

Министерство науки и высшего образования Российской Федерации
Тольяттинский государственный университет

Д.Г. Токарев, А.В. Прядилов, Е.С. Глибин

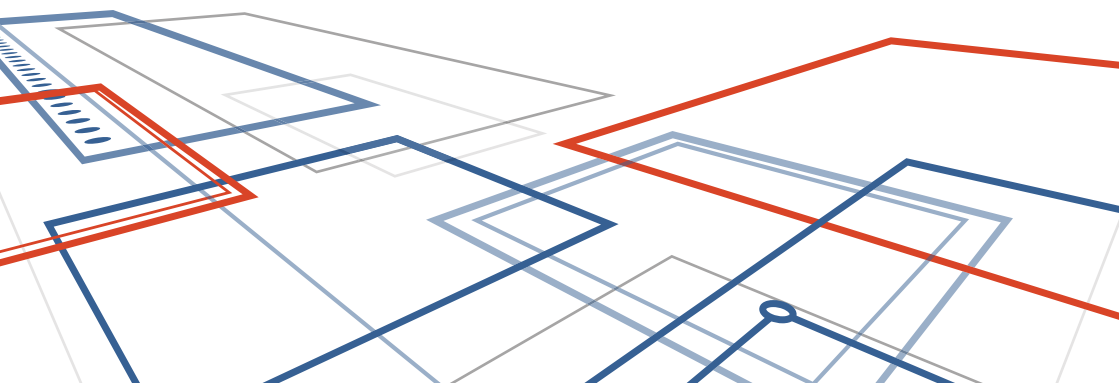
АВТОМАТИЗАЦИЯ ДИСКРЕТНЫХ И НЕПРЕРЫВНЫХ ПРОИЗВОДСТВЕННЫХ СИСТЕМ

Лабораторный практикум

© Токарев Д.Г., Прядилов А.В., Глибин Е.С., 2024

© ФГБОУ ВО «Тольяттинский
государственный университет», 2024

ISBN 978-5-8259-1634-7



УДК 004.31(075.8)
ББК 32.973.26-04

Рецензенты:

д-р техн. наук, профессор, зав. кафедрой «Сервис технических и технологических систем» Поволжского государственного университета сервиса *Б.М. Горшков*;
д-р техн. наук, профессор Тольяттинского государственного университета *В.П. Певчев*.

Токарев, Д.Г. Автоматизация дискретных и непрерывных производственных систем : лабораторный практикум / Д.Г. Токарев, А.В. Прядилов, Е.С. Глибин. – Тольятти : Издательство ТГУ, 2024. – 1 оптический диск. – ISBN 978-5-8259-1634-7.

Лабораторный практикум описывает работу со средой CoDeSys для программирования ПЛК «ОВЕН» для автоматизированных систем управления. Содержит краткие теоретические сведения по устройству программируемого логического контроллера ПЛК 150-220.А-М и языкам программирования, описанным в ГОСТ Р МЭК 61131-3–2016. Приводятся методические указания по выполнению шести лабораторных работ.

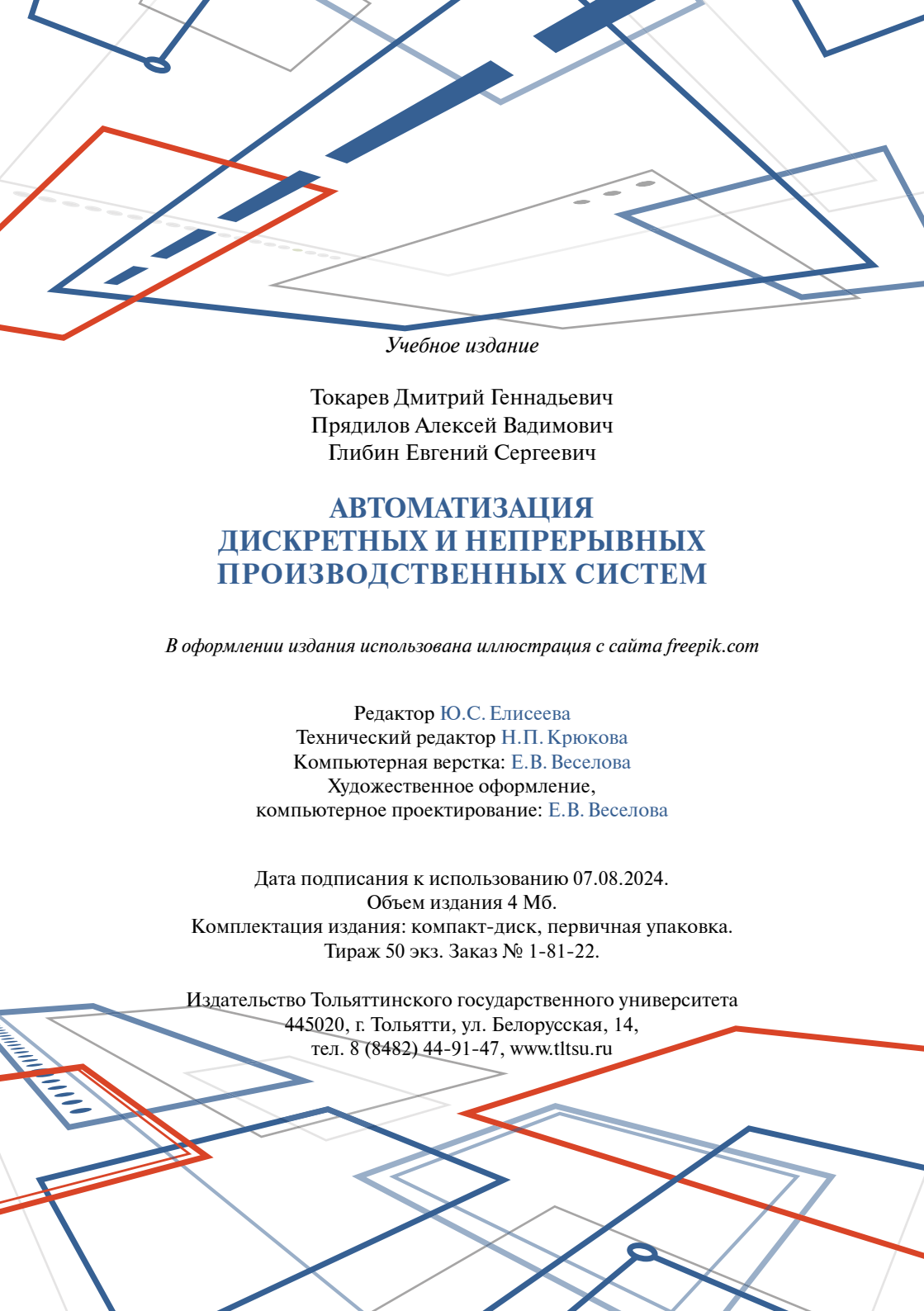
Предназначен для студентов бакалавриата, обучающихся в очной форме (в том числе с использованием ДОТ) по направлению подготовки 11.03.04 «Электроника и нанoeлектроника» (профили «Промышленная электроника» и «Электроника и робототехника»).

Текстовое электронное издание.

Рекомендовано к изданию научно-методическим советом Тольяттинского государственного университета.

Минимальные системные требования: IBM PC-совместимый компьютер: Windows XP/Vista/7/8/10; ПИИ 500 МГц или эквивалент; 128 Мб ОЗУ; SVGA; CD-ROM; Adobe Acrobat Reader.

© Токарев Д.Г., Прядилов А.В., Глибин Е.С., 2024
© ФГБОУ ВО «Тольяттинский
государственный университет», 2024



Учебное издание

Токарев Дмитрий Геннадьевич
Прядилов Алексей Вадимович
Глибин Евгений Сергеевич

АВТОМАТИЗАЦИЯ ДИСКРЕТНЫХ И НЕПРЕРЫВНЫХ ПРОИЗВОДСТВЕННЫХ СИСТЕМ

В оформлении издания использована иллюстрация с сайта [freepik.com](https://www.freepik.com)

Редактор Ю.С. Елисеева
Технический редактор Н.П. Крюкова
Компьютерная верстка: Е.В. Веселова
Художественное оформление,
компьютерное проектирование: Е.В. Веселова

Дата подписания к использованию 07.08.2024.
Объем издания 4 Мб.
Комплектация издания: компакт-диск, первичная упаковка.
Тираж 50 экз. Заказ № 1-81-22.

Издательство Тольяттинского государственного университета
445020, г. Тольятти, ул. Белорусская, 14,
тел. 8 (8482) 44-91-47, www.tltsu.ru

Содержание

Введение	5
Лабораторная работа 1	
Установка и знакомство с CoDeSys V2.3	7
Лабораторная работа 2	
Знакомство с ПЛК «ОВЕН»	15
Лабораторная работа 3	
Знакомство с языками программирования	22
Лабораторная работа 4	
Релейный регулятор температуры.....	43
Лабораторная работа 5	
Пропорциональный регулятор температуры.....	49
Лабораторная работа 6	
ПИД-регулятор температуры	51
Заключение	53
Библиографический список	54
Приложение А.....	55

Введение

Цель освоения дисциплины «Автоматизация дискретных и непрерывных производственных систем» — закрепить у студентов знания о методах и средствах автоматизации производственных процессов и производств в различных отраслях промышленности.

Достижение поставленной цели осуществляется путем решения следующих задач:

- 1) изучение устройства программируемых логических контроллеров;
- 2) знакомство с основными схемами подключения датчиков и исполнительных устройств к контроллеру;
- 3) освоение основных применяемых языков программирования.

В рамках лабораторного практикума предполагается изучение систем автоматизации на примере программируемого логического контроллера ПЛК150-220.А-М фирмы «ОВЕН».

В ходе выполнения лабораторных работ изучается инструментальное программное обеспечение CoDeSys. Осуществляется написание программ, подключение ПЛК к персональному компьютеру для загрузки и отладки программ, создание визуализаций. Рассматриваются различные способы регулирования температуры с помощью контроллера, в том числе ПИД-регулирование.

После выполнения заданий лабораторного практикума студент должен знать синтаксис языков программирования МЭК 6-1131/3: LD, FBD, ST, SFC, IL, а также устройство регуляторов различных типов, уметь разрабатывать непрерывные и дискретные системы автоматического управления, владеть навыками программирования ПЛК «ОВЕН» в среде разработки CoDeSys.

Требования по технике безопасности

1. По способу защиты от поражения электрическим током используемые в лабораторных работах контроллеры соответствуют классу III в соответствии с ГОСТ 12.2.007.0–75

- (в цепях отсутствует опасное для жизни обслуживающего персонала напряжение).
2. При эксплуатации и техническом обслуживании необходимо соблюдать требования ГОСТ 12.3.019–80, «Правил эксплуатации электроустановок потребителей» и «Правил охраны труда при эксплуатации электроустановок потребителей».
 3. Открытые контакты клемм контроллера при эксплуатации находятся под напряжением.
 4. Любые подключения к контроллеру и работы по его техническому обслуживанию производятся только при отключенном питании контроллера и подключенных к нему устройств.
 5. Не допускается попадание влаги на контакты выходных соединителей и внутренние элементы контроллера. Запрещается использование контроллера при наличии в атмосфере кислот, щелочей, масел и иных агрессивных веществ.
 6. Подключение и регулировка программируемого логического контроллера должны производиться только после изучения руководства по его эксплуатации.

Лабораторная работа 1

Установка и знакомство с CoDeSys V2.3

Цель работы – установить программный комплекс CoDeSys V2.3 на рабочее место, проверить его работоспособность, научиться запускать простейшие программы для программируемых логических контроллеров.

Программа работы

1. Скачать и установить программу CoDeSys.
2. Создать и отладить в режиме эмуляции программу, выполняющую управление двумя поочередно мигающими светодиодами – красным и зеленым.
3. Создать визуализацию для данной программы.

Используемое оборудование и материалы

1. Персональный компьютер с установленной операционной системой Windows.
2. Среда разработки CoDeSys.

Указания по выполнению работы

Программный комплекс промышленной автоматизации CoDeSys распространяется бесплатно и может быть без ограничений установлен на нескольких рабочих местах.

Скачайте программу с сайта производителя оборудования для автоматизации «ОВЕН» по ссылке owen.ru. Для этого перейдите в каталог продукции и выберите «Среда программирования CoDeSys», далее перейдите в раздел CoDeSys V2 или воспользуйтесь ссылкой (рис. 1.1): https://owen.ru/product/codesys_v2

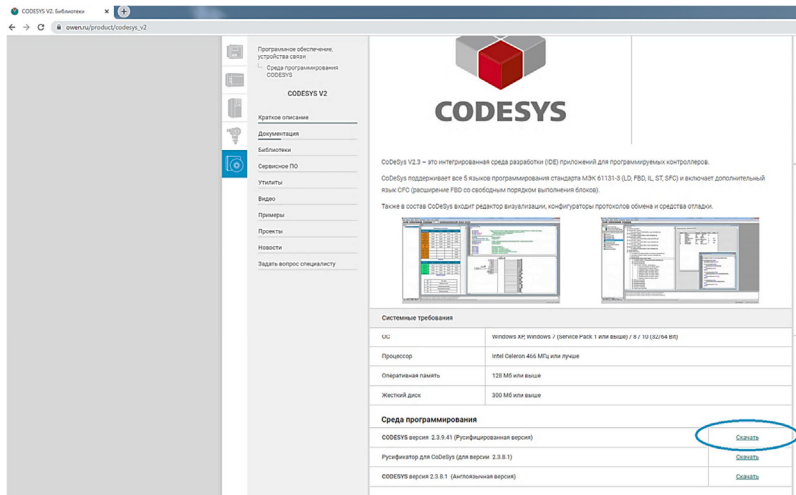


Рис. 1.1. Загрузка CoDeSys

Выполните установку русскоязычной версии с настройками по умолчанию. Хотя вы можете использовать версию 3, а также англоязычную версию для выполнения лабораторных работ, далее будет рассматриваться именно русская версия CoDeSys V2.3.

После установки выполните *Пуск* → *Все программы* → *3S Software* → *CoDeSys V2.3* → *CoDeSys V2.3*. Программа запустится, а на экране появится окно, показанное на рис. 1.2.



Рис. 1.2. Главное окно CoDeSys

Примечание. Если программа на компьютере была установлена до вас, то откроется последний проект. Просто закройте его, выполнив команду меню *Файл* → *Закрыть*.

Выполните команду меню *Файл* → *Создать*, появится диалоговое окно с выбором платформы, показанное на рис. 1.3.

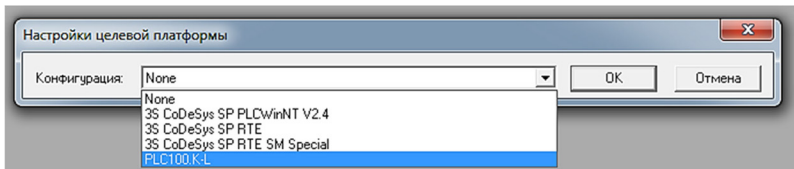


Рис. 1.3. Варианты платформы для нового проекта

Список конфигурации в «чистой» CoDeSys V2.3 будет включать только 4 пункта из 5, показанных на рисунке. Дополнительные конфигурации под используемый программируемый логический контроллер «ОВЕН» будут добавлены вами в последующих лабораторных работах. Первые программы будут отлаживаться в режиме эмуляции, поэтому выберите любую конфигурацию и нажмите *OK*.

Далее создайте новый программный компонент с настройками, показанными на рис. 1.4. Убедитесь, что выбрана «Программа» в качестве типа POU, название **PLC_PRG** менять на надо (аналог main() из C++), в качестве языка реализации выберите LD (Ladder Diagram). Нажмите *OK*.

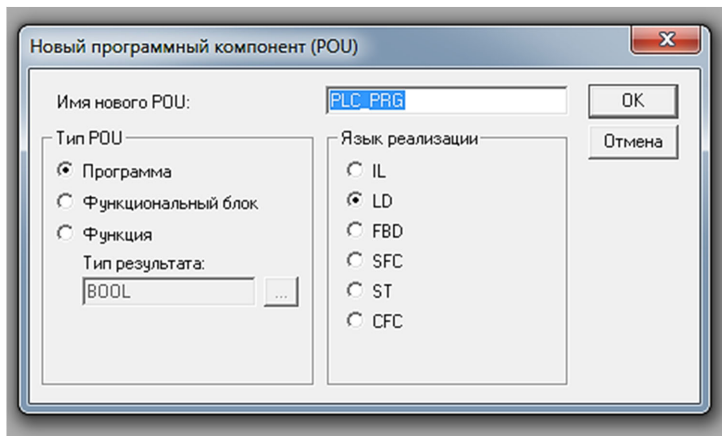


Рис. 1.4. Создание нового программного компонента

Появится окно программы, как показано на рис. 1.5.

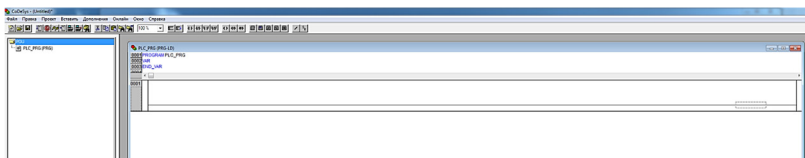


Рис. 1.5. Окно программы на языке LD

Рассмотрим простейшую программу управления двумя поочередно мигающими светодиодами, условно красного и зеленого.

Поочередно добавьте 2 элемента – контакт и таймер, нажав на соответствующие кнопки на панели инструментов или выполнив пункты меню *Вставить* (рис. 1.6). Синими линиями подчеркнуты нужные кнопки.

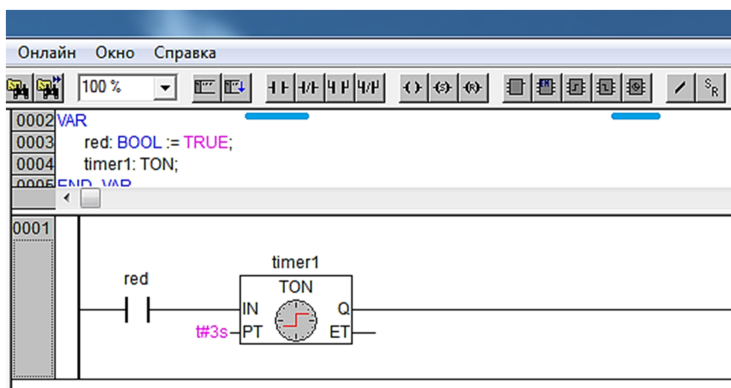


Рис. 1.6. Добавление элементов в программу

Щёлкните на «???» контакта и напишите название переменной – red. После ввода появится окно объявления переменной (рис. 1.7).

Введите TRUE в поле «Нач. значение» и нажмите OK. Это добавит новую переменную логического типа BOOL в программу и присвоит ей значение TRUE при ее запуске. Переменная будет хранить состояние красного светодиода (FALSE – не горит, TRUE – горит).

Аналогичным образом введите timer1 вместо «???» над элементом таймера, все настройки переменной оставьте по умолчанию. Щёлкните возле вывода PT таймера и наберите #3s. Это означает, что после подачи TRUE на вход IN таймера через 3 секунды на выход Q поменяет состояние с FALSE на TRUE.

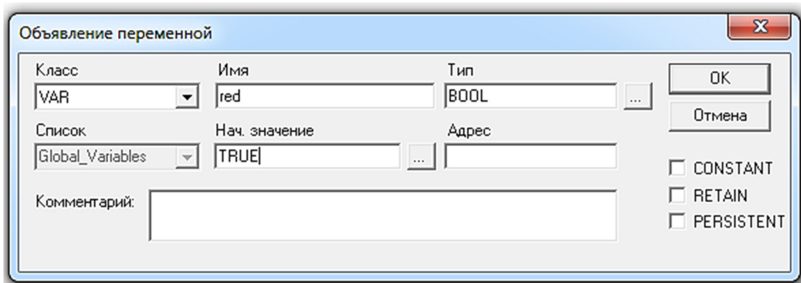


Рис. 1.7. Объявление новой переменной в программе

Добавьте элементы программы, как показано на рис. 1.8.

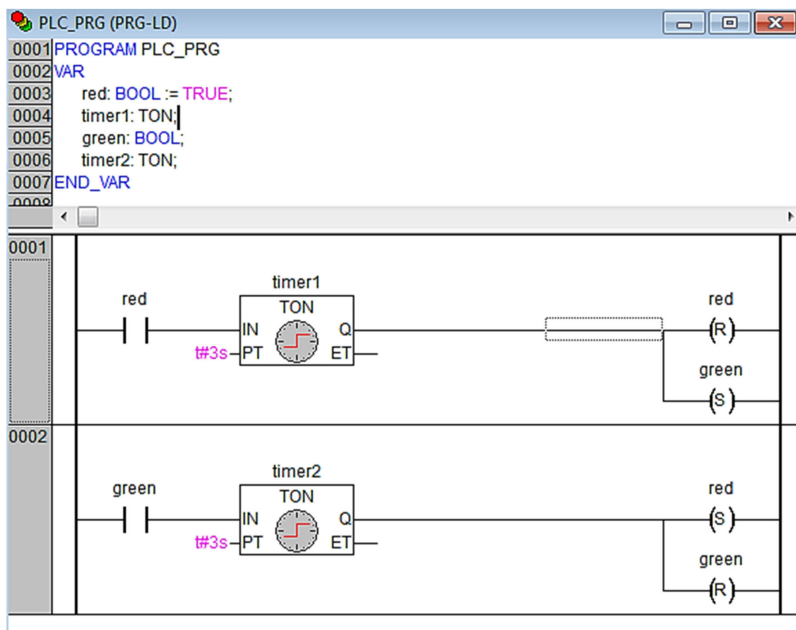


Рис. 1.8. Готовая программа «Светофор»

Сначала добавьте 2 обмотки, нажав на соответствующие кнопки на панели инструментов. Они добавятся параллельно. Присвойте им имена red и green. При создании обе обмотки будут обозначаться как две круглые скобки – (), выделите обмотку red и нажмите на R на панели инструментов, это поменяет тип обмотки на Reset,

и обозначение сменится на (R). Аналогичным образом поменяйте тип обмотки green.

Далее добавьте вторую цепь, нажав на кнопку *Цепь (после)* на панели инструментов. Вы можете выделять цепь, щелкая на серую область с обозначениями «0001» или «0002» в левой части. После выделения цепи все новые элементы будут добавляться в нее. Цепь также можно удалить, нажав на клавишу *Delete*.

Отредактируйте программу, как показано на рис. 1.8. При объявлении всех переменных настройки оставьте по умолчанию.

