется использовать элемент *Индикатор*, для численных значений – элемент *Прямоугольник*.

6.1.10 Компиляция, загрузка и отладка проекта.

Компиляция проекта выполняется клавишей F11. При отсутствии ошибок в режиме эмуляции загружается проект клавишами Alt+F8. Старт проекта производится клавишей F5. Следует показать преподавателю реализацию задания обмена данными. Остановка отладки клавишами Shift+F8. Выгрузка проекта выполняется нажатием Ctrl+F8.

6.1.11 Оформление отчета.

В текстовом редакторе создается электронный документ отчета, содержание которого соответствует подразделу 5.2. Отчет показывается преподавателю и в случае его одобрения выполняется защита путем ответа на контрольные вопросы.

6.1.12 Завершение работы со средой программирования и выход из ПК.

Перед завершением работы следует сохранить проект и отчет на ПК. Рекомендуется выполнить их резервное копирование на мобильный носитель.

6.2 Содержание отчета

Отчет по работе № 6 оформляется индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист;
- текст индивидуального задания;
- данные настройки конфигурации интерфейсов используемых всех устройств в виде экранных копий;
- текст раздела описания переменных;
- копии экранов с параметрами настройки каналов передачи данных;
- текст программы обмена данными;
- копии экрана визуализации программы обмена данными.

Контрольные вопросы

1 Дать определение понятия «промышленный интерфейс».

2 Какие виды интерфейса используются для обмена данными между элементами системы управления МС?

3 Какие основные характеристики имеет промышленный интерфейс, используемый в мехатронных системах?

4 Какие режимы работы имеют устройства, входящие в состав промышленной сети?

5 Какие способы передачи данных между устройствами используются в среде программирования CODESYS?

6 Каким образом выполняется конфигурация интерфейса обмена данными в среде CODESYS?

7 Дать определение понятия «канал».

8 Какие основные параметры используются при настройке канала обмена данными на ведомых устройствах?

9 Для каких целей выполняется инициализация регистров обмена данными?

10 Какую физическую реализацию имеет последовательный интерфейс?

Дополнительно могут задаваться контрольные вопросы к работам № 3–5.

7 Лабораторная работа № 7. Программирование в SCADA

Цель работы:

– изучение структуры и основных характеристик SCADA-систем, используемых для управления мехатронных систем;

– освоение интерфейса SCADA TraceMode;

– получение практических навыков работы со SCADA TraceMode при разработке интерфейса оператора для мехатронных систем.

Задание

Магистрант получает индивидуальное задание у преподавателя, проводящего лабораторную работу, на создание интерфейса оператора для взаимодействия с системой управления МС, исходные данные совпадают с таблицей 3.1.

7.1 Ход выполнения работы

7.1.1 Загрузка среды SCADA.

После входа в аккаунт пользователя на ПК запускается интегрированная среда SCADA TraceMode.

7.1.2 Создание нового проекта SCADA.

В среде TraceMode создается новый проект, используя клавиши Ctrl + N. Далее выполняется сохранение проекта в своем рабочем каталоге на ПК. Если нужно загрузить ранее созданный проект, то используются клавиши Ctrl + O и в диалоговом окне указывается имя нужного проекта.

7.1.3 Создание узла.

В навигаторе (левое окно с деревом проекта) следует выполнить клик правой кнопкой мыши по ветке *Система*. В раскрывшемся всплывающем меню следует выбрать пункт *Создать узел/RTM*. В свойствах узла (отдельном диалоговом окне, вызываемом одноименном пунктом всплывающего меню) указывается его имя, кодировка и атрибуты.

7.1.4 Создание каналов.

На ветке созданного узла кликом по пиктограмме *Каналы* в дереве проекта вызывается всплывающее меню, в котором выбирается пункт *Создать компонент/Канал_HEX16* (для целочисленного или логического типа данных) или *Канал_FLOAT* (для вещественного типа данных).

7.1.5 Создание элемента программы.

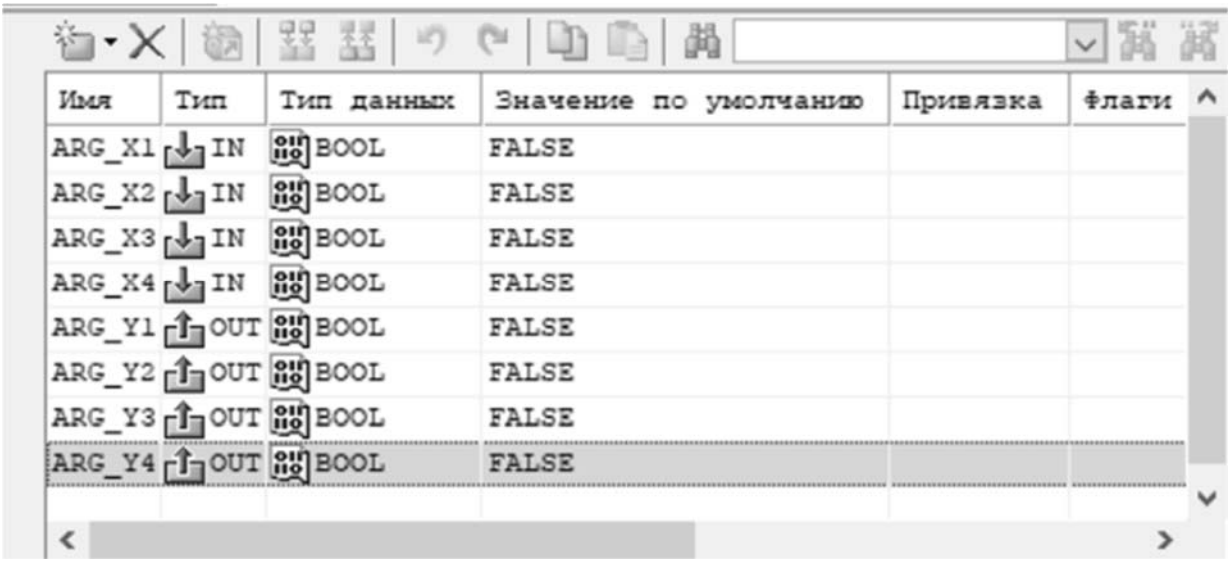
В навигаторе проекта выполняется клик по пиктограмме созданного в п. 7.1.3 узла, чем вызывается всплывающее меню. В меню выбирается пункт *Создать компонент/Программа*. Настройка параметров программы выполняется в специальном диалоговом окне, которое вызывается выбором во всплывающем меню пункта *Редактировать*.

7.1.6 Описание переменных и программирование алгоритма.

Для создания программы выполняется клик мышкой по наименованию

созданной программы, что распахнет в рабочей области отдельную закладку, в которой отражается структура выбранной программы. Следует выполнить клик по ветке *Аргументы* в структуре программы, что распахнет в правой части экрана таблицу, в которую записываются входные и выходные аргументы программы.

Для каждого аргумента в отдельной колонке таблицы последовательно указывается имя аргумента, выбирается из выпадающего списка тип аргумента (для входных – `IN`, а для выходных – `OUT`), указывается из списка тип данных и начальное значение. Пример описания аргументов для решения задачи, показанной на рисунке 3.2, дается на рисунке 7.1.



Имя	Тип	Тип данных	Значение по умолчанию	Привязка	Флаги
ARG_X1	IN	BOOL	FALSE		
ARG_X2	IN	BOOL	FALSE		
ARG_X3	IN	BOOL	FALSE		
ARG_X4	IN	BOOL	FALSE		
ARG_Y1	OUT	BOOL	FALSE		
ARG_Y2	OUT	BOOL	FALSE		
ARG_Y3	OUT	BOOL	FALSE		
ARG_Y4	OUT	BOOL	FALSE		

Рисунок 7.1 – Пример заполнения аргументов программы в табличном редакторе

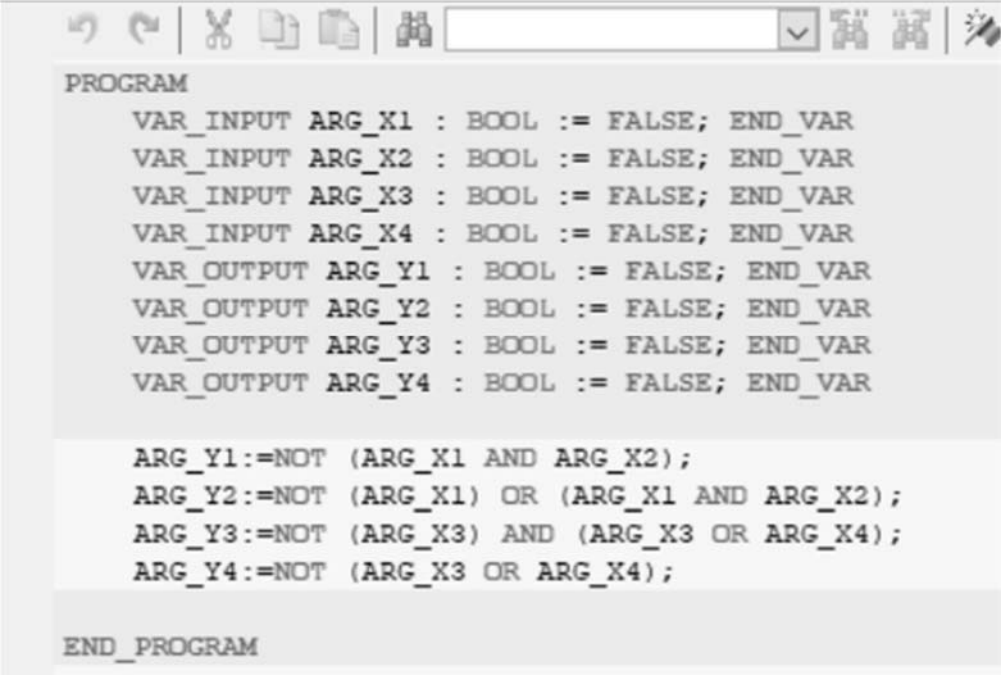
7.1.7 Программирование алгоритма задания.

