УНИВЕРСИТЕТ ИТМО

М. Я. Афанасьев, А. А. Крылова, С. А. Шорохов

ОСНОВЫ ПРОЕКТИРОВАНИЯ СИСТЕМ АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ С ПРИМЕНЕНИЕМ ПЛК

Санкт-Петербург
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

М. Я. Афанасьев, А. А. Крылова, С. А. Шорохов

ОСНОВЫ ПРОЕКТИРОВАНИЯ СИСТЕМ АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ С ПРИМЕНЕНИЕМ ПЛК

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 12.04.01, 15.04.04
в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования магистратуры

Санкт-Петербург
2020

Афанасьев М. Я., Крылова А. А., Шорохов С. А., Основы проектирования систем автоматизации технологических процессов с применением ПЛК – СПб: Университет ИТМО, 2020. – 75 с.

Рецензент(ы):
Федосов Юрий Валерьевич, кандидат технических наук, доцент факультета систем управления и робототехники, Университета ИТМО.

В пособии приведено описание лабораторных работ по тематике разработки компонентов систем автоматизации технологических процессов с применением программируемых логических контроллеров.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Оглавление

Введение

На сегодняшний день практически в любой отрасли промышленности присутствуют элементы автоматизации технологических процессов. Термин «автоматизация» в первую очередь необходимо отделить от термина «автоматика». Автоматические производства не предполагают участие человека, в то время как автоматизированные подразумевают его, как минимум, в отдельных операциях. С другой стороны, не следует путать автоматизацию и механизацию. Последняя является более широким понятием, обозначающим отказ от ручного труда в пользу использования широкой номенклатуры механических устройств. При этом автоматизация, естественно, невозможна без механизации и является ее логическим продолжением.

Автоматизация предприятий может быть полной (охватывающей все технологические процессы предпрядения) или частичной (только в рамках определенных производственных участков). Тем не менее, процесс автоматизации является комплексным мероприятием и включает в себя создание единой системы управления и мониторинга промышленным оборудованием, выбор программных и аппаратных средств, а также протокола взаимодействия всех компонентов технологического процесса.

Наиболее сложным этапом автоматизации промышленных предприятий является создание единой управляющей сети, способной объединить всё технологическое оборудование, используемое в производственном процессе. На сегодняшний день наиболее распространенным способом создания подобной сети является применение программируемых логических контроллеров (ПЛК), объединенных в общую сеть по одному из стандартных промышленных протоколов. Программируемый логический контроллер является вычислительной системой, способной решать задачи управления и мониторинга как в автономном режиме, так и в качестве компонента распределенной управляющей сети.

ПЛК является специализированной электронно-вычислительной машиной, имеющей ряд особенностей, отличающих его от других контроллеров, используемых для автоматизации. В наиболее общем смысле ПЛК является вычислительной системой специального назначения, обладающей своей собственной операционной системой с возможностью программирования и подключения к различным периферийным устройствам и коммуникационным сетям. ПЛК может быть построен на программной и аппаратной базе персональных компьютеров общего назна-

чения или микроконтроллеров. Однако нельзя путать эти вычислительные системы.

В отличие от персональных компьютеров общего назначения, ПЛК способен взаимодействовать с периферийными устройствами напрямую, то есть непосредственно формировать и принимать электрические сигналы. Последнее позволяет ПЛК работать в режимах, близких к «реальному времени». Необходимо отметить, что термины «реальное время», «мягкое реальное время» и даже «жесткое реальное время» вовсе не означают, что ПЛК выполняет программные инструкции мгновенно.

Под «реальным временем» исполнения в данном случае подразумевается то, что вычислительный процесс, реализуемый электронно-вычислительной машиной, синхронизирован с астрономическим временем. То есть, если какая-то инструкция должна быть выполнена через 20 мс, то это реальное время, через которое она будет исполнена. Очевидно, что в задачах управления будет одинаково опасно как слишком быстрое, так и слишком медленное выполнение данной инструкции.

Безусловно, такая идеальная ситуация практически трудноосуществима, поэтому и появились термины «мягкое» и «жесткое» реальное время, которые по факту просто обозначают величину допуска на каждый временной интервал, заданный в программе. Чем меньше данный допуск (дельта), тем точнее будут исполняться инструкции и тем более ответственные задачи сможет решать электронно-вычислительная машина.

Реализация реального времени на персональных компьютерах общего назначения крайне сложна. Дело в том, что подавляющее большинство существующих на сегодняшний день операционных систем являются многозадачными. При этом условно можно считать, что несколько параллельных вычислительных задач должны синхронно и параллельно утилизировать ресурсы всего одного одноядерного процессора управления.