Запустите программу. Для этого в меню *Онлайн* поставьте галочку в пункте *Режим эмуляции*. Выполните команду *Онлайн* → *Подключение*. На этом этапе, если в программе имеются ошибки, они будут отображены в появившемся окне. Если подключение прошло успешно, выполните команду *Онлайн* → *Старт*. Чтобы иметь возможность отредактировать программу, необходимо ее остановить, а также выполнить команду меню *Онлайн* → *Отключение*.

В завершение для отладки добавьте визуализацию (рис. 1.9).

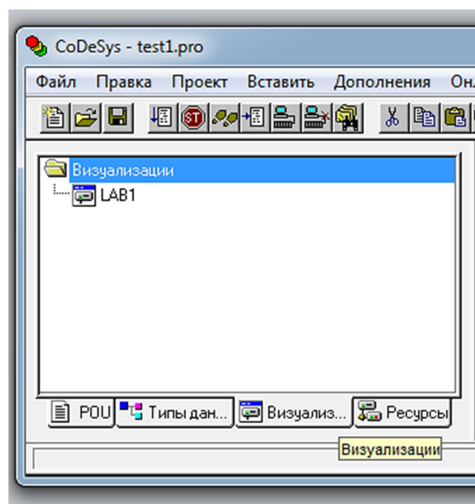


Рис. 1.9. Создание новой визуализации

Перейдите на вкладку *Визуализации* на левой панели. Щелкните правой кнопкой мыши на *Визуализации* и выберите *Добавить объект...* Дайте название, например LAB1.

В окне редактирования добавьте 2 круга и, щелкнув на них дважды, откройте диалог конфигурирования элемента (рис. 1.10).

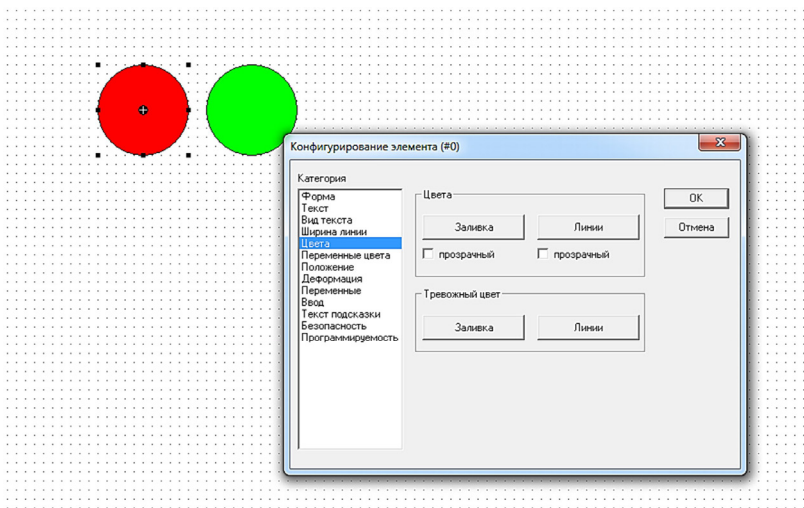


Рис. 1.10. Создание визуализации программы

Залейте один круг красным цветом, другой – зеленым. Далее свяжите видимость кругов с переменными в программе. Перейдите в раздел *Переменные* и в поле «Невидимость» введите PLC_PRG.green для красного круга и PLC_PRG.red – для зеленого (рис. 1.11).

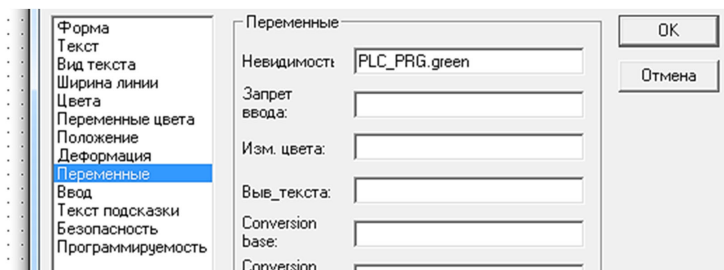


Рис. 1.11. Назначение переменных

Запустите программу.

Выводы

В ходе лабораторной работы выполнена установка и настройка инструментального программного обеспечения CoDeSys, необходимого для разработки программ для ПЛК «ОВЕН». Написана первая программа на языке релейной логики, и создана визуализация.

Лабораторная работа 2

Знакомство с ПЛК «ОВЕН»

Цель работы – научиться загружать программы в программируемые логические контроллеры.

Программа работы

1. Скачать и установить target-файл предоставленного ПЛК в CoDeSys.
2. Создать и отладить в режиме эмуляции программу, выполняющую управление двумя поочередно мигающими светодиодами по алгоритму согласно выбранному варианту.
3. Загрузить программу в ПЛК и запустить ее.

Используемое оборудование и материалы

1. Персональный компьютер с установленной операционной системой Windows.
2. Среда разработки CoDeSys.
3. ПЛК «ОВЕН» 150/160/63 с проводом питания.
4. Преобразователь USB-RS232.
5. Кабель прошивки KC1.

Указания по выполнению работы

На рис. 2.1 приведено изображение ПЛК 150-220.А-М с обозначением выводов. Данный контроллер имеет дискретные (цифровые) и аналоговые входы, дискретные (цифровые, релейные) и аналоговые выходы.

С цифровыми входами и выходами связаны индикаторные светодиоды, указывающие на состояние выводов.

В данной лабораторной работе необходимо загрузить в конкретный контроллер программу, подобную рассмотренной в первой

работе, выполняющую переключение выхода по заданному алгоритму. Модель используемого контроллера может отличаться от ПЛК 150.

На странице, с которой вы загружали среду разработки в лабораторной работе 1, необходимо скачать файл конфигурации предоставленного ПЛК (рис. 2.2).

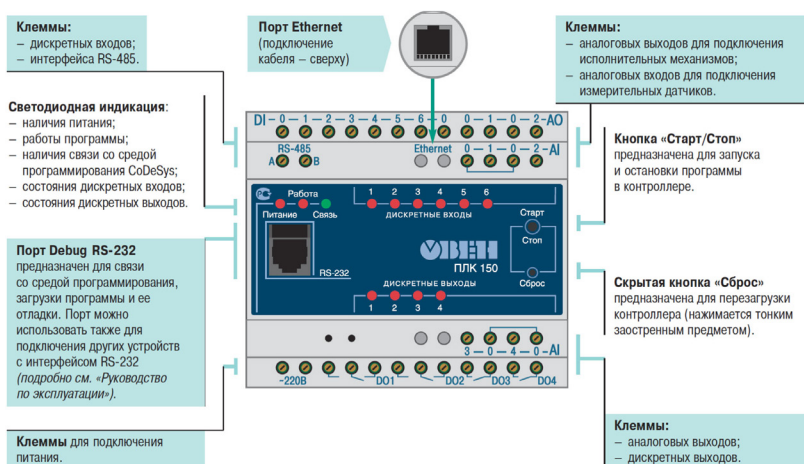


Рис. 2.1. ПЛК «ОВЕН» 150

Для установки target-файла:

1. Распакуйте содержимое архива.
2. Откройте папку, соответствующую модели вашего контроллера.
3. Запустите файл InstallTarget.bat.
4. Установка завершится автоматически.

Или воспользуйтесь утилитой InstallTarget для менеджмента установленных конфигураций (рис. 2.3). Выберите *Пуск – Все программы – 3S Software – CoDeSys V2.3 – InstallTarget*. Укажите папку с установленной CoDeSys и выберите файл формата .INF из архива. В случае успешной установки выбранный ПЛК отобразится в Installed Targets.

После установки ПЛК его можно выбрать в конфигурации при создании нового проекта. Если у вас уже написана программа, то смена ПЛК осуществляется следующим образом. В *Ресурсах* выберите пункт *Настройки целевой платформы* (рис. 2.4) и смените ПЛК. Далее перейдите к пункту *Конфигурация ПЛК* и выберите пункт меню *Дополнения – Стандартная конфигурация* (рис. 2.5).

Сервисное ПО			
Контроллер	Прошивка	Таргет	Драйвер USB
ПЛК110 [M02]	1.1.0	3.3	скачать*
ПЛК160 [M02]	1.1.0		
ПЛК110	2.17.0	2.12	скачать**
ПЛК160	2.17.0		
ПЛК100	2.17.0		
ПЛК150	2.17.0		
ПЛК154	2.17.0		
ПЛК63	2.12	2.01	
ПЛК73	2.16		
Архив			

Рис. 2.2. Загрузка target-файлов и драйверов

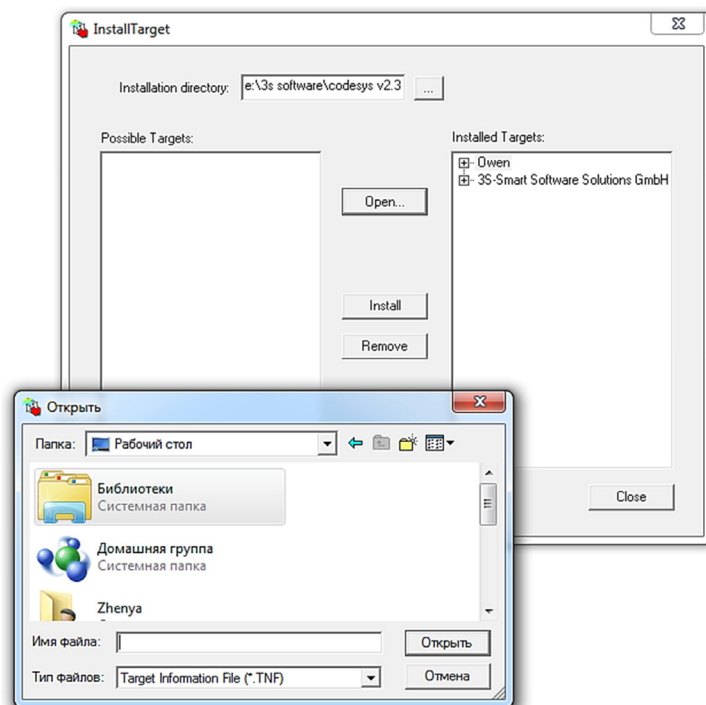


Рис. 2.3. Установка target-файла вручную

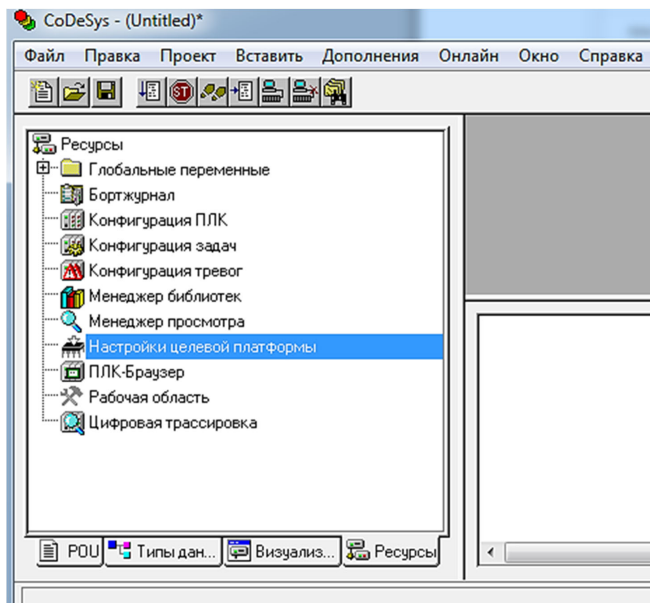


Рис. 2.4. Смена ПЛК в проекте. Шаг 1

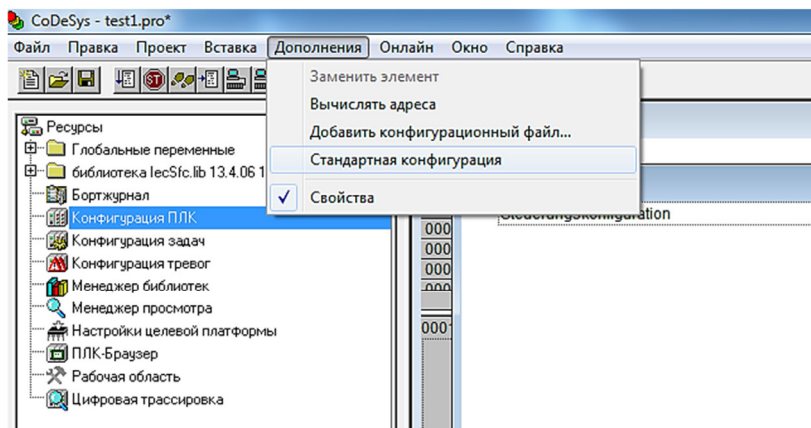


Рис. 2.5. Смена ПЛК в проекте. Шаг 2

Далее напишите свою программу на языке LD согласно варианту, указанному в табл. 2.

Варианты заданий

№ бригады	Последовательность световых импульсов
1	Короткий – длинный – длинный
2	Короткий – короткий – длинный
3	Короткий – длинный
4	Короткий – короткий – длинный – длинный
5	Длинный – длинный – длинный – короткий
6	Короткий – короткий – короткий – длинный

Составьте программу, циклически формирующую световые импульсы светодиодом № 1 в последовательности, указанной в табл. 2.

Длительность паузы всегда постоянна и равна 3 секундам, длительность короткого импульса составляет 5 секунд, длинного импульса – 10 секунд. В каждый момент времени светодиод № 2 должен находиться в противоположном состоянии относительно светодиода № 1 (гореть, если второй не горит).

Проверьте работу с помощью визуализации. Если всё работает корректно, переходите к загрузке программы в ПЛК.

Для этого необходимо связать переменные в программе с выводами ПЛК: в окне *Конфигурация ПЛК* дважды щелкнуть на аббревиатуру АТ и набрать имя переменной (рис. 2.6).

Обратите внимание, что если в программе вы объявляли переменную `red` в `VAR... END_VAR`, то ее необходимо удалить, если вы добавили переменную с таким же именем в конфигурации ПЛК. Необходимость удалить ее объясняется тем, что это будут разные переменные и программа будет изменять переменную, не связанную с выходом контроллера. Или же вы можете поменять объявление с `red: BOOL := TRUE;` на `red АТ %QX1.0: BOOL;`. Во втором случае в конфигурации переменную можно не указывать.

Выполните пункт меню *Проект – Компилировать все*. Убедитесь, что ошибок в программе нет. Далее подключите ПЛК к компьютеру, соблюдая правила техники безопасности.

Создайте новое подключение, выполнив пункт меню *Онлайн – Параметры связи...* (рис. 2.7). Выберите RS-232.

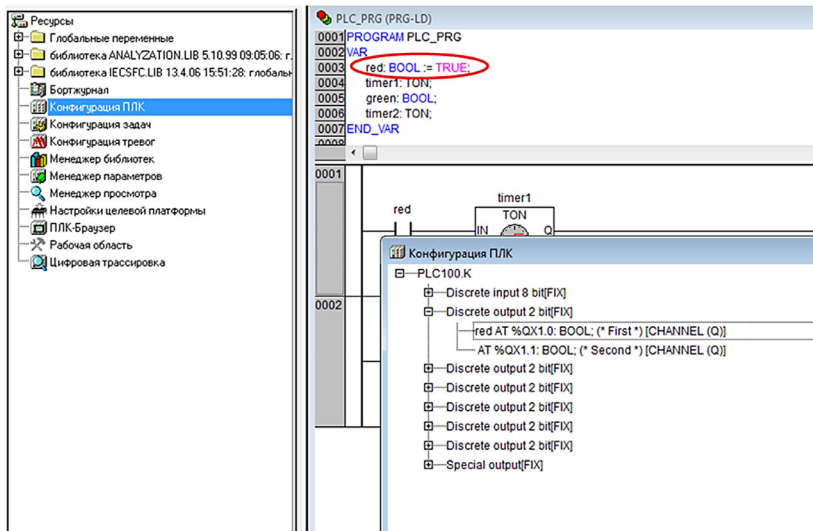


Рис. 2.6. Связывание переменной red из программы (Лабораторная работа 1) с выходом %QX1.0

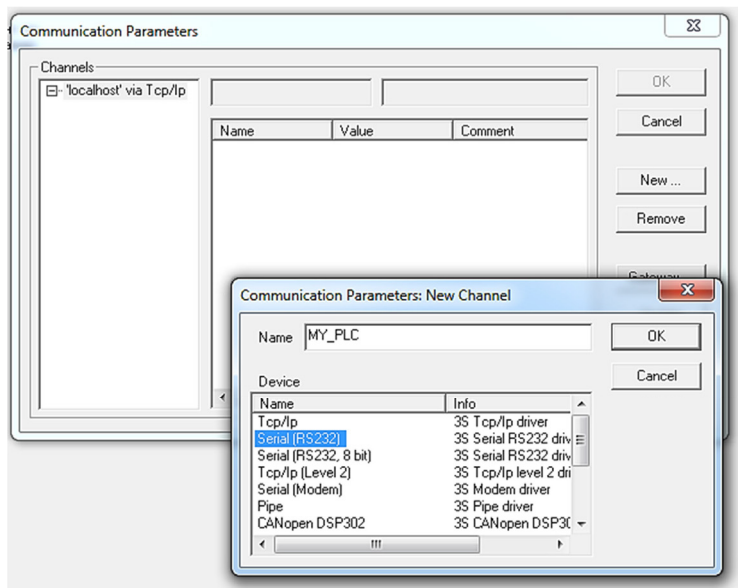


Рис. 2.7. Создание нового подключения к ПЛК

В завершение, выполнив пункт меню *Онлайн – Подключение*, вы сможете автоматически загрузить программу в контроллер. Запуск и прекращение программы осуществляются или нажатием кнопки на самом ПЛК, или в программе CoDeSys через меню *Онлайн – Старт (Стоп)*.

Выводы

В ходе лабораторной работы выполнено подключение ПЛК к компьютеру, установка необходимого target-файла, настройка проекта под конкретный контроллер. Написана программа согласно представленному варианту, загружена в контроллер, отлажена ее работа.

Лабораторная работа 3

Знакомство с языками программирования

Цель работы – ознакомиться с работой ПЛК и его языками программирования.

Программа работы

1. Изучить подключение кнопок к контроллеру.
2. Ознакомиться с основами программирования на языках, описанных ГОСТ Р МЭК 61131-3–2016.
3. Реализовать программу, считывающую состояния входов DI1, DI2, подключенных к кнопкам SB1, SB2 (рис. 3.1), и формирующую сигнал на выходах DO1 и DO2, в соответствии с вариантом из таблицы.
4. Загрузить программу в ПЛК и запустить ее.

Используемое оборудование и материалы

1. Персональный компьютер с установленной операционной системой Windows.
2. Среда разработки CoDeSys.
3. ПЛК «ОВЕН» 150/160/63 с проводом питания.
4. Преобразователь USB-RS232.
5. Кабель прошивки KC1.
6. Кнопки.
7. Соединительные провода.

Варианты заданий

Предлагается выполнить задание в соответствии с вариантами, представленными в табл. 3.1.

Варианты заданий для выполнения лабораторной работы 3

Вариант	DO1	DO2
1	2И-НЕ	RS-триггер
2	2ИЛИ-НЕ	Импульс длительностью 3 с при нажатии на DI1
3	2исКЛИЛИ	Импульс П с задержкой включения 4 с
4	2И	D-триггер

ВНИМАНИЕ! Данное задание необходимо выполнить на всех шести языках программирования. После написания каждой из программ необходимо осуществить онлайн-подключение ПЛК к персональному компьютеру (ПК), запустить и протестировать программу в реальном времени, наблюдая на мониторе ПК за ее работой.

Указания по выполнению работы

Схема лабораторной установки приведена на рис. 3.1.

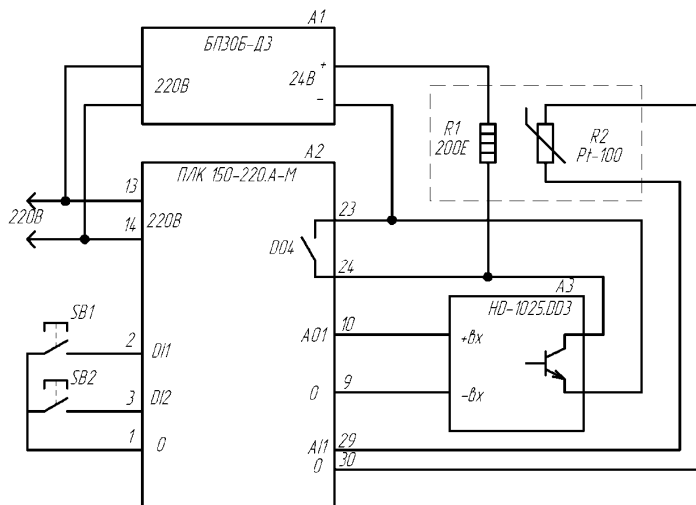


Рис. 3.1. Схема лабораторной установки

На схеме присутствуют:

1. Блок питания 24 В (A1).
2. ПЛК150 (A2).
3. Твердотельное реле (A3).
4. Кнопки SB1 (зеленая), SB2 (красная).
5. Электронагревательный элемент (R1) и физически присоединенный к нему датчик температуры Pt-100 (R2).

Необходимо связать переменные в программе с кнопками SB1 и SB2, соединенными со входами DI1, DI2. Далее с помощью конструкций программы осуществить управление выходами DO1 и DO2 согласно варианту. Например, если значение, связанное с DO1 представляет собой логическую операцию OR, то первый операнд этой операции – переменная, связанная с DI1, а второй – переменная, связанная с DI2.

Изучите основы программирования на графических языках LD, FBD, SFC, CFC и текстовых ST и IL.

Программирование ПЛК на языке LD

Язык LD (Ladder Diagram), или РКС (релейно-контактные схемы), представляет собой графическую форму записи логических выражений в виде контактов и обмоток реле.

LD-язык предназначен для программирования промышленных контроллеров (ПЛК). Синтаксис языка удобен для замены логических схем, выполненных на релейной технике.

Слева и справа схема ограничена вертикальными линиями – шинами питания. Между ними расположены цепи, образованные контактами и обмотками реле, по аналогии с обычными электронными цепями. Слева любая цепь начинается набором контактов, которые посылают слева направо состояние ON или OFF, соответствующие логическим значениям TRUE (истина) или FALSE (ложь). Каждому контакту соответствует логическая переменная. Если переменная имеет значение TRUE, то состояние передается через контакт, иначе правое соединение получает значение «выключено» (OFF).

Контакты могут быть соединены параллельно, тогда соединение передает состояние OR (логическое «ИЛИ»). Если контакты соединены последовательно, то соединение передаёт AND (логическое «И»).

Контакт может быть инвертируемым. Такой контакт обозначается с помощью символа $|/|$ и передает состояние ON, если значение переменной – FALSE.

LD-язык позволяет:

- выполнять последовательное соединение контактов;
- выполнять параллельное соединение контактов;
- применять нормально разомкнутые или замкнутые контакты;
- использовать переключаемые контакты;
- записывать комментарии;
- включать Set/Reset-выходы;
- осуществлять переходы;
- включать в диаграмму функциональные блоки;
- управлять работой блоков по входам EN.

При написании программы в рабочей зоне вкладки POUs последовательно вводятся типы компонентов и их обозначения, как это представлено на рис. 3.2.

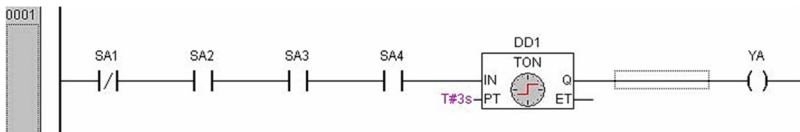


Рис. 3.2. Пример программы на языке LD


Пользователь при составлении виртуальной схемы может следовать приведенной ниже инструкции.

1. Создание нормально замкнутого контакта: в контекстном меню выбрать команду Contact (negated) или нажать кнопку $|/|$ на панели инструментов. Вопросительные знаки (рис. 3.3, а) необходимо заменить именем, например SA1.

Описывать переменную в данном случае не требуется, так как она уже была указана в окне PLC Configuration и связана с конкретным дискретным входом.



Рис. 3.3. Создание нормально замкнутого (а) и разомкнутого (б) контактов

2. Создание нормально разомкнутого контакта осуществляется аналогичным образом, только используется команда контекстного меню Contact или кнопка  на панели инструментов (рис. 3.3, б).

3. Функциональный блок: из контекстного меню выбирается команда Function Block..., в появившемся окне Input Assistant (рис. 3.4) из раздела Standard Function Blocks в библиотеке с именем STANDARD.LIB в папке Timer выбирается вид таймера – TON (FB).

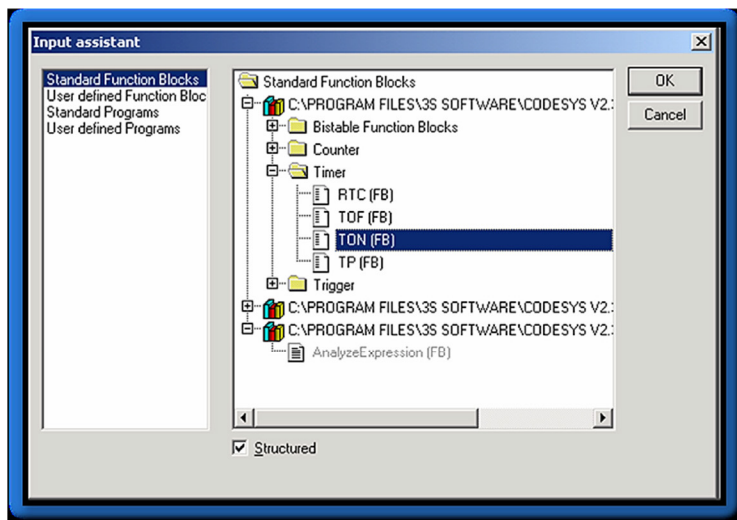


Рис. 3.4. Выбор таймера

На схеме перед входом РТ указывается время задержки в формате Т#3с. Над блоком вводится имя, например DD1, и на клавиатуре нажимается клавиша «стрелка вправо», подтверждаются свойства функционального блока.

Программирование ПЛК на языке ST

Язык ST (Structured Text) – это язык высокого уровня. Синтаксически ST представляет собой адаптированный язык Паскаль. Вместо процедур Паскаля в ST используются компоненты программ стандарта МЭК. На основе ST можно создавать гибкие процедуры обработки данных.

Основой ST-программы служат выражения. Результат вычисления выражения присваивается переменной при помощи оператора «:=», как и в Паскале. Каждое выражение обязательно заканчивается точкой с запятой «;». Выражение состоит из переменных констант и функций, разделенных операторами, например:

```
iVar1 := 1 + iVar2 / ABS(iVar2);
```

Стандартные операторы в выражениях ST имеют символическое представление, например, математические действия: +, -, *, /, операции сравнения и т. д.

Помимо операторов, элементы выражения можно отделять пробелами и табуляциями для лучшего восприятия.

Таблица 3.2

Операторы в ST

Операция	Выражение	Приоритет
Выражение в скобках	(Выражение)	Самый высокий
Вызов функции	Имя функции (список параметров)	
Возведение в степень	EXPT	
Замена знаков	-	
Числовое дополнение	NOT	
Умножение	*	
Деление	/	
Абсолютная величина	MOD	
Сложение	+	
Вычитание	-	
Сравнение	<, >, <=, >=	
Неравенство	<, >	
Равенство	=	
Логическое «И»	AND	
Логическое исключающее «ИЛИ»	XOR	
Логическое «ИЛИ»	OR	Самый низкий

Основные типовые конструкции в ST

Перед оператором присваивания находится операнд (переменная или адрес), которому присваивается значение выражения, стоящего после оператора присваивания. Пример:

```
Var1 := Var2 * 10;
```

После выполнения этой операции Var1 принимает значение в десять раз большее, чем Var2.

Функциональный блок вызывается с помощью имени экземпляра функционального блока и списка входных параметров с присваиванием данных в круглых скобках. В следующем примере вызывается таймер с параметрами IN и PT. Значение выходной переменной Q присваивается переменной A. К выходной переменной можно обратиться с помощью имени экземпляра функционального блока, точки, следующей за ним, и имени выходной переменной:

```
CMD_TMR (IN := %IX5, PT := 300);  
A := CMD TMR.Q
```

Оператор выбора позволяет выполнить различные группы выражений в зависимости от условий, обозначенных логическими выражениями. Полный синтаксис оператора IF (если) выглядит следующим образом:

```
IF <логическое выражение IF>  
THEN  
<выражения IF>  
[  
ELSIF <логическое выражение ELSEIF 1>  
THEN  
<выражения ELSEIF 1> ;  
...  
ELSIF <логическое выражение ELSEIF n>  
THEN  
<выражения ELSEIF n> ;  
ELSE  
<выражения ELSE> ;  
]  
END_IF
```

Если <логическое выражение IF> – TRUE, то выполняются выражения первой группы – <выражения IF>. Прочие выражения пропускаются, альтернативные условия не проверяются. Часть конструкции в квадратных скобках является необязательной и может отсутствовать.

Если <логическое выражение IF> – FALSE, то одно за другим проверяются условия ELSIF. Первое истинное условие приведет к выполнению соответствующей группы выражений. Прочие условия ELSIF анализироваться не будут.

Групп ELSIF может быть несколько или не быть совсем. Если все логические выражения дали ложный результат, то выполняются выражения группы ELSE при ее наличии. Если группы ELSE нет, то ничто не выполняется. В простейшем случае оператор IF содержит только одно условие:

```
IF bReset THEN
  iVar1 := 1;
  iVar2 := 0;
END_IF
```

Оператор множественного выбора CASE позволяет выполнить различные группы выражений в зависимости от значения одной целочисленной переменной или выражения.

Синтаксис:

```
CASE <целочисленное выражение> OF
  <значение 1> :
  <выражения 1> ;
  <значение 2> , значение 3> :
  <выражения 3> ;
  <значение 4>..значение 5> :
  <выражения 4> ; ...
[
  ELSE
  <выражения ELSE>;
]
END_CASE
```

Если значение выражения совпадает с заданной константой, то выполняется соответствующая группа выражений. Прочие условия не анализируются (<значение 1>: <выражения 1>;). Если

несколько значений констант должны соответствовать одной группе выражений, их можно перечислить через запятую (<значение 2>, <значение3>: <выражения 3>;). Диапазон значений можно определить через две точки (<значение 4>..<>значение 5> : <выражения 4>;). Группа выражений ELSE является необязательной. Она выполняется при несовпадении ни одного из условий (<выражения ELSE> ;).

Пример:

```
CASE byLeft/2 OF
0,127:
bReset := TRUE;
Var1 :=0;
16..24:
Var1 :- 1;
ELSE
Var1 := 2;
END_CASE
```

Значениями выбора CASE могут быть только целые константы, переменные использовать нельзя. Одинаковые значения в альтернативах выбора задавать также нельзя, даже в диапазонах.

С помощью *цикла FOR* можно программировать повторяющиеся процессы.

Синтаксис:

```
FOR <Целый счетчик> := <Начальное значение>
TO <Конечное значение>
[BY <Шаг>] DO
<Выражения – тело цикла>
```

Перед выполнением цикла счетчик получает начальное значение. Далее тело цикла повторяется, пока значение счетчика не превысит конечного значения. Счетчик увеличивается в каждом цикле. Начальное и конечное значения и шаг могут быть как константами, так и выражениями. Счетчик изменяется после выполнения тела цикла. Поэтому если задать конечное значение, меньшее в сравнении с начальным, то при положительном приращении цикл не будет выполнен ни разу. При одинаковых начальном и конечном значениях тело цикла будет выполнено один раз.

Часть конструкции **BY** в скобках необязательна, она определяет шаг приращения счетчика. По умолчанию счетчик увеличивается на единицу в каждой итерации. В качестве счетчика можно использовать переменную любого целого типа.

Пример:

```
Var1 := 0;  
FOR cw := 1 TO 10 DO  
  Var1 := Var1 + 1;  
END_FOR
```

Данный цикл будет выполнен 10 раз, и, соответственно, *Var1* будет иметь значение 10.

Шаг изменения счетчика итераций может быть и отрицательным. Начальное условие в этом случае должно быть больше конечного. Цикл будет закончен, когда значение счетчика станет меньше конечного значения.

Цикл **FOR** очень удобен для итераций с заранее известным числом повторов.

Для построения правильного цикла достаточно соблюдать два простых формальных требования:

- не изменять счетчик цикла и условие окончания в теле цикла. Счетчик и переменные образующие конечное условие в цикле, можно использовать только для чтения;
- не задавать в качестве конечного условия максимальное для типа переменной счетчика значение. Так, если для однобайтного целого без знака задать константу 255, то условие окончания не будет выполнено никогда. Цикл станет бесконечным.

Циклы *WHILE* и *REPEAT* обеспечивают повторение группы выражений, пока верно условное логическое выражение. Если условное выражение всегда истинно, то цикл становится бесконечным.

Синтаксис **WHILE**:

```
WHILE <Условное логическое выражение> DO  
<Выражения – тело цикла>  
END_WHILE
```

Условие в цикле **WHILE** проверяется до начала цикла. Если логическое выражение изначально имеет значение **FALSE**, тело цикла не будет выполнено ни разу.

Пример:

```
ci := 64;
WHILE ci > 1 DO
Var1 := Var1 + 1;
ci := ci/2;
END_WHILE
```

Синтаксис REPEAT:

```
REPEAT
<Выражения – тело цикла >
UNTIL <Условное логическое выражение> 17
END_REPEAT
```

Правильно построенный цикл WHILE или REPEAT обязательно должен изменять переменные, составляющие условие окончания в теле цикла, постепенно приближаясь к условию завершения. Если этого не обеспечить, цикл не закончится никогда.

При написании программы в рабочей зоне вкладки POU вводится текст, состоящий из имен переменных и операторов (табл. 3.2). При необходимости используются приведенные выше конструкции IF, FOR, CASE, WHILE и REPEAT.

В тексте программы необходимо учесть технологические требования, предъявляемые к системе управления.

Программирование ПЛК на языке IL

Язык IL (Instruction list, дословный перевод – «список инструкций») – это типичный ассемблер с аккумулятором и переходами по меткам. Набор инструкций стандартизован и не зависит от конкретной целевой платформы. Поскольку IL – самый простой в реализации язык, он получил очень широкое распространение.

Наибольшее влияние на формирование современного IL оказал язык программирования STEP контроллеров фирмы Siemens. Язык IL позволяет работать с любыми типами данных, вызывать функции и функциональные блоки, реализованные на любом языке. Таким образом, на IL можно реализовать алгоритм любой сложности, хотя текст будет достаточно громоздким.

В составе МЭК-языков IL применяется при создании компактных компонентов, требующих тщательной проработки, на которую

не жалко времени. При работе с IL гораздо адекватнее, чем с другими языками, можно представить, как будет выглядеть оттранслированный код, благодаря чему IL выигрывает там, где нужно достичь наивысшей эффективности. К компиляторам это относится в полной мере. В системах исполнения с интерпретатором промежуточного кода выигрыш не столь значителен.

Текст на IL – это текстовый список последовательных инструкций. Каждая инструкция записывается на отдельной строке. Инструкция может включать 4 поля, разделенные пробелами или знаками табуляции:

Метка: *Оператор* *Операнд* *Комментарий*

Метка инструкции не является обязательной, она ставится только там, где нужно. Оператор присутствует обязательно. Операнд необходим почти всегда. Комментарий – необязательное поле, он записывается в конце строки. Ставить комментарии между полями инструкции нельзя.

Пример IL-программы:

```

МЕТКА1:            LD                    Sync                (*пример IL*)
                    AND                    Start
                    S                        Q

```

Для лучшего восприятия строки IL выравнивают обычно в колонки по полям. Редактор CoDeSys выравнивает текст автоматически. Помимо этого, редактор «налету» выполняет синтаксический контроль и выделение цветом. Так, корректно введенные операторы выделяются голубым цветом.

Абсолютное большинство инструкций IL выполняют некоторую операцию с содержимым аккумулятора. Операнд, конечно, тоже принимает участие в инструкции, но результат опять помещается в аккумулятор. Например, инструкция SUB 10 отнимает число 10 от значения аккумулятора и помещает результат в аккумулятор. Команды сравнения сравнивают значения операнда и аккумулятора, результат сравнения TRUE или FALSE вновь помещается в аккумулятор. Команды перехода на метку способны анализировать аккумулятор и принимать решение – выполнять переход или нет.

Аккумулятор IL является универсальным контейнером, способным сохранять значения переменных любого типа.

В аккумулятор можно поместить значение типа BOOL, затем INT или REAL, притом транслятор не будет считать это ошибкой. Впрочем, такая гибкость не означает, что аккумулятор способен одновременно содержать несколько значений разных типов: может содержаться только одно, причем тип значения также фиксируется в аккумуляторе. Если операция требует значение другого типа, транслятор выдаст ошибку.

Следует отметить, что в стандарте МЭК вместо термина «аккумулятор» используется термин «результат» (result). Так, инструкция берет «текущий результат» и формирует «новый результат». Тем не менее почти все руководства по программированию различных фирм широко используют термин «аккумулятор».

Программа на языке PL выполняется последовательно сверху вниз. Для изменения порядка выполнения и организации циклов применяется переход на метку. Переход на метку может быть безусловным – JMP. Такой переход выполняется всегда, независимо от чего-либо. Условный переход – JMPC – выполняется только при значении аккумулятора TRUE.

Переход можно выполнять как вверх, так и вниз. Метки являются локальными, другими словами, переход на метку в другом POU не допускается.

Переходы нужно организовывать достаточно аккуратно, чтобы не получить бесконечный цикл.

Последовательный порядок выполнения команд PL можно изменять при помощи скобок. Открывающая скобка ставится в инструкции после операции. Закрывающая скобка ставится в отдельной строке. Инструкции, заключенные в скобки, выполняются в первую очередь. Результат вычисления инструкций в скобках помещается в дополнительный аккумулятор, после чего выполняется команда, содержащая открывающую скобку.

Пример:

LD	1	
ST	Counter	
LD	5	
MUL	(2	
SUB	1	
)		
ST	y	(*y = 5 * (2 - 1) = 5*)

LD	5	
MUL	2	
SUB	1	
ST	y	(*y = 5 * 2 - 1 = 9*)

Скобки могут быть вложенными. Каждое вложение требует организации некоего временного аккумулятора. Это вызывает неоднозначность при выходе из блока скобок командами JMP, RET, CAL и LD. Применять эти команды в скобках нельзя.

Добавление к мнемонике некоторых операторов символов-модификаторов C и N модифицирует смысл инструкции.

Символ N (negation) вызывает диверсию значения операнда до выполнения инструкции. Операнд должен относиться к типам BOOL, BYTE, WORD или DWORD.

Символ C (condition) добавляет проверку условий к командам перехода, вызова и возврата. Команды JMPC, CALC, RETC будут выполняться только при значении аккумулятора TRUE. Добавление символа N приводит к сравнению условия с инверсным значением аккумулятора. Команды JMPCN, CALCN, RETCN будут выполняться только при значении аккумулятора FALSE. Модификатор N без C не имеет смысла в данных операциях и не применяется.

Стандартные операторы IL с допустимыми модификаторами представлены в табл. 3.3.

Таблица 3.3

Операторы IL

Оператор	Модификатор	Описание
LD	N	Загрузить значение операнда в аккумулятор
ST	N	Присвоить значение аккумулятора операнду
S		Если аккумулятор имеет значение TRUE, установить логический операнд (TRUE)
R		Если аккумулятор имеет значение TRUE, сбросить логический операнд (FALSE)
AND	N, (Поразрядное «И»
OR	N, (Поразрядное «ИЛИ»
XOR	N, (Исключающее «ИЛИ»

Окончание табл. 3.3

Оператор	Модификатор	Описание
NOT		Отрицание «НЕ»
ADD	(Сложение
SUB	(Вычитание
MUL	(Умножение
DIV	(Деление
MOD	(Деление по модулю
GT	(>
GE	(=>
QE	(=
NE	(< >
LE	(<=
LT	(<
JMP	CN	Переход к метке
CAL	CN	Вызов функционального блока
RET	CN	Выход из POU и возврат в вызывающую программу

Операторы S и R применяются только с операндами типа BOOL. Прочие операторы работают с любыми переменными базовых типов.

Приведенный список содержит операторы, поддерживаемые в обязательном порядке. Трансляторы кода CoDeSys для различных аппаратных платформ реализуют различные подмножества дополнительных операторов.

Программирование ПЛК на языке FBD

FBD – это графический язык программирования. Он работает с последовательностью цепей, каждая из которых содержит логическое или арифметическое выражение, вызов функционального блока, переход или инструкцию возврата.

Диаграмма FBD строится из компонентов, отображаемых на схеме прямоугольниками (рис. 3.5). Входы ROU изображаются слева от прямоугольника, выходы справа. Внутри прямоугольника указывается тип ROU и наименования входов и выходов. Для экземпляра функционального блока его наименование указывается сверху, над прямоугольником. В графических системах программирования прямоугольник компонента может содержать картинку, отражающую его тип. Размер прямоугольника зависит от числа входов и выходов и устанавливается графическим редактором автоматически.

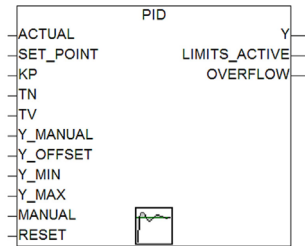


Рис. 3.5. Пример графического представления экземпляра функционального блока PID

Программа в FBD не обязательно должна представлять большую единую схему. Как и в LD, диаграмма образуется из множества цепей, которые выполняются одна за другой.

В CoDeSys все цепи одного ROU отображаются в едином графическом окне, они пронумерованы и разделены горизонтальными линиями (рис. 3.6). Значения переменных, вычисленные в одной цепи, доступны в последующих цепях сразу в том же рабочем цикле.

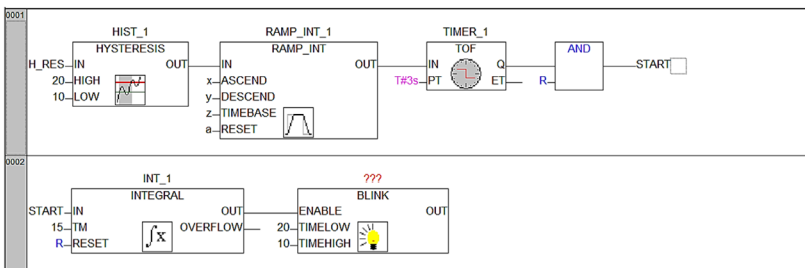


Рис. 3.6. Диаграмма FBD из двух цепей

Прямоугольники POU в FBD соединены линиями связи. Соединения имеют направленность слева направо. Вход блока может быть соединен с выходом блока, расположенного слева от него. Помимо этого, вход может быть соединен с переменной или константой. Соединение должно связывать переменные или входы и выходы одного типа. В отличие от компонента переменная изображается на диаграмме без прямоугольной рамки. Ширина соединительной линии в FBD роли не играет. Стандарт допускает использование соединительных линий разной ширины и стиля для соединений разного типа.

Выполнение FBD-цепей осуществляется слева направо, сверху вниз. Блоки, расположенные левее, выполняются раньше. Блок начинает вычисляться только после вычисления значений всех его входов. Дальнейшие вычисления не будут продолжены, пока не будут вычислены значения на всех выходах. Другими словами, значения на всех выходах графического блока появляются одновременно. Вычисление цепи считается законченным только после вычисления значений на выходах всех входящих в нее элементов.

В некоторых системах программирования пользователь имеет возможность свободно передвигать блоки с сохранением связей. В этом случае ориентироваться нужно исходя из порядка соединений. Редактор FBD CoDeSys автоматически расставляет блоки в порядке выполнения.

Инверсия логического сигнала в PBD изображается в виде окружности на соединении, перед входом или переменной. Инверсия не является свойством самого блока и может быть легко добавлена или отменена непосредственно в диаграмме.

Порядок выполнения FBD-цепей диаграммы можно принудительно изменять, используя метки и переходы, точно так же, как и в релейных схемах. Метка ставится в начале любой цепи, являясь, по сути, названием данной цепи.

Цепь может содержать только одну метку.

Имена меток подчинены общим правилам наименования идентификаторов МЭК.

Графический редактор автоматически нумерует цепи диаграммы. Эта нумерация применяется исключительно для документирования и не может заменять метки.

Переход обязательно связан с логической переменной и выполняется, если переменная имеет значение TRUE. Для создания безусловного перехода используется константа TRUE, связанная с переходом.

Оператор возврата RETURN можно использовать в FBD так же, как и переход на метку, т. е. в связке с логической переменной. Возврат приводит к немедленному окончанию работы программного компонента и возврату на верхний уровень вложений. Для основной программы это начало рабочего цикла ПЛК.

Программирование ПЛК на языке SFC

SFC – это графический язык, который позволяет описать хронологическую последовательность различных действий в программе. Для этого действия связываются с шагами (этапами), а последовательность работы определяется условиями переходов между шагами.

Любая SFC-схема составляется из элементов, представляющих шаги и условия переходов (рис. 3.7).