Кликом по наименованию программы вызывается окно выбора языка программирования, в котором выбирается установкой маркера язык ST. В правой части окна в отдельной закладке отображается шаблон для формирования текста программы. В начале шаблона после слова `PROGRAM` уже автоматически прописаны, созданные в п. 7.1.6 аргументы программы.

После описания переменных-аргументов при необходимости можно задавать локальные переменные в секции `VAR…END_VAR`. После описания переменных на языке ST записываются выражения выходных аргументов, реализующие заданные математические выражения. Пример программы на языке ST, описывающий схему рисунка 3.2, дана на рисунке 7.2.

7.1.8 Создание экрана узла.

Во всплывающем меню, вызываемом кликом по имени узла в дереве проекта, выбирается пункт *Создать компонент/Экран*. Это распахнет в правой части экран графического редактора и раскроет дополнительную панель графических элементов.



```
PROGRAM
    VAR_INPUT ARG_X1 : BOOL := FALSE; END_VAR
    VAR_INPUT ARG_X2 : BOOL := FALSE; END_VAR
    VAR_INPUT ARG_X3 : BOOL := FALSE; END_VAR
    VAR_INPUT ARG_X4 : BOOL := FALSE; END_VAR
    VAR_OUTPUT ARG_Y1 : BOOL := FALSE; END_VAR
    VAR_OUTPUT ARG_Y2 : BOOL := FALSE; END_VAR
    VAR_OUTPUT ARG_Y3 : BOOL := FALSE; END_VAR
    VAR_OUTPUT ARG_Y4 : BOOL := FALSE; END_VAR

    ARG_Y1:=NOT (ARG_X1 AND ARG_X2);
    ARG_Y2:=NOT (ARG_X1) OR (ARG_X1 AND ARG_X2);
    ARG_Y3:=NOT (ARG_X3) AND (ARG_X3 OR ARG_X4);
    ARG_Y4:=NOT (ARG_X3 OR ARG_X4);

END_PROGRAM
```

Рисунок 7.2 – Пример программы на языке ST

В свойствах экрана устанавливается цвет его фона. С помощью следующих специальных кнопок из панели графических элементов формируется на экране изображение интерфейса оператора:

– текст – формирование текстовой статической надписи;

– линия – рисование прямой линии;

– прямоугольник – изображение прямоугольника с имитацией трехмерности;

– ломаные и кривые – отображение линий, состоящих из прямоугольных и кривых сегментов;

– плоские фигуры – рисование различных геометрических фигур.

Используя кнопки инструментальной панели редактирования экрана, создается и настраивается заданный вид интерфейса оператора узла. Для дискретных входных сигналов используются прямоугольные кнопки, для индикации сигналов рисуются эллипсы из плоских фигур. При настройке цвета индикаторов устанавливается свойство *Вид индикации* в вариант Arg = Const.

Пример реализации интерфейса дан на рисунке 7.3.

7.1.9 Привязка фигур к переменным SCADA.

Каждая из кнопок привязывается к созданным входным аргументам программы, а каждая из окружностей – к выходным.

7.1.10 Компиляция программы и эмуляция проекта.

Для компиляции проекта в навигаторе выполняется переход на созданную программу. После раскрытия в рабочем окне программы в главном меню добавится пункт *Программа*. В меню программы выбором пункта *Компиляция* (или F7) выполняется компиляция. При отсутствии сообщения об ошибках выполняется запуск откладки пунктом *Старт* или клавишей F5, а останов отладки – Shift + F5.

Правильность срабатывания индикаторов проверяется на основе аналогичных данных лабораторной работы № 4.

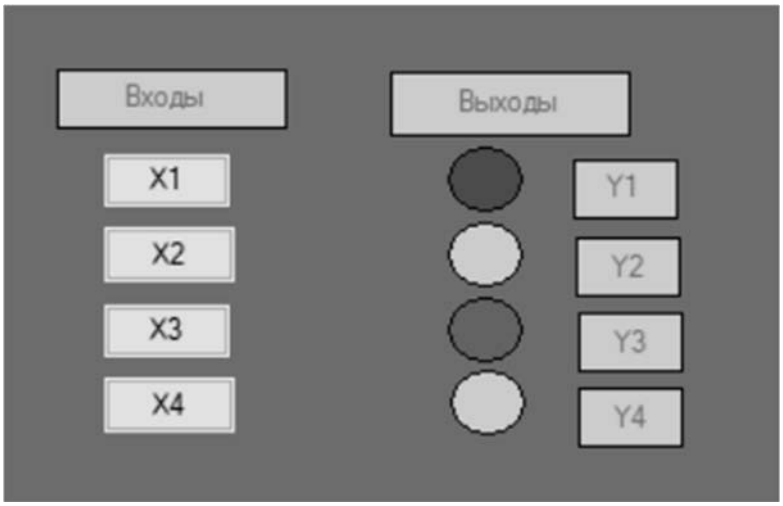


Рисунок 7.3 – Пример реализации интерфейса экрана узла

7.1.11 Оформление отчета.