На практике это реализуется постоянным переключением контекста, когда каждому процессу, управляемому операционной системой, выделяется некоторое количество квантов времени в соответствии с некоторыми правилами, определёнными типом и приоритетом решаемой им задачи. Таким образом, пропорция времени, выделенная на каждую задачу, определяет «загруженность» центрального процессора, которая ipso facto не может превышать 100 %.

Распределением времени занимается специальная программа-диспетчер, которая следит за назначением определённого количества машинного времени каждому процессу. Когда время, отведенное данному

конкретному процессу, истекает, диспетчер принудительно отбирает у него управление, сохраняет его текущее состояние в памяти[1], после чего передает управление следующему процессу в своем списке. При этом управление отнимается ультимативно вне зависимости от выполняемой в данный момент процедуры. Следовательно, в любой произвольный момент времени состояние вычислительной системы не является абсолютно детерминированным и определяется количеством выполняемых задач, их сложностью, частотой процессора и т. д.

В дополнение к этому необходимо отметить, что диспетчер оперирует только машинными тактами, которые, строго говоря, нельзя назвать настоящими временными метками. Тактирование центрального процессора может осуществляться от внутреннего тактового генератора, который зачастую является обычной RC-цепью с цифровым управлением. Частота данного источника может изменяться в зависимости от напряжения питания. Естественно, брать за основу только частоту центрального процессора нельзя, необходим какой-то внешний источник точного времени.

Программируемые логические контроллеры, в свою очередь, также реализуют некоторое подобие многозадачной операционной системы. Однако при программировании ПЛК именно время выполнения той или иной процедуры является определяющим. Операционная система ПЛК реализует следующие виды задач:

- Циклические – задачи, которые выполняются постоянно с момента старта системы с заданным временным интервалом. Временной интервал задается программистом.
- По событию – выполняются только в том случае, если произошло некоторое внешнее событие, которое привело к изменению внутреннего состояния ПЛК. Например, была нажата кнопка, связанная с цифровым входом ПЛК.
- Свободные (англ. freewheeling) – задачи, которые запускаются только в том случае, если центральные процессор ПЛК не занят выполнением других задач. Аналог фоновых задач в операционных системах общего назначения. Как правило, эти задачи реализуют наименее ответственные функции управляющей системы, например, протоколирование событий и графическое отображение результатов работы ПЛК.

Таким образом, работа в режиме, близкому к «реальному времени» в ПЛК, достигается за счёт правильного расчёта времени выполнения

[1] Текущим состоянием могут, например, быть значения указателя стека, счетчика команд и системных регистров.

каждой задачи, не допускающего их пересечения. Безусловно, это требует дополнительных знаний от программиста, но за счёт этого программы, написанные для ПЛК, могут стабильно управлять различными быстродействующими промышленными устройствами.

Что касается микроконтроллеров, некорректно сравнивать их с ПЛК, потому что микроконтроллеры не могут выполнять функции по управлению техническими объектами автономно. Любой микроконтроллер представляет собой систему на кристалле, к которой нужно спроектировать так называемую «обвязку»: схему питания, схему сопряжения с внешними устройствами, схему безопасности и т. д. ПЛК может быть создан на основе микроконтроллера, который вместе с электрической «обвязкой» будет выступать в роли вычислительного ядра программируемого логического контроллера.

Впрочем, одного вычислительного ядра недостаточно. Чтобы вычислительная система стала настоящим ПЛК, необходимо привести его программное обеспечение в соответствие со стандартом Международной электротехнической комиссии (МЭК[2]). МЭК 61131 является стандартом, описывающим работу программируемых логических контроллеров. До изменения в системе нумерации МЭК был известен как МЭК 1131. Части стандарта МЭК 61131 разрабатываются и сопровождаются рабочей группой 7 по программируемым системам управления подкомитета SC 65B Технического комитета TC65 МЭК.

Стандарт МЭК 61131 разделен на несколько частей.

Часть 1. Общая информация. Содержит определения терминов, которые используются в последующих частях стандарта, и описывает основные функциональные свойства и характеристики ПЛК.

Часть 2. Требования к оборудованию и тесты – устанавливает требования и связанные с ними тесты для программируемых контроллеров и их периферийных устройств. Этот стандарт предписывает:

– нормальные условия эксплуатации и требования (например, требования, связанные с климатическими условиями, транспортировкой и хранением, электрическим обслуживанием и т. д.);
– функциональные требования (источники питания, оперативная и постоянная память, цифровые и аналоговые входы/выходы);
– функциональные тесты и проверки (состояние окружающей среды, вибрации, удары, падения с высоты, ввод-вывод, силовые порты и т. д.);

[2] англ. *International Electrotechnical Commission – IEC*.

– требования электромагнитной совместимости (ЭМС) и тесты, которые должны быть реализованы программируемыми контроллерами.

Отметим, что данная часть может служить основой для оценки безопасности программируемых контроллеров в соответствии со стандартом МЭК 61508.

Часть 3. Языки программирования. Стандарт устанавливает пