МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
АГРАРНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА СИСТЕМ АВТОМАТИЗАЦИИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано Учебно-методическим объединением высших учебных заведений Республики Беларусь по образованию в области автоматизации технологических процессов, производств и управления в качестве учебно-методического пособия для студентов учреждений высшего образования по специальности 1-53 01 01-09 Автоматизация технологических процессов и производств (сельское хозяйство)

Минск
БГАТУ
2017

УДК 681.51(07)
ББК 32.966я7
М59

Составители:

доктор технических наук, доцент *И. И. Гируцкий,*
кандидат технических наук, доцент *А. Г. Сеньков*

Рецензенты:
кафедра проектирования информационно-компьютерных систем Учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»,
заведующий кафедрой кандидат технических наук, доцент *И. Н. Цырельчук*;
заведующий лабораторией РУП «НПЦ НАН Беларуси по механизации сельского хозяйства» кандидат технических наук, доцент *В. В. Чумаков*

М59 **Микропроцессорная** техника систем автоматизации. Лабораторный практикум : учебно-методическое пособие / сост.: И. И. Гируцкий, А. Г. Сеньков. – Минск : БГАТУ, 2017. – 136 с.
ISBN 978-985-519-883-4.

Содержится учебно-методический материал для выполнения лабораторных и практических работ по дисциплине «Микропроцессорная техника систем автоматизации».

Предназначено для студентов, обучающихся по специальности 1-53 01 01-09 Автоматизация технологических процессов и производств (сельское хозяйство).

УДК 681.51(07)
ББК 32.966я7

ISBN 978-985-519-883-4

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное агропромышленное, и, особенно, сельскохозяйственное производство нуждаются в высокоэффективных системах управления технологическими процессами. Отсутствие надежных и функционально достаточных систем управления технологическими и производственными процессами в сельскохозяйственном производстве приводит к существенному невыполнению агротребований и, как следствие, значительному перерасходу кормов и энергии на единицу продукции. Эффективным инструментом решения этой проблемы должны стать автоматизация и компьютеризация.

Надежность, адаптивность (гибкость), возможность обеспечения согласованного управления разнородными технологическими объектами и построения распределенных многоуровневых систем управления – основные требования, предъявляемые к современным системам управления. Наиболее полно этим требованиям отвечают системы управления, построенные на базе микропроцессорной техники.

Удовлетворение потребности производства в современных технологиях управления может осуществляться двумя путями. Наиболее простым является использование достижений передовых западных технологий «под ключ». Однако при этом мы будем финансировать и развивать научный и интеллектуальный потенциал и без того развитых стран. Более сложным, но и более перспективным является развитие собственного научного потенциала.

Поэтому в настоящее время является актуальной подготовка специалистов в области микропроцессорных систем управления технологическими процессами. Учебная программа обучения направлена на выработку у студентов методологических основ построения современных микропроцессорных систем управления, таких как децентрализованность, многоуровневость, открытость, унифицированность программно-технических средств, экономическая и социальная эффективность.

Основой специализированных стендов, на использование которых сориентирован цикл лабораторных работ, является микропроцессорный контроллер общепромышленного применения типа Simens S7-1200. Выполнение лабораторных работ позволит студенту освоить не только программно-технические средства автоматизации технологических процессов, но и получить навыки разработки и отладки микропроцессорных систем управления реальными технологическими процессами агропромышленного комплекса.

ЛАБОРАТОРНОЕ ОБОРУДОВАНИЕ

Цикл лабораторных работ по курсу «Микропроцессорная техника систем автоматизации» проводится на базе специализированного класса, включающего 6 рабочих мест (рис. 1).

Рис. 1. Общий вид учебного класса

На стенде реализована конфигурация, включающая в себя ПЛК Simatic S7-1200, компьютер с установленной на нем средой программирования и конфигурирования аппаратных средств, коммутационное оборудование с использованием сетевых интерфейсов Profibus и Industrial Ethernet, блоки ввода/вывода дискретных и аналоговых электрических сигналов для имитации технологического процесса. Так же на стенде представлена сенсорная панель оператора KTP700 Basic Color PN – устройство ввода и отображения информации.

В качестве программатора используется ПЭВМ с лицензионной системой программирования TIA (Totally Integrated Automation) Portal (V13).

Программируемый логический контроллер Simatic S7-1200 (рис. 2) объединяет в компактном корпусе микропроцессор, встроенный блок питания, входные и выходные цепи. ПЛК серии S7-1200 имеют модульную конструкцию. Т.е. к центральному процессору (ЦПУ) ПЛК S7-1200 при необходимости увеличения числа обрабатываемых информационных сигналов могут быть подключены коммуникационные модули (CM), сигнальные модули (SM) и сигнальные платы (SB) ввода/вывода дискретных и аналоговых сигналов (табл. 1).

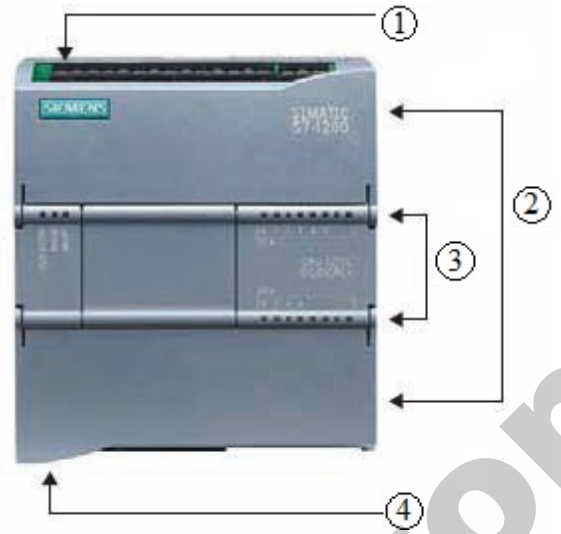


Рис. 2. Внешний вид ПЛК Simatic S7-1200: 1 – разъем питания; 2 – съемный клеммный блок для подключения пользователя (за дверцами); гнездо для карты памяти (под верхней дверцей); 3 – светодиоды состояния для встроенных входов/выходов; 4 – разъем PROFINET

Таблица 1

Технические характеристики ПЛК Simatic S7-1200

Напряжение питания	+24 В постоянного тока
Встроенная загружаемая память	2 МБ
Встроенная рабочая память	50 КБ
Энергонезависимая память для сохранения данных при перебоях в питании	2 КБ
Битовая память (М)	8192 байта
ПИД-регулирование	Поддержка 16 контуров
Скоростные счетчики	3×100 кГц + 3×30 кГц
Импульсные выходы	2×100 кГц
Встроенные аппаратные часы реального времени	Запас хода без подключения питания – 240 часов
Интерфейс Ethernet	1xRJ45, 10/100 Мбит/с

Окончание таблицы 1

Встроенные входы и выходы	2 аналоговых входа 0 .. 10 В/10 бит
	14 цифровых входов, 24 В
	10 цифровых выходов, 24 В/0.5А на основе транзисторных ключей
Количество каналов системы локального ввода/вывода:	
▪ каналов ввода/вывода дискретных сигналов, не более:	144/140
▪ каналов ввода/вывода аналоговых сигналов, не более:	34/17

Для того чтобы обеспечить поступление информации о состоянии объекта управления, необходимо подключить различные датчики параметров технологических процессов к входам контроллера. С точки зрения схем подключения, датчики можно разделить на датчики с унифицированным выходом и специализированные. Унифицированные выходы включают три типа: выход типа «ключ»; непрерывный, с информационным сигналом в виде тока, напряжения или частоты импульсов; сетевые, когда для передачи данных используются протоколы локальных вычислительных сетей. Если датчик не имеет унифицированного выхода, как, например, термометр сопротивления металлический ТСП, то необходимо наличие специализированных входных модулей в составе контроллера.

Широко распространены датчики с выходом типа «ключ», т. е. имеющие два состояния: технологический параметр достиг заданного значения или нет. В качестве примера можно привести датчики конечного положения с выходом в виде механического контакта или транзисторного ключа. Такие датчики называют бинарными или дискретными. Схемы подключения датчиков и исполнительных механизмов к программируемым контроллерам унифицированы. Дискретные входы (discreet input, DI), как правило, рассчитаны на напряжение 24 В и потребляют ток в диапазоне 4..10 mA (рис. 3).

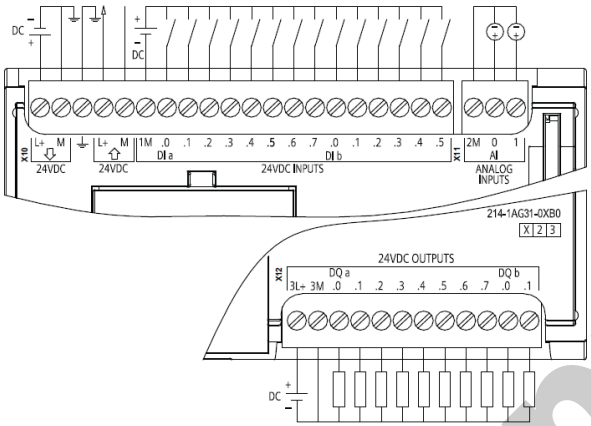


Рис. 3. Схема подключения внешних устройств к входам и выходам контроллера

При замыкании контакта напряжение 24 В поступает на дискретный вход контроллера, например DIa.0. В этом случае операционная система контроллера записывает в соответствующую однобитовую ячейку памяти входов состояние входа DIa.0 как логическую 1 (TRUE), при размыкании контакта и отсутствии напряжения на входе – записывается состояние логический 0 (FALSE). Такие типы сигналов являются широко распространенными. Причем, в качестве источника дискретного сигнала может выступать не только механический контакт, имеющий два состояния «замкнут/разомкнут», но и любой электронный прибор. Таким прибором наиболее часто является транзистор в ключевом режиме. Но обычные дискретные входы контроллеров рассчитаны на сигналы, длительность которых должна превышать время выполнения цикла прикладной программы. Цикл выполнения программы является внутренней характеристикой конкретного контроллера и может изменяться в широких диапазонах, например, 10..500 мс. Если необходимо воспринимать информацию с датчиков, имеющих высокочастотный импульсный выход, например, в диапазоне 0..2000 кГц, то используются специализированные входы контроллера, способные работать в режиме высокочастотного счетчика. Таким образом, подключение датчиков с дискретным сигналом является достаточно тривиальной задачей, не требующей глубоких знаний электроники.

Пользователь имеет доступ только к клеммникам контроллера, однако полезно владеть информацией о входном каскаде, который и осуществляет первичную обработку сигнала (рис. 4). Для повышения помехоустойчивости связь внешних электрических сигналов с внутренней схемой контроллера осуществляется через оптрон, выполняющий функцию гальванической развязки цепей питания внешних датчиков от цепей питания внутренних микросхем ПЛК. Наличие гальванической развязки препятствует проникновению внешних помех во внутренние электрические схемы контроллера и повышает надежность его функционирования.

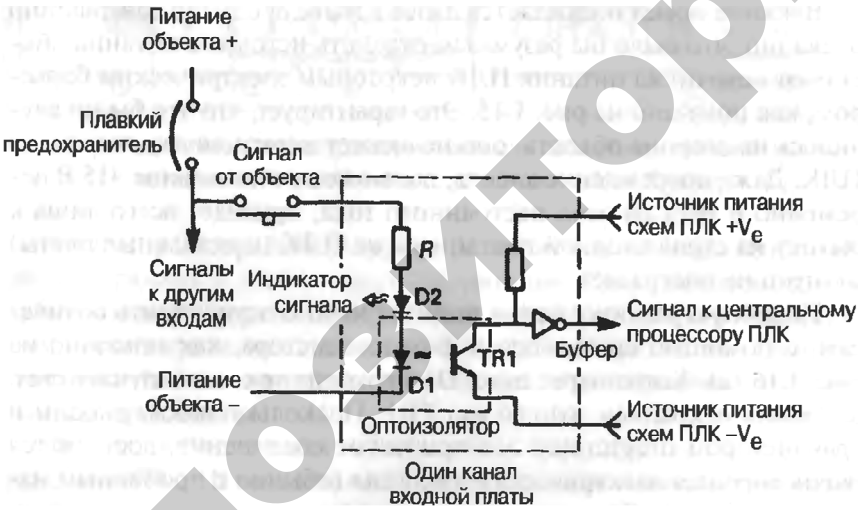


Рис. 4. Типовая схема первичного каскада дискретного входа

Вывод дискретных сигналов используется для управления состоянием включено/выключено исполнительных устройств. Устройства вывода отличаются большим многообразием. Знание структуры выходных каскадов необходимо для правильного их применения. Выходные платы также нуждаются в некотором изолирующем барьере, чтобы ограничить ущерб от возможных неисправностей на стороне объекта и исключить электрические помехи, нарушающие работу процессора (рис. 5).

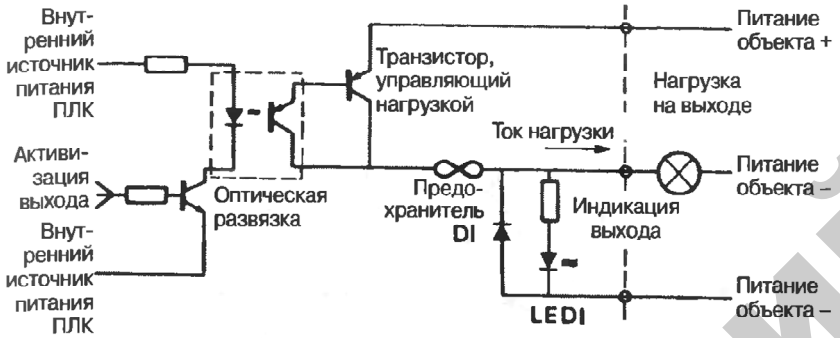


Рис. 5. Типовая схема каскада транзисторного выхода ПЛК

На лабораторном стенде имеются модули ввода/вывода цифровых и аналоговых сигналов, предназначенные для учебной имитации подачи на контроллер внешних входных и снятия с контроллера управляющих выходных сигналов (рис. 6).

Рис. 6. Модули ввода/вывода цифровых и аналоговых сигналов

Модуль Festo 8DIn (рис. 6, а) – источник входных цифровых сигналов – позволяет подавать на цифровые входы контроллера одновременно до 8 дискретных сигналов постоянного напряжения +24 В.

Модуль Festo 8DOut (рис. 6, б) – приемник выходных цифровых сигналов. Содержит 8 светодиодных индикаторов уровней выходных цифровых сигналов контроллера и позволяет передавать их на внешнюю нагрузку через соединительные разъемы.

Модуль Festo 4AIn/2AOut (рис. 6, в) – модуль ввода/вывода аналоговых сигналов. Модуль обеспечивает подачу на аналоговые входы контроллера до 4 аналоговых сигналов в диапазоне от 0 до 10 В постоянного напряжения. При переключении соответствующего тумблера в верхнее положение значение аналогового напряжения задается вращением ручки потенциометра. При переключении тумблера в нижнее положение аналоговое напряжение может быть подано от внешнего источника при подключении его через 4 мм электрический разъем. Модуль обеспечивает также отображение значений двух сигналов с аналоговых выходов контроллера в диапазоне от -10 до + 10 В постоянного напряжения с возможностью подключения нагрузки. Значение аналогового напряжения отображается на цифровом вольтметре.

В качестве выхода аналогового сигнала контроллера на его передней стороне имеется аналоговая сигнальная плата SB 1223 AQ 1x12bit. Схема подключения нагрузки к единственному аналоговому выходу показана на рис. 7.

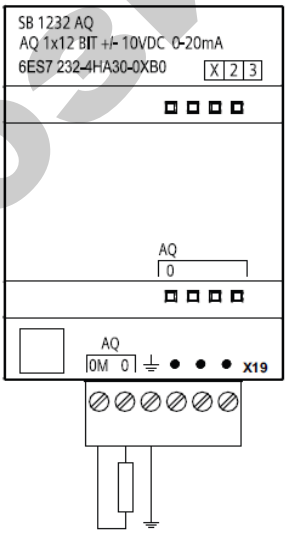


Рис. 7. Схема подключения нагрузки к аналоговому выходу ПЛК

Сенсорная панель оператора KTP700 Basic – предназначена для визуализации параметров процесса (объекта) и/или осуществления операторского управления. Сенсорная панель оператора позволяет реализовать следующие функции человеко-машинного интерфейса:

– отображение параметров технологического процесса (или объекта) в текстовом или графическом режимах;

– управление и обработка аварийных сообщений, регистрация времени и даты возникновения аварийных сообщений;

– ручное управление с помощью функциональных кнопок или сенсорного экрана;

– построение диаграмм и трендов, отображение сводных отчетов;

– возможность программирования, графики и настройки функциональных клавиш.

Аппаратная архитектура панели устроена по подобию обычных персональных компьютеров, только вместо жесткого диска используется Flash-память.

Типовая панель состоит из следующих аппаратных компонентов: процессор; оперативная память небольшого объема; встроенная Flash-EEPROM память (ПЗУ) для хранения ОС и накопления пользовательских данных; различные слоты расширения и интерфейсы для подключения программатора и/или сети передачи данных.

Сенсорная панель KTP700 Basic имеет следующие характеристики:

– напряжение питания – + 24 В постоянного тока;

– разрешение экрана – 800×480 точек;

– объем доступной для хранения пользовательских данных оперативной памяти – 10 Мб;

– 1 слот для подключения устройств USB объемом до 16 Гб;

– 1 слот для сетевого подключения по стандарту Industrial Ethernet;

– поддерживаемые протоколы передачи данных: PROFINET, TCP/IP, DHCP, SNMP, DCP, LLDP, MODBUS.

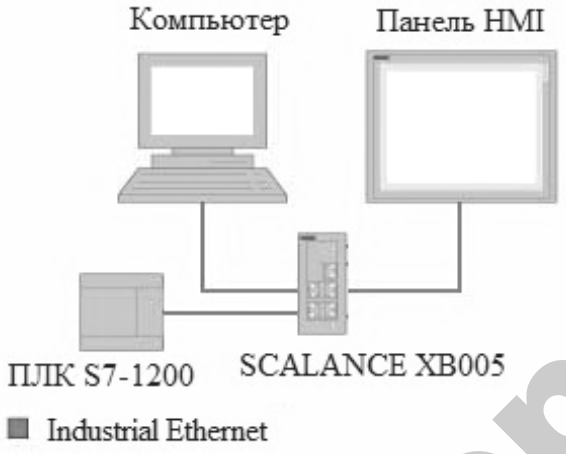


Рис. 8. Звездообразная топология сети на лабораторном стенде

Коммутатор SCALANCE XB005 используется для сетевого соединения компьютера (устройства программирования), ПЛК Simatic S7-1200 и сенсорной панели оператора по схеме, представленной на рис. 8.

По включению питания встроенная в ПЛК операционная система выполняет самотестирование и настройку аппаратных ресурсов, очистку оперативной памяти данных (ОЗУ), контроль целостности прикладной программы пользователя. Если прикладная программа сохранена в памяти программ, ПЛК переходит к основной работе, которая состоит из постоянного повторения последовательности действий, входящих в рабочий цикл. В самом начале цикла ПЛК производит физическое чтение входов. Считанные значения входных сигналов размещаются в области памяти входов. Таким образом, создается одномоментная зеркальная копия значений входов. Далее выполняется код пользовательской программы. Пользовательская программа работает с копией значений входных и выходных сигналов, размещенной в соответствующих областях (I-область и Q-область) системной памяти данных. После выполнения кода пользовательской программы операционная система ПЛК

выполняет копирование расчетных значений выходных сигналов из Q-области памяти на физические выходы ПЛК (рис. 9).

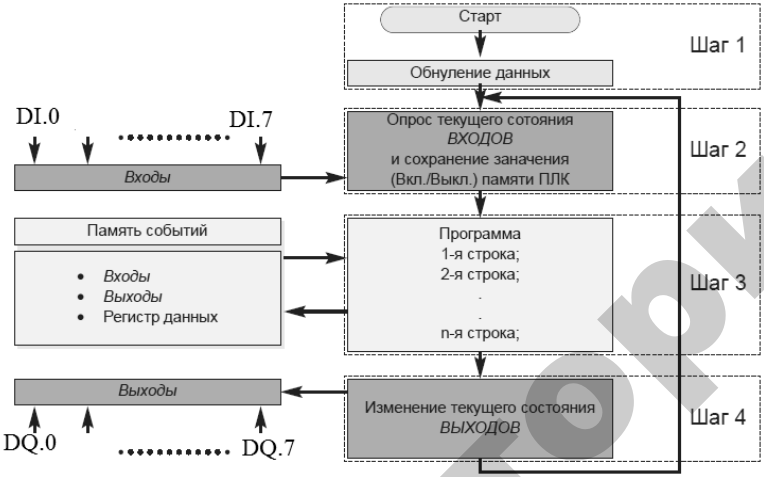


Рис. 9. Цикл выполнения управляющей программы пользователя в ПЛК

Такая последовательность действий, включающая замер, обсчет и выработку воздействия, называется рабочим циклом ПЛК или прогоном программы. Время выполнения цикла программы может варьироваться в широких пределах, например, 10..500 мс. Таким образом, вычисления в ПЛК всегда повторяются циклически. Выполняемые действия зависят от значения входов контроллера, предыдущего состояния и определяются пользовательской программой.

ЛАБОРАТОРНАЯ РАБОТА № 1
СИСТЕМА ПРОГРАММИРОВАНИЯ TIA PORTAL V13. СОЗДАНИЕ ПРОЕКТА

Цель работы: изучение структуры микропроцессорной системы управления и приобретение навыков создания проекта автоматизации в среде программирования TIA Portal.

Информация для выполнения лабораторной работы

Для разработки программ для ПЛК в настоящее время используются интегрированные среды разработки (ИСР, англ. IDE – Integrated development environment), содержащие в своем составе текстовые редакторы, компиляторы, редакторы связей, загрузчики и симуляторы. ИСР обычно представляет собой единственную программу, в которой проводится вся разработка. Она, как правило, содержит много функций для создания, изменения, компилирования, развертывания и отладки программы ПЛК.

TIA Portal (Totally Integrated Automation Portal) – интегрированная среда разработки программного обеспечения систем автоматизации технологических процессов на основе оборудования производства фирмы Siemens. В TIA Portal объединены три основных программных пакета:

- Simatic Step 7 V.11 для программирования контроллеров S7-1200, S7-300, S7-400 и WinAC;
- Simatic WinCC V.11 для разработки человеко-машинного интерфейса (программирование сенсорных панелей и SCADA-систем);
- Sinamics StartDrive V.11 для программирования преобразователей частоты Sinamics.

В среде TIA Portal предусмотрено два способа отображения структуры проекта автоматизации: портальное представление (portal view) и проектно-ориентированное представление (project view). Портальное представление (рис. 10) отображает структуру проекта с точки зрения задач и функций, которые могут быть

выполнены в проекте, например, создание нового проекта – Create new project, открытие уже созданного ранее и сохраненного на жестком диске проекта – Open existing project, отображение используемых в проекте устройств (контроллеров, панелей оператора, модулей ввода/вывода и др.) и настройку сетевых соединений между устройствами – Devices & networks, мониторинг и диагностика доступных для программирования в данном проекте устройств – Online & Diagnostics и др.

Рис. 10. Портальное представление структуры проекта (portal view):
1 – панели задач; 2 – действия для выбранной задачи;
3 – панель выбора вариантов для указанного действия;
4 – переход к проектно-ориентированному представлению;
5 – название открытого проекта

Проектно-ориентированное представление (рис. 11) отображает все компоненты внутри проекта и позволяет получить быстрый доступ к любому из них. В процессе работы над проектом при необходимости в любой момент можно переключиться от портального к проектно-ориентированному представлению структуры проекта и обратно.

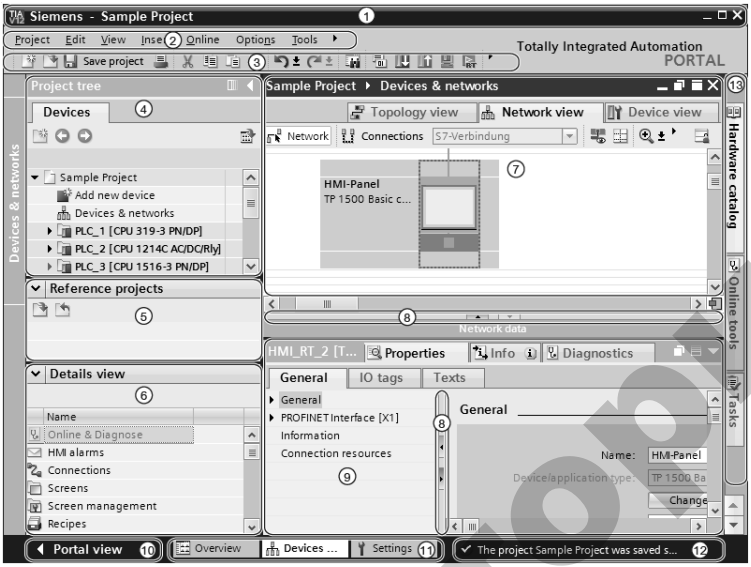


Рис. 11. Проектно-ориентированное представление структуры проекта (project view):
1 – панель заголовка (название проекта); 2 – главное меню;
3 – панель кнопок управления; 4 – дерево проекта;
5 – отображение других проектов, связанных с данным проектом;
6 – подробные данные об объекте, выбранном в дереве проекта;
7 – рабочая область окна; 8 – разделители; 9 – окно инспектора свойств объектов;
10 – переход к портальному представлению;
11 – панель переключения между задачами;
12 – строка состояния; 13 – панель вкладок библиотек компонентов

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы необходимо разработать в среде TIA Portal первый простейший проект для ПЛК Simatic S7-1200. Созданную пользовательскую программу необходимо откомпилировать, загрузить в ПЛК и исследовать ее работу на лабораторном стенде.

Задача управления лампой

Пусть имеется некоторая лампа, подключенная к цифровому выходу ПЛК DQa.0, и имеется кнопка «Start», подключенная к цифровому входу DIa.0.

Постановка задачи

Лампа может быть включена или выключена с помощью данной кнопки. При нажатии на кнопку «Start» лампа включается, а при отпускании – выключается.

Создание проекта

Запуск интегрированной среды разработки TIA Portal V13 выполняется по соответствующему значку на рабочем столе компьютера либо из меню Пуск→Все программы→Siemes automation→TIA Portal V13. Изначально рабочее окно среды разработки TIA Portal находится в режиме портального представления.

Для создания нового проекта необходимо выбрать левой кнопкой мыши действие Create new project (создать новый проект), после чего указать название проекта, задать каталог на диске, в котором будут храниться файлы проекта, и нажать на экране кнопку Create.

Следующий шаг – конфигурация используемого аппаратного оборудования – Devices & Networks→Configure a device. Необходимо добавить в проект (Add new device) контроллер Simatic S7-1200 модели CPU→1214C DC/DC/DC с номером 6ES7 1214-1AG31-0XB0. При этом в поле Device name необходимо указать имя контроллера либо оставить автоматически предложенное имя (PLC_1). Нажать на экране кнопку Add. В результате этого представление проекта автоматически изменится на проектно-ориентированное, в котором в рабочей области окна появится графическое изображение добавленного в проект контроллера. Щелкнув на нем правой кнопкой мыши и выбрав во всплывающем меню пункт Properties (свойства), в нижней части экрана в окне инспектора свойств объекта можно просматривать и нужным образом настраивать свойства ЦПУ. В окне свойств можно установить следующие параметры:

- интерфейс PROFINET: установка IP-адреса для ЦПУ и синхронизации времени;
- DI, DO, и AI: настройка поведения встроенных цифровых и аналоговых входов и выходов;
- скоростные счетчики и генераторы импульсов: активизация и настройка быстрых счетчиков (HSC) и генераторов импульсов, используемых для операций с последовательностями импульсов

(pulse-train operations, PTO) и широтно-импульсной модуляции (pulse-width modulation, PWM);

- запуск: настройка поведения ЦПУ после выключения и последующего включения;
- время суток: установка времени, часового пояса и переключения между летним и зимним временем;
- защита: установка защиты от чтения/записи и пароля для доступа к ЦПУ;
- системная и тактовая битовая память (тактовые меркеры): установка байта для функций «системной памяти» (для битов «первый цикл», «всегда включен» и «всегда выключен») и установка байта для функций «тактовой памяти» (где каждый бит включается и выключается с заранее заданной частотой);
- время цикла: установка максимального времени цикла или фиксированного минимального времени цикла;
- коммуникационная нагрузка: назначение процентной доли времени ЦПУ для коммуникационных задач.

Далее в дереве проекта (Project tree) для добавленного контроллера необходимо найти и открыть таблицу символьных имен переменных – пункт PLC_1→PLC tags→Default tag table, в которой следует определить символьные имена для следующих используемых в программе переменных (табл. 2).

Таблица 2

Символьные имена используемых в проекте переменных

Name	Data Type	Logical Address	Comment
Start	Bool	%I0.0	Кнопка "Старт"
LampOn	Bool	%Q0.0	Цифровой выход управления лампой: FALSE - выключена, TRUE - включена

Далее необходимо добавить в проект кодовый блок, в который будет введен текст программы. Для этого в дереве проекта (Project tree) нужно выбрать пункт

PLC_1→Program blocks→Add new block. В появившемся диалоговом окне (рис. 12) следует сначала указать тип добавляемого кодового блока – Organization Block и Program cycle. Такой блок в процессе работы программы будет выполняться в каждом цикле программы. Далее в поле Language следует выбрать язык программирования – SCL, и, наконец, в поле Name – задать название блока, например, LampControl, либо оставить название, автоматически предложенное системой. Нажать кнопку OK.

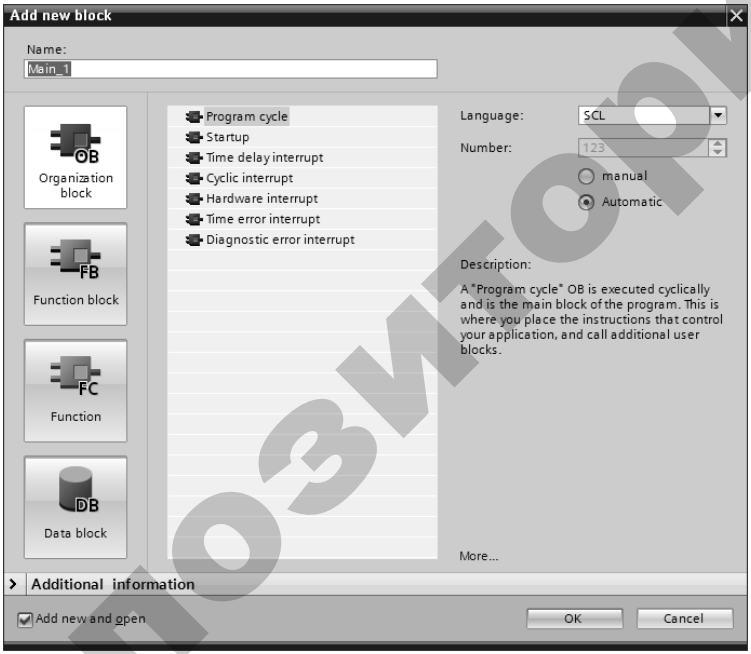


Рис. 12. Диалоговое окно добавления кодового блока в проект

В появившемся на экране рабочем окне для ввода текста программы нужно ввести следующий текст.

Текст программы на языке SCL

```
LampOn =: Start;
```

При загрузке программы пользователя с устройства программирования в ЦПУ она сохраняется в постоянной памяти ЦПУ. Для загрузки проекта в ЦПУ необходимо в главном меню

выбрать команду Online→Download to device (Загрузить в устройство). Альтернативный способ: на панели инструментов щелкнуть на символе (Загрузить в устройство). При этом в появившемся диалоговом окне (рис. 13) необходимо указать интерфейс связи между ЦПУ и компьютером (Type of the PG/PC interface) – PN/IE, а также название сетевого интерфейса устройства программирования (PG/PC interface) – в нашем случае это название сетевой карты компьютера Realtek PCIe GBE Family Controller. После нажатия кнопки Start search (начать поиск) компьютер выполняет поиск и установление связи с подключенным к нему оборудованием, соответствующим заданному в проекте (в нашем случае – только с одним ЦПУ PLC_1). После установления такой связи и нажатия кнопки Load (загрузить) выполняется загрузка программы пользователя в память ЦПУ и его запуск.

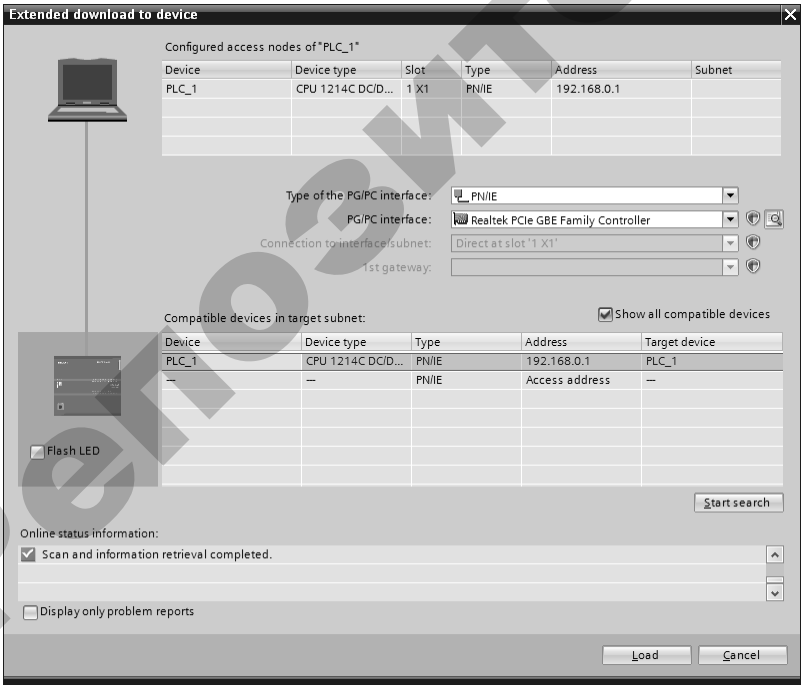


Рис. 13. Диалоговое окно загрузки программы пользователя в ЦПУ

Содержание отчета

1. Название и цель работы.
2. Схемы подключения дискретного входа и дискретного выхода к контроллеру.
3. Выводы.

Контрольные вопросы

1. Раскройте понятие «микропроцессорная система управления». Какие существуют синонимы данного понятия?
2. Что такое дискретный вход или выход? Как определяются их состояния?
3. Понятие и состав программного обеспечения. Операционная система. Система программирования. Прикладная программа.
4. Как создать проект в системе программирования TIA (Totally Integrated Automation) Portal (V13)?
5. В чем вы видите достоинства и (или) недостатки микропроцессорных систем управления?
6. Дайте характеристику дискретным входам и выходам контроллера S7-1200.
7. Нарисуйте схему подключения кнопки с замыкающим контактом к дискретному входу DIa.2.
8. Нарисуйте схему подключения светодиода к дискретному выходу DQb.1.

ЛАБОРАТОРНАЯ РАБОТА № 2
ОСНОВЫ АЛГОРИТМИЧЕСКОГО ЯЗЫКА STRUCTURED CONTROL LANGUAGE. СТРУКТУРА ПРОГРАММЫ. ПОНЯТИЕ ПЕРЕМЕННОЙ. ОСНОВНЫЕ ОПЕРАТОРЫ

Цель работы: изучить основы алгоритмического языка программирования контроллеров Structured Control Language (SCL).

Информация для выполнения лабораторной работы

Структура программы

В зависимости от сложности решаемой задачи управления для программы пользователя может быть выбрана линейная либо модульная структура (рис. 14).

В линейной программе все команды программы выполняются последовательно друг за другом и находятся в одном кодовом блоке, называемом организационным блоком (OB1).

Модульная программа вызывает специальные кодовые блоки, которые выполняют отдельные подзадачи общей задачи управления. Для создания модульной структуры сложная задача автоматизации делится на более простые подзадачи, соответствующие технологическим функциям процесса.

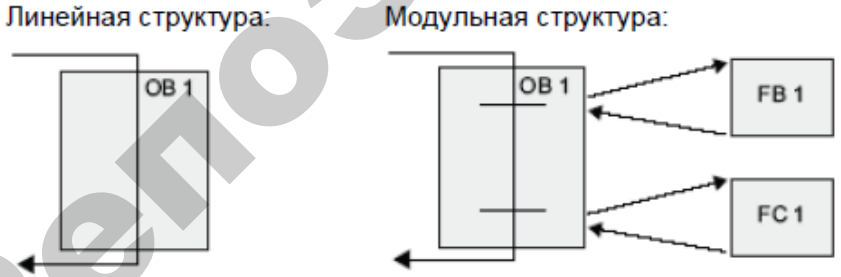


Рис. 14. Линейная и модульная структуры программы пользователя

В ПЛК Simatic S7-1200 имеется три типа кодовых блоков:

- Организационный блок (OB) реагирует на определенное событие в CPU и может прервать исполнение программы

пользователя. Стандартный блок для исполнения программы пользователя (OB1) предоставляет основную структуру пользовательской программы и является единственным кодовым блоком, необходимым для пользовательской программы. Если вы вставите другие OB в свою программу, то эти OB прерывают исполнение OB1. Другие OB выполняют специфические функции, например, для задач запуска, для обработки прерываний и ошибок или для исполнения конкретного программного кода через определенные интервалы времени.

- Функциональный блок (FB) – это подпрограмма, которая исполняется при вызове из другого кодового блока (OB, FB или FC). Вызывающий блок передает параметры в FB, а также определяет некоторый блок данных (DB), который сохраняет данные для этого вызова или экземпляра этого FB. Изменение экземплярного DB позволяет родовому FB управлять работой группы устройств. Например, один FB может управлять несколькими насосами или вентилями с помощью различных экземплярных DB, содержащих конкретные рабочие параметры для каждого насоса или вентиля.
- Функция (FC) – это подпрограмма, которая исполняется при вызове из другого кодового блока (OB, FB или FC). У FC нет связанного с ней кодового блока данных DB. Вызывающий блок передает параметры в FC. Выходные значения FC должны быть записаны в адреса памяти или в глобальный DB.

В качестве примера модульной структуры программы можно привести подробно рассмотренную ниже задачу управления температурой и уровнем воды в водогрейном котле. Данная задача может быть структурно разделена на две подзадачи:

- регулирование уровня воды в котле – обеспечивается управлением работой входного и выходного клапанов на основе сигналов датчиков уровня;
- регулирование температуры воды – обеспечивается периодическим включением и выключением электронагревателя с учетом сигналов датчиков температуры.

Соответственно, программа может быть представлена в виде двух кодовых организационных блоков: TemperatureControl [OB1] (контроль температуры) и WaterLevelControl [OB123] (контроль

уровня воды), которые в процессе прогона программы будут выполняться последовательно один за другим, как это схематически показано на рис. 15.

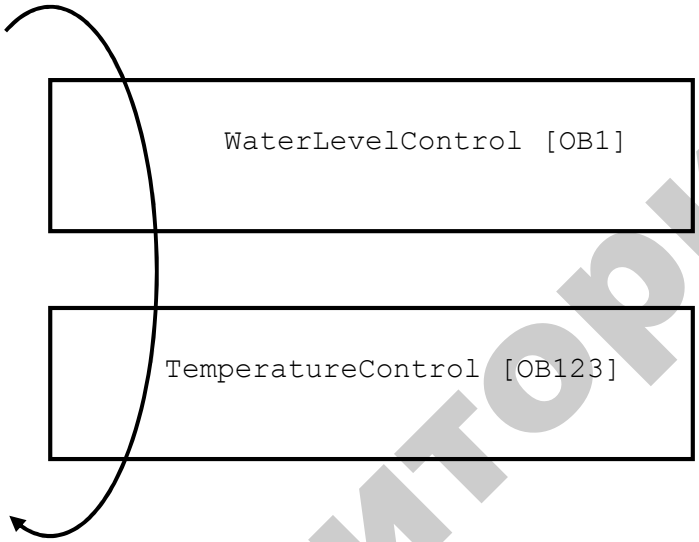


Рис. 15. Модульная структура программы управления водогрейным котлом

Понятие переменной. Адресация переменных, типы данных

Переменная – это область памяти контроллера для хранения данных определенного типа. Каждая переменная характеризуется своим адресом и длиной, которая зависит от типа хранимых данных. При обращении в программе к переменной для считывания или записи данных нужно указать адрес этой переменной. В языках программирования высокого уровня есть два способа указания адреса переменной: абсолютный адрес и символьный адрес. При указании абсолютного адреса переменной нужно указать область памяти контроллера, в которой она находится, размер переменной, номер начального байта, и при необходимости – номер бита. Например, I0.1, QB0, MW20. Первая буква определяет область памяти контроллера (подробнее об этом будет рассказано чуть ниже). Вторая буква определяет размер переменной в байтах: B (byte) – 1 байт, W (word) – 2 байта,

D (double word) – 4 байта. Однако такой способ указания адреса не всегда удобен и при использовании в программе большого количества переменных может привести к путанице при написании программы. Поэтому можно использовать символьное указание адреса, при котором с абсолютным адресом переменной связывается некоторое состоящее из символов слово (имя переменной), имеющее для программиста определенный понятный ему смысл. Так, например, если к дискретному входу контроллера I0.1 подключена кнопка пуска двигателя, то с адресом I0.1 можно связать символьное имя данной переменной ButtonStart. Для одной переменной может быть задано только одно символьное имя.

Для задания символьных имен переменных в Simatic Step 7 предусмотрены таблицы символьных имен – PLC tags. Для открытия таблицы символьных имен необходимо в дереве проекта (Project tree) выбрать пункт PLC_1→PLC tags→Default tag table, в результате чего откроется таблица Default tag table, создаваемая средой Simatic Step 7 автоматически для каждого проекта. Структура таблицы символьных имен показана на стр. 37, табл. 13**Ошибка! Источник ссылки не найден.**. Для определения символьного имени переменной необходимо задать само символьное имя (Name), указать ее тип данных (Data type), абсолютный адрес (Adress) и при необходимости – комментарий.

ЦПУ предоставляет несколько возможностей для сохранения данных во время исполнения программы пользователя (табл. 3).

Глобальная память: ЦПУ предоставляет ряд специализированных областей памяти, включая входы (I), выходы (Q) и битовую память (меркеры) (M). Эта память доступна для всех кодовых блоков без ограничения.

Блок данных (DB): может быть включен в программу для сохранения данных для кодовых блоков. Эти данные сохраняются после исполнения соответствующего кодового блока. В «глобальном» DB сохраняются данные, которые могут быть использованы всеми кодовыми блоками, тогда как в экземплярном DB хранятся данные только для конкретного FB, и они структурированы в соответствии с параметрами этого FB.

Временная память: при вызове кодового блока операционная система ЦПУ выделяет временную, или локальную, память (L) для использования во время исполнения этого блока. Когда исполнение кодового блока заканчивается, ЦПУ выделяет эту локальную память для исполнения другого блока.

Таблица 3

Области памяти ПЛК для хранения данных

Область памяти	**Описание**
I Образ процесса на входах ПЛК	В начале каждого цикла (прогона) копируется из физических входов
I__:P Физический вход	Непосредственное чтение физических входов ЦПУ, сигнальной платы (SB) или сигнального модуля (SM)
Q Образ процесса на выходах	В начале каждого цикла (прогона) копируется в физические выходы
Q__:P Физический выход	Непосредственная запись в физические выходы ЦПУ, SB или SM
M Битовая память	Управление и память данных
L Временная память	Временные, локальные данные для блока
DB Блок данных	Память данных, а также память параметров для FB

I (образ процесса на входах): CPU опрашивает периферические (физические) входы в каждом цикле непосредственно перед исполнением циклического OB и записывает эти значения в образ процесса на входах. Можно обращаться к образу процесса на входах побитно, побайтно, пословно или используя двойные слова. Разрешается доступ, как на чтение, так и на запись, но обычно входы образа процесса только считываются (табл. 4).

Таблица 4

Обращение к переменной из I-области по абсолютному адресу

Размер переменной	**Адрес**	**Пример**
Бит	I[адрес байта].[адрес бита]	I0.1
Байт, слово или двойное слово	I[размер][адрес начального байта]	IB4, IW5, ID12

Добавляя к адресу ":P", можно непосредственно считывать цифровые и аналоговые входы ЦПУ, SB или SM. Доступ через I_:P отличается от доступа через I тем, что данные получаются непосредственно с входов, к которым производится обращение, а не из образа процесса на входах. Доступ через I_:P называется также прямым доступом на чтение, так как данные считываются прямо из источника, а не из его копии, которая была сделана при последнем обновлении образа процесса на входах.

Так как физические входы получают свои значения непосредственно из подключенных к ним полевых устройств, то запись в эти входы запрещена. То есть доступ через I_:P является доступом только на чтение, в отличие от доступа к I, который возможен как на считывание, так и на запись (табл. 5).

Таблица 5

Обращение к переменной из I__:P области по абсолютному адресу

Размер переменной	**Адрес**	**Пример**
Бит	I[адрес байта].[адрес бита]:P	I0.1:P
Байт, слово или двойное слово	I[размер][адрес начального байта]:P	IB4:P, IW5:P, ID12:P

Q (образ процесса на выходах): CPU копирует значения, хранящиеся в образе процесса на выходах в физические выходы. К образу процесса на выходах можно обращаться побитно, побайтно, пословно или используя двойные слова. К выходам образа процесса разрешается доступ, как на чтение, так и на запись (табл. 6).

Таблица 6

Обращение к переменной из P области по абсолютному адресу

Размер переменной	**Адрес**	**Пример**
Бит	Q[адрес байта].[адрес бита]	Q0.1
Байт, слово или двойное слово	Q[размер][адрес начального байта]	QB4, QW5, QD12

Добавляя к адресу ":P", можно осуществлять непосредственную запись в физические цифровые и аналоговые выходы ЦПУ, SB или SM. Доступ через Q_:P отличается от доступа через Q тем, что данные поступают непосредственно на выходы, к которым осуществляется обращение, и, кроме того, в образ процесса на выходах (запись осуществляется в оба места). Доступ через Q_:P иногда называют прямым доступом, так как данные посылаются прямо на целевой адрес, которому не приходится ждать следующего обновления образа процесса на выходах.

Так как физические выходы непосредственно управляют полевыми устройствами, подключенными к этим выходам, то чтение с этих выходов запрещено. Т. е. доступ через Q_:P является доступом только на запись, в отличие от доступа через Q, при котором возможно как чтение, так и запись (табл. 7).

Таблица 7

Обращение к переменной из Q__:P области по абсолютному адресу

Размер переменной	**Адрес**	**Пример**
Бит	Q[адрес байта].[адрес бита]:P	Q0.1:P
Байт, слово или двойное слово	Q[размер][адрес начального байта]:P	QB4:P, QW5:P, QD12:P

M (область битовой памяти, M-память): Эту область памяти можно использовать для управляющих реле и данных, чтобы хранить промежуточные результаты операций или другую управляющую информацию. К области битовой памяти можно обращаться побитно, побайтно, пословно или используя двойные слова. Для битовой памяти возможен доступ, как на чтение, так и на запись (табл. 8).

Таблица 8

Обращение к переменной из M-области по абсолютному адресу

Размер переменной	**Адрес**	**Пример**
Бит	M[адрес байта].[адрес бита]	M0.1
Байт, слово или двойное слово	M[размер][адрес начального байта]	MB4, MW5, MD12

Temp (временная память): CPU выделяет временную память по мере необходимости. CPU выделяет временную память кодовому блоку в момент его запуска (для OB) или вызова (для FC или FB). При выделении временной памяти кодовому блоку могут повторно использоваться те же адреса временной памяти, которые перед этим были использованы другим OB, FC или FB. CPU не инициализирует временную память в момент выделения, поэтому она может содержать любые значения.

Временная память подобна M-памяти за одним важным исключением: область действия M-памяти "глобальна", а область действия временной памяти "локальна":

M-память: Любой OB, FB и любая FC может обратиться к данным в M-памяти, т. е. данные находятся глобально в распоряжении всех элементов программы пользователя.

Временная память: доступ к данным во временной памяти ограничен тем OB, FB или той FC, где были созданы или объявлены адреса во временной памяти. Адреса временной памяти остаются локальными и не могут быть использованы другими кодовыми блоками, даже если кодовый блок вызывает другой кодовый блок. Например: Если OB вызывает FC, то FC не может обратиться к временной памяти OB, вызвавшего эту функцию.

К временной памяти можно обращаться только с использованием символической адресации.

DB (блок данных) обычно используются для хранения различных типов данных, включая промежуточные результаты операций или другие управляющие параметры для FB, и структуры данных, необходимые для многих команд, например, таймеров и счетчиков. Для обращения к переменным, находящимся в блоке данных, необходимо использовать символическую адресацию (табл. 9).

Таблица 9

Обращение к переменной из блока данных с помощью символической адресации

Адрес	**Пример**
[имя блока данных][имя переменной]	"IEC_Timer_0_DB".PT, "My_DB".MyVar

Каждая переменная имеет определенный тип. Тип переменной определяет ее размер в байтах и описывает свойства переменной, то есть диапазон значений или точности числа, сохраненного в переменной, или какие операции являются возможными с этой переменной. ПЛК Simatic S7-1200 поддерживает следующие простые числовые типы данных, перечисленные в табл. 10.

Таблица 10

Типы данных

Тип данных	**Размер**	**Диапазон**	**Примеры значений**
Bool	1 бит	от 0 до 1	TRUE, FALSE, 0, 1
Byte	1 байт	от 0 до 255	10, 107, 220
Word	2 байта	от 0 до 65535	607, 1112
DWord	4 байта	от 0 до 4294967295	13042014, 14121981
Char	1 байт	от 0 до 255	‘A’, ‘t’, ‘@’
SInt	1 байт	от -128 до 127	-123, 123
Int	2 байта	от -32768 до 32767	-3505, 26648
Dint	4 байта	от -2147483648 до 2147483647	-25, 6071952, 11121942
USInt	1 байт	от 0 до 255	33
UInt	2 байта	от 0 до 65535	44444
UDInt	4 байта	от 0 до 4294967295	555555555
Real	4 байта	от $\pm 1.18 \cdot 10^{-38}$ до $\pm 3.40 \cdot 10^{38}$	-0.1, 10.25
LReal	8 байт	от $\pm 2.23 \cdot 10^{-308}$ до $\pm 1.79 \cdot 10^{308}$	-0.0001, 123.456789

Кроме того, есть специальные типы данных для хранения данных, представляющих собой строки символов (STRING), данных о дате (DATE) и времени суток (TIME_OF_DAY).

Основы алгоритмического языка Structured Control Language

Несмотря на значительное разнообразие производителей контроллеров и систем программирования, международный стандарт МЭК 61131-3 определяет пять основных языков программирования контроллеров (табл. 11).

Таблица 11

Языки программирования промышленных контроллеров, принятые международным стандартом МЭК 61131-3

Название	**Название полное**		**Описание**
IL	Instruction List	Список инструкций	Текстовый ассемблероподобный язык низкого уровня
LD	Ladder Diagram	**Р**елейно-**к**онтактные **с**хемы	Графический язык, программная реализация релейно-контактных схем
FBD	Function Block Diagram	Функциональные блоковые диаграммы	Графический язык. Каждый ФБ имеет входы (слева) и выходы (справа).
SFC	Sequential Function Chart	Последовательные функциональные диаграммы	Графический высокоуровневый язык. Создан на базе математического аппарата сетей Петри
ST	Structured Text	Структури-рованный текст	Текстовый Паскалеподобный язык программирования высокого уровня

Выбор конкретного языка программирования зависит от субъективного опыта разработчика. Если ставится задача логического управления, наглядным является использование графических языков программирования. При этом программа внешне напоминает принципиальную электрическую схему: релейно-контактную или электронной логики. Но использование текстовых языков программирования позволяет решать не только логические задачи на основе алгебры Буля, но и использовать весь арсенал математики, начиная с арифметики, включая интегральное и дифференциальное исчисление и базы данных.

Step7-SCL (Structured Control Language – Структурный управляющий язык) – это текстовый язык высокого уровня, разработанный компанией Siemens для программирования контроллеров серии Simatic. Язык SCL является текстовым паскалеподобным языком программирования высокого уровня и соответствует определенному в международном стандарте МЭК 61131-3 текстовому языку ST.

Для языка SCL определены следующие общие правила синтаксиса:

- одна команда может быть введена в несколько строк;
- каждая команда завершается символом “;” (точка с запятой);
- нет разницы между строчными и прописными буквами;
- комментарии в программе служат только для ее пояснения и не влияют на ее исполнение.

Язык SCL включает следующие виды команд:

- команды присвоения значений (Value assignments) – присваивают переменной некоторое значение (явно указанное значение – константу, результат некоторого выражения либо значение другой переменной);
- команды контроля последовательности выполнения программы (Instructions for program control) – для организации условных переходов, ветвлений (IF … THEN … ELSE …), циклов (FOR … TO … DO …) и т. д.;
- прочие команды – арифметические, логические операции, операции преобразования и округления, перемещения блоков данных и др., представленные на панели Instructions;
- команды вызова программных блоков (block calls) – таймеров, счетчиков, ПИД-регуляторов и т. д.

Арифметические операторы

SCL обеспечивает стандартные арифметические функции (операторы):

=	**присваивание**
+	сложение
–	вычитание и отрицательное значение
*	умножение
/	деление

Эти операторы могут использоваться на числовых переменных в выражениях для выполнения вычислений и должны быть помещены между двумя переменными (как в обычной математике).

```
"x" := 5;
"y" := -10;
"z" := "x";
"out" := "x" - "y";
"out" := 2 * "x" - "z" * "y";
"out" := (10 * "x" + "y") / 4;
```

Операторы двоичной логики

Основными операторами двоичной логики являются операторы NOT, AND, OR, XOR, которые определяют классические логические функции (табл. 12) – инверсия, логическое умножение (И), логическое сложение (ИЛИ) и исключающее ИЛИ.

```
"out" := NOT "in1";
"out" := "in1" AND "in2";
"out" := "in1" OR "in2";
"out" := "in1" XOR "in2";
```

Таблица 12

Таблица истинности логических операторов

in1	NOT
0	1
0	1
1	0
1	0

in1	in2	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Оператор присваивания и операторы двоичной логики являются наиболее часто используемыми операторами и позволяют решать несложные задачи автоматизации на основе позиционных законов управления. Примером такой задачи является упомянутая выше задача управления водогрейным котлом.

Остальные операторы и возможности текстового алгоритмического языка SCL будут изучены в последующих лабораторных работах.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для ознакомления с основами алгоритмического языка программирования SCL необходимо разработать в среде TIA Portal проект для ПЛК Simatic S7-1200, реализующий решение задачи управления водогрейным котлом. Созданную пользовательскую программу необходимо откомпилировать, загрузить в ПЛК и исследовать ее работу на лабораторном стенде.

Задача управления водогрейным котлом

Имеется котел для подогрева воды (рис. 16).

Входы:

I0.0 – датчик минимального уровня воды;

I0.1 – датчик максимального уровня воды.

I0.2 – минимальная температура;

I0.3 – максимальная температура.

Выходы:

Q0.0 – входной клапан (наполнение);

Q0.1 – выходной клапан (слив);

Q0.2 – нагревательный элемент.

Постановка задачи

Необходимо циклически наполнять и опустошать резервуар, непрерывно регулируя температуру воды между максимальным и минимальным значениями.

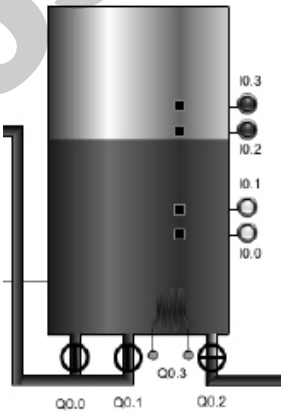


Рис. 16. Бак с водой как объект регулирования уровня воды и температуры

Клапан слива может быть открыт только при условии, когда температура воды находится в допустимых пределах. Клапан слива открывается (Q0.1 = TRUE), если уровень воды в баке превышает максимальный, и выключается (Q0.1 = FALSE), если уровень воды ниже минимального.

Клапан наполнения (Q0.0) включается, если уровень воды в баке ниже минимального, и выключается, если уровень воды выше максимального.

Нагрев воды (Q0.2) должен включаться, если уровень воды выше минимального уровня (I0.0) и температура воды ниже минимальной. Если температура воды достигла максимума, то нагрев воды должен прекратиться до снижения температуры до минимального значения.

Создание проекта

Запуск интегрированной среды разработки TIA Portal V13, создание в ней нового проекта и конфигурация используемого аппаратного оборудования выполняются аналогично тому, как это делалось в лабораторной работе № 1.

Для определения символьных имен используемых в программе переменных выбрать в дереве проекта (Project tree) пункт PLC_1→PLC tags→Default tag table, открыть таблицу символьных имен переменных, в которой определить символьные имена для следующих переменных (табл. 13).

Таблица 13

Символьные имена используемых в проекте переменных

Name	Data Type	Logical Address	Comment
MinLevel	Bool	%I0.0	Датчик минимального уровня воды
MaxLevel	Bool	%I0.1	Датчик максимального уровня воды
MinTemperature	Bool	%I0.2	Датчик минимальной температуры
MaxTemperature	Bool	%I0.3	Датчик максимальной температуры
InFlowValve	Bool	%Q0.0	Входной клапан – наполнение
OutFlowValve	Bool	%Q0.1	Выходной клапан – слив
Heating	Bool	Q0.2	Нагревательный элемент

Далее необходимо добавить в проект кодовый блок WaterLevelControl, в котором будет введена подпрограмма управления уровнем воды в котле. Для этого в дереве проекта (Project tree) нужно выбрать пункт PLC_1→Program blocks→Add new block. В появившемся диалоговом окне (см. рис. 12) следует сначала указать тип добавляемого кодового блока – Organization Block и Program cycle. Такой блок в процессе работы программы будет выполняться в каждом цикле программы. Далее в поле Language следует выбрать язык программирования – SCL, и, наконец, в поле Name – задать название блока: WaterLevelControl. Нажать кнопку OK.

В появившемся на экране рабочем окне для ввода текста программы нужно ввести следующий текст.

Код программы на языке SCL: организационный блок управления уровнем воды

```
  IF "MaxLevel" THEN  // если уровень воды выше
максимального,
  "OutFlowValve" := TRUE; // клапан слива
включается
  "InFlowValve" := FALSE; // клапан наполнения
выключается
  END_IF;
  IF NOT "MinLevel" THEN  //если уровень воды
ниже минимального,
  "OutFlowValve" := FALSE; // клапан слива
выключается
  "InFlowValve" := TRUE;   // клапан наполнения
включается
  END_IF;
  IF ("MaxTemperature" OR (NOT
"MinTemperature")) THEN
  // клапан слива выключается,
  // если температура вне заданного интервала,
  // т.е. выше максимальной ИЛИ ниже минимальной
  "OutFlowValve" := FALSE;
END_IF;
```

Загрузить созданную программу в контроллер и исследовать ее работу на лабораторном стенде.

После этого необходимо добавить в проект кодовый блок TemperatureControl, в котором будет введена подпрограмма управления температурой воды в котле. Текст подпрограммы управления температурой необходимо продумать и ввести самостоятельно, руководствуясь требованиями задачи и используя сведения об основных операторах языка SCL.

Загрузить созданную программу в контроллер, исследовать и продемонстрировать преподавателю ее работу на лабораторном стенде.

Содержание отчета

1. Название и цель работы.
2. Схемы подключения дискретных входов и выходов к контроллеру.
3. Текст разработанной программы управления водогрейным котлом.
4. Выводы.

Контрольные вопросы

1. Определите понятие бит/байт. Какое максимальное число можно записать в 1 байт?
2. Что такое дискретный вход или выход? Как определяются их состояния?
3. На каких языках могут создаваться прикладные программы в системе программирования TIA Portal (V13)?
4. Как объявляются переменные и присваиваются символьные имена дискретным входам и выходам?
5. Каков принцип работы логического оператора NOT?
6. Каков принцип работы логического оператора AND?
7. Каков принцип работы логического оператора OR?

ЛАБОРАТОРНАЯ РАБОТА № 3
РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С РЕАЛИЗАЦИЕЙ СТАНДАРТНЫХ ФУНКЦИЙ ТАЙМЕРА

Цель работы: изучение функций таймера и принципов его использования в программах для ПЛК.

Информация для выполнения лабораторной работы

Время всегда играет важную роль в системах управления. Вот несколько типичных примеров:

- при включении электропривода движения кормораздатчика от бункера с кормом к кормушкам необходимо в течение 2 с подавать предупреждающую световую и звуковую сигнализацию;
- в процессе запаривания картофеля при достижении температуры в чане 120 ºС необходимо выдержать интервал времени 20 мин. И только после этого отключать подачу пара и начинать выгрузку готовой массы картофеля.

Поэтому ПЛК должен иметь в своем составе таймеры как часть языка программирования. Как показывает опыт, таймеры являются практически незаменимыми и одними из наиболее часто используемых команд при программировании ПЛК. С помощью таймерных команд можно создавать программные запаздывания.

В ПЛК S7-1200 представлены в распоряжение программиста следующие программные блоки, реализующие функции таймеров.

TP (Generate pulse) – импульсный таймер – генерирует однократный импульс заданной длительности. Когда на дискретном входе IN имеет место положительный фронт импульса (значение изменяется с FALSE в TRUE), значение на выходе Q устанавливается равным TRUE и начинается отсчет интервала времени, указанного на входе PT. В течение всего указанного интервала времени значение на выходе Q сохраняется равным TRUE (независимо от возможных изменений сигнала на входе IN), а по окончании интервала времени – устанавливается в FALSE. При этом значение времени, прошедшего с момента начала отсчета, сохраняется на выходе ET (elapsed time – прошедшее время).

TON (Generate on-delay) – задержка включения на заданное время. Когда на дискретном входе IN имеет место положительный фронт импульса (значение изменяется с FALSE в TRUE), начинается отсчет указанного интервала времени, по истечении которого значение на выходе Q устанавливается в TRUE. После установления выхода Q = TRUE это значение сохраняется в течение всего времени, пока значение на входе IN = TRUE. Как только IN = FALSE, значение на выходе Q сразу же сбрасывается в FALSE.

TOF (Generate off-delay) – задержка выключения на заданное время. Когда значение IN=TRUE, значение на выходе также сразу устанавливается Q=TRUE. Когда значение на входе IN изменяется с TRUE на FALSE, начинается отсчет указанного интервала времени, по прошествии которого значение на выходе Q сбрасывается в FALSE.

TONR (Timer accumulator) – запоминающий таймер с запаздыванием включения. Выход запоминающего таймера с запаздыванием включения устанавливается в состояние TRUE по истечении заранее заданного времени. Истекшее время накапливается в переменной на выходе ET все время, пока вход IN = TRUE. Если на вход IN подать FALSE, накопление прекратится, а при подаче TRUE на IN – возобновится. Когда накопленное значение времени достигнет требуемого интервала PT, значение на выходе Q установится равным TRUE. Подача сигнала TRUE на вход R (reset) сбрасывает накопленное значение времени ET в 0, а значение на выходе Q – в FALSE независимо от сигнала на входе IN.

Важнейшим параметром для всех перечисленных блоков таймеров является длительность задаваемого интервала времени PT (preset time), которую часто называют *уставкой*. Значение уставки PT, а также значение истекшего времени ET (elapced time) выражаются в миллисекундах и хранятся в памяти ПЛК в виде двойного целого числа со знаком. Такой тип данных имеет название TIME. Значение типа данных TIME использует идентификатор T# и может быть введено как простая единица времени, например, ″T#200ms″, или в виде комбинированных единиц времени, например, ″T#5s_200ms″.

RESET_TIMER – сбрасывает указанный таймер программно в исходное состояние.

PRESET_TIMER – устанавливает заданное значение временной уставки для указанного таймера.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления сведений о стандартных таймерных блоках TP, TON, TOF и TONR, реализованных в ПЛК Simatic S7-1200, необходимо создать новую программу, в которой задействовать все указанные таймерные блоки, и исследовать ее работу на лабораторном стенде.

Запуск интегрированной среды разработки TIA Portal V13, создание в ней нового проекта и конфигурация используемого аппаратного оборудования описаны на стр. 19.

Для определения символьных имен используемых в программе переменных выбрать в дереве проекта (Project tree) пункт PLC_1→PLC tags→Default tag table, открыть таблицу символьных имен переменных, в которой определить символьные имена для следующих переменных (табл. 14).

Таблица 14

Символьные имена используемых в проекте переменных

Name	Data Type	Logical Address	Comment
in0	Bool	%I0.0	Дискретный вход 0
in1	Bool	%I0.1	Дискретный вход 1
in2	Bool	%I0.2	Дискретный вход 2
in3	Bool	%I0.3	Дискретный вход 3
reset_in	Bool	%I0.4	Дискретный вход сброса таймера TONR
out0	Bool	%Q0.0	Дискретный выход 0
out1	Bool	%Q0.1	Дискретный выход 1
out2	Bool	%Q0.2	Дискретный выход 2
out3	Bool	%Q0.3	Дискретный выход 3
t0	Time	%MD0	Значение времени на выходе ET

Окончание таблицы 14

Name	Data Type	Logical Address	Comment
t1	Time	%MD4	Значение времени на выходе ET
t2	Time	%MD8	Значение времени на выходе ET
t3	Time	%MD12	Значение времени на выходе ET

Далее необходимо добавить в проект новый кодовый блок типа Organization Block, Program cycle – организационный блок циклического исполнения с вводом программы на языке SCL. Необходимые для этого действия описаны на стр. 20.

Так как таймерные блоки являются готовыми к использованию системными блоками, реализованными в пакете Simatic Step 7, то для их ввода в программу не следует набирать вручную соответствующий код, а нужно воспользоваться ***одним из двух*** вариантов:

- поместив курсор в рабочем окне ввода программы в нужное место, перейти на панель инструкций Instructions, открыть вкладку Instructions→Basic instructions→Timer operations и дважды щелкнуть левой клавишей мыши на значке нужного блока;
- сначала перейти на панель инструкций Instructions, открыть вкладку Instructions→Basic instructions→Timer operations и перетащить мышью значок нужного блока в нужное место программы.

При этом среда Simatic Step 7 автоматически предложит создать новый системный блок данных, связанный с функциональным блоком таймера, и в появившемся диалоговом окне ″Call options″ предложит ввести название блока данных. В данном случае необходимо согласиться с предлагаемым по умолчанию вариантом и просто нажать OK. В результате этого в рабочем окне ввода программы появится кодовый блок таймера, имеющий примерно следующий вид:

```
"IEC_Timer_0_DB".TP(IN:=_bool_in_,
                    PT:=_time_in_,
                    Q=>_bool_out_,
                    ET=>_time_out_);
```

В данный кодовый блок в скобках нужно ввести переменные, являющиеся входными и выходными параметрами. Так, справа от параметра ″IN″ помещена подсказка _bool_in_, что говорит о том, что данный параметр является входным параметром блока таймера и на его месте должна быть указана переменная типа Bool. В качестве параметра ″PT″ (уставка) может быть указана как переменная типа Time, так и константа того же типа, например, T#5s. Для выходного параметра Q нужно указать выходную переменную типа Bool, а для выходного параметра ET – переменную типа Time, в которой будет автоматически сохраняться накопленное значение времени, т. е. время, прошедшее с момента подачи сигнала на вход блока таймера. Параметр ET можно не указывать, если в этом нет нужды.

Таким образом, в программу нужно добавить все четыре исследуемых таймерных блока и в качестве их входных и выходных параметров задать определенные в проекте переменные так, как это показано ниже.

Код программы исследования работы таймеров на языке SCL

```
//однократный импульс
"IEC_Timer_0_DB".TP(IN:="in0",
                    PT:=T#3s,
                    Q=>"out0",
                    ET=>"t0");
//включение с запаздыванием
"IEC_Timer_0_DB_1".TON(IN:="in1",
                       PT:=T#5s,
                       Q=>"out1",
                       ET=>"t1");
//выключение с запаздыванием
"IEC_Timer_0_DB_2".TOF(IN:="in2",
                       PT:=T#2s,
                       Q=>"out2",
                       ET=>"t2");
//включение с запаздыванием с накоплением
времени
"IEC_Timer_0_DB_3".TONR(IN:="in3",
                        R:="reset_in",
                        PT:=T#10s,
```

```
Q=>"out3",
ET=>"t3");
```

Загрузить программу в ПЛК и исследовать ее работу на стенде.

Для закрепления полученных знаний о работе таймерных блоков в ПЛК Simatic S7-1200 предлагается самостоятельно продумать решение приведенной ниже задачи управления электродвигателем ленточного транспортера, дополнить созданную программу необходимым кодом, реализующим это решение, и продемонстрировать преподавателю ее работу.

Задача управления ленточным транспортером

Имеется ленточный транспортер для транспортировки сухого комбикорма из бункера в смесительный бак для последующего приготовления жидкого корма. Транспортер приводится в движение электродвигателем, который включается и отключается через промежуточное реле и магнитный пускатель сигналом с дискретного выхода Q0.7 контроллера: Q0.7 = TRUE соответствует включенному состоянию транспортера, а Q0.7 = FALSE – выключенному. Включение и выключение электродвигателя производится кнопкой, подключенной к дискретному входу I0.7 контроллера. При нажатой кнопке (I0.7 = TRUE) двигатель должен быть включен, при отпущенной кнопке (Q0.7 = FALSE) – выключен. Имеется также предупредительная звуковая сигнализация, управляемая сигналом с дискретного выхода Q0.6 контроллера.

Вход: I0.7 – кнопка включения/выключения электродвигателя.

Выходы:

Q0.6 – звуковая сигнализация;

Q0.7 – электродвигатель.

Постановка задачи

1. При нажатии кнопки включения сначала включается и в течение 2 с работает звуковая сигнализация. После отключения звуковой сигнализации включается электродвигатель транспортера.

2. При отключении кнопки электродвигатель отключается с задержкой 15 с, необходимой для очищения ленты транспортера от остатков транспортируемого корма.

Содержание отчета

1. Название и цель работы.
2. Схемы подключения дискретных входов и выходов к контроллеру.
3. Временные диаграммы изменения сигналов на входах и выходах исследуемых таймерных блоков.
4. Выводы.

Контрольные вопросы

1. Для каких целей используются таймерные блоки в программах ПЛК?
2. Какие таймерные блоки реализованы в ПЛК S7-1200?
3. Каково назначение входов и выходов блоков TP, TON, TOF?
4. Каково назначение входа R блока TONR?
5. В чем состоит отличие блока TONR от блока TON&?

ЛАБОРАТОРНАЯ РАБОТА № 4
РАЗРАБОТКА ПРОГРАММЫ ПЛК С РЕАЛИЗАЦИЕЙ СТАНДАРТНЫХ ФУНКЦИЙ СЧЕТЧИКА

Цель работы: изучение функций счетчика и принципов его использования в программах для ПЛК.

Информация для выполнения лабораторной работы

Во многих задачах автоматизации есть необходимость в подсчете чего-либо. Например, от ПЛК можно потребовать подсчитать число предметов партии или зарегистрировать, сколько раз произойдет некоторое событие, а для больших электродвигателей – зафиксировать количество включений. Для этих целей в ПЛК реализованы системные функциональные блоки, выполняющие функции счетчиков.

В ПЛК Simatic S7-1200 реализованы следующие функциональные блоки счета:

CTUD – реверсивный счетчик;

CTU – суммирующий счетчик;

CTD – вычитающий счетчик.

Каждый счетчик для сохранения всех необходимых данных счета использует структуру, хранящуюся в связанном с ним системном (т.е. автоматически создаваемом системой Simatic Step 7) блоке данных. В табл. 15 перечислены все переменные, входящие в системный блок данных счета.

Таблица 15

Переменные системного блока данных счетчика

Параметр	Тип данных	В каких счетчиках используется	Описание
CU	Bool	CTU, CTUD	При подаче положительного фронта сигнала на вход CU текущее значение счета CV увеличивается на 1

Окончание таблицы 15

Параметр	Тип данных	В каких счетчиках используется	Описание
CD	Bool	CTD, CTUD	При подаче положительного фронта сигнала на вход CD текущее значение счета CV уменьшается на 1
R	Bool	CTU, CTUD	Сборс значения счетчика в 0 : при подаче положительного фронта сигнала на вход R текущее значение счета CV становится равным 0
LOAD	Bool	CTD, CTUD	Загрузка предустановленного значения: при подаче положительного фронта сигнала на вход LOAD текущее значение счета CV становится равным значению уставки PV
PV	SInt, Int, Dint, USInt, UInt, UDInt	во всех трех	Уставка
QU	Bool	CTU, CTUD	Истина (TRUE), если CV≥PV
QD	Bool	CTD, CTUD	Истина (TRUE), если CV≤0
CV	SInt, Int, Dint, USInt, UInt, UDInt	во всех трех	Текущее значение счета

При написании программы, когда функциональный блок счетчика помещается в текст программы, связанный с ним блок данных создается автоматически в памяти контроллера.

Принцип организации работы счетчика приведен на рис. 17. Работа счетчика характеризуется двумя числами. Первое из них – это текущее значение (current value – CV). Текущее значение CV увеличивается на 1 при переходе сигнала 0→1 на суммирующем входе и уменьшается при переходе сигнала 0→1 на вычитающем входе. Текущее значение CV может быть сброшено в 0 путем подачи сигнала 1 на вход сброса счетчика R (reset).

Следующее число – это уставка (preset value – PV), которую можно рассматривать как заданную цель для счетчика. Если текущее значение CV достигает значения уставки, сигнал на выходе QU устанавливается равным 1. Если же текущее значение счетчика CV меньше либо равно 0, то в 1 устанавливается сигнал на выходе QD.

Текущее значение CV может быть установлено равным значению уставки PV путем подачи сигнала 1 на вход LOAD загрузки предустановленного значения.

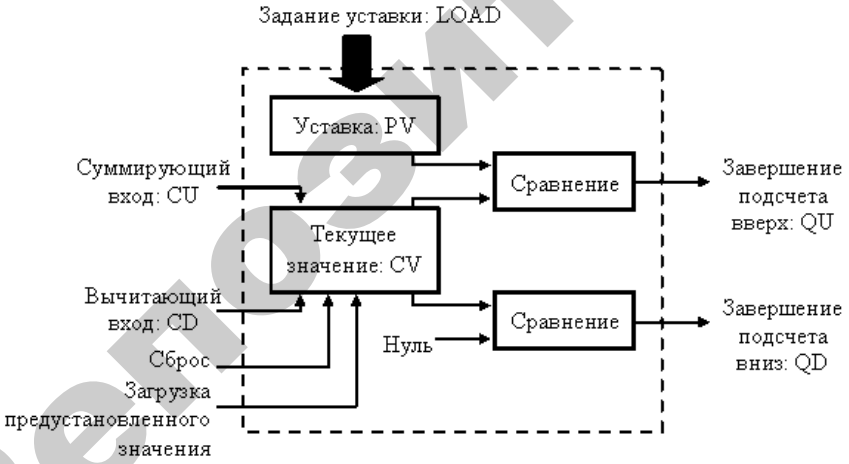


Рис. 17. Принцип организации работы счетчика

Значение уставки PV может быть изменено программно, например, путем ввода нужного числа пользователем с помощью сенсорной панели оператора.

Реализованный в Simatic S7-1200 суммирующий счетчик CTU позволяет выполнять счет вверх (count up – CU) и сброс CV в нулевое значение. Вычитающий счетчик CTD позволяет выполнять счет вниз (count down – CD) и установку CV равным значению уставки PV. Реверсивный счетчик CTUD (count up and down – CTUD) выполняет счет в обоих направлениях.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления сведений о функциональных блоках счетчиков CTU, CTD и CTUD, реализованных в ПЛК Simatic S7-1200, необходимо создать новую программу, в которой задействовать все указанные блоки, и исследовать ее работу на лабораторном стенде.

Запуск интегрированной среды разработки TIA Portal V13, создание в ней нового проекта и конфигурация используемого аппаратного оборудования описаны на стр. 19.

Для определения символьных имен используемых в программе переменных выбрать в дереве проекта (Project tree) пункт PLC_1→PLC tags→Default tag table, открыть таблицу символьных имен переменных, в которой определить символьные имена для следующих переменных (табл. 16).

Таблица 16

Символьные имена используемых в проекте переменных

Name	Data Type	Logical Address	Comment
in0	Bool	%I0.0	Дискретный вход 0
in1	Bool	%I0.1	Дискретный вход 1
in2	Bool	%I0.2	Дискретный вход 2
in3	Bool	%I0.3	Дискретный вход 3
In4	Bool	%I0.4	Дискретный вход 4

Окончание таблицы 16

Name	Data Type	Logical Address	Comment
In5	Bool	%I0.5	Дискретный вход 5
In6	Bool	%I0.6	Дискретный вход 6
In7	Bool	%I0.7	Дискретный вход 7
out0	Bool	%Q0.0	Дискретный выход 0
out1	Bool	%Q0.1	Дискретный выход 1
out2	Bool	%Q0.2	Дискретный выход 2
out3	Bool	%Q0.3	Дискретный выход 3
cv0	Int	%MW0	Текущее значение суммирующего счетчика
cv1	Int	%MW2	Текущее значение вычитающего счетчика
cv2	Int	%MW4	Текущее значение реверсивного счетчика

Далее необходимо добавить в проект новый кодовый блок типа Organization Block, Program cycle – организационный блок циклического исполнения с вводом программы на языке SCL. Необходимые для этого действия описаны на стр. 20.

Так как счетчики являются готовыми к использованию системными функциональными блоками, реализованными в пакете Simatic Step 7, для их ввода в программу следует не набирать вручную соответствующий код, а подобно таймерам (см. стр. 43), перетащить нужный счетчик в нужное место программы мышью с вкладки Instructions→Basic instructions→Counter operations на панели Instruction.

При этом среда Simatic Step 7 автоматически предложит создать новый системный блок данных, связанный с функциональным блоком счетчика, и в появившемся диалоговом окне ″Call options″

предложит ввести название блока данных. В данном случае необходимо согласиться с предлагаемым по умолчанию вариантом и просто нажать OK. В результате этого в рабочем окне ввода программы появится кодовый блок счетчика, имеющий примерно следующий вид:

```
"IEC_Counter_0_DB".CTU(CU:=_bool_in_,
                       R:=_bool_in_,
                       PV:=_in_,
                       Q=>_bool_out_,
                       CV=>_out_);
```

В данный кодовый блок в скобках нужно ввести переменные, являющиеся входными и выходными параметрами. Так, справа от параметра ″CU″ помещена подсказка _bool_in_, что говорит о том, что данный параметр является входным параметром блока счетчика и на его месте должна быть указана переменная типа Bool. В качестве параметра ″PV″ (уставка) может быть указана как переменная неотрицательная (≥0) переменная одного из целочисленных типов данных (SInt, Int, Dint, USInt, UInt, UDInt), так и просто число-константа того же типа, например, 5. Для выходного параметра Q нужно указать выходную переменную типа Bool, а для выходного параметра CV – переменную того же целочисленного типа, что и для PV. Параметр CV можно не указывать, если в программе в этом нет нужды.

Таким образом, в программу нужно добавить все три исследуемых счетчика и в качестве их входных и выходных параметров задать определенные в проекте переменные так, как это показано ниже.

Код программы исследования работы таймеров на языке SCL

```
// суммирующий счетчик
"IEC_Counter_0_DB".CTU(CU:="in0",
                       R:="in1",
                       PV:=4,
                       Q=>"out0",
                       CV=>"cv0");
// вычитающий счетчик
```

```
"IEC_Counter_0_DB_1".CTD(CD:="in2",
                                 LD:="in3",
                                 PV:=7,
                                 Q=>"out1",
                                 CV=>"cv1");
// реверсивный счетчик
"IEC_Counter_0_DB_2".CTUD(CU:="in4",
                                 CD:="in5",
                                 R:="in6",
                                 LD:="in7",
                                 PV:=6,
                                 QU=>"out2",
                                 QD=>"out3",
                                 CV=>"cv2");
```

Загрузить программу в ПЛК и исследовать ее работу на стенде.

Для закрепления полученных знаний о работе счетчиков в ПЛК Simatic S7-1200 предлагается самостоятельно продумать решение приведенной ниже задачи упаковки предметов, создать в TIA Portal новый проект, в котором реализовать решение этой задачи на языке SCL, загрузить созданную программу в контроллер и продемонстрировать преподавателю ее работу.

Задача упаковки кубиков

Имеется линия для упаковки кубиков в коробки (рис. 18). Кубики упаковываются по коробкам. Кубики автоматически подаются по конвейеру, если он включен. В каждую коробку необходимо разместить строго определенное количество кубиков. Лишние кубики будут выпадать из коробки. Необходимо так производить упаковку кубиков, чтобы ни один из них не выпал из коробки.

Входы:

I0.0 – кубик на конвейере;

I0.1 – коробка не заполнена.

Выходы:

Q0.0 – включить конвейер;

Q0.1 – установить новую коробку.

Постановка задачи

В каждую коробку должно попадать по 5 кубиков. Как только нужное число кубиков поступит в коробку, конвейер останавливается. После заполнения очередной коробки поступает сигнал (Q0.1) на установку новой коробки, и конвейер включается снова.

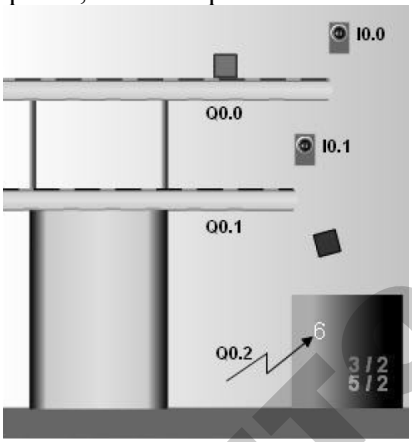


Рис. 18. Конвейер для упаковки кубиков

Содержание отчета

1. Название и цель работы.
2. Схемы подключения дискретных входов и выходов к контроллеру.
3. Временные диаграммы изменения сигналов на входах и выходах блоков CTU, CTD, CTUD.
4. Выводы.

Контрольные вопросы

1. Для каких целей используются счетчики в программах ПЛК?
2. Какие счетчики реализованы в ПЛК S7-1200?

ЛАБОРАТОРНАЯ РАБОТА № 5
РАЗРАБОТКА ПРОГРАММЫ ПЛК С РЕАЛИЗАЦИЕЙ ФУНКЦИЙ ОБРАБОТКИ ДАННЫХ О ВРЕМЕНИ И ДАТЕ

Цель работы: изучение функций ПЛК для обработки данных о времени и дате.

Информация для выполнения лабораторной работы

В Simatic Step 7 для хранения сведений о времени и дате предусмотрены следующие типы данных.

DATE.

Переменные этого типа данных используются для хранения даты в формате «гггг-мм-дд». Переменная типа DATE фактически представляет собой беззнаковое (неотрицательное) целое число, равное числу дней, прошедших от даты 01-01-1990. Диапазон возможных значений переменной – от 01-01-1990 до 31-12-2168. Значение такой переменной может быть записано в программе одним из двух способов:

- с явным указанием типа данных в формате «гггг-мм-дд», например, D#1990-01-01, или DATE#2016-05-23;
- в виде шестнадцатиричного целого числа дней, прошедших, начиная с 01-01-1190, например, 16#00F2.

TIME_OF_DAY (TOD).

Переменные этого типа имеют размер двойного слова (т. е. 4 байта) и представляют собой число миллисекунд, прошедших с момента начала суток. Значение такой переменной в программе записывается в формате «TOD#часы:минуты:секунды.миллисекунды», например, TOD#10:20:30.400. Диапазон возможных значений переменной типа TOD, очевидно, равен от TOD#00:00:00.000 до TOD#23:59:59.999.

DATE_AND_TIME.

Переменные типа DATE_AND_TIME (date and time of day) содержат сведения о дате и времени суток в формате BCD. Длина переменной – 8 байт. Структура данных, хранимых в каждом из байтов, приведена в табл. 17.

Таблица 17

Структура переменной типа DATE_AND_TIME

Номер байта	Содержимое байта	Диапазон значений
0	год	от 0 до 99 (годы от 1990 до 2089)
1	месяц	от 0 до 12
2	день	от 1 до 31
3	час	от 0 до 23
4	минута	от 0 до 59
5	секунда	от 0 до 59
6	две старшие цифры разряда миллисекунд	от 0 до 99
7 (старшие 4 бита)	младшая (третья) цифра разряда миллисекунд	от 0 до 9
7 (младшие 4 бита)	день недели	от 1 до 7

Диапазон возможных значений – от DT#1990-01-01-00:00:00.000 до DT#2089-12-31-23:59:59.999.

DTL (date and time long).

Переменная такого типа имеет длину 12 байт и хранит информацию о времени и дате в виде следующей структуры (табл. 18).

Таблица 18

Структура переменной типа DT

Номер байта	Содержимое байта	Тип данного	Диапазон значений
0 1	год	UINT	1970 – 2200
2	месяц	USINT	1 – 12
3	день	USINT	1 – 31
4	день недели	USINT	1 (вс.) – 7 (сб.) день недели не указывается явно при записи переменной в программе
5	час	USINT	0 – 23

Окончание таблицы 18

Номер байта	Содержимое байта	Тип данного	Диапазон значений
6	минута	USINT	0 – 59
7	секунда	USINT	0 – 59
8	наносекунды	UDINT	0 – 999999999
9			
10			
11			

Диапазон возможных значений – от DTL#1970-01-01-00:00:00.0 до DTL#2200-12-31-23:59:59.999999999. Пример записи в программе: DTL#2008-12-31-20:15:45.250.

Для обработки данных о времени и дате в Simatic Step 7 реализованы следующие основные системные функции, расположенные в среде TIA Portal на вкладке Instructions→Extended instructions→Date and time-of-day.

RD_SYS_T.

Считывание системного времени (read system time) – данная функция считывает текущее системное время со встроенных в ПЛК часов реального времени. Данная функция имеет выходной параметр OUT типа DTL: в переменную, указанную в качестве этого параметра, записывается считанное значение системного времени. Результатом выполнения функции RD_SYS_T является возвращаемое ей значение RET_VAL – целое число типа Int, по смыслу представляющее собой код ошибки выполнения функции (табл. 19**Ошибка! Источник ссылки не найден.**).

Таблица 19

Возможные значения RET_VAL

Код ошибки: число типа Int в 16-ричной форме (W#16#)	**Описание**
0000	Нет ошибки
8081	Полученное значение времени не соответствует интервалу допустимых значений для типа DTL, т. е. меньше DTL#1970-01-01-00:00:00.0 либо больше DTL#2200-12-31-23:59:59.999999999

Значение системного времени является внутренней характеристикой ПЛК, по смыслу представляет собой скоординированное универсальное (всемирное) время (Universal Time Coordinated, UTC) и не учитывает ни местного часового пояса, ни переходов на зимнее время и обратно. Поэтому для практических задач большее значение имеет следующая функция:

RD_LOC_T.

Считывание местного времени (read local time) – данная функция считывает текущее местное время со встроенных в ПЛК часов реального времени. При этом сначала происходит считывание системного времени, которое затем преобразуется программой в местное время с учетом данных настроек ПЛК о часовом поясе и режиме автоматического перехода на зимнее время и обратно. Данная функция имеет выходной параметр OUT типа DTL: в переменную, указанную в качестве этого параметра, записывается считанное значение местного времени. Результатом выполнения функции RD_LOC_T является возвращаемое ей значение RET_VAL – целое число типа Int, по смыслу представляющее собой код ошибки выполнения функции (табл. 20).

Таблица 20

Возможные значения RET_VAL

Код ошибки: число типа Int в 16-ричной форме (W#16#)	**Описание**
0000	Нет ошибки
0001	Нет ошибки. Включен автоматический переход на зимнее время и обратно
8080	Местное время не может быть считано
8081	Недопустимое значение года
8082	Недопустимое значение месяца
8083	Недопустимое значение дня
8084	Недопустимое значение часа
8085	Недопустимое значение минуты
8086	Недопустимое значение секунды
8087	Недопустимое значение наносекунды
80B0	Часы реального времени вышли из строя

T_CONV.

Преобразование времени (Time Convertation) – эта функция выполняет преобразование данных о времени и дате из одного типа данных в другой. Например, определения текущего времени суток сначала необходимо с помощью функции RD_LOC_T считать локальное время и дату в переменную типа DTL, а затем с помощью функции T_CONV преобразовать эту переменную к переменной типа TIME_OF_DAY (TOD).

T_ADD.

Функция T_ADD (сложение времен) складывает входное значение IN1 (типа данных DTL или Time) с входным значением IN2 (тип Time). Возвращаемое функцией выходное значение OUT имеет тип данных DTL или Time. Возможны операции с двумя типами данных:

Time + Time = Time;

DTL + Time = DTL.

T_SUB.

Функция T_ SUB (вычитание времени) вычитает значение типа Time (входной параметр IN2) из значения типа DTL или Time (входной параметр IN1). Возвращаемое функцией выходное значение OUT выдает значение разности, используя тип данных DTL или Time.

Возможны операции с двумя типами данных:

Time – Time = Time;

DTL – Time = DTL.

T_DIFF.

Функция T_ DIFF (разность времен) вычитает значение типа DTL (входной параметр IN2) из значения типа DTL (входной параметр IN1). Возвращаемое функцией выходное значение OUT выдает значение разности, используя тип данных Time:

DTL – DTL = Time.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для приобретения навыков использования функций обработки информации о времени и дате необходимо разработать в среде TIA Portal проект для ПЛК

Simatic S7-1200, реализующий решение описанной ниже задачи управления отоплением производственного служебного помещения. Созданную программу необходимо откомпилировать, загрузить в ПЛК и исследовать ее работу на лабораторном стенде.

Задача управления отоплением

Для экономии тепловых ресурсов, затрачиваемых на обогрев служебного помещения (например, офиса или цеха), необходимо, чтобы система отопления работала в номинальном режиме (Q0.0 = TRUE) только в холодный сезон года (с октября по март, включительно), в рабочие дни недели в рабочее время суток, т. е. когда в помещении могут находиться сотрудники. В остальное время система отопления должна работать в режиме экономии (Q0.0 = FALSE).

Создание проекта

Запуск интегрированной среды разработки TIA Portal V13, создание в ней нового проекта и конфигурация используемого аппаратного оборудования описаны на стр. 19.

Для определения символьных имен используемых в программе переменных выбрать в дереве проекта (Project tree) пункт PLC_1→PLC tags→Default tag table, открыть таблицу символьных имен переменных, в которой определить символьные имена для следующих переменных (табл. 21).

Таблица 21

Символьные имена используемых в проекте переменных

Name	Data Type	Logical Address	Comment
t	Time_Of_Day	%MD11	Время суток
WorkTime	Bool	%M16.0	Рабочее время суток
WorkDay	Bool	%M16.1	Рабочий день недели
HeatingOn	Bool	%Q0.0	Включение отопления
ColdSeason	Bool	%M16.2	Холодное время года

Далее в этой же таблице символьных имен Default tag table нужно перейти с вкладки Tags (для переменных) на вкладку

User constants (постоянные пользователя) и определить символьные имена и значения следующих констант, которые будут необходимы в программе (табл. 22).

Таблица 22

Символьные имена используемых в проекте констант

Name	Data Type	Value	Comment
BeginTime	Time_Of_Day	TOD#08:00:00.000	Время включения отопления
EndTime	Time_Of_Day	TOD#18:00:00.000	Время отключения отопления
Monday	USInt	2	Понедельник
Friday	USInt	6	Пятница

Далее необходимо добавить в проект новый кодовый блок типа Organization Block, Program cycle – организационный блок циклического исполнения с вводом программы на языке SCL. Необходимые для этого действия описаны на стр. 20.

В программе, очевидно, придется использовать функцию RD_LOC_T считывания текущего местного времени и даты. Для работы этой функции, как было сказано выше, необходима переменная типа DTL, которая должна быть указана в качестве параметра функции и в которую будет помещаться считанное значение. Однако поскольку переменная типа DTL представляет собой сложную структуру данных (см. табл. 18), она не может быть определена в М-области памяти контроллера. Поэтому данную переменную следует определить в области временной локальной памяти (см. стр. 31) созданного организационного кодового блока. Таблица для определения символьных имен локальных переменных и констант кодового блока находится над рабочим окном ввода текста программы (рис. 19). В данной таблице, как показано на рисунке, нужно определить имена двух локальных переменных, необходимых для работы функции RD_LOC_T: переменной RET_VAL типа Int и переменной d типа DTL.

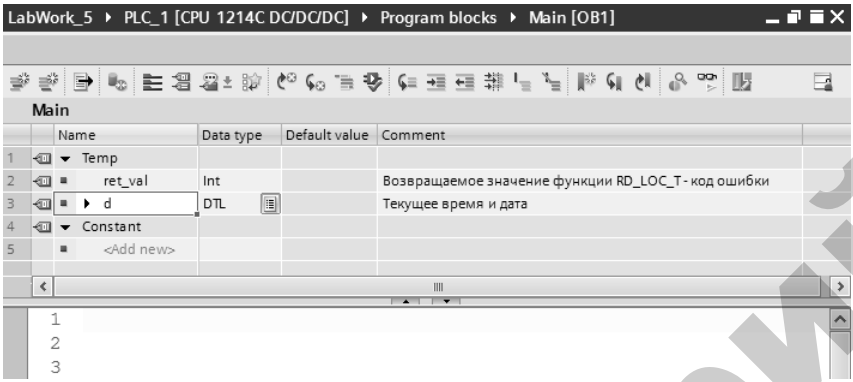


Рис. 19. Рабочее окно ввода программы и таблица локальных данных кодового блока

Как видно на рис. 19, слева от имени переменной ″d″ имеется маленький треугольный указатель вправо. Этот указатель свидетельствует о том, что переменная ″d″ является сложной структурой данных. Нажав на него левой клавишей мыши, можно «развернуть» структуру ″d″ и получить подстрочное отображение информации обо всех составляющих ее компонентах.

Далее в программу нужно добавить следующий код с комментариями. Необходимо помнить, что вводить названия системных функций, в данной программе это RD_LOC_T и DTL_TO_TOD, следует методом перетаскивания их мышью с вкладки инструкций Instrucions→Extended instructions→Date and time-of-day в нужное место программы. Так, для ввода функции DTL_TO_TOD (преобразование из типа данных DTL к типу TOD) во второй строке программы нужно вставить в это место описанную выше функцию T_CONV. При вставке появится маленькое диалоговое окно T_CONV, в котором потребуется указать тип входного и выходного параметров: DTL в поле Source type и TOD в поле Target type.

Код программы исследования работы таймеров на языке SCL

```
#ret_val := RD_LOC_T(#d); // определение
текущего времени и даты
"t" := DTL_TO_TOD(#d); // определение времени
суток
```

```
// рабочий день недели - WorkDay=TRUE, выходной
- FALSE
"WorkDay" := (#d.WEEKDAY>="Monday") AND
(#d.WEEKDAY<="Friday");
// рабочее время суток - WorkTime=TRUE,
нерабочее - FALSE
"WorkTime" := ("t">="BeginTime") AND
("t"<="EndTime");
// холодное время года - с октября по март
включительно
"ColdSeason" := (#d.MONTH>9) OR (#d.MONTH<4);
// включение отопления - в холодное время года,
//                       в рабочий день недели,
//                       в рабочее время суток
"HeatingOn" := "ColdSeason" AND "WorkDay" AND
"WorkTime";
```

Загрузить программу в ПЛК и исследовать ее работу на стенде.

Переопределить значения используемых в программе констант (табл. 22**Ошибка! Источник ссылки не найден.**) таким образом, чтобы включение либо отключение отопления можно было бы наблюдать в настоящий момент. Повторно загрузить программу в контроллер и понаблюдать за включением и отключением отопления. Продемонстрировать работу программы преподавателю.

Содержание отчета

1. Название и цель работы.
2. Схема подключения дискретного выхода к контроллеру.
3. Выводы.

Контрольные вопросы

1. Какие типы данных для хранения информации о времени и дате предусмотрены в ПЛК S7-1200?

2. Какие системные функции обработки данных о времени и дате реализованы в ПЛК S7-1200?

ЛАБОРАТОРНАЯ РАБОТА № 6
СРЕДСТВА ВИЗУАЛИЗАЦИИ ЧЕЛОВЕКО-МАШИННОГО ИНТЕРФЕЙСА. ОРГАНИЗАЦИЯ ЦИФРОВЫХ ПОЛЕЙ ВВОДА/ВЫВОДА НА ДИСПЛЕЕ ПАНЕЛИ

Цель работы: знакомство с сенсорными панелями фирмы Siemens, получение навыков создания систем визуализации с помощью среды WinCC.

Информация для выполнения лабораторной работы

Для визуализации процессов измерения, регулирования различных параметров в промышленности используют сенсорные панели. Преимуществом использования этих устройств является их компактность, простота программирования, невысокая стоимость. Сенсорные панели разных производителей и разных поколений обладают разными функциональными возможностями.

Типовая панель предоставляет пользователю следующую функциональность:

- визуализация параметров технологического процесса (или объекта) в текстовом или графическом режимах;
- управление и обработка аварийных сообщений, регистрация времени и даты возникновения аварийных сообщений;
- ручное управление с помощью функциональных кнопок или сенсорного экрана;
- возможность программирования графики и настройки функциональных клавиш;
- построение диаграмм и трендов, отображение сводных отчетов.

Пример подключения и настройки сенсорной панели

Методику включения в проект и настройки сенсорной панели рассмотрим на примере задачи визуализации работы функционального блока счетчика CTU. По условию, счетчик по нарастанию CTU должен подсчитывать число включений тумблера, подключенного к дискретному входу контроллера I0.0. При достижении текущим значением числа включений тумблера

(Current_Value) некоторого определенного предустановленного значения Preset (уставки) сигнал на дискретном выходе контроллера out (Q0.0) должен становиться равным TRUE. При подаче же положительного сигнала TRUE на дискретный вход count_reset (I0.1) происходит сброс счетчика, т. е. текущее значение счетчика становится равным нулю, а сигнал на дискретном выходе out (Q0.0) – равным FALSE. По условию визуализация работы счетчика состоит в том, чтобы на экране сенсорной панели отображались текущее значение счетчика и предустановленное значение (уставка), а также имелась возможность задания пользователем нового значения уставки.

Решение этой задачи необходимо начинать с программирования контроллера. Запуск интегрированной среды разработки TIA Portal V13, создание в ней нового проекта, добавление в проект контроллера и его настройка описаны на стр. 19.

Для определения символьных имен используемых в программе переменных выбрать в дереве проекта (Project tree) пункт PLC_1→PLC tags→Default tag table, открыть таблицу символьных имен переменных, в которой определить символьные имена для следующих переменных (табл. 23).

Таблица 23

Символьные имена используемых в проекте переменных контроллера

Name	Data Type	Logical Address	Comment
count_input	Bool	%I0.0	Вход счетчика
count_reset	Bool	%I0.1	Сброс счетчика
Preset	Int	%MW0	Установленное значение (уставка)
Current_Value	Int	%MW2	Текущее (накопленное) значение
out	Bool	%Q0.0	Двоичный выход счетчика

Далее необходимо добавить в проект новый кодовый блок типа Organization Block, Program cycle – организационный блок циклического исполнения с вводом программы на языке SCL. Код программы будет содержать в данном случае один-единственный функциональный блок – счетчик по возрастанию CTU. Добавление счетчиков в программу выполнялось ранее и описано на стр. 51. В качестве входных и выходных параметров блока должны быть указаны переменные из табл. 23.

Код программы для ПЛК

```
"IEC_Counter_0_DB".CTU(CU:="count_input",
                       R:="count_reset",
                       PV:="Preset",
                       Q=>"out",
                       CV=>"Current_Value");
```

Для программирования человеко-машинного интерфейса на сенсорной панели оператора нужно добавить ее в проект и выполнить ее настройку. Добавление в проект сенсорной панели (на лабораторном стенде имеется панель KTP700 Basic) выполняется путем выбора в дереве проекта (Project tree) пункта Add new device (добавить новое устройство). В появившемся диалоговом окне необходимо выбрать тип устройства – HMI, и его конкретную модель: HMI→SIMATIC basic panel→7″ Display→KTP700 Basic→ 6AV2 123-2GB03-0AX0. При этом запустится Мастер добавления сенсорной панели (HMI Device Wizard), который предложит пользователю выполнить настройку панели. Первым шагом такой настройки является задание связи добавляемой панели с имеющимся в проекте контроллером (в общем случае в проекте может быть как несколько контроллеров, так и несколько панелей). В поле Select PLC нужно нажать кнопку Browse и в появившемся всплывающем окошке указать имя единственного в данном проекте контроллера.

Следующим шагом настройки является настройка внешнего вида экрана панели: указание цвета фона (Background color) и параметров «шапки» в верхней части экрана (Date/time, Logo).

Последующие шаги настройки можно пока пропустить, нажав кнопку Finish. При этом в дереве проекта (Project tree) появится

пункт HMI_1[KTP700 Basic panel], при развертывании которого можно получить доступ ко всем свойствам добавленной панели, а в рабочей области среды TIA Portal появится изображение корневого экрана сенсорной панели. Корневой экран (Root screen) отображается автоматически при подаче питания и включении сенсорной панели. На нем нужно размещать компоненты графического интерфейса: кнопки, поля ввода/вывода, текстовые поля, геометрические фигуры и др. Имеющиеся в распоряжении программиста компоненты графического интерфейса сгруппированы в правой части рабочего окна среды TIA Portal на панели вкладок библиотек компонентов Toolbox. Нужные компоненты помещаются на экран панели путем перетаскивания мышью соответствующей пиктограммы с панели Toolbox в нужное место экрана.

Для отображения текстовых данных служит компонент Text field, имеющийся на вкладке Toolbox→Basic objects в виде пиктограммы с изображением буквы «A». Перетащив данный компонент в нужное место экрана, необходимо нужным образом задать его свойства в окне инспектора свойств объекта Properties. Основным свойством является поле Text, в котором непосредственно задается отображаемый текст. Помимо этого, можно указать размер и наименование шрифта (поле Style), размер компонента и его координаты на экране панели (Position & size) и другие свойства.

Для отображения и ввода числовых данных предназначен компонент I/O field, расположенный на вкладке Toolbox→Elements в виде пиктограммы  . Основные свойства для настройки данного компонента сгруппированы в окне инспектора свойств объекта Properties на вкладке General. В поле Process→Tag необходимо указать переменную из памяти контроллера либо HMI панели, значение которой будет связано с данным компонентом. Нажав маленькую пиктограмму с тремя точками  , можно указать путь к нужной переменной в появившемся окне задания пути. В поле Type→Mode задается тип элемента: Output – вывод текущего значения на экран, Input/output – вывод текущего и ввод нового значения переменной вручную, Input – ввод значения переменной. В поле Format можно задать требуемый формат

отображения данных – задать число десятичных знаков, число знаком после запятой и др.

Для решения поставленной задачи на экране панели необходимо разместить два элемента типа Text field для отображения текстовых надписей «Уставка» и «Текущее значение». Также необходимо разместить два элемента типа I/O field и связать их с переменными Preset и Current_Value из памяти контроллера. Для элемента, отображающего значение уставки Preset, необходимо задать режим Input/output, что даст возможность пользователю изменять вручную значение уставки счетчика. Для элемента, отображающего текущее значение счетчика Current_Value, режим должен быть задан как Output.

Примерный внешний вид корневого экрана Root screen сенсорной панели показан на рис. 20.

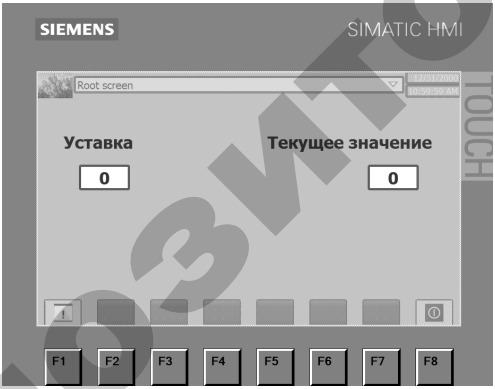


Рис. 20. Внешний вид корневого экрана панели управления и используемые в нем элементы: 2 элемента типа Text field – содержат текстовые надписи «Уставка» и «Текущее значение», 2 элемента типа I/O field – выполняют отображение числовых значений уставки (переменная Preset) и текущего значения счетчика (переменная Current_Value)

Загрузка готового проекта производится отдельно в память контроллера и в память сенсорной панели. Для этого в дереве проекта следует выбрать вкладку Devices & networks. В появившемся окне Devices & networks будет схематическое изображение сетевого соединения контроллера PLC_1 и сенсорной панели HMI_1. Для

загрузки проекта в память контроллера необходимо выделить мышью схематическое изображение контроллера и выбрать команду главного меню Online→Download to device. Затем необходимо проделать то же самое, выделив мышью схематическое изображение сенсорной панели.

Загрузить созданный проект в ПЛК и сенсорную панель и исследовать их работу на стенде.

Содержание отчета

1. Название и цель работы.
2. Схема сетевого соединения контроллера и сенсорной панели.
3. Выводы.

Контрольные вопросы

1. Каково назначение сенсорной панели в проекте автоматизации?

2. Какова последовательность действий по включению в проект TIA Portal сенсорной панели оператора?

ЛАБОРАТОРНАЯ РАБОТА № 7
РАЗРАБОТКА ПРОГРАММЫ ПЛК С РЕАЛИЗАЦИЕЙ ШИРОТНО-ИМПУЛЬСНОЙ МОДУЛЯЦИИ ВЫХОДНОГО УПРАВЛЯЮЩЕГО СИГНАЛА

Цель работы: изучение функции ПЛК для реализации широтно-импульсной модуляции.

Информация для выполнения лабораторной работы

Широтно-импульсная модуляция – это способ влияния на управляющее воздействие при дискретном выходе. Сигналы на дискретных выходах микроконтроллера, как известно, могут принимать только два значения напряжения: напряжение питания (например, 5 В), либо 0 В. ШИМ (широтно-импульсная модуляция) представляет собой импульсный сигнал постоянной частоты и переменной скважности. Скважностью или коэффициентом заполнения (англ. duty cycle) называется отношение полного периода к времени включения. С помощью задания скважности можно менять среднее напряжение на выходе ШИМ, а постоянно меняя скважность – формировать сигнал любой формы.

Примером использования широтно-импульсной модуляции может служить плавное регулирование яркости светодиода. Сигнал на дискретном выходе микроконтроллера, управляющем включением/выключением светодиода, переключается между землей и напряжением питания с достаточно большой частотой, например, 100 Гц. Глаз не замечает мерцания с частотой более 50 Гц, поэтому кажется, что светодиод не мерцает, а горит непрерывно. Уровень яркости светодиода будет зависеть от значения скважности импульсов (рис. 21). Аналогично, разогнанный мотор не может остановить вал за миллисекунды, поэтому ШИМ-сигнал заставит вращаться его в неполную силу.

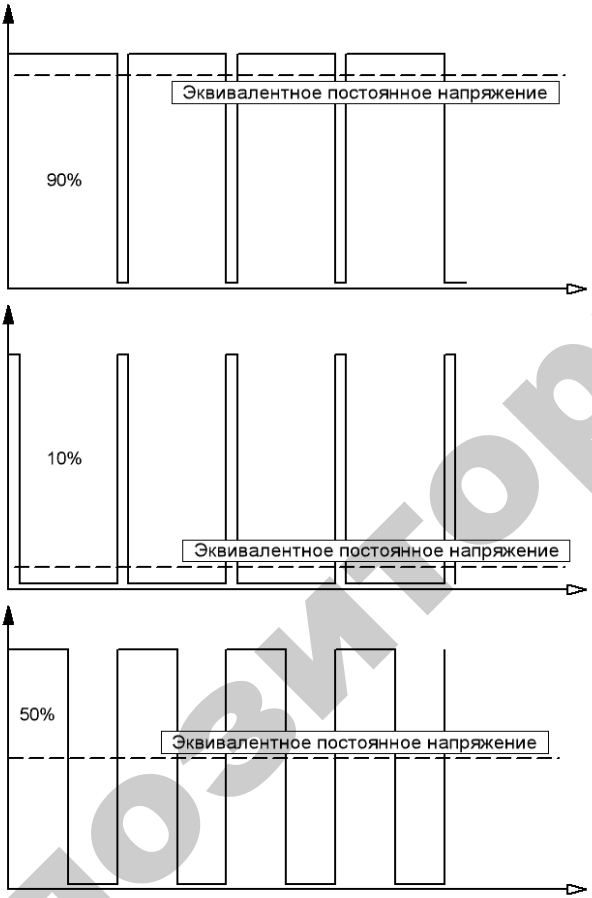


Рис. 21. Принцип управления яркостью светодиода с помощью ШИМ:
а – скважность 50 %; *б* – скважность 10 %; *в* – скважность 90 %

Основное достоинство импульсных регуляторов напряжения с ШИМ – высокий КПД работы, который достигается за счет работы силовых преобразователей в ключевом режиме. В данном случае силовой преобразователь (обычно в роли преобразователя выступает транзистор) работает не в режиме сопротивления, а в режиме ключей, т. е. транзистор либо полностью закрыт, при этом ток через него практически не течет, либо открыт так, что сопротивление его канала

минимальное, и, соответственно, минимальное падение напряжения. В обоих случаях выделяемая на ключе мощность значительно меньше мощности, передаваемой в нагрузку.

Для формирования ШИМ-сигнала в контроллере S7-1200 предусмотрена команда CTRL_PWM. Команда CTRL_PWM (Pulse Width Modulation – широтно-импульсная модуляция) выдает последовательность импульсов с фиксированным временем цикла, но с переменным коэффициентом заполнения. Выход PWM работает непрерывно после запуска с заданной частотой (временем цикла). Ширина импульсов меняется по потребности, чтобы достичь желаемого управления.

Ширина импульса может быть задана в сотых долях времени цикла (0 – 100), в тысячных долях (0 – 1000), в десятитысячных долях (0 – 10000) или в аналоговом формате S7. Ширина импульса может меняться от 0 (отсутствие импульсов, всегда выключено) до полного заполнения (отсутствие импульсов, всегда включено).

Для управления быстрыми импульсными выходами в контроллере S7-1200 имеется два импульсных генератора: ШИМ и последовательность импульсов (PTO). PTO используется командами управления перемещением и в данной работе не рассматривается. Можно назначить каждый импульсный генератор PWM или PTO, но не обоим в одно и то же время.

Эти два импульсных генератора поставлены в соответствие конкретным цифровым выходам, как это показано в табл. 24.

Таблица 24

Возможные значения RET_VAL

Генератор импульсов	**Дискретный выход**
PWM 1	Q0.0
PWM 2	Q0.2

Чтобы подготовить функционирование PWM, сначала нужно сконфигурировать импульсный канал в конфигурации контроллера. Для этого нужно проделать следующие шаги:

1. В дереве проекта для имеющегося в проекте контроллера (PLC_1) выбрать пункт Device configuration, в появившемся рабочем окне выделить правой кнопкой мыши изображение

контроллера и во всплывающем меню выбрать пункт Properties (свойства). В результате в нижней части экрана откроется окно инспектора свойств контроллера PLC_1.

2. В окне инспектора свойств на вкладке General найти пункт Pulse generators (PTO/PWM), развернуть его и выделить подпункт PTO1/PWM1. Разблокировать данный генератор импульсов, поставив галочку в триггерной кнопке Enable this pulse generator. Если генератор импульсов разблокирован, то этому конкретному импульсному генератору назначается уникальное имя по умолчанию. Вы можете изменить это имя, редактируя поле "Name [Имя]:", но оно должно быть уникальным именем. Имена разблокированных генераторов импульсов становятся переменными в таблице переменных "constant", и будут предоставлены для использования в качестве параметра PWM команды CTRL_PWM.

3. Далее нужно назначить параметры (Parameter assignment) следующим образом:

Используемый генератор импульсов Pusle options→Signal type: PWM или PTO – выберите PWM.

Источник вывода Output source: встроенный в CPU или сигнальная плата – в нашем случае на лабораторном стенде нет дополнительной сигнальной платы с дискретными выходами, поэтому данная опция заблокирована, и источником вывода служит встроенный дискретный выход Q0.0.

База времени Time base: миллисекунды или микросекунды.

Формат ширины импульсов: сотые доли (от 0 до 100), тысячные (от 0 до 1000), десятитысячные (от 0 до 10000) либо аналоговый формат S7 (от 0 до 27648).

Время цикла Cycle time – введите значение своего времени цикла. Это значение может быть изменено только в конфигурации устройств, а затем в процессе выполнения программы оно остается неизменным.

Начальная ширина импульсов Initial pulse duration – введите значение своей начальной ширины импульсов. Это значение может быть изменено во время исполнения.

Значение ширины импульса занимает в памяти CPU 2 байта и хранится по адресу: Q1000 – Q1001. Это называется адрес выходного

слова генератора импульсов. Адресом по умолчанию является QW1000 для PWM1 и QW1002 для PWM2. Значение по этому адресу управляет шириной импульса и инициализируется на указанное выше значение для Initial pulse duration (Начальная ширина импульса) при каждом переходе процессора ПЛК из режима STOP в режим RUN. Значение этого выходного (Q) слова можно изменять во время исполнения, чтобы изменить ширину импульса.

После настройки параметров генератора импульсов для управления его работой в программе нужно использовать команду CTRL_PWM. Данная команда вводится в программу не вручную, а путем перетаскивания в нужное место программы мышью с вкладки Instructions→Extended instructions→Pulse на панели Instruction. Команда CTRL_PWM имеет следующие параметры (табл. 25).

Таблица 25

Параметры команды CTRL_PWM

Параметр	**Тип параметра**	**Тип данных**	**Начальное значение**	**Описание**
PWM	входной	Word	0	Идентификатор PWM: имена разблокированных генераторов импульсов становятся переменными в таблице переменных "constant" и предоставляются для использования в качестве параметра PWM
ENABLE	входной	Bool		1 = запустить генератор импульсов 0 = остановить генератор импульсов
BUSY	выходной	Bool	0	Функция занята
STATUS	выходной	Word	0	Код условия выполнения

Для хранения информации о параметрах команда CTRL_PWM использует блок данных (DB). Когда вы вставляете команду CTRL_PWM в программный редактор, ей назначается DB. Параметры этого блока данных не изменяются отдельно пользователем, а управляются командой CTRL_PWM.

В параметре PWM следует указать настроенный предварительно генератор импульсов PWM. Когда вход EN принимает значение ИСТИНА, команда PWM_CTRL запускает или останавливает указанный PWM на основе значения на входе ENABLE. Ширина импульсов определяется значением в соответствующем адресе выходного (Q) слова. Так как S7-1200 обрабатывает запрос, когда команда CTRL_PWM исполняется, то параметр BUSY у моделей CPU S7-1200 всегда принимает значение ЛОЖЬ. Если обнаружена ошибка, то параметр STATUS содержит код ошибки.

Ширина импульса устанавливается на начальное значение, установленное в конфигурации устройств, когда ПЛК впервые переходит в режим RUN. Чтобы изменить ширину импульсов, необходимо записать желаемые значения в адрес выходного (Q) слова.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления сведений о реализации функции широтно-импульсной модуляции на контроллере Simatic S7-1200, необходимо создать в среде TIA Portal новый проект.

В данном проекте нужно добавить контроллер Simatic S7-1200 (см. стр. 19), в котором нужно сконфигурировать широтно-импульсный генератор PWM1. При настройке в поле Name (имя) – оставить предлагаемое по умолчанию имя Pulse_1. Время цикла задать равным 1000 мс.

Значение же ширины импульсов необходимо задавать с экрана сенсорной панели. Для этого в таблице символьных имен переменных контроллера (пункт PLC_1→PLC tags→Default tag table в дереве проекта) нужно определить переменную PulseDuration типа Int по адресу QW1000, который совпадает с адресом выходного слова генератора импульсов PWM1. Значение этой переменной будет

задаваться с экрана сенсорной панели. Для этого в проект следует включить сенсорную панель (см. стр. 66). На экране панели необходимо разместить элемент типа Text field для отображения текстовой надписи «Ширина импульса, %». Также необходимо разместить элемент типа I/O field и связать его с переменной PulseDuration из памяти контроллера. Для элемента I/O field задать режим Input/output, а в поле Format pattern – задать формат отображаемого числового значения в виде «999», что означает неотрицательное целое число длиной от 1 до 3 знаков (ширина в процентах от 0 до 100).

Далее в программу контроллера нужно вставить единственную команду – CTRL_PWM. В качестве параметра PWM следует указать имя сконфигурированного генератора импульсов – Pulse_1. В качестве параметра ENABLE – задать дискретный вход I0.0. Таким образом, генератор будет включаться и работать при включении тумблера I0.0. Оставшиеся два параметра оставить пустыми.

Загрузить созданный проект в ПЛК и сенсорную панель и исследовать их работу на стенде.

Изменить в свойствах генератора импульсов значение времени цикла на 10 мс. Повторно загрузить измененный проект в память ПЛК и пронаблюдать, как изменился режим работы светодиода на выходе Q0.0.

Содержание отчета

1. Название и цель работы.
2. Графики, иллюстрирующие принцип управления яркостью светодиода с помощью ШИМ.
3. Выводы.

Контрольные вопросы

1. В чем заключается принцип управления яркостью светодиода (скорость вращения электродвигателя) с помощью ШИМ?

2. Почему при времени цикла ШИМ, равном 1000 мс, светодиод мигает, а при 10 мс – горит непрерывно?

ЛАБОРАТОРНАЯ РАБОТА № 8
ИЗУЧЕНИЕ ПРИНЦИПОВ ОБРАБОТКИ ПРЕРЫВАНИЙ В ПЛК

Цель работы: изучение принципов обработки прерываний в ПЛК.

Информация для выполнения лабораторной работы

Исполнение пользовательской программы в ПЛК происходит по циклическому типу, что иллюстрирует схема на рис. 22. Операционная система контроллера:

1. Осуществляет считывание значений из области входов.
2. Делает вызов и выполнение необходимой программы.
3. После прохождения алгоритма от начала и до конца осуществляет запись результатов его работы в память выходов.

Рис. 22. Цикл выполнения программы пользователя в ПЛК

«Циклическое сканирование» – это обычный метод выполнения пользовательских программ в ПЛК. Подавляющее большинство систем управления использует только такой способ выполнения программ.

На практике при управлении каким-либо объектом ПЛК должен одновременно следить за сигналами десятков различных

дискретных и аналоговых датчиков (конечные выключатели, кнопки ручного управления, датчики уровня и т.д.) и в зависимости от их значений выполнять определенные управляющие действия. Один из путей такого одновременного слежения состоит в том, что в пользовательской программе для каждого из опрашиваемых входов контроллера необходимо поместить свой отдельный условный оператор.

Пример организации опроса состояния дискретного входа

```
IF ("input" = TRUE) THEN
        (* какие-то действия *)
        …
        …
END_IF;
```

Другой подход основан на использовании прерываний.

Прерыванием называется временное прекращение выполнения микропроцессором текущей основной (фоновой) программы и переход к специальной подпрограмме – обработчику. Прерывание обычно происходит вне всякой связи с фоновой программой при поступлении сигнала от внешних выводов или от внутренних устройств микроконтроллера.

Выполнение основной программы (рис. 23) останавливается, данные, необходимые для дальнейшего продолжения работы, сохраняются в отдельную область памяти (стек) и далее начинается выполнение подпрограммы обработки прерывания.

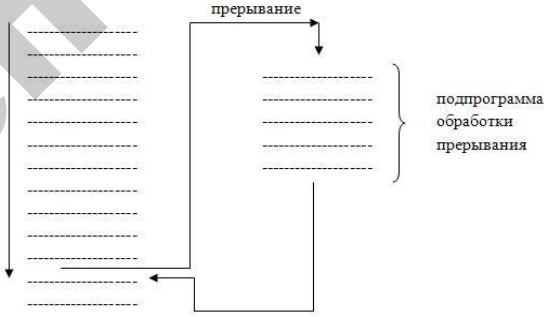


Рис. 23. Выполнение прерывания

После завершения исполнения процедуры обработки прерывания процессор контроллера возвращается к выполнению основной программы с того места, где оно было прервано.

ПЛК Simatic S7-1200 поддерживает следующие события, вызывающие аппаратные прерывания:

– события типа нарастающих фронтов (все встроенные цифровые входы CPU плюс цифровые входы сигнальной платы). Нарастающий фронт возникает, когда цифровой вход переходит из состояния ВЫКЛ в состояние ВКЛ как реакция на изменение сигнала от полевого устройства, подключенного к этому входу;

– события типа падающих фронтов (все встроенные цифровые входы CPU плюс цифровые входы сигнальной платы). Падающий фронт возникает, когда цифровой вход переходит из состояния ВКЛ в состояние ВЫКЛ;

– события типа Текущее значение скоростного счетчика (HSC) = эталонному значению (CV = RV) (HSC 1 … 6). Прерывание CV = RV для HSC генерируется, когда текущее значение переходит от соседнего значения к значению, точно совпадающему с предварительно установленным эталонным значением. Принцип работы и назначение высокоскоростных счетчиков будут подробнее рассмотрены в следующей лабораторной работе;

– события типа Изменение направления счета HSC (HSC 1 … 6). Событие типа Изменение направления счета происходит, когда обнаружено, что HSC перешел от прямого счета к обратному или от обратного к прямому;

– события типа Внешний сброс HSC (HSC 1 … 6). Некоторые режимы HSC допускают назначение цифрового входа для внешнего сброса значения счетчика HSC в ноль. Событие типа Внешний сброс происходит для такого HSC, когда этот вход переходит из состояния ВЫКЛ в состояние ВКЛ.

Аппаратные прерывания должны быть разблокированы при конфигурировании ПЛК.

Опции триггерных кнопок в конфигурации устройств ПЛК:

– цифровой вход;

– разблокировать обнаружение нарастающего фронта;

– разблокировать обнаружение падающего фронта;

– разблокировать этот скоростной счетчик для использования;

– генерировать прерывание при совпадении значения счетчика с эталонным значением;

– генерировать прерывание при изменении направления счета.

Например, чтобы разблокировать аппаратное прерывание (событие) при обнаружении нарастающего фронта для встроенного цифрового входа I0.0 контроллера, необходимо открыть инспектор задания свойств контроллера. Для этого в дереве проекта следует выбрать пункт PLC_1→Device configuration, в появившемся рабочем окне Device view выделить правой кнопкой мыши изображение контроллера и во всплывающем меню выбрать пункт Properties, в результате чего в нижней части экрана появится окно инспектора свойств контроллера PLC_1→Properties. Далее в этом окне на вкладке General следует выбрать пункт DI14/DO10→Digital inputs→Channel0 и поставить галочку в триггерной кнопке Enable rising edge detection (разблокировать определение нарастающего фронта сигнала). Соответственно, для разблокирования определения падающего фронта сигнала нужно поставить галочку в триггерной кнопке Enable falling edge detection.

Когда разблокируется вызывающее аппаратное прерывание событие, то ему присваивается по умолчанию уникальное имя. Вы можете изменить имя этого события, редактируя поле ввода "Event name [Имя события]:", но это имя должно быть уникальным. Имена этих событий становятся именами переменных в таблице переменных "Constants [Константы]" и появляются в ниспадающем списке параметра EVENT для блоков команд ATTACH и DETACH. Значением этой переменной является внутренний номер, используемый для идентификации события.

Далее необходимо связать с разблокированным событием программный организационный блок (OB) обработки данного прерывания – обработчик. По умолчанию при первом разблокировании событию не ставится в соответствие никакой OB. На это указывает метка "– – –" в поле Hardware interrupt [Аппаратное прерывание]. Событию, вызывающему аппаратные прерывания, может быть поставлен в соответствие только OB аппаратных прерываний. Все существующие OB аппаратных прерываний выводятся в ниспадающем списке "HW interrupt:".

Если в этом списке ОВ отсутствуют, то вы должны создать ОВ типа "Hardware interrupt [Аппаратное прерывание]" следующим образом.

1. В ветви "Program blocks [Программные блоки]" дерева проекта: дважды щелкните на "Add new block [Добавить новый блок]", выберите "Organization block [Организационный блок] (ОВ)", а затем "Hardware interrupt".

2. Вы имеете возможность переименовать ОВ, выбрать язык программирования (LAD, FBD или SCL) и задать номер блока (переключитесь в ручной режим и выберите другой номер блока вместо предложенного).

3. Отредактируйте ОВ и добавьте реакцию программы на возникновение события.

Общий принцип действия

Каждое аппаратное прерывание может быть поставлено в соответствие ОВ аппаратных прерываний, который будет поставлен в очередь на исполнение, когда происходит событие, вызывающее это аппаратное прерывание. Установление соответствия между ОВ и событием может происходить во время конфигурирования или во время исполнения.

Вы можете назначать или отменять назначение ОВ разблокированному событию во время конфигурирования. Для назначения ОВ событию во время конфигурирования вы должны использовать ниспадающий список "HW interrupt [Аппаратное прерывание]:" (щелкните на направленной вниз стрелке справа) и выбрать ОВ из списка имеющихся ОВ аппаратных прерываний. Выберите подходящее имя ОВ из этого списка или выберите "<not connected [не присоединено]>" для отмены назначения.

Вы можете также назначать или отменять назначение разблокированного события, вызывающего аппаратные прерывания, во время исполнения. Для этого используйте в программе во время исполнения команды ATTACH или DETACH (при желании - несколько раз) для назначения или отмены назначения разблокированного события, вызывающего аппаратные прерывания, подходящему ОВ. Если никакой ОВ в настоящее время не назначен (из-за выбора "<not connected>" в конфигурации устройств или в

результате выполнения команды DETACH), то разблокированное событие, вызывающее аппаратное прерывание, игнорируется.

Команды ATTACH и DETACH

С помощью команд ATTACH и DETACH вы можете во время исполнения программы связать или отвязать определенный организационный блок OB обработки прерывания с нудным событием. Эти команды вводятся в программу не вручную, а путем перетаскивания в нужное место программы мышью с вкладки Instructions→Extended instructions→Interrupts на панели Instruction.

Команда ATTACH активизирует исполнение подпрограммы OB прерываний для событий, вызывающих аппаратные прерывания.

Команда DETACH деактивизирует исполнение подпрограммы OB прерываний для событий, вызывающих аппаратные прерывания (табл. 26).

Таблица 26

Параметры команд ATTACH и DETACH

Параметр	Тип параметра	Тип данных	Описание
OB_NR	входной	Int	Идентификатор организационного блока: выберите его из имеющихся OB аппаратных прерываний, которые были созданы с помощью опции "Add new block [Добавить новый блок]
EVENT	входной	DWord	Идентификатор (название) события: выберите его из имеющихся событий, вызывающих аппаратные прерывания, которые были разблокированы в конфигурации устройств ПЛК для цифровых входов или скоростных счетчиков

Окончание таблицы 26

Параметр	Тип параметра	Тип данных	Описание
ADD (только ATTACH)	входной	Bool	ADD = 0 (по умолчанию): это событие заменяет все предыдущие назначения событий для этого OB. ADD = 1: это событие добавляется к предыдущим назначениями событий для этого OB
RET_VAL	выходной	Int	Возвращаемое значение – код условия выполнения: 0000 – нет ошибки; 0001 – назначения отсутствуют (только для команды DETACH); 8090 – OB не существует; 8091 – OB неверного типа; 8093 – событие не существует

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления сведений о настройке и использовании аппаратных прерываний ПЛК необходимо создать в среде TIA Portal новый проект.

В данном проекте нужно добавить контроллер Simatic S7-1200 (см. стр. 19). В добавленном контроллере необходимо активировать аппаратное прерывание по нарастающему (либо падающему) фронту сигнала на встроенном дискретном входе I0.x (номер входа x – также уточнить у преподавателя). В качестве обработчика прерывания следует добавить в программу организационный блок

OB типа Hardware interrupt, в котором запрограммировать реакцию программы на данное событие. Требуемая реакция программы состоит в инвертировании (изменении на противоположное) состояния сигнала на встроенном дискретном выходе Q0.x (номер x такой же, как и для указанного выше дискретного входа).

В процессе выполнения программы должно быть реализовано связывание и отвязка блока обработки прерывания с самим прерыванием в зависимости от состояния сигнала на дискретном входе I0.y контроллера (номер входа y – задается преподавателем): при I0.y = TRUE блок обработки связан с самим прерыванием, при I0.y = FALSE – данная связь разрывается.

Содержание отчета

1. Название и цель работы.
2. Графическая схема, иллюстрирующая принцип обработки аппаратного прерывания.
3. Выводы.

Контрольные вопросы

1. В чем заключается принцип обработки прерываний?
2. Какие аппаратные прерывания поддерживаются ПЛК Simatic S7-1200?
3. Каково назначение команд ATTACH и DETACH?

ЛАБОРАТОРНАЯ РАБОТА № 9
РАЗРАБОТКА ПРОГРАММЫ ПЛК С РЕАЛИЗАЦИЕЙ ФУНКЦИИ СЧЕТА БЫСТРЫХ ИМПУЛЬСОВ

Цель работы: изучение функции ПЛК для реализации счета быстрых импульсов.

Информация для выполнения лабораторной работы

Команда CTRL_HSC управляет скоростными счетчиками, которые используются для счета событий, происходящих чаще, чем частота исполнения ОВ. Частота выполнения операций счета командами CTU, CTD и CTUD ограничена частотой исполнения ОВ, в которых они находятся. Так, время прогона программы в ПЛК Simatic S7-1200 по умолчанию равно 150 мс. Это значит, что счетчики CTU, CTD и CTUD могут в данном случае выполнять счет импульсов, частота следования которых не превышает 6 Гц. Минимально допустимое время прогона программы ПЛК Simatic S7-1200 составляет 1 мс. Следовательно, максимально возможная частота следования импульсов для счетчиков CTU, CTD и CTUD составляет 1000 Гц или 1 кГц. Высокоскоростной же счетчик HCS (High speed counter) позволяет правильно подсчитывать импульсы с частотой следования до 100 кГц.

Типичным использованием скоростных счетчиков является счет импульсов, посылаемых на дискретный вход контроллера датчиком скорости вращения в системах управления перемещением.

Конфигурирование скоростного счетчика

В контроллере Simatic S7-1200 имеется возможность сконфигурировать до 6 скоростных счетчиков. Конфигурирование параметров каждого из скоростных счетчиков осуществляется через свойства (Properties) ЦПУ контроллера. Для активизации (разблокирования) скоростного счетчика в дереве проекта следует выбрать пункт PLC_1→Device configuration, в появившемся рабочем окне Device view выделить правой кнопкой мыши изображение контроллера и во всплывающем меню выбрать пункт Properties, в результате чего в нижней части экрана появится окно

инспектора свойств контроллера PLC_1→Properties. Далее в этом окне на вкладке High speed counters (HCS) следует выбрать счетчик с нужным номером и поставить галочку в триггерной кнопке Enable this high speed counter for use (Разблокировать этот скоростной счетчик для использования).

После активизации HSC следует сконфигурировать следующие его параметры (табл. 27).

Таблица 27

Параметры настройки работы скоростного счетчика HSC

Параметр	**Описание**
Параметры настройки режима работы счетчика	
Type of counting	Функция счетчика – тип счета: Count – подсчет импульсов; Frequency – измерение частоты следования импульсов
Operating phase	Алгоритм счета: Single phase – однофазный; Two phase – двухфазный; A/B counter – квадратурный A/B-счетчик; A/B counter fourfold - квадратурный A/B-счетчик 4-тактный
Counting direction is specified by:	Направление счета определяется: User program – программой пользователя (внутреннее управление направлением); Input – сигналом на соответствующем дискретном входе (внешнее управление направлением)
Initial counting direction	Начальное направление счета: Count up – счет импульсов с возрастанием; Count down – счет импульсов с убыванием

Окончание таблицы 27

Параметр	Описание
Initial counter value	Начальное значение счетчика
Initial reference value	Начальное опорное значение счетчика (уставка)
Use external reset input	Использовать внешний дискретный сигнал для сброса счетчика
Параметры настройки аппаратных прерываний счетчика	
Generate interrupt for counter value equals reference value event	Генерировать аппаратное прерывание в том случае, когда текущее значение счетчика становится равным опорному значению (уставке)
Generate interrupt for external reset event	Генерировать аппаратное прерывание при сбросе текущего значения счетчика внешним дискретным сигналом (в том случае, когда для сброса используется внешний дискретный сигнал)
Generate interrupt for change of direction event	Генерировать аппаратное прерывание при смене направления счета

При настройке аппаратных прерываний счетчика необходимо с каждым разблокированным прерыванием связать соответствующий программный блок обработки прерывания – организационный блок OB типа Hardware interrupt, внутри которого следует запрограммировать нужную реакцию ПЛК на данное прерывание[*].

После конфигурирования счетчика, в процессе выполнения программы для управления его работой необходимо использовать команду CTRL_HSC, которая обычно помещается в соответствующие программные блоки обработки прерываний счетчика.

Принцип действия скоростного счетчика

Принцип работы скоростного счетчика в общем случае – при однофазовом алгоритме счета – состоит в том, что при поступлении импульса дискретного сигнала на дискретный вход счетчика контроллер (а точнее, его операционная система), в зависимости от установленного направления счета, увеличивает

[*] Методика настройки аппаратных прерываний контроллера Simatic S7-1200 подробно рассматривалась в предыдущей лабораторной работе (№ 8, стр. 77 – 84)

либо уменьшает на 1 текущее значение счетчика. Текущее значение скоростного счетчика имеет тип данных Dint и занимает в памяти контроллера 4 байта. Процессор ПЛК сохраняет текущее значение каждого скоростного счетчика в области памяти входных данных I по следующим адресам (табл. 28).

Таблица 28

Адреса текущих значений HSC в ОЗУ ПЛК Simatic S7-1200

Скоростной счетчик	**Тип данных**	**Адрес по умолчанию**
HSC1	DInt	ID1000
HSC2	DInt	ID1004
HSC3	DInt	ID1008
HSC4	DInt	ID1012
HSC5	DInt	ID1016
HSC6	DInt	ID1020

Адреса памяти для хранения текущих значений скоростных счетчиков при необходимости могут быть изменены в процессе разработки программы (но не в процессе ее исполнения) при настройке свойств ПЛК.

При подаче питания на ПЛК и запуске программы пользователя в скоростной счетчик операционной системой контроллера загружаются указанные при конфигурации начальное значение (Initial counter value) и значение уставки (Initial reference value). В процессе подсчета импульсов, когда текущее значение счетчика становится равным значению уставки, генерируется аппаратное прерывание и вызывается соответствующий программный блок обработки прерывания. В этом блоке должны быть запрограммированы действия, определяющие реакцию контроллера на данное событие. Например, с помощью команды CTRL_HSC, помещенной в данный блок, счетчику могут быть присвоены новое текущее значение и/или значение уставки.

Если при сбросе текущего значения счетчика внешним дискретным сигналом (при конфигурации была активирована опция Use external reset input), то в счетчик загружается начальное текущее значение (Initial counter value), указанное при конфигурации. Для направления счета и уставки могут быть загружены соответствующие начальные значения, указанные при конфигурации, либо с помощью команды CTRL_HSC, помещенной в блок обработки данного прерывания, могут быть заданы новые значения.

Команда CTRL_HSC

Команда CTRL_HSC управляет скоростными счетчиками. Команда CTRL_HSC вводится в программу путем перетаскивания в нужное место программы мышью с вкладки Instructions→Technology→Counting на панели Instruction. Каждая команда CTRL_HSC использует для сохранения данных структуру, хранящуюся в связанном с ней блоке данных. Блок данных создается и связывается с командой CTRL_HSC, когда команда вставляется в программу. Команда CTRL_HSC имеет следующие параметры (табл. 29).

Таблица 29

Параметры команды CTRL_HSC

Параметр	**Тип параме-тра**	**Тип данных**	**Описание**
HSC	входной	HW_HSC	Идентификатор скоростного счетчика HSC – целое число, автоматически связываемое системой программирования с данным счетчиком при его конфигурации. Данное значение отображается на панели свойств ЦПУ контроллера на вкладке High speed counters (HCS)→HSCi→Hardware identifier
DIR	входной	Bool	1 = Запрос на установку нового направления счета
CV	входной	Bool	1 = Запрос на установку нового значения счетчика
RV	входной	Bool	1 = Запрос на установку нового опорного значения счетчика (уставки)
PERIOD	входной	Bool	1 = Запрос на установку нового интервала времени (только при измерении частоты)
NEW_DIR	входной	Int	Новое направление счета: 1 = вперед, с возрастанием; –1 = назад, с убыванием
NEW_CV	входной	Dint	Новое текущее значение счетчика

Окончание таблицы 29

Параметр	**Тип параме-тра**	**Тип данных**	**Описание**
NEW_RV	входной	Dint	Новое опорное значение счетчика (уставка)
NEW_PERIOD	входной	Int	Новое значение интервала времени в миллисекундах (только при измерении частоты): 10 = 0.01 с; 100 = 0.1 с; 1000 = 1 с
BUSY	выходной	Bool	Функция занята
STATUS	выходной	Word	Код результата выполнения команды (в 16-ричном формате W#16#): 0 = нет ошибки; 80A1 – идентификатор HSC не обращается к HSC; 80B1 – недопустимое значение в NEW_DIR; 80B2 – недопустимое значение в NEW_CV; 80B3 – недопустимое значение в NEW_RV; 80B4 – недопустимое значение в NEW_PERIOD; 80D0 – данный счетчик HSC не активирован в конфигурации ЦПУ

Многие конфигурационные параметры скоростных счетчиков устанавливаются только в конфигурации устройств проекта. Некоторые параметры скоростных счетчиков инициализируются в конфигурации устройств проекта, но позже могут быть изменены под управлением программы. Параметры команды CTRL_HSC обеспечивают программное управление процессом счета, а именно, позволяют в процессе выполнения программы выполнять следующие действия:

– установку нового значения направления счета NEW_DIR;

– установку текущего значения счетчика на новое значение NEW_CV;

– установку опорного значения счетчика на новое значение NEW_RV;

– установку значения интервала времени (для режима измерения частоты) на новое значение NEW_PERIOD.

Если при исполнении команды CTRL_HSC следующие биты установлены в 1, то соответствующее значение NEW_xxx загружается в счетчик. Несколько запросов (более одного бита установлено одновременно) обрабатываются за одно исполнение команды CTRL_HSC.

– DIR = 1 – запрос на загрузку нового значения направления счета NEW_DIR, DIR = 0 – нет изменения;

– CV = 1 – запрос на загрузку нового значения NEW_CV, CV = 0 – нет изменения;

– RV = 1 – запрос на загрузку нового значения NEW_RV, RV = 0 – нет изменения;

– PERIOD = 1 – запрос на загрузку нового значения NEW_PERIOD, PERIOD = 0 – нет изменения.

Если нет запроса на обновление параметра, то соответствующие входные значения игнорируются.

Команда CTRL_HSC обычно помещается в OB аппаратных прерываний, который исполняется, когда запускается аппаратное прерывание, связанное со счетчиком. Например, если событие CV = RV запускает прерывание, связанное со счетчиком, то кодовый блок OB аппаратных прерываний исполняет команду CTRL_HSC и может изменить эталонное значение, загрузив значение NEW_RV.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления сведений о настройке и использовании скоростных счетчиков импульсов необходимо создать в среде TIA Portal новый проект.

В данном проекте нужно добавить контроллер Simatic S7-1200 (см. стр. 19). В добавленном контроллере необходимо активировать скоростной счетчик HSC1 и настроить его свойства следующим образом:

– функция счета (Type of counting) – Count, подсчет числа импульсов;

– алгоритм счета (Operating phase) – Single phase, однофазный;

– направление счета определяется программой пользователя;

– начальное направление счета – с возрастанием;
– начальное значение счетчика 0;
– начальное опорное значение 15.

Необходимо активировать аппаратное прерывание при равенстве текущего значения счетчика опорному значению. При этом данному прерыванию нужно поставить в соответствие программный блок обработки прерывания – OB типа Hardware interrupt, в котором с помощью команды CTRL_HSC выполнить сброс в 0 текущего значения счетчика.

Далее необходимо добавить в проект сенсорную панель оператора, на экране которой организовать вывод текущего значения счетчика HSC1. Для отображения значения счетчика можно использовать графический компонент I/O field (см. стр. 67).

Загрузить созданный проект в ПЛК и сенсорную панель оператора. Исследовать и объяснить работу созданной программы.

Далее необходимо исследовать работу скоростного счетчика в соответствии с тремя оставшимися алгоритмами счета: двухфазным (Two phase), квадратурным A/B (A/B counter) и квадратурным A/B 4-тактным (A/B counter fourfold).

Содержание отчета

1. Название и цель работы.
2. Временные диаграммы изменения дискретных сигналов на входах скоростного счетчика и его текущего значения, иллюстрирующие его работу для каждого из доступных алгоритмов счета.
3. Выводы.

Контрольные вопросы

1. В чем заключается принцип работы скоростного счетчика HSC?
2. В чем состоит отличие скоростного счетчика HSC от счетчиков типа CTU, CTD, CTUD?
3. Для чего предназначена команда CTRL_HSC?
4. Объяснить с физической точки зрения особенности работы созданной программы на лабораторном стенде.

ЛАБОРАТОРНАЯ РАБОТА № 10
ИЗУЧЕНИЕ СЛОЖНЫХ ТИПОВ ДАННЫХ. МАССИВЫ. ЦИФРОВОЙ ВВОД ДАННЫХ С ПАНЕЛИ ЧЕЛОВЕКО-МАШИННОГО ИНТЕРФЕЙСА

Цель работы: изучение сложных типов данных и основ использования и работы с ними в программе ПЛК.

Информация для выполнения лабораторной работы

Тип данных определяет размер элемента данных в памяти, а также то, какие именно данные он может хранить и каким образом этот элемент данных интерпретируется программой. Так, например, тип данных Int имеет размер 16 бит (2 байта). Переменная типа Int содержит целое число, при этом старший из 16 бит определяет знак числа (“+” – если бит равен 0, и “–” – если 1), а остальные 15 бит – абсолютное значение числа. Таким образом, переменная типа Int может содержать целое число в диапазоне от -2^{15} до $2^{15}-1$.

Типы данных в памяти ПЛК могут быть:

– предопределенные (стандартные), т.е. определенные и реализованные разработчиками системы программирования S7;

– пользовательские (User Datatype), размер и структура их данных определяются пользователем в процессе разработки и написания программы.

Так, тип данных Int, очевидно, относится к предопределенным типам данных.

В качестве примера пользовательского типа данных можно привести некий тип данных «Person», содержащий определенную информацию о человеке: фамилию, имя, отчество, пол, возраст, идентификационный номер. Для определения пользовательского типа данных нужно в дереве проекта выбрать ветвь PLC_1→PLC data types→Add new data type и в появившемся рабочем окне построчно задать нужные поля, указав для каждого из них свой тип данных (табл. 30).

Таблица 30

Структура пользовательского типа данных «Person»

Name (название поля данных)	**Data type** (тип данных)	**Default value** (значение по умолчанию)	**Comment** (комментарий)
Family	String	‘’	Фамилия
Name	String	‘’	Имя
SecondName	String	‘’	Отчество
Sex	Bool	TRUE	Пол: TRUE – муж., FALSE – жен.
Age	Byte	0	Возраст
ID	UInt	0	Идентификационный номер

Кроме того, помимо деления на стандартные и пользовательские, по размеру и структуре типы данных могут быть простые и сложные.

К простым относятся стандартные типы данных, размер которых в памяти ПЛК не превышает 4 байт. Соответственно, к сложным относятся стандартные типы данных длиной более 4 байт и все пользовательские типы данных.

Одним из примеров сложного типа данных является описанный на стр. 56 тип DTL (date and time long), имеющий длину 12 байт и используемый для хранения и обработки данных о времени и дате.

Еще одним примером сложного типа данных, которому будет уделено основное внимание в данной лабораторной работе, является ***массив***. Массив, или Array, представляет собой структуру данных, состоящую из некоторого фиксированного числа элементов одного типа.

Свойства типа данных Array приведены в табл. 31.

Таблица 31

Свойства типа данных Array

Длина	Число элементов × длину типа данных
Формат	Array [low limit…high limit] of <data type>
Тип данных элементов массива	Любой, кроме Array

Ниже приведены несколько примеров определения массивов:

Array [1..20] of Real – одномерный массив из 20 элементов вещественного типа;

Array [– 5..5] of Int – одномерный массив из 11 элементов целочисленного типа;

Array [1..2,4..6] of Char – двумерный массив из 6 элементов символьного типа.

Обращение к отдельно взятому элементу массива при чтении или записи в него данных выполняется по его номеру (или индексу). Например, a[3] означает элемент массива a под номером 3.

Создание массивов

Так как массив относится к сложным типам данных, то его нельзя создать в таблице описания переменных ПЛК. В таблице переменных ПЛК можно определять только глобальные переменные ПЛК простых типов данных. Для создания же массива в памяти ПЛК его следует определить либо в отдельно взятом глобальном блоке данных DB, либо внутри одного из остальных кодовых блоков – организационного блока OB, функционально блока FB или функции FC.

В качестве примера определим в памяти ПЛК массив с именем V беззнаковых целых чисел (типа UInt), содержащий 8 элементов с номерами от 1 до 8. Для этого в дереве проекта следует выбрать ветвь PLC_1→Program blocks→Add new block – добавление нового программного блока. В появившемся диалоговом окне следует выбрать тип добавляемого программного блока – Data block и Global DB, указать символьное имя блока (или оставить предложенное по умолчанию). Далее внутри созданного блока данных необходимо определить новую переменную – массив, указав его имя (в столбце Name), тип элементов и диапазон их номеров (в столбце Data type). Активировав триггерную кнопку (поставив «галочку») в столбце Retain, можно обеспечить автоматическое сохранение данного массива в энергонезависимой памяти ПЛК в случае сбоя напряжения питания и непредвиденного отключения ПЛК. При восстановлении питания сохраненные значения элементов массива будут автоматически восстановлены. Слева от имени массива имеется пиктограмма в виде маленького черного треугольника, направленного вправо. Щелкнув по нему левой кнопкой мыши, можно развернуть полный список элементов массива, для каждого из которых при необходимости можно указать в столбце Start value начальное значение, которое будет иметь этот элемент в начале работы программы ПЛК.

Далее в качестве примера работы с массивами рассмотрим задачу нахождения суммы всех элементов созданного массива. Результат суммирования нужно сохранить в отдельной переменной VSum типа UInt, которую можно определить как в таблице глобальных переменных ПЛК (в М-области памяти), так и в созданном уже блоке данных, в котором определен сам массив. Последуем второму варианту и определим переменную VSum в блоке данных.

В простейшем случае программный код для нахождения суммы элементов массива может быть размещен в главном организационном блоке программы Main [OB1] и будет иметь следующий вид.

Код программы нахождения суммы элементов массива

```
"Data_block_1".VSum := 0;
FOR #i := 1 TO 8 DO
    // Statement section FOR
    "Data_block_1".VSum := "Data_block_1".VSum +

    "Data_block_1".V[#i];
END_FOR;
```

В коде программы использован оператор FOR…TO…DO… для организации цикла, который находится на вкладке Instructions→Basic instructions→Program control operations.

В качестве альтернативного варианта рассмотрим случай, когда суммирование элементов массива выделено в отдельную пользовательскую функцию, для которой сам массив служит входным параметром. Для добавления функции в проект программы в дереве проекта выберем ветвь добавления нового программного блока PLC_1→Program blocks→Add new block. На этот раз тип блока следует указать как Function, язык программирования – SCL, имя блоку присвоим FindArrraySum. Входным параметром (Input) данной функции укажем параметр IN типа Array [1..8] of UInt. В качестве выходного (Output) укажем параметр с OUT типа UInt. В числе промежуточных переменных (Temp) данной функции укажем целочисленную переменную i, которая будем содержать значение индекса элемента массива при обращении к нему. Тогда программный код функции FindArrraySum будет иметь вид.

Код программы функции FindArraySum нахождения суммы элементов массива

```
#OUT := 0;
FOR #i := 1 TO 8 DO
     // Statement section FOR
     #OUT := #OUT + #IN[#i];
END_FOR;
```

В данном случае мы пока что описали лишь прототип функции FindArraySum, т. е. указали ее входные и выходные параметры, а также действия, которые она производит. Теперь, чтобы вычислить сумму созданного нами массива V, необходимо в главном организационном блоке программы Main [OB1] выполнить вызов функции FindArraySum. Для этого блок FindArraySum нужно из дерева проекта мышью перетащить в нужное место блока Main [OB1] и далее в качестве параметра IN указать массив "Data_block_1".V, а в качестве параметра OUT – переменную "Data_block_1".VSum.

Код вызова функции FindArraySum в главном блоке программы Main [OB1]

```
"FindArraySum"(IN:="Data_block_1".V,
OUT=>"Data_block_1".VSum);
```

Цифровой ввод данных с панели человеко-машинного интерфейса

Для отображения и ввода числовых данных с панели человеко-машинного интерфейса предназначен компонент I/O field, расположенный на вкладке Toolbox→Elements в виде пиктограммы  . Основные свойства для настройки данного компонента сгруппированы в окне инспектора свойств объекта Properties на вкладке General. В поле Process→Tag необходимо указать переменную из памяти контроллера либо HMI панели, значение которой будет связано с данным компонентом. Нажав маленькую пиктограмму с тремя точками  , можно указать путь к нужной переменной в появившемся окне задания пути. В поле Type→Mode задается тип элемента: Output – вывод текущего значения на экран, Input/output – вывод текущего и ввод нового значения переменной

вручную, Input – ввод значения переменной. В поле Format можно задать требуемый формат отображения данных – задать число десятичных знаков, число знаком после запятой и др.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления полученных сведений об использовании массивов в программе ПЛК необходимо создать в среде TIA Portal новый проект.

В данном проекте нужно добавить контроллер Simatic S7-1200 (см. стр. 19). В памяти ПЛК нужно определить массив целых чисел типа UInt, состоящий из 8 элементов с номерами от 1 до 8, а также реализовать в виде отдельных пользовательских функций

- вычисление суммы элементов массива,
- нахождение максимального значения его элементов,
- нахождение минимального значения его элементов.

Для хранения результатов суммирования, максимального и минимального значений определить отдельные переменные типа UInt.

Далее необходимо добавить в проект сенсорную панель оператора, на экране которой организовать ввод и отображение значений элементов массива, а также отображение суммы, максимального и минимального значений его элементов. Использовать для этого графический компонент I/O field.

Содержание отчета

1. Название и цель работы.
2. Формат определенного в памяти ПЛК массива.
3. Программный код функций определения максимального и минимального значений элементов массива.
4. Выводы.

Контрольные вопросы

1. В чем состоит отличие простых и сложных типов данных?
2. Что называется массивом?
3. В каких областях памяти ПЛК может быть определен массив?
4. Каков принцип действия оператора FOR…TO…DO..?

ЛАБОРАТОРНАЯ РАБОТА № 11
ИЗУЧЕНИЕ МЕТОДОВ ОТЛАДКИ ПРОГРАММ ПЛК

Цель работы: изучение методов отладки и тестирования программ ПЛК.

Информация для выполнения лабораторной работы

Светодиоды состояния

ЦПУ и модули вода/вывода используют светодиоды для предоставления информации о рабочем состоянии модуля или входов/выходов. У ЦПУ имеются следующие индикаторы состояния:

· STOP/RUN

– Постоянно горящий оранжевый свет указывает на состояние STOP.

– Постоянно горящий зеленый свет указывает на режим RUN.

– Мигающий (попеременно зеленый и оранжевый) указывает, что ЦПУ находится в состоянии запуска.

· ERROR

– Мигающий красный указывает на ошибку, например, внутреннюю ошибку в ЦПУ, ошибку карты памяти или ошибку конфигурирования (несогласованные модули).

– Постоянно горящий красный указывает на неисправность аппаратуры.

· MAINT (обслуживание) мигает всякий раз, как вы вставляете карту памяти. Затем CPU переходит в состояние STOP. После того как ЦПУ перешел в состояние STOP, выполните одно из следующих действий, чтобы инициировать анализ карты памяти:

– Переведите ЦПУ в режим RUN.

– Выполните полное стирание памяти (MRES).

– Выключите ЦПУ и включите его снова.

ЦПУ предоставляет также два светодиода, которые указывают состояние связи через PROFINET. Откройте нижнюю крышку клеммного блока, чтобы увидеть светодиоды PROFINET.

– Link [соединение] (зеленый) включается, чтобы показать, что соединение выполнено успешно.

– Rx/Tx (желтый) включается, чтобы показать активность передачи.

ЦПУ и каждый цифровой сигнальный модуль (SM) имеют по одному светодиоду канала ввода/вывода для каждого из цифровых входов и выходов. Светодиод канала ввода/вывода (зеленый) включается или выключается, чтобы показать состояние отдельного входа или выхода.

Кроме того, каждый цифровой SM имеет светодиод DIAG, который указывает состояние модуля:

– Зеленый указывает, что модуль готов к работе.

– Красный указывает, что модуль неисправен или не готов к работе.

Каждый аналоговый SM имеет по одному светодиоду канала ввода/вывода для каждого из аналоговых входов и выходов.

– Зеленый указывает, что канал сконфигурирован и активен.

– Красный указывает на состояние ошибки отдельного входа или выхода.

Кроме того, каждый аналоговый SM имеет светодиод DIAG, который указывает состояние модуля:

– Зеленый указывает, что модуль готов к работе.

– Красный указывает, что модуль неисправен или не готов к работе.

Таблицы наблюдения для контроля программы пользователя

Таблица наблюдений позволяет осуществлять функции контроля и управления в информационных точках, когда ЦПУ выполняет вашу программу (рис. 24). Этими информационными точками могут быть элементы образа процесса (I или Q), физические входы или выходы (I_:P или Q_:P), M или DB в зависимости от функции контроля и управления.

Функция контроля не изменяет процесс исполнения вашей программы. Она снабжает вас информацией об исполнении программы и данных программы в ЦПУ. Функции управления позволяют пользователю управлять последовательностью исполнения и данными программы. При использовании функций

управления следует соблюдать осторожность. Эти функции могут существенно влиять на исполнение пользовательской или системной программы. Этими тремя функциями являются изменение, принудительное задание и разблокирование выходов в состоянии STOP.

С помощью таблицы наблюдения вы можете выполнять следующие онлайновые функции:

– контроль состояния переменных;

– изменение значений отдельных переменных;

– принудительное присваивание переменной определенного значения.

Для создания таблицы наблюдения:

1. Дважды щелкните на "Add new watch table [Добавить новую таблицу наблюдения]", чтобы открыть новую таблицу наблюдения.

2. Введите имя переменной, чтобы добавить переменную в таблицу наблюдения.

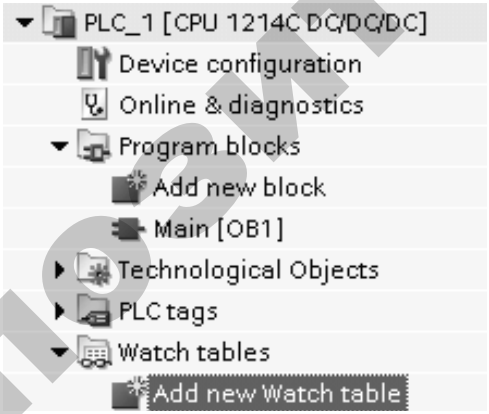


Рис. 24. Добавление в проект таблицы наблюдения

Для контроля переменных имеются следующие возможности:

– Monitor all [Контролировать все]: Эта команда запускает контроль видимых переменных в активной таблице наблюдения.

– Monitor now [Контролировать теперь]: Эта команда запускает контроль видимых переменных в активной таблице наблюдения.

Таблица наблюдения выполняет контроль переменных немедленно и только один раз.

Для изменения переменных имеются в распоряжении следующие возможности:

– "Modify to 0 [Изменить на 0]" устанавливает значение выбранного адреса в "0".

– "Modify to 1 [Изменить на 1]" устанавливает значение выбранного адреса в "1".

– "Modify now [Изменить сейчас]" немедленно изменяет значение выбранных адресов на время одного цикла.

– "Modify with trigger [Инициирование изменений]" изменяет значение выбранных адресов. Эта функция не обеспечивает обратной связи, чтобы показать, что выбранные адреса были действительно изменены. Если требуется ответная реакция на изменения, используйте функцию "Modify now [Изменить сейчас]".

– "Enable peripheral outputs [Разблокировать периферийные выходы]" деактивизирует команду на блокировку выходов и имеется в распоряжении только тогда, когда ЦПУ находится в состоянии STOP.

Порядок выполнения лабораторной работы

Для закрепления полученных сведений о программных инструментах отладки и тестирования пользовательских программ ПЛК необходимо выполнить отладку и тестирование программы управления водогрейным котлом, разработанной при выполнении лабораторной работы № 2 (стр. 36–39). Для этого следует открыть сохраненный ранее на диске проект TIA Portal, созданный при выполнении лабораторной работы № 2. В открытом проекте следует добавить таблицу наблюдения, в которую, в свою очередь, необходимо добавить все используемые в проекте переменные. Загрузить программу в ПЛК, затем установить онлайн соединение с ПЛК и в режиме RUN протестировать правильность исполнения пользовательской программы путем изменения в таблице наблюдения значений всех входных переменных и просмотра значений сигналов на дискретных выходах.

Содержание отчета

1. Название и цель работы.
2. Список переменных, используемых в программе управления водогрейным котлом.
3. Графическая схема алгоритма управления водогрейным котлом.
4. Выводы.

Контрольные вопросы

1. Какие существуют режимы работы ЦПУ ПЛК?
2. Какие программные инструменты отладки и тестирования реализованы в среде разработки TIA Portal?
3. Какие возможности по тестированию и отладке программ реализуются с помощью таблиц наблюдений?

ЛАБОРАТОРНАЯ РАБОТА № 12
ОРГАНИЗАЦИЯ ЛОКАЛЬНОЙ СЕТИ И ОБМЕН ДАННЫМИ МЕЖДУ КОНТРОЛЛЕРАМИ

Цель работы: изучение методов сетевого обмена данными между контроллерами.

Информация для выполнения лабораторной работы

ЦПУ может выполнять обмен с другими ЦПУ, с устройством программирования (т. е. с компьютером), с устройствами человеко-машинного интерфейса (сенсорные панели HMI) и с устройствами, произведенными не фирмой Siemens, используя стандартные сетевые коммуникационные протоколы TCP/IP через встроенный сетевой порт PROFINET. Для этого в ПЛК Simatic S7-1200 имеется один сетевой разъем для подключения кабеля Ethernet (рис. 25).

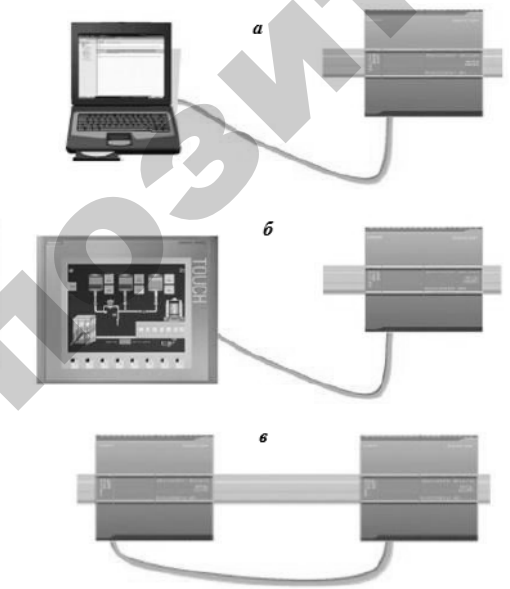


Рис. 25. Коммуникационное подключение к ЦПУ устройства программирования (а), сенсорной панели оператора (б), другого ЦПУ (в)

Для организации сети, включающей в себя более двух ЦПУ или устройств HMI, требуется дополнительный блок – Ethernet-коммутатор (рис. 26).

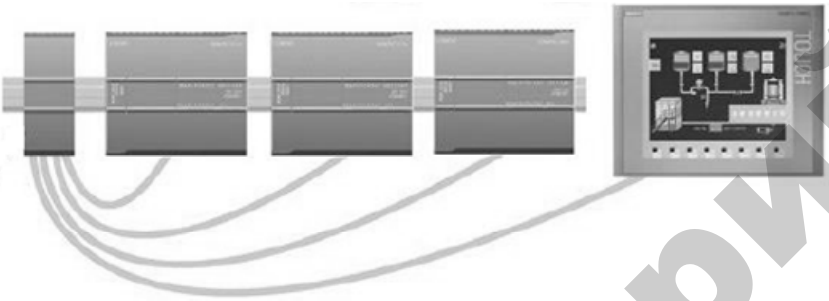


Рис. 26. Организация сети в составе нескольких ЦПУ и HMI-устройств с использованием Ethernet-коммутатора

Встроенный в ЦПУ сетевой порт PROFINET поддерживает следующие одновременные соединения для обмена данными:

– 3 соединения для обмена данными между ЦПУ и устройством HMI, т. е. к одному ЦПУ одновременно может быть подключено до 3 сенсорных панелей;

– 1 соединение для обмена данными между устройством программирования (PG) и ЦПУ;

– 8 активных коммуникационных соединений ЦПУ с другими устройствами с помощью команд TSEND_C и TRCV_C;

– 3 соединения типа S7 connection для пассивного ЦПУ S7-1200, обменивающегося данными с активным ЦПУ из серии моделей S7 с помощью команд GET и PUT.

Последовательность настройки параметров связи между задействованными в проекте ЦПУ, сенсорными панелями HMI и компьютером как устройством программирования включает следующие шаги:

1. Создание аппаратного коммуникационного соединения – заключается в физическом соединении устройств кабелем Ethernet в соответствии с рис. 25, 26.

2. Назначение устройствам соответствующих IP-адресов.

3. Конфигурирование логических сетевых соединений между двумя CPU.

4. Конфигурирование параметров передачи и приема.

Назначение IP-адресов

Каждому устройству производителем для идентификации назначается MAC-адрес (Media Access Control address [адрес протокола управления доступом к передающей среде]). MAC-адрес состоит из шести групп по две шестнадцатиричных цифры, разделенных дефисами (-) или двоеточиями (:) в порядке передачи (например, 01-23-45-67-89-AB или 01:23:45:67:89:AB).

Каждое устройство должно также иметь адрес протокола Интернет (Internet Protocol, IP) – IP-адрес. Этот адрес позволяет устройству поставлять данные через более сложную сеть с маршрутизацией. Каждый IP-адрес делится на четыре 8-битовых сегмента и представляется в десятичном формате с разделительными точками (например, 211.154.184.16). Первая часть IP-адреса используется для идентификатора, а вторая часть адреса является идентификатором хоста (уникальным для каждого устройства в сети).

IP-адрес 192.168.x.y является стандартным обозначением, которое распознается как часть ведомственной или частной *сети, которая находится вне сети Интернет*.

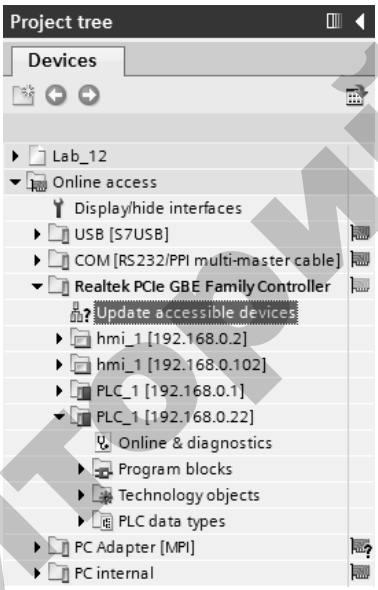
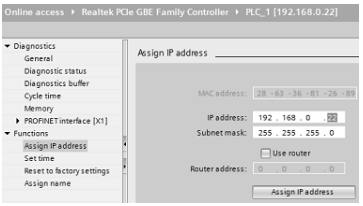
Всем задействованным в проекте и участвующим в сетевом обмене устройствам, включая и устройство программирования, должны быть назначены уникальные IP-адреса во избежание конфликта с другими пользователями сети. Для назначения IP-адреса ЦПУ или HMI панели можно использовать следующие два способа:

- назначить IP-адрес в режиме online;
- сконфигурировать IP-адрес в созданном проекте.

Для назначения IP-адреса в режиме online следует действовать пошагово в соответствии с табл. 32.

Таблица 32

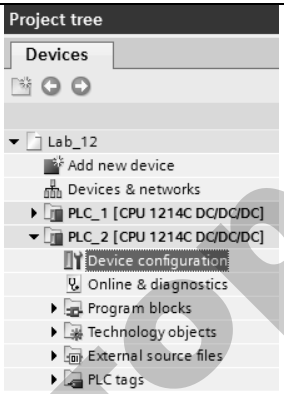
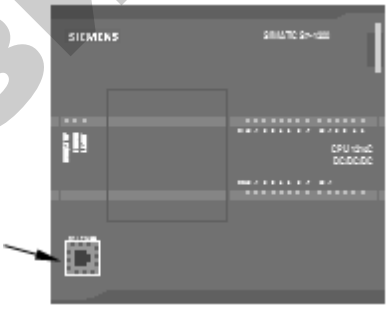
Действия по назначению IP-адреса для ЦПУ в режиме online

В дереве проекта (Project tree) выбрать команды меню • Online access (онлайновый доступ); • Realteck PCIe GBE Family Controller – название сетевой адаптерной платы устройства программирования (сетевой карты компьютера); • Update accessible devices (обновить доступные устройства) При этом должны отобразиться все устройства, доступные компьютеру по сети.	
Выбрать нужный ЦПУ и для него выбрать команду Online & diagnostics (онлайновый режим и диагностика)	
В появившемся диалоговом окне Online access выбрать команды меню • Functions (функции); • Assign IP adress (назначить IP-адрес) В поле IP adress ввести нужный IP-адрес и подтвердить ввод нажатием кнопки Assign IP adress.	
Проверить, назначен ли новый IP-адрес для ЦПУ, повторно выбрав в дереве проекта (Project tree) команды • Online access (онлайновый доступ); • Realteck – название сетевой адаптерной платы устройства программирования (компьютера); • Update accessible devices (обновить доступные устройства)	

Для назначения IP-адреса в созданном проекте следует действовать пошагово в соответствии с табл. 33, 34.

Таблица 33

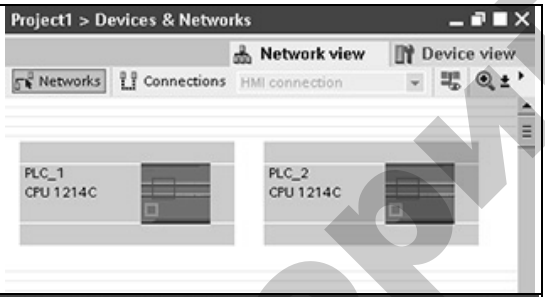
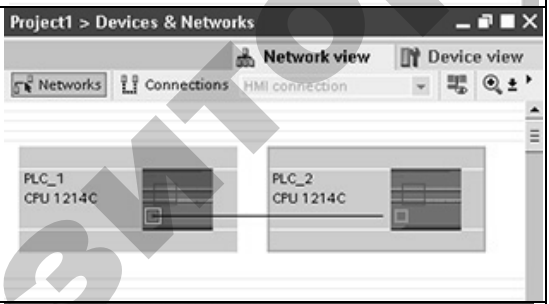
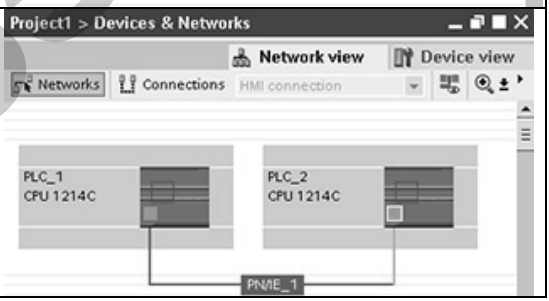
Действия по назначению IP-адреса для ЦПУ в созданном проекте

В дереве проекта (Project tree) выделить требуемый ЦПУ и для него выбрать команду Device configuration. При этом в рабочей области окна появится графическое изображение выбранного ЦПУ	
Выполнить двойной щелчок левой клавишей мыши на изображении сетевого порта PROFINET на ЦПУ. В результате в нижней части экрана в окне просмотра параметров на вкладках Properties, General отобразятся параметры сетевых настроек ЦПУ	
Выбрав компонент конфигурации Ethernet address, в диалоговом окне для конфигурирования адреса Ethernet следует ввести нужные IP-адрес в поле Set IP address in the project и маску подсети в поле Subnet mask. Данные параметры будут загружены в ЦПУ при загрузке готового откомпилированного проекта в ЦПУ	

Конфигурирование логических сетевых соединений между двумя ЦПУ

Таблица 34

Действия по конфигурированию логического сетевого соединения двух ЦПУ

Выберите "Network view [Отображение сети]" для отображения устройств, подлежащих соединению	
Выберите порт на одном устройстве и протяните линию к порту на втором устройстве	
Отпустите клавишу мыши, чтобы создать сетевое соединение	

Конфигурирование параметров передачи и приема

Для установления соединений между двумя CPU используются команды TSEND_C и TRCV_C.

Команда TSEND_C устанавливает связь с партнерской станцией и посылает ей данные. Пареметры, которые должны быть указаны для команды TSEND_C, следующие (табл. 35).

Таблица 35

Параметры команды TSEND_C

Параметр	Тип данных	Описание
REQ	Bool	Начало отправки данных – в момент положительного фронта сигнала на данном входе (изменение сигнала FALSE на TRUE) начинается и выполняется отправка данных партнерскому устройству
CONT	Bool	Контроль поддержания соединения с партнерским устройством: • 0 (FALSE) – соединение разрывается; • 1 (TRUE) – соединение поддерживается
LEN	UINT	Размер передаваемых данных в байтах
DATA	Variant	Указатель на передаваемые данные – символическое имя либо абсолютный адрес передаваемого блока данных. В случае, если в качестве параметра DATA указано символическое имя блока данных (переменной), параметр LEN должен быть равен 0.
CONNECT	TCON_Param	Указатель на сконфигурированное в проекте логическое сетевое соединение устройств и на соответствующий системный блок данных, содержащий параметры сетевого соединения
COM_RST	Bool	Сброс установленного соединения с партнерской станцией и повторное установление соединения – выполняется в момент положительного фронта сигнала на данном входе
DONE	Bool	• 0 – передача данных еще не завершена; • 1 – передача данных завершена без ошибок

Окончание таблицы 35

Параметр	Тип данных	Описание
BUSY	Bool	• 0 – передача данных еще не начиналась либо уже завершена; • 1 – передача данных выполняется в текущий момент, передача новой порции данных пока невозможна
ERROR	Bool	• 0 – нет ошибок; • 1 – ошибка при передаче
STATUS	Word	Код результата выполнения команды

Параметры REQ, CONT, DATA и CONNECT обязательно должны быть заданы. Остальные параметры могут быть заданы при необходимости.

Команда TRCV_C устанавливает связь с партнерской станцией и принимает от нее данные. Параметры, которые должны быть указаны для команды TRCV_C, следующие (табл. 36).

Таблица 36

Параметры команды TRCV_C

Параметр	Тип данных	Описание
EN_R	Bool	• 0 – прием данных не разрешен; • 1 – прием данных разрешен
CONT	Bool	Контроль поддержания соединения с партнерским устройством: • 0 – соединение разрывается; • 1 – соединение поддерживается
LEN	UINT	Размер принимаемых данных в байтах
DATA	Variant	Адрес области памяти, куда следует поместить принимаемые данные, передаваемые данные. В случае, если в качестве параметра DATA указано символическое имя блока данных (переменной), параметр LEN должен быть равен 0.

Окончание таблицы 36

Параметр	Тип данных	Описание
CONNECT	TCON_Param	Указатель на сконфигурированное в проекте логическое сетевое соединение устройств и на соответствующий системный блок данных, содержащий параметры сетевого соединения
COM_RST	Bool	Сброс установленного соединения с партнерской станцией и повторное установление соединения – выполняется в момент положительного фронта сигнала на данном входе
DONE	Bool	• 0 – передача данных еще не завершена; • 1 – передача данных завершена без ошибок
BUSY	Bool	• 0 – передача данных еще не начиналась либо уже завершена; • 1 – передача данных выполняется в текущий момент, передача новой порции данных пока невозможна
ERROR	Bool	• 0 – нет ошибок; • 1 – ошибка при передаче
STATUS	Word	Код результата выполнения команды
RCVD_LEN	UDINT	Размер (в байтах) реально принятых данных

Параметры REQ, CONT, DATA и CONNECT обязательно должны быть заданы. Остальные параметры могут быть заданы при необходимости.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления полученных сведений об организации сетевого обмена данными между контроллерами необходимо создать в среде TIA Portal

новый проект. В данном проекте следует запрограммировать сетевой обмен данными между двумя контроллерами S7-1200, посредством которого состояние сигналов на дискретных входах DI а контроллера PLC_1 (адреса входов I0.0 – I0.7) должно по сети передаваться на дискретные выходы DQ а контроллера PLC_2 (адреса Q0.0 – Q0.7).

В начале выполнения работы следует в среде TIA Portal с использованием режима online убедиться в том, что устройству программирования (компьютеру) доступны по сети минимум два контролера. Этим контроллерам необходимо в режиме online назначить IP-адреса. Для назначения IP-адреса в режиме online следует действовать пошагово в соответствии с табл. 32.

Далее в среде TIA Portal следует создать новой проект и добавить в него два контроллера S7-1200 (см. стр. 19). В проекте контроллерам также необходимо назначить эти же IP-адреса, действуя в соответствии с табл. 33**Ошибка! Источник ссылки не найден.**.

Далее нужно определиться, в какие моменты времени или с какой частотой контроллер PLC_1 будет передавать данные. Для этого в дереве проекта (Project tree) нужно выделить PLC_1, для него выбрать команду Device configuration, сделать двойной клик мышью на изображении контроллера в рабочем окне. На вкладке свойств Properties в нижней части экрана следует выбрать пункт General→System and clock memory и активировать (поставить галочку) пункт Enable the use of clock memory byte. При этом активируется clock memory – байт так называемой часовой памяти. Каждый бит этого байта изменяет свое состояние с FALSE на TRUE и обратно периодически в виде импульсов со значением скважности импульсов равным 0.5, т. е. половину периода бит равен FALSE, другую половину – TRUE. Частота импульсов всех восьми битов различна и изменяется в пределах от 0.5 Гц до 10 Гц. За всем этим следит операционная система ЦПУ. Таким образом, биты часовой памяти могут использоваться в качестве опорного тактового сигнала, инициирующего периодическую отправку данных по сети от PLC_1 к PLC_2.

После этого необходимо сконфигурировать логическое сетевое соединение между двумя контроллерами, действуя в соответствии с табл. 34.

Следующим шагом будет создание в памяти контроллеров символьных переменных, определяющих адрес пересылаемого блока данных в памяти контроллера – отправителя PLC_1 и адрес блока, куда данные будут помещаться в память контроллера – получателя PLC_2. Переменные, которые необходимо создать, показаны в табл. 37.

Таблица 37

Символьные имена используемых в проекте переменных контроллера

ПЛК	Name	Data Type	Logical Address
PLC_1 – отправитель	inputs	Byte	%IB0
PLC_2 – получатель	outputs	Byte	%QB0

Переменная inputs должна быть создана в таблице тегов контроллера PLC_1 (Project tree→PLC_1→PLC tags→Default tag table), а переменная outputs – в таблице тегов контроллера PLC_2 (Project tree→PLC_2→PLC tags→Default tag table). Таким образом, переменная inputs объединяет в себе 8 дискретных входов DI а контроллера PLC_1 с адресами %I0.0 – %I0.7, а переменная outputs объединяет в себе восемь дискретных выходов DQ а контроллера PLC_2 с адресами %Q0.0 – %Q0.7. Очевидное преимущество такого подхода состоит в том, что вместо пересылки каждого из восьми битов по отдельности удобнее переслать их все вместе одним блоком типа Byte.

Далее следует использовать команды TSEND_C, TRCV_C в программных блоках Main, соответственно, контроллеров PLC_1 и PLC_2.

Для вставки команды TSEND_C в блок Main контроллера PLC_1 следует перейти на панель инструкций Instructions, открыть вкладку Instructions→Communication→Open user communications и перетащить мышью значок TSEND_C в начало блока Main. При

этом среда Simatic Step 7 автоматически предложит создать новый системный блок данных, связанный с командой TSEND_C, и в появившемся диалоговом окне ″Call options″ предложит ввести название блока данных. В данном случае необходимо согласиться с предлагаемым по умолчанию вариантом, нажать OK. В результате этого в рабочем окне ввода программы появится следующий код:

```
"TSEND_C_DB"(REQ:=false,
             CONT:=false,
             LEN:=0,
             DONE=>_bool_out_,
             BUSY=>_bool_out_,
             ERROR=>_bool_out_,
             STATUS=>_word_out_,
             CONNECT:=_struct_inout_,
             DATA:=_variant_inout_,
             COM_RST:=_bool_inout_);
```

Для настройки и ввода всех необходимых параметров команды следует выполнить щелчок левой клавишей мыши на названии команды “TSENC_C_DB” и в появившемся маленьком всплывающем меню выбрать команду Start configuration. При этом в нижней части экрана активируется панель Configuration, реализующая интерфейс настройки параметров. Необходимые для заполнения поля будут выделены розовым цветом. На этой панели на вкладке Connection parameter нужно:

− в поле Partner (устройство-партнер) выбрать PLC_2;

− в поле Connection data для устройства PLC_1 (Local) выбрать new, при этом система автоматически создаст системный блок данных PLC_1_Send_DB, содержащий все параметры сконфигурированного ранее сетевого соединения между контроллерами: наименование протокола передачи данных (по умолчанию – TCP), IP-адрес получателя и др.;

− в поле Connection ID (dec) ввести номер соединения – 1;

− повторно в поле Partner указать PLC_2 и далее в поле Connection data для PLC_2 (Partner) также выбрать new, при этом

система создаст уже в памяти контроллера PLC_2 аналогичный системный блок PLC_2_Receive_DB, также содержащий параметры сетевого соединения;

на вкладке Block parameter:

− в поле Start request (REQ) ввести “Clock_1Hz” – символьное имя бита часовой памяти, реализующего тактовые импульсы с частотой 1 Гц (можно выбрать бит с другим значением частоты);

− в поле CONT – ввести 1;

− в поле Send area (DATA), Start указать адрес пересылаемых данных – “inputs”.

Далее аналогичным образом нужно вставить команду TRCV_C в блок Main контроллера PLC_2. Для настройки параметров команды следует выполнить щелчок левой клавишей мыши на названии команды “TRCV_C_DB” и выбрать Start configuration. При этом в нижней части экрана активируется панель Configuration, реализующая интерфейс настройки параметров. На панели Configuration нужно

на вкладке Connection parameter:

− в поле Partner (устройство-партнер) выбрать PLC_1;

− в поле Connection data для устройства PLC_1 (Local) выбрать блок PLC_2_Receive_DB;

на вкладке Block parameter:

− в поле Enable request (EN_R) ввести TRUE;

− в поле CONT – ввести 1;

− в поле Receive area (DATA), Start указать адрес памяти, куда будет помещен принимаемый блок данных – “outputs”.

Сохранить проект.

Загрузка сохраненного проекта производится отдельно в память каждого из контроллеров. Для этого в дереве проекта следует выбрать вкладку Devices & networks. В появившемся окне Devices & networks будет схематическое изображение соединенных сетевым соединением контроллеров PLC_1 и ЗДС_2. Для загрузки проекта в память контроллера необходимо выделить мышью схематическое изображение контроллера и выбрать команду главного меню Online→Download to device. Затем необходимо проделать то же самое для другого контроллера.

Загрузить созданный проект в контроллеры и убедиться в правильности работы программы.

Содержание отчета

1. Название и цель работы.
2. Принципиальная электрическая схема сетевого соединения контроллеров.
3. Список используемых в проекте значений параметров команд TSEND_C, TRCV_C.
4. Выводы.

Контрольные вопросы

1. Какой сетевой интерфейс используется в выполненной лабораторной работе для обмена данными между контроллерами?
2. Какова последовательность действий по настройке логического сетевого соединения между двумя ЦПУ?

ЛАБОРАТОРНАЯ РАБОТА № 13
РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ПРОЦЕССОМ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ПИД-РЕГУЛЯТОРА

Цель работы: приобретение навыков использования программной реализации ПИД-регулятора в программе ПЛК.

Информация для выполнения лабораторной работы

В среде программирования Step 7 имеются в распоряжении программиста следующие компоненты, представляющие собой программную реализацию ПИД-регулятора:

– инструкция PID_Compact используется в качестве стандартного непрерывного ПИД-регулятора;

– инструкция PID_3Step – используется для регулирования объектов, в состав которых входят электродвигатели, с точки зрения динамических характеристик представляющие собой интегрирующее звено.