В текстовом редакторе создается электронный документ отчета, содержание которого соответствует подразделу 7.2. Отчет вместе с созданным проектом в SCADA показывается преподавателю и в случае его одобрения, выполняется его защита путем ответа на контрольные вопросы.

7.1.12 Завершение работы со SCADA и выход из аккаунта ПК.

Перед завершением работы со SCADA следует сохранить разработанный проект. Сохраняется при выходе из текстового редактора электронный отчет. Рекомендуется выполнить их резервное копирование на мобильный носитель.

7.2 Содержание отчета

Отчет по работе № 7 оформляется индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист;
- текст индивидуального задания;
- исходные данные задания работы;
- копия экрана SCADA с созданным узлом и его ресурсами: каналом, экраном и программой;
- копия экрана SCADA с составом аргументов и иных переменных;
- копия программы, реализующей алгоритм задания;
- копия экрана с интерфейсом оператора.

Контрольные вопросы

1 Дать определение понятия «SCADA».
2 Дать определение понятия «канал» в SCADA.

3 Дать определение понятия «узел» в SCADA.

4 Какие типы узлов используются в SCADA TraceMode?

5 Какой состав интерфейса имеет SCADA TraceMode?

6 Какие типы данных переменных используются при программировании в SCADA TraceMode?

7 Какие виды переменных используются при составлении программы в SCADA TraceMode?

8 Какие языки программирования поддерживает SCADA TraceMode?

9 Каким образом в программе SCADA TraceMode задаются двоичные и шестнадцатеричные константы?

10 Каким образом в программе SCADA TraceMode задаются временные константы?

11 Каким образом в программе SCADA TraceMode задаются символьные константы?

12 Какое назначение имеют в составе SCADA группы источников (приемников)?

13 Какие элементы используются для проектирования интерфейса экрана оператора в SCADA TraceMode?

Дополнительно могут задаваться контрольные вопросы к работам № 3–6.

8 Лабораторная работа № 8. Оптимизация МС в среде MATLAB/Simulink

Цель работы:

– изучение методов составления математических моделей МС для оптимизации параметров регуляторов;

– получение практических навыков работы по оптимизации параметров регуляторов цифровых систем управления позиционных устройств МС в среде MATLAB/Simulink.

Задание

Составить динамическую модель МС в среде *MATLAB/Simulink* с целью параметрической оптимизации настроек цифровых регуляторов двухконтурной системы управления положением исполнительного механизма.

Исследуемая МС включает: два цифровых пропорционально-интегрально-дифференцирующих (ПИД) регулятора с датчиками скорости и положения, преобразователь частоты, управляющий асинхронным двигателем с короткозамкнутым ротором (АД), выход которого вращает через редуктор линейно перемещающийся исполнительный механизм. Математическая модель системы в виде структурной схемы показана на рисунке 8.1, значения параметров передаточных функций и воздействий указываются в таблице 8.1.

В ходе работы следует:

1) спроектировать модель системы в среде MATLAB/Simulink (S-модель);
2) выполнить настройку параметров ПИД-регуляторов;

3) исследовать режим отработки заданного положения с пуском на холостом ходу и включением ступенчатого возмущающего воздействия, имеющего реактивный характер и изменяющегося скачком от нуля до заданного значения M_c;

4) определить параметры дискретных передаточных функций оптимизированных регуляторов, динамические характеристики положения, скорости, момента на валу АД, частота на выходе ПЧ, величина напряжения управления на выходе системы управления;

5) по результатам моделирования определить показатели качества регулирования положения механизма и построить статическую механическую характеристику исследуемой системы (зависимость положения механизма от момента нагрузки на валу двигателя).

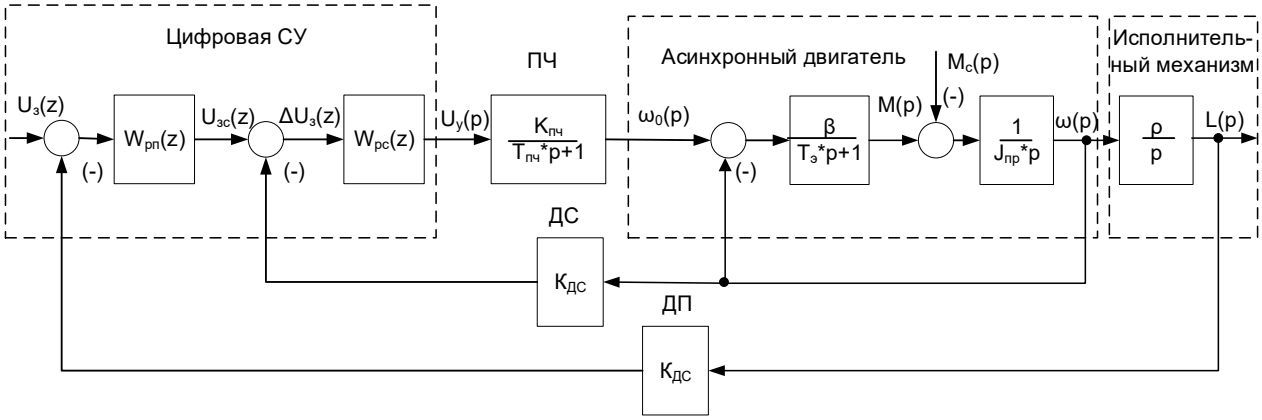


Рисунок 8.1 – Структурная схема исследуемой мехатронной системы

Таблица 8.1 – Вариант заданий параметров исходной математической модели МС

Вариант	1	2	3	4	5	6	7	8	9	10
$U_з$, В	8	8	8	8	9	8	8	8	8	7
$U_{умах}$, В	10	10	10	10	10	10	10	10	10	10
$J_{пр}$, кг·м^2	0,04	0,05	0,055	0,06	0,08	0,1	0,12	0,14	0,18	0,34
$K_{пч}$	32	32	32	32	32	32	32	32	32	32
$T_{пч}$, с	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
$K_{дс}$, В·с	0,0318									
$K_{дп}$, В/м	1	1	1	1	1	1	1	1	1	1
β, Н·м·с	0,240	0,452	0,632	0,837	1,468	1,463	5,642	5,643	8,663	10,52
$T_э$, с	0,092	0,123	0,141	0,142	0,246	0,253	0,381	0,419	0,353	0,414
M_c, Н·м	3	4	5	8	10	12	15	24	35	42
ρ, м	0,02	0,025	0,04	0,05	0,05	0,1	0,08	0,04	0,05	0,05
$K_{дп}$	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
$t_н$, с	2	2,5	3	3	3	3	3	3	3	3
$t_{сч}$, с	4	4,5	5	5	5	5	5	5	5	5

8.1 Ход выполнения работы

8.1.1 Загрузка среды MATLAB/Simulink.

После входа в аккаунт ПК следует выполнить запуск среды системы MATLAB. После загрузки которой выполняется запуск интерфейса моделирования *Simulink*.

8.1.2 Создание файла Simulink-модели.

Следует создать новую Simulink-модель (S-модель). В главном окне MATLAB формируется новый m-файл для задания исходных данных.

8.1.3 Создание модели непрерывной части системы.

В S-модель добавляются блоки *Transfer Fcn* (из библиотеки *Continuous*), описывающие динамические звенья непрерывной части МС. В окнах настройки параметров полиномов числителя и знаменателя передаточной функции указываются имена глобальных переменных, значения которых задаются оператором присваивания в m-файле исходных данных. Пропорциональные динамические звенья моделируются блоком *Gain* из *Math operations.* Для моделирования задающего воздействия используется блок *Constant*, а для возмущающего *Step* из библиотеки *Sources*. Суммирование сигналов выполняется блоком *Sum* из библиотеки *Math operations*. Реактивный характер нагрузки моделируется блоком *Dot Product* (*Math operations*), который перемножает сигналы нагрузки и выход с блока Sign выделения знака сигнала скорости.

8.1.4 Формирование цифровой части модели системы.

В цифровой части системы для цифровых регуляторов устанавливается блоки *Discrete PID Controller* (из *Discrete*). В модель добавляются блоки дискретизации *Quantizer* (из *Discontinues*). В поле блока *Quantizer* задается значение интервала дискретизации – 0,01 с. Выходы *Quantizer* подаются на отрицательные входы сумматоров. S-модель примет вид, показанный на рисунке 8.2.