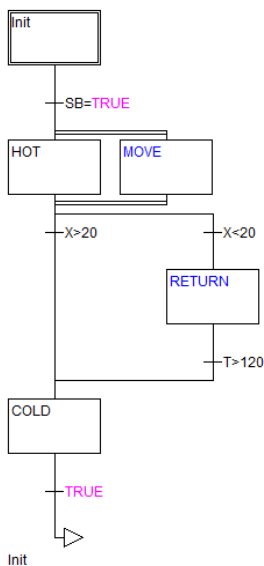


Рис. 3.7. Пример SFC-диаграммы

Шаги показаны на схеме прямоугольниками. Реальная работа шага (действия) описывается в отдельном окне системы программирования и не отражается на диаграмме. О назначении шага SFC говорит только его название или, если этого недостаточно, краткое текстовое описание (комментарий).

Шаги на схеме могут быть пустыми, что не вызывает ошибки при компиляции проекта. Пустые шаги являются нормой при применении программирования сверху вниз, характерного для SFC. Определить действия, соответствующие шагу, можно в любое время. Нет ничего удивительного, если пустые шаги останутся и в законченном проекте. Задачей пустого шага является ожидание перехода.

Ниже шага на соединительной линии присутствует горизонтальная черта, обозначающая переход. Условием перехода может служить логическая переменная, логическое выражение, константа или прямой адрес.

Переход выполняется при соблюдении двух условий:

- 1) он разрешен (соответствующий ему шаг активен);
- 2) условие перехода имеет значение TRUE.

Простые условия отображаются непосредственно на диаграмме справа от черты, обозначающей переход. В CoDeSys на диаграмме можно записывать только выражения на языке ST. Для громоздких условий применяется другой подход. Вместо условия на диаграмме записывается только идентификатор перехода. Само же условие описывается в отдельном окне с применением языка IL, ST, LD или FBD. Переменные или прямые адреса используются в условии перехода только для чтения. В условном выражении перехода нельзя:

- вызывать экземпляры функциональных выражений перехода;
- вызывать экземпляры функциональных блоков;
- использовать операцию присваивания.

Признаком того, что идентификатор перехода на диаграмме является отдельно реализованным условием, а не простой логической переменной, служит закрашенный угол перехода. В качестве условия перехода может быть задана логическая константа. Если задано TRUE, то шаг будет выполнен однократно за один рабочий цикл, далее управление перейдет к следующему шагу. Если задано условие FALSE, то шаг будет выполняться бесконечно.

Каждая SFC-схема начинается с шага, выделенного графически двойными вертикальными линиями или по всему периметру. Это начальный шаг. Наименование начального шага может быть произвольным (по умолчанию Init). Начальный шаг присутствует обязательно, хотя и может быть пустым.

Несколько ветвей SFC могут быть параллельными. Признаком параллельных ветвей на схеме является двойная горизонтальная линия. Каждая параллельная ветвь начинается и заканчивается шагом. Таким образом, условие входа в параллельность всегда одно, условие выхода тоже одно на всех.

Теоретически параллельные ветви выполняются одновременно, практически – в одном рабочем цикле, слева направо. Условие перехода, завершающее параллельность, проверяется только в случае, если в каждой параллельной ветви активны последние шаги.

Несколько ветвей SFC могут быть альтернативными ветвями. Признаком таких ветвей на схеме является одинарная горизонтальная линия. Каждая альтернативная ветвь начинается и заканчивается собственным условием перехода. Проверка альтернативных условий выполняется слева направо. Если верное условие найдено, то прочие альтернативы не рассматриваются. В альтернативных ветвях всегда работает только одна из ветвей, поэтому ее окончание и будет означать переход к следующему за альтернативной группой шагу.

При создании альтернативных ветвей желательно задавать взаимоисключающие условия. В этом случае вероятность допустить ошибку при анализе или в процессе доработки диаграммы значительно ниже.

В общем случае SFC-схема выполняется сверху вниз. Стандартом допускается создание переходов на произвольный шаг. Для этого применяются соединительные линии с промежуточными стрелками или поименованные переходы, то есть переход выполняется на шаг, имя которого указано под стрелкой. В англоязычных источниках переход на произвольный шаг называется «прыжок» (jump). Прыжок из одной ветви параллельного блока наружу вызывает эффект размножения маркера. Прыжок внутрь параллельного блока нарушает параллельность ветвей. Подобных «трюков» необходимо избегать.

Программирование ПЛК на языке CFC

В этом редакторе нет сетки, поэтому элементы могут располагаться где угодно. К элементам языка CFC относятся блоки, входы, выходы, возвраты, произвольные переходы, метки и комментарии.

Входы и выходы этих элементов можно соединять, перетаскивая линии соединения мышкой. Эти линии будут перерисовываться автоматически при перемещении элементов. В случае, если линия соединения не может быть перерисована, она выделяется красным. Соответственно, как только элемент будет переставлен так, чтобы можно было соединить вход и выход линией без пересечений с другими элементами, выделение красным исчезает.

Данный редактор похож на редактор FBD. Основное преимущество CFC-редактора перед FBD заключается в том, что в схемы можно непосредственно добавлять линии обратной связи.

Выводы

В ходе лабораторной работы обеспечено знакомство с основными языками программирования ПЛК, изучены схемы подключения к контроллеру периферийных устройств. Написана программа согласно представленному варианту, которая загружена в контроллер и работа которой отлажена.

Лабораторная работа 4

Релейный регулятор температуры

Цель работы – ознакомиться с работой релейного регулятора температуры.

Программа работы

1. Изучить подключение датчика и реле к контроллеру.
2. Ознакомиться с конфигурацией ПЛК для работы с датчиком температуры и реле.
3. Реализовать программу, осуществляющую двухпозиционное регулирование температуры.
4. Загрузить программу в ПЛК и запустить ее.

Используемое оборудование и материалы

1. Персональный компьютер с установленной операционной системой Windows.
2. Среда разработки CoDeSys.
3. ПЛК «ОВЕН» 150/160/63 с проводом питания.
4. Преобразователь USB-RS232.
5. Кабель прошивки KC1.
6. Кнопки.
7. Соединительные провода.
8. Блок питания 220–24 В БП30Б-Д3.
9. Твердотельное реле HD-1025.DD3.
10. Термистор.
11. Резисторы.

Варианты заданий

Необходимо реализовать программу, осуществляющую релейное регулирование температуры по заданным порогам. Коммутацию

нагрузки R1 осуществлять посредством выходного реле DO4. Регулирование начинается по нажатию на кнопку SB1 («Старт») и прекращается при нажатии на кнопку SB2 («Стоп»). Варианты заданий приведены в табл. 4.

Таблица 4

Варианты заданий для лабораторной работы 4

Вариант	Нижний порог, °С	Верхний порог, °С
1	33	35
2	35	40
3	40	45
4	45	50

Указания по выполнению работы

Для программирования реле необходимо установить программное обеспечение CoDeSys V2.3. Рекомендуется устанавливать только минимально необходимые компоненты (оставив галочку напротив CoDeSys V2.3 и убрав напротив остальных компонентов, где это возможно).

При создании программы используется среда программирования CoDeSys V2.3 (далее – CoDeSys). Перед созданием проекта пользователь, используя утилиту Install Target в составе CoDeSys, устанавливает для применяемого контроллера файл целевых задач (target-файл), который обеспечивает программный доступ к ресурсам ПЛК. Необходимо скачать эти файлы с официального сайта «ОВЕН» и запустить файл InstallTarget.bat в папке PLC150.A-M.

Создание проекта программы

CoDeSys запускается последовательным выбором:

Пуск → Все программы → 3S Software CoDeSys → CoDeSys V2.3.

Новый проект открывается из главного меню (*File → New*). В открывшемся окне (рис. 4.1) выбирается тип контроллера PLC150.A-M, выбор подтверждается нажатием кнопки ОК.

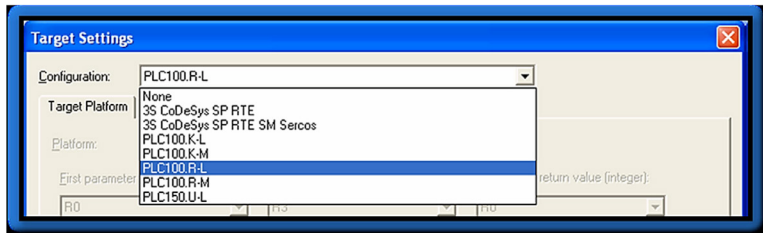


Рис. 4.1. Окно конфигурации Target Settings программы

После выбора проекта выводится экранная форма, задающая тип, имя и язык программирования первичного компонента *New POU*, главной программы контроллера.

В CoDeSys предусмотрено 6 языков программирования, которые будут рассмотрены в последующих пунктах. Необходимо выбрать язык программирования (например, LD), установив соответствующие флажки в позициях, указанных на рис. 4.2.

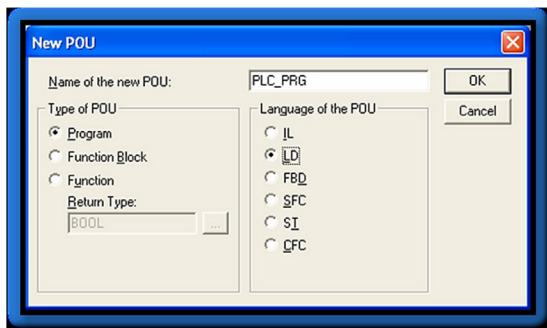


Рис. 4.2. Вид окна New POU с отмеченными параметрами

Примечание. Имя главной программы PLC_PRG и ее тип менять нельзя.

После подтверждения выбора нажатием кнопки ОК откроется окно нового проекта с именем по умолчанию Untitled. В нем присутствует одна вкладка *POUs*.

Для каждого проекта рекомендуется создавать отдельную папку, поскольку кроме самого файла проекта (*.pro) в нее сохраняются дополнительные (вспомогательные) файлы.

Параметры входов и выходов контроллера

Цепям контроллера, используемым в разрабатываемой электрической схеме, присваиваются имена переменных. В дальнейшем эти имена используются в программе для работы с конкретным входом или выходом контроллера.

Для присвоения имени какому-либо ресурсу ввода/вывода контроллера необходимо на вкладке ресурсов (*Resources*) *Организатора объектов* CoDeSys запустить утилиту *PLC Configuration* (*Конфигурация ПЛК*).

В появляющейся иерархической структуре – дереве *Конфигурации ПЛК* – пользователь открывает папки (модули) входов (*Discrete input*) и выходов (*Discrete output*) ПЛК и именуется необходимые каналы. Перед адресом указывается имя (идентификатор переменной) для цепей входов и выходов схемы созданного проекта.

Именованье канала (входа или выхода) производится следующим образом: двойным щелчком мыши при курсоре, установленном в начале строки названия канала, осуществляется переход в режим редактирования и вводится имя переменной канала.

В указанном на рис. 4.3 примере цифровые входы обозначены как DI1–6, цифровые выходы DO1–4, аналоговые выходы AO1–2.

Датчик температуры Pt100, представляющий собой сопротивление с положительным коэффициентом сопротивления, подключен к третьему аналоговому входу (рис. 4.3). Для его настройки нужно выбрать правой кнопкой мыши на соответствующем входе *Replace element* → *RTD sensor* и в открывшемся меню выбрать градуировку термосопротивления. Для термосопротивления Pt100 необходимо выбрать тип сенсора r385_100.

В случае использования датчика температуры ДРТС014-100_Ом.50/2 необходимо замерять его сопротивление с последующим пересчетом в температуру. Для задания режима замера сопротивления выбирается тип входа Unifid signal sensor [slot], тип сенсора – R0_5000 (будет замеряться сопротивление в Ом). Для автоматического пересчета в температуру указывается значение температуры, соответствующее сопротивлению 0 Ом ($A_{in\ low} = -86$) и сопротивлению 5000 Ом ($A_{in\ high} = 470$). Данные значения получены пересчетом результатов экспериментально тестирования датчика (выявлено соответствие сопротивлений температурам: 1000 Ом – 25 °С, 1450 Ом – 75 °С).

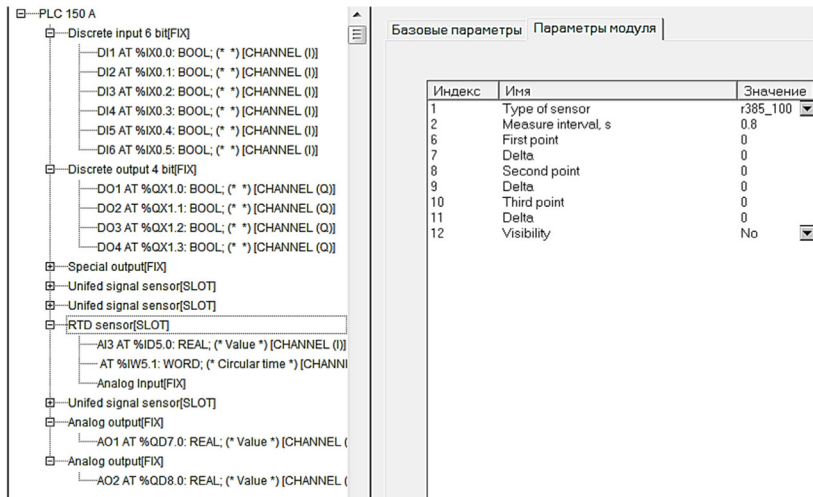


Рис. 4.3. Экранная форма для именованния входов и выходов при работе программы с цепями ПЛК

После написания программы необходимо соединить ПЛК с ПК, записать программу на ПЛК и запустить ее выполнение. Настройка соединения ПК с ПЛК «ОВЕН» для загрузки и проверки работы программы в автономном режиме производится следующим образом.

Для информационного обмена ПК с ПЛК «ОВЕН» используется кабель прошивки (КС1). Им соединяются СОМ-порт компьютера и порт Debug RS-232 контроллера (на лицевой панели). Соединение с ПК может осуществляться через адаптер USB-RS232. Если не удастся установить соединение при подключении адаптера к ПК спереди, рекомендуется попробовать подключать сзади без удлинителя.

Для настройки канала соединения из основного меню CoDeSys выбирается команда *Online* → *Communication parameters* (Онлайн – Параметры связи). В диалоговом меню командой *New...* открывается диалоговое окно, в котором соединению присваивается имя (например, СОМ) и выбирается (из перечня) вид соединения *Serial* (RS-232). Выбор подтверждается нажатием кнопки *OK*.

После указанных действий в окне коммуникационных параметров появляется канал соединения с именем COM. В зоне настроек (*Value*) для параметра *Baudrate* устанавливается значение 115 200 (бит/с – скорость соединения с компьютером). Значение может быть изменено двойным щелчком левой кнопкой мыши по значению. Сохранение установок подтверждается нажатием кнопки *OK*.

Программное соединение с ПЛК «ОВЕН» включается из главного меню CoDeSys командой *Online → Login*. При этом флажок перед строкой меню *Simulation Mode* должен быть снят.

Как только система устанавливает связь с ПЛК «ОВЕН», появляется запрос на подтверждение загрузки новой программы, пользователь подтверждает загрузку: «Да».

После завершения записи проекта в оперативную память ПЛК «ОВЕН» запуск работы программы осуществляется выбором команды *Online → Run* (или нажатием на лицевой панели ПЛК «ОВЕН» кнопки «Старт»). ПЛК начинает выполнение программы, и на мониторе ПК в режиме реального времени отображаются состояния всех элементов программы. При необходимости можно изменять значения переменных, загрузка изменений в ПЛК производится командой *Online → Write values*. Остановка выполнения программы: *Online → Stop*. Для дальнейшего изменения программы необходимо отключение станда от CoDeSys (*Online → Logout*). Отметим, что при остановке программы состояния выходов ПЛК перестают изменяться (остаются в своем последнем состоянии). Иначе говоря, если в этот момент нагревательный элемент (R1 на рис. 3.1) был включен – он останется во включенном состоянии. Для выключения всех выходов необходимо выполнение команды сброса: *Online → Reset*.

Выводы

В ходе лабораторной работы обеспечено знакомство с подключением датчика температуры к контроллеру, алгоритмом работы релейного регулятора. Написана программа, реализующая нагрев детали до заданной пользователем температуры, и отлажена ее работа.

Лабораторная работа 5

Пропорциональный регулятор температуры

Цель работы – ознакомиться с работой ПИД-регулятора температуры и формированием ШИМ управления нагревателем.

Программа работы

1. Модифицировать программу из лабораторной работы 4, изменив способ управления на пропорциональный регулятор.
2. Подобрать пропорциональный коэффициент регулирования.
3. Загрузить программу в ПЛК и запустить ее.

Используемое оборудование и материалы

1. Персональный компьютер с установленной операционной системой Windows.
2. Среда разработки CoDeSys.
3. ПЛК «ОВЕН» 150/160/63 с проводом питания.
4. Преобразователь USB-RS232.
5. Кабель прошивки KC1.
6. Кнопки.
7. Соединительные провода.
8. Блок питания 220–24 В БП30Б-Д3.
9. Твердотельное реле HD-1025.DD3.
10. Термистор.
11. Резисторы.

Варианты заданий

Варианты заданий приведены в табл. 5.

Таблица 5

Варианты заданий

Вариант	Заданная температура, °С
1	35
2	40
3	45
4	50

Указания по выполнению работы

Необходимо реализовать программу, осуществляющую регулирование температуры согласно пропорциональному алгоритму. Подобрать пропорциональный коэффициент регулирования для максимально быстрого установления заданного значения при условии отсутствия перерегулирования (превышения температуры над заданной). Коммутацию нагрузки R1 осуществлять посредством твердотельного реле АЗ (см. рис. 3.1).

Регулирование начинается нажатием на кнопку SB1 («Старт») и прекращается нажатием на кнопку SB2 («Стоп»).

Выводы

В ходе лабораторной работы обеспечено знакомство с пропорциональным регулятором для управления нагревом детали. Написана программа, отлажена ее работа, подобран коэффициент регулятора.

Лабораторная работа 6

ПИД-регулятор температуры

Цель работы – ознакомиться с работой ПИД-регулятора температуры и формированием ШИМ управления нагревателем.

Программа работы

1. Модифицировать программу из лабораторной работы 5, добавив интегральное и дифференциальное звенья в регулятор температуры.
2. Подобрать коэффициенты регулирования.
3. Загрузить программу в ПЛК и запустить ее.

Используемое оборудование и материалы

1. Персональный компьютер с установленной операционной системой Windows.
2. Среда разработки CoDeSys.
3. ПЛК «ОВЕН» 150/160/63 с проводом питания.
4. Преобразователь USB-RS232.
5. Кабель прошивки KC1.
6. Кнопки.
7. Соединительные провода.
8. Блок питания 220–24 В БП30Б-Д3.
9. Твердотельное реле HD-1025.DD3.
10. Термистор.
11. Резисторы.

Варианты заданий

Варианты заданий приведены в табл. 6.

Таблица 6

Варианты заданий

Вариант	Заданная температура, °С
1	35
2	40
3	45
4	50

Указания по выполнению работы

Необходимо реализовать программу, осуществляющую ПИД-регулирование температуры. Подобрать коэффициенты ПИД-регулирования для максимально быстрого установления заданного значения при условии отсутствия перерегулирования (превышения температуры над заданной).

Регулирование начинается нажатием на кнопку SB1 («Старт») и прекращается нажатием на кнопку SB2 («Стоп»).

Выводы

В ходе лабораторной работы обеспечено знакомство с ПИД-регулятором для управления нагревом детали. Написана программа, отлажена ее работа, подобраны пропорциональный, интегральный и дифференциальный коэффициенты регулятора. Выполнено сравнение работы регуляторов различных типов для управления процессом нагрева.

Заключение

В рамках шести лабораторных работ мы познакомились с устройством программируемых логических контроллеров на примере контроллеров «ОВЕН». Изучили основные языки программирования, используемые для написания программ для ПЛК. Используя среду разработки CoDeSys, реализовали регуляторы различных типов для управления процессом нагрева деталей до заданной температуры и поддержания требуемой температуры с необходимой точностью.

Бибблиографический список

1. *Шишов, О. В.* Программируемые контроллеры в системах промышленной автоматизации : учебник / О. В. Шишов. – Москва : ИНФРА-М, 2019. – 365 с. – (Высшее образование – Бакалавриат). – URL: new.znaniium.com/catalog/product/1007990 (дата обращения: 09.12.2019). – Режим доступа: по подписке. – ISBN 978-5-16-103331-9.
2. Настройка и программирование цифровых систем управления с использованием контроллеров, панелей оператора и частотных преобразователей (теория и практика) : учеб. пособие / В. С. Кудряшов, А. В. Иванов, М. В. Алексеев [и др.] ; Воронежский государственный университет инженерных технологий. – Воронеж : ВГУИТ, 2020. – 215 с. – URL: www.iprbookshop.ru/106446.html (дата обращения: 03.06.2021). – Режим доступа: по подписке. – ISBN 978-5-00032-459-2.
3. *Петров, И. В.* Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / И. В. Петров ; под ред. В. П. Дьяконова. – Москва : СОЛОН-Пресс, 2016. – 254 с. – ISBN 978-5-91359-151-7.
4. *Иванов, А. А.* Модернизация промышленных предприятий на базе современных систем автоматизации и управления : учеб. пособие / А. А. Иванов. – Москва : Форум [и др.], 2020. – 383 с. – (Высшее образование – Бакалавриат). – URL: znaniium.com/catalog/document?id=359721 (дата обращения: 14.02.2020). – Режим доступа: по подписке. – ISBN 978-5-16-108665-0.
5. *Фурсенко, С. Н.* Автоматизация технологических процессов : учеб. пособие / С. Н. Фурсенко, Е. С. Якубовская, Е. С. Волкова. – Москва : ИНФРА-М, 2022. – 376 с. – (Высшее образование). – URL: znaniium.com/catalog/document?id=390468 (дата обращения: 02.12.2022). – Режим доступа: по подписке. – ISBN 978-5-16-102249-8.

Приложение А

Оформление отчета по лабораторным работам

Отчеты по лабораторным работам должны содержать следующую информацию:

- 1) титульный лист с указанием учебного заведения и кафедры, темы работы, номера группы, фамилии, и., о. студента, фамилии преподавателя, календарного года;
- 2) цель работы;
- 3) программу работы;
- 4) задание вашей бригады;
- 5) ход работы;
- 6) вывод по результатам лабораторной работы.

В ходе работы должны быть приведены:

- 1) алгоритм работы разрабатываемой программы в виде текстового описания или блок-схемы;
- 2) скриншоты конфигурации ПЛК с текстовым описанием изменяемых значений;
- 3) визуализация с текстовым описанием настраиваемых свойств объектов;
- 4) скриншоты или листинг программы (в зависимости от языка программирования);
- 5) скриншоты или фото, показывающие работу программы.