Важным моментом при использовании инструкций ПИД-регулирования в программе является то, что эти команды должны выполняться через равные и постоянные промежутки времени. Поэтому обычно команды ПИД-регулирования помещают внутри блока циклического прерывания (Cyclic interrupt OB). Организационный блок циклического прерывания вызывается операционной системой контроллера через строго равные интервалы времени. По этой же причине команды ПИД-регулирования нельзя размещать в обычном циклическом организационном блоке, так как время прогона программы не является постоянным значением.

Алгоритм ПИД-регулирования

Значение выходного сигнала ПИД-регулятора включает три компонента:

– пропорциональная составляющая (P) – выходное значение данной составляющей сигнала пропорционально разности между

заданным значением (Setpoint) и реальным значением (Input) управляемой величины;

– интегральная составляющая (I) – выходное значение данной составляющей сигнала пропорционально интегралу по времени от разности между заданным значением (Setpoint) и реальным значением (Input) управляемой величины;

– дифференциальная составляющая (D) – выходное значение пропорционально скорости изменения во времени разности между заданным значением (Setpoint) и реальным значением (Input) управляемой величины.

Таким образом, инструкция PID_Compact работает в соответствии со следующей формулой:

$$y = K_{\mathrm{P}} \cdot \left(\left(b \cdot w - x\right) + \frac{1}{T_{\mathrm{I}} \cdot s} \cdot \left(w - x\right) + \frac{T_{\mathrm{D}} \cdot s}{a \cdot T_{\mathrm{D}} \cdot s + 1} \cdot \left(c \cdot w - x\right) \right),$$

где y – выходной сигнал;

w – заданное значение (Setpoint) управляемой величины;

K_{P} – коэффициент усиления пропорциональной составляющей выходного сигнала;

T_{I} – постоянная времени интегрирования, с;

T_{D} – постоянная времени дифференцирования, с;

x – входной сигнал;

s – комплексный оператор Лапласа;

a – коэффициент неидеальности дифференциального звена;

b – весовой коэффициент пропорциональной составляющей;

c – весовой коэффициент дифференциальной составляющей.

Математическая формула, описывающая работу компонента PID_3Step, имеет вид:

$$y = K_{\mathrm{P}} \cdot s \cdot \left(\left(b \cdot w - x\right) + \frac{1}{T_{\mathrm{I}} \cdot s} \cdot \left(w - x\right) + \frac{T_{\mathrm{D}} \cdot s}{a \cdot T_{\mathrm{D}} \cdot s + 1} \cdot \left(c \cdot w - x\right) \right) \cdot$$

Вставка команд PID_Compact и PID_3Step выполняется путем перетаскивания мышью со вкладки

Instructions→Technology objects. После этого в окне инспектора свойств объекта следует задать численные значения коэффициентов блока ПИД-регулятора.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для закрепления полученных сведений об использовании программной реализации ПИД-регулятора в программе пользователя создать в среде TIA Portal новый проект.

В данном проекте нужно добавить контроллер Simatic S7-1200 (см. стр. 19). В состав пользовательской программы контроллера следует добавить организационный блок циклического прерывания Cyclic interruptd OB, в котором поместить команду ПИД-регулирования PID_Compact. В инспекторе свойств блока PID_Compact ввести указанные преподавателем числовые значения коэффициентов усиления и постоянных времени. В качестве задающего (Setpoint) и входного (Input) сигналов указать сигналы с аналоговых входов контроллера. Загрузить программу в контроллер и исследовать с помощью имеющегося на лабораторном стенде цифрового вольтметра изменение выходного сигнала ПИД-регулятора в зависимости от изменений входных сигналов.

Содержание отчета

1. Название и цель работы.
2. Принципиальная электрическая схема подключения аналогового датчика к контроллеру.
3. Выводы.

Контрольные вопросы

1. В чем состоит физический смысл действия ПИД-регулятора?
2. С какой целью команды ПИД-регулирования следует помещать в организационном блоке циклического прерывания?

ЛАБОРАТОРНАЯ РАБОТА № 14
ОРГАНИЗАЦИЯ ОБМЕНА ДАННЫМИ С SCADA СИСТЕМОЙ НА ОСНОВЕ ПРОТОКОЛА OPC

Цель работы: знакомство с сенсорными панелями фирмы Siemens и приобретение начальных навыков их программирования.

Информация для выполнения лабораторной работы

SIMATIC WinCC является модульной масштабируемой системой визуализации процесса (SCADA-системой) для приложений различного уровня. WinCC обеспечивает полный набор функциональных возможностей для визуализации и управления процессом. Кроме того, WinCC предоставляет в распоряжение пользователя ряд редакторов и интерфейсов, которые можно использовать для индивидуального определения функциональных возможностей пользовательского приложения. Поддерживаемые инструментальные средства проектирования включают в себя средства создания кадров (Graphics Designer [Графический дизайнер]), редакторы для конфигурирования системы сообщений (Alarm Logging [Регистрация аварийных сообщений]) и системы архивирования значений процесса (Tag Logging [Регистрация тегов]), систему формирования отчетов (Report Designer [Дизайнер отчетов]), создание сценариев (Global Script [Глобальный сценарий]), систему управления пользователями (User Administrator [Администратор пользователей]), определение связей для обмена данными (Tag Management [Управление тегами]).

Графический редактор WinCC – WinCC Graphics Designer [Графический дизайнер WinCC] – является векторно-ориентированной программой для рисования. В редакторе имеются функции для точного позиционирования и выравнивания, вращения и зеркального отражения, переноса свойств графических объектов, а так же возможности группировки, создания библиотечных объектов и импорт или встраивание текстов и графики, отредактированных внешним редактором с

использованием различных форматов (BMP, GIF, JPG, WMF, EMF) или с помощью технологии OLE.

Сообщения предоставляют оператору информацию о рабочих и аварийных состояниях технологического процесса. Сообщения информируют оператора о критических ситуациях на самых ранних стадиях их возникновения, поэтому позволяют сократить время простоя. В процессе проектирования определяются события процесса, которые будут инициировать появление соответствующих сообщений. Например, событием может быть установка определенного бита в контроллере системы автоматизации или превышение предельных значений параметров (выход за уставку значения параметра).

Система сообщений состоит из компонента проектирования и компонента исполнения.

Компонентом проектирования системы сообщений является редактор Alarm Logging [Регистрация аварийных сообщений]. В редакторе Alarm Logging [Регистрация аварийных сообщений] определяются события, при которых появляются соответствующие сообщения, а также содержание сообщений. В графическом редакторе Graphic Designer [Графический дизайнер] есть специальный графический объект WinCC Alarm Control [Окно отображения сообщений WinCC], в котором могут отображаться сообщения, сконфигурированные в редакторе Alarm Logging [Регистрация аварийных сообщений].

Компонентом исполнения системы сообщений является Alarm Logging Runtime [Система исполнения регистрации аварийных сообщений]. При исполнении проекта Alarm Logging Runtime [Система исполнения регистрации аварийных сообщений] отвечает за выполнение определенных задач контроля. Система исполнения также осуществляет управление операциями вывода сообщений и квитирования отображаемых сообщений.

Система архивирования значений процесса конфигурируется в редакторе Tag Logging [Регистрация тегов]. Здесь определяется, когда и какие значения процесса должны быть заархивированы. В WinCC для архивирования значений процесса можно создавать архив значений процесса (англ. process value archive) и вторичный архив (англ. Compressed archive).

В редакторе Tag Logging [Регистрация тегов] конфигурируются архивы, определяются циклы сбора и архивирования и значения процесса, которые будут архивироваться. Кроме того, в Tag Logging [Регистрации тегов] конфигурируются буферы данных на жестком диске и параметры выгрузки значений процесса. Для запуска Tag Logging [Регистрации тегов], так же как любого другого редактора WinCC, необходимо дважды щелкнуть кнопкой мыши на имени редактора в WinCC Explorer [Проводнике WinCC].

В режиме исполнения требуется непрерывное обновление значений процесса. Логическое соединение определяет и предоставляет WinCC информацию о том, из памяти какого контроллера должны считываться значения тегов процесса, и какой канал используется для управления потоком данных. Значения процесса передаются по этому каналу и записываются в рабочую память сервера WinCC. Управление обменом, выполняемое каналом связи, позволяет оптимизировать необходимые этапы обмена данными таким образом, что поток данных через соответствующую шину сокращен до минимума.

Теги WinCC позволяют обращаться к определенным данным системы автоматизации (англ. AS). Теги, для определения значений которых необходима связь с контроллером, называют внешними тегами. Теги, не имеющие связи с контроллером, называются внутренними тегами.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы для приобретения начальных навыков программирования графического интерфейса пользователя на основе сенсорной панели оператора необходимо создать в среде TIA Portal новый проект.

В данном проекте нужно добавить контроллер Simatic S7-1200 (см. стр. 19) и сенсорную панель, установив между ними сетевое соединение. В качестве задачи необходимо реализовать управление состоянием дискретных выходов контроллера с помощью графических элементов типа Button (кнопка) на сенсорной панели.

Содержание отчета

1. Название и цель работы.
2. Принципиальная электрическая схема сетевого соединения контроллера и сенсорной панели.
3. Выводы.

Контрольные вопросы

1. Что такое WinCC?
2. Какие основные инструментальные средства проектирования входят в состав WinCC?

ЛАБОРАТОРНАЯ РАБОТА № 15
МИКРОПРОЦЕССОРНАЯ СИСТЕМА УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ПРОЦЕССОМ ПРИГОТОВЛЕНИЯ ЖИДКИХ КОРМОВ НА СВИНОВОДЧЕСКОМ КОМПЛЕКСЕ

Цель работы: – разработка автоматизированной системы управления технологическим процессом на примере технологического процесса приготовления жидкого корма для кормления свиней.

Информация для выполнения лабораторной работы

Описание технологического процесса кормоприготовления

Жидкая кормовая смесь, влажностью 75...80 %, получается путем смешивания полнорационного комбикорма и воды в определенной пропорции. Влажность кормовой смеси определяется выражением:

$$W = \frac{\left(m_{\text{В}} + 0.14 \cdot m_{\text{К}}\right)}{\left(m_{\text{В}} + m_{\text{К}}\right)} \cdot 100\%, \qquad (1)$$

где $m_{\text{В}}$ и $m_{\text{К}}$ – массы, соответственно, воды и комбикорма, кг;

14 % – стандартная влажность комбикорма.

Разделив числитель и знаменатель выражения (1) на $m_{\text{К}}$, получим преобразованное выражение

$$W = \frac{\left(K + 0.14\right)}{\left(K + 1\right)} \cdot 100\%, \qquad (2)$$

где $K = \frac{m_{\text{В}}}{m_{\text{К}}}$ – отношение воды и комбикорма при приготовлении жидкой кормовой смеси.

Кормоприготовление на технологическом оборудовании промышленного свинокомплекса осуществляется следующим образом (рис. 27). Первоначально в смесительную ванну, установленную на

тензовесах, поступает вода. После набора заданного количества воды осуществляется подача комбикорма из бункера для хранения комбикорма, с помощью шнека-извлекателя. Управление оборудованием осуществляется микропроцессорным контроллером. Алгоритм представлен на рис. 28.

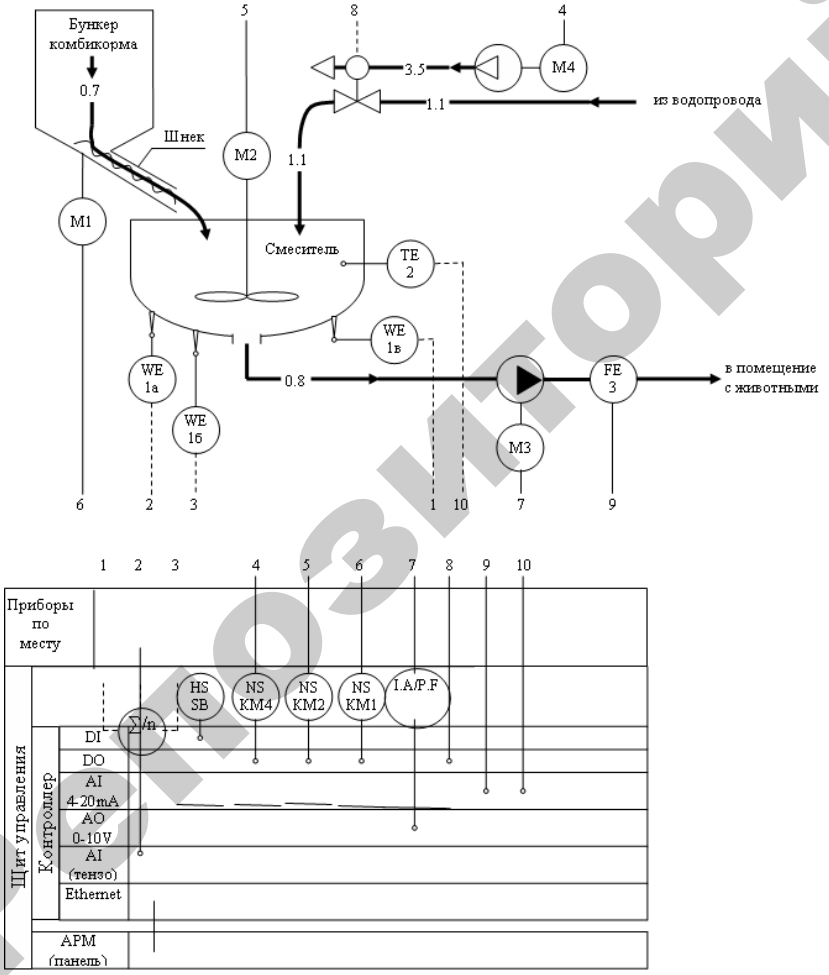


Рис. 27. Схема автоматизации приготовления жидкого корма

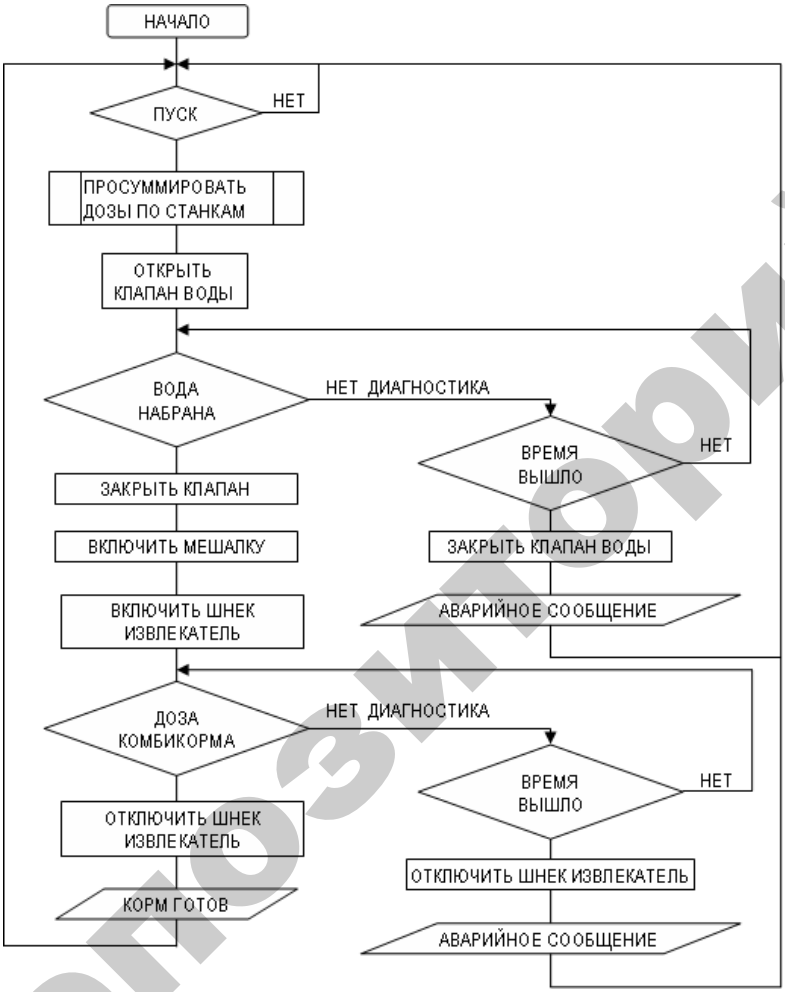


Рис. 28. Блок-схема алгоритма кормоприготовления

Устройство тензовесов

Датчики силоизмерительные тензорезисторные ДСТ Датчики силоизмерительные, с клеевыми фольговыми тензорезисторами, соединенными по мостовой схеме, предназначены для работы в силоизмерительных системах, с целью измерения статических или медленно изменяющихся усилий (сжатия и/или растяжения) (рис. 29).

Рис. 29. Монтажный чертеж тензорезистивного датчика

Изменение сопротивления является очень малым (обычно менее 0.1 Ом при номинальном сопротивлении 100 Ом) и по величине сравнимо с изменением под влиянием температуры. Поэтому обычно на нагрузочной ячейке закрепляются четыре тензодатчика, из которых два воспринимают нагрузку (деформируются), а два, расположенные под углом 90°, являются ненагруженными. Включение этих тензодатчиков в мост Уитстона (рис. 30) позволяет получить выходной сигнал, зависящий только от нагрузки и не зависящий от температуры.

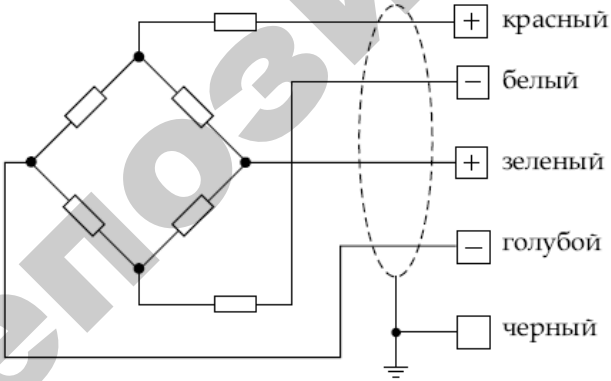


Рис. 30. Схема электрическая принципиальная тензорезистивного датчика

При подключении тензодатчиков к унифицированным аналоговым входам контроллера необходимо использовать промежуточный усилитель (рис. 31).

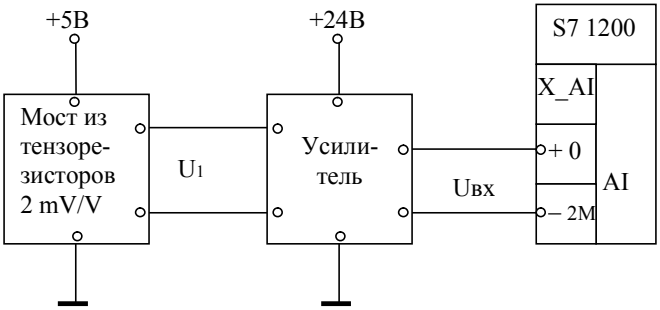


Рис. 31. Схема электрическая промежуточного усилителя

Порядок выполнения лабораторной работы

Разработать схемы подключения тензовесов, клапана воды и шнека-извлекателя к контроллеру. Тензовесы – устройство с унифицированным сигналом (см. табл. 38).

Таблица 38

Варианты параметров для разных бригад

Номер бригады	Используемый выход тензовесов	Масса корма, кг	Влажность жидкого корма, %	Время контроля, сек.
1	Аналог 0..10 В, 0..1500 кг	800	75	30
2	Аналог 0..10 В, 0..2000 кг	1000	78,6	40
3	Аналог 0..20 мА, 0..1500 кг	1000	80	25
4	Аналог 4..20 мА, 0..1000 кг	800	75	30
5	Аналог 0..10 В, 0..1500 кг	1200	81	40
6	Аналог 4..20 мА, 0..1500 кг	800	75	25

Обеспечить приготовление заданного объема жидкого корма заданной влажности в соответствии с алгоритмом (см. рис. 28).

Средства визуализации. Ввод задания, пуск и вывод текущего состояния весов осуществлять с помощью сенсорной панели.

Оформить отчет.

Содержание отчета

1. Название и цель работы.
2. Блок-схема алгоритма управления технологическим процессом кормоприготовления.
3. Выводы.

Контрольные вопросы

1. Устройство и принцип действия тензодатчика.
2. Почему в тензодатчике используются 4 тензорезистора?
3. Тип переменной аналогового входа контроллера.

ЛАБОРАТОРНАЯ РАБОТА № 16
МИКРОПРОЦЕССОРНАЯ СИСТЕМА УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ПРОЦЕССОМ РАЗДАЧИ ЖИДКИХ КОРМОВ НА СВИНОВОДЧЕСКОМ КОМПЛЕКСЕ

Цель работы: изучить технологический процесс, алгоритм управления, разработать программное и техническое обеспечение АСУ ТП раздачи жидких кормов.

Информация для выполнения лабораторной работы

Краткое описание технологического оборудования и средств автоматизации

Типовой свинарник-откормочник, рассчитанный на откорм 10.8 тыс. голов свиней в год, состоит из шести изолированных секторов. В каждом секторе находится 24 групповых станка, в которых содержится до 25 свиней. Каждый свинарник-откормочник имеет свою линию раздачи жидких кормов (рис. 32). Линия раздачи жидкого корма включает смеситель (1), на выходе которого установлен электромагнитный расходомер и центробежный насос с электроприводом. Магистральный кормопровод (2) крепится на опорах в технологическом проходе свинарника. От него в каждую кормушку станка отходит опуск с быстродействующим электропневмоклапаном. Клапан открывается и закрывается по команде с системы управления (3). Групповая кормушка (4) изготовлена из нержавеющей стали и снабжена электродным датчиком наличия жидкого корма. Управление процессом осуществляется с помощью сенсорной панели (5). Дозы корма выдаются в соответствии с количеством и массой свиней в групповом станке (6).

Рис. 32. Оборудование для раздачи жидких кормов свиньям: *1* – смеситель; *2* – элемент кормопровода с клапаном; *3* – шкаф управления; *4* – кормушка из нержавейки с опусом и датчиком наличия корма; *5* – сенсорная панель; *6* – станок с животными

Алгоритм управления процессом кормораздачи

После завершения процесса приготовления в смесителе жидкого корма в объеме, соответствующем сумме запланированных доз по групповым кормушкам, по сигналу разрешения на раздачу включается привод кормораздаточного насоса. Затем открывается первоначальный клапан и осуществляется выдача запланированной

дозы жидкого корма в групповую кормушку. Измерение расхода корма может осуществляться или встроенным в кормопровод электромагнитным расходомером, или тензовесами. После выдачи дозы клапан закрывается и открывается клапан следующего группового станка. Процесс периодически повторяется до завершения кормления.

Порядок выполнения лабораторной работы

В процессе выполнения лабораторной работы следует создать в среде TIA Portal новый проект, в котором реализовать описанный в работе алгоритм автоматического управления технологическим процессом кормораздачи. Создать средства визуализации с использованием возможностей сенсорной панели (табл. 39).

Таблица 39

Варианты параметров для разных бригад

Номер бригады		
1	4 кормушки (30, 40, 40,10)	тензовесы
2	3 кормушки (30, 40, 10)	расходомер
3	4 кормушки (30, 40, 40,10)	расходомер
4	3 кормушки (30, 40,10)	тензовесы
5	5 кормушек (30, 40, 40,0, 10)	тензовесы
6	5 кормушек (30, 40, 40,0,10)	тензовесы

Содержание отчета

1. Название и цель работы.
2. Блок-схема алгоритма управления технологическим процессом кормораздачи.
3. Выводы.

Контрольные вопросы

1. Общее понятие о промышленной технологии откорма свиней.
2. Состав технологического оборудования для обеспечения раздачи жидкого корма по групповым станкам.
3. Состав средств автоматизации, необходимых для управления процессом раздачи жидкого корма.
4. Принципы и технические средства дозирования жидкого корма.
5. Объяснить работу фрагментов программного обеспечения проекта раздачи жидких кормов.

СПИСОК ЛИТЕРАТУРЫ

Основная

1. Новиков, Ю. В. Основы микропроцессорной техники : учебное пособие / Ю. В. Новиков, П. К. Скоробогатов. – 4-е изд., испр. – М. : Интернет-Университет Информационных Технологий : БИНОМ. Лаборатория знаний, 2012. – 358 с. : ил. – (Основы информационных технологий).

2. Минаев, И. Г. Программируемые логические контроллеры: практ. руковод. для начинающего инженера / И. Г. Минаев, В. В. Самойленко. – Ставрополь : АГРУС, 2009. – 100 с.

3. Парр, Э. Программируемые контроллеры: руковод. для инженера / Э. Парр. – М : БИНОМ, 2007. – 516 с.

4. Митин, Г. Л. Системы автоматизации с использованием программируемых логических контроллеров / Г. Л. Митин, О. В. Хазанова. – М. : Изд. МГТУ «Станкин», 2005. – 136 с.

Дополнительная

5. Деменков, Н. П. Языки программирования промышленных контроллеров / Н. П. Деменков. – М.: Изд. МГТУ им. Н. Э. Баумана, 2004. – 172 с.

6. Петров, И. В. Программируемые контроллеры. Стандартные языки и инструменты / И. В. Петров. – М. : СОЛОН-Пресс, 2003. – 256 с.

7. Гируцкий, И. И. Компьютеризированные системы управления в сельском хозяйстве / И. И. Гируцкий, А. Г. Сеньков. – Минск : БГАТУ, 2014. – 212 с.

Учебное издание

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА СИСТЕМ АВТОМАТИЗАЦИИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Учебно-методическое пособие

Составители:

Гируцкий Иван Иванович,
Сеньков Андрей Григорьевич

Ответственный за выпуск *А. Г. Сеньков*
Корректор *Г. В. Анисимова*
Компьютерная верстка *Е. А. Хмельницкой*
Дизайн и оформление обложки *Д. О. Бабаковой*

Подписано в печать 29.12.2017. Формат $60\times84^{1}/_{16}$.
Бумага офсетная. Ризография.
Усл. печ. л. 7,9. Уч.-изд. л. 6,18. Тираж 40 экз. Заказ 418.

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный аграрный технический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
№ 1/359 от 09.06.2014.
№ 2/151 от 11.06.2014.
Пр-т Независимости, 99-2, 220023, Минск.