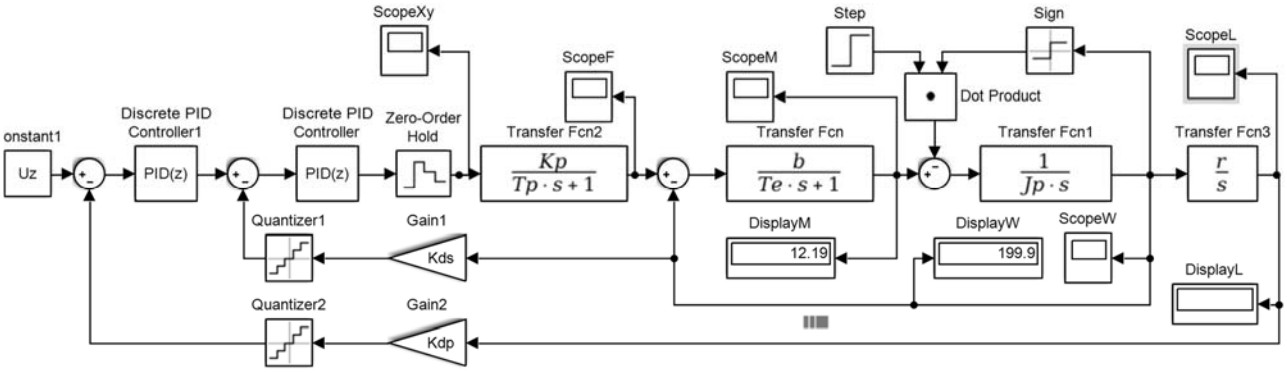


Рисунок 8.2 – S-модель исследуемой системы

8.1.5 Использование блоков фиксации результатов эксперимента.

Фиксация численных значений сигналов выполняется блоком *Display*, а графиков динамических характеристик блоком *Scope* из библиотеки *Sinks.*

К существующему набору блоков фиксации следует добавить два элемента фиксации данных из *Sinks*:

1) блок *Scope* для хранения динамической характеристики положения, в окне параметров которого на вкладке *Logging*, устанавливается маркер в позицию *Log data to workspace*, задается имя глобальной переменной в поле *Variable name*, устанавливается в списке *Save format* значение *Array*;

2) блок *Display* для отображения численного значения положения.

S-модель исследуемой системы примет окончательный вид (рисунок 8.3).

8.1.6 Оптимизация регулятора скорости.

Настройка регуляторов выполняется с внутреннего контура скорости. Для оптимизации РС следует модифицировать модель, временно отключив регулятор положения и оборвав обратную связь по положению, как это показано на рисунке 8.3.

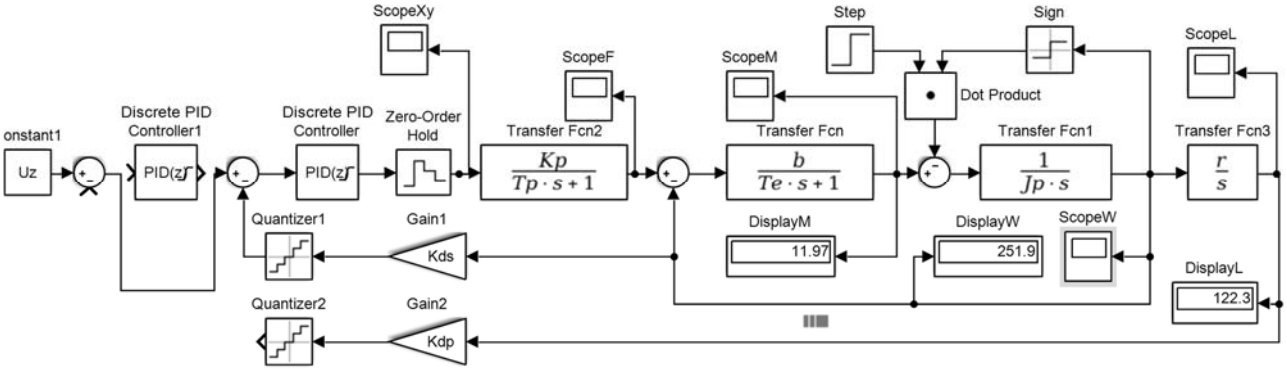


Рисунок 8.3 – Преобразование модели для настройки регулятора скорости

Перед настройкой регуляторов следует обязательно рассчитать m-файл исходных данных, нажав клавишу F5. Для настройки регулятора выполняется клик по блоку *Discrete PID Controller*, вызвав диалоговое окно настройки параметров. В распахнувшемся окне сначала следует перейти на вкладку *PID Advances* и установить маркер в позицию *Limit output* для задания насыщения на выходе на уровне $U_{ymax.}$ Имя глобальной переменной из файла исходных данных с этим значением должно записываться в поле *Upper saturation limit*, а в поле *Lower saturation limit* – как отрицательное значение.

Затем следует перейти в окне на вкладку *Main* и нажать кнопку настройки *Tune*. После этого отобразится окно настройки параметров передаточной функции с графиком переходной характеристики, показанное на рисунке 8.4.

Настройка параметров регулятора выполняется с помощью двух ползунковых переключателей в верхней части окна. Сначала перемещается верхний переключатель между значениями *slower* и *faster* таким образом, чтобы число колебаний переходной характеристики не превышало три. Время регулирования должно быть не более чем половина значения времени $t_н$ из таблицы 8.1. С помощью нижнего переключателя формируется желаемый вид переходной характеристики, величина перерегулирования которой не должна превы-

шать 50 %. Скорректированная переходная характеристика отображается на графике сплошной линией, а предыдущий вариант настройки – штриховой. При изменении настроек параметры регулятора отображаются в строке статуса окна. В случае получения приемлемого вида переходной характеристики выполняется сохранение параметров настроек регулятора нажатием кнопки *Update Block*. Окно настроек параметров может быть закрыто.

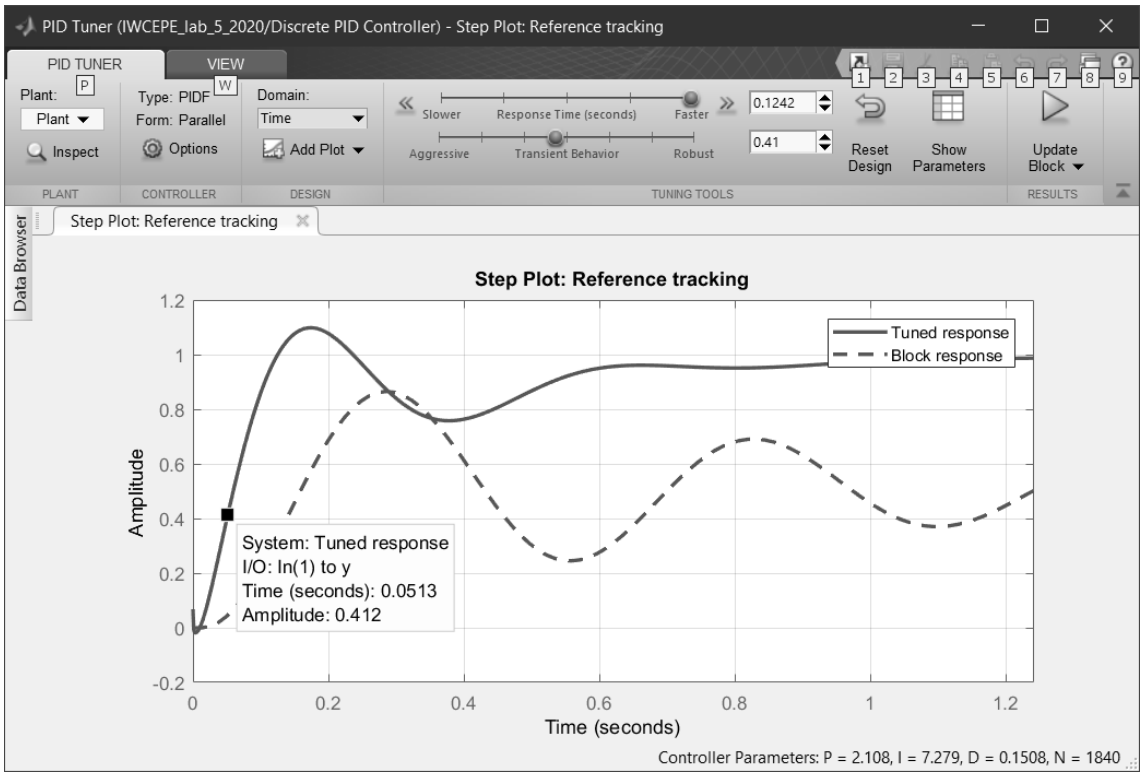


Рисунок 8.4 – Окно настройки регулятора по переходной характеристике системы

Для проверки настройки регулятора выполняется тестовый вычислительный эксперимент с настроенным временем счета, равным значению $t_н$, которое можно указать в поле *Simulation Stop Time* в окне S-модели. Запуск расчета выполняется нажатием кнопки *Run* или клавиш Ctrl + T. Если вид динамической характеристики скорости удовлетворяет вышеперечисленным требованиям, тогда график характеристики копируется через буфер обмена в электронный файл отчета с помощью пункта меню *File/Copy to Clipboard*. Также в отчет помещается изображение S-модели, использованной отладки регулятора и выполняется копия главного окна настроек регулятора или переписываются данные полей *Proportional*, *Integral*, *Derivative* и *Filter coefficient*.

8.1.7 Настройка регулятора положения.

Для настройки регулятора положения S-модель возвращается к виду, показанному на рисунке 8.2. Процедура настройки ПИД-регулятора положения аналогична описанной в п. 8.1.6. Отличием является то, что время регулирования переходной характеристики положения должно быть больше, чем время регулирования контура скорости не менее чем в 1,5 раза. Также выполняется

тестирование переходной характеристики регулирования положения без наброса нагрузки путем проведения вычислительного эксперимента. В отчет копируется график переходной характеристики положения, а также полученные параметры регулятора положения.

8.1.8 Проведение вычислительного эксперимента с S-моделью.

Настройка параметров времени моделирования согласно варианту таблицы 8.1 выполняется в окне S-модели в поле *Simulation Stop Time*. При необходимости более детальной настройки выбирается пункт *Simulation/Model Configuration Parameters* (*Ctrl+E*). В разделе окна *Solve options* следует уменьшить порядок точности численного метода с фиксированным шагом интегрирования (*Fixed-step*) в списке *Solve* до двух или трех значений. Значение шага интегрирования 0,0001, показанное ранее в поле *Fixed-step size*, изменять не рекомендуется.

8.1.9 Фиксация и анализ результатов моделирования.

Запуск вычислительного эксперимента с S-моделью производится кнопкой *Run* или нажатием клавиш Ctrl+T. Если за указанное в таблице 8.1 время расчета система не успела отработать заданное перемещение, тогда изменяется настройка регулятора положения. Увеличить время расчета можно лишь в том случае, если после перенастройки регулятора динамический процесс положения имеет большое перерегулирование или число его колебаний за время регулирования превышает 3. Полученные графики динамических характеристик напряжения управления, момента, скорости и положения копируются в отчет с помощью *Edit/Copy Figure* с использованием векторного формата (метафайла).

8.1.10 Окончание работы со средой MATLAB.

По окончании работы со средой моделирования выполняется сохранение всех открытых файлов и закрывается главное окно MATLAB. Выполняется резервное копирование файлов на мобильный носитель или в сеть. Следует удалить все ненужные файлы из внешней памяти компьютера.

8.2 Содержание отчета

Отчет по работе № 8 оформляется на листах формата А4 в соответствии с требованиями ГОСТ 7.32 со следующим содержанием:

- m-файл исходных данных модели;
- S-модель, использованная для настройки регулятора скорости МС;
- S-модель всей мехатронной системы;
- полученные параметры настройки регулятора скорости;
- параметры настройки регулятора положения;
- динамическая характеристика настройки контура скорости;
- динамическая характеристика настройки контура положения;
- динамические характеристики положения, скорости, момента, частоты сети и изменения напряжения задания.

Контрольные вопросы

1 Дать определение понятия «среда компьютерной математики».
2 Дать определение понятия «оптимизация».
3 Дать определение понятия «параметрическая оптимизация».
4 Каким образом моделируется ПИД-регулятор в среде Simulink?
5 Каким образом выполняется настройка ПИД-регулятора скорости?
6 Как устанавливается ограничение выходного сигнала блока *Discrete PID Controller*?
7 Какие параметры используются для оптимизации настроек составляющих регулятора в блоке *Discrete PID Controller*?
8 Как в среде MATLAB/Simulink моделируется исполнительный механизм?

Список литературы

1 **Шишов, О. В.** Программируемые контроллеры в системах промышленной автоматизации : учебник / О. В. Шишов. – Москва: ИНФРА-М, 2021. – 365 с.
2 **Гагарина, Л. Г.** Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул; под ред. Л. Г. Гагариной. – Москва : ФОРУМ; ИНФРА-М, 2019. – 400 с.
3 **Гагарина, Л. Г.** Введение в теорию алгоритмических языков и компиляторов : учебное пособие / Л. Г. Гагарина. Е. В. Кокорева; под ред. Л. Г. Гагариной. – Москва : ФОРУМ, 2018. – 176 с.
4 **Мякишев, Д. В.** Принципы и методы создания надежного программного обеспечения АСУТП : учебное пособие / Д. В. Мякишев. – 2-е изд. – Москва; Вологда : Инфра-Инженерия, 2021. – 116 с.
5 **Ананьева, Т. Н.** Стандартизация, сертификация и управление качеством программного обеспечения : учебное пособие / Т. Н. Ананьева, Н. Г. Новикова, Г. Н. Исаев. – Москва : ИНФРА-М, 2021. – 232 с.
6 **Павлов, В. П.** Автоматизация моделирования мехатронных систем транспортно-технологических машин: учебное пособие / В. П. Павлов, А. Ю. Ахпашев. – Красноярск : СФУ, 2016. – 144 с.
7 **Матюшин, А. О.** Программирование микроконтроллеров: стратегия и тактика / А. О. Матюшин. – Москва : ДМК Пресс, 2017. – 356 с.
8 **Солем, Я. Э.** Программирование компьютерного зрения на языке Python: пер. с англ. / Я. Э. Солем. – Москва : ДМК Пресс, 2016. – 312 с.
9 **Кангин, В. В.** Разработка SCADA-систем : учебное пособие / В. В. Кангин, М. В. Кангин, Д. Н. Ямолдинов. – Москва ; Вологда : Инфра-Инженерия, 2019. – 564 с.
10 **Башлыков, А. А.** Основы конструирования интеллектуальных систем поддержки принятия решений в атомной энергетике : учебник / А. А. Башлыков, А. П. Еремеев. – Москва : ИНФРА-М, 2021. – 351 с.

11 **Осипова, Н. В.** Программное обеспечение систем управления : учебное пособие / Н. В. Осипова. – Москва : МИСиС, 2019. – 74 с.

12 **Шишов, О. В.** Современные средства АСУ ТП: учебник / О. В. Шишов. – Москва ; Вологда : Инфра-Инженерия, 2021. – 532 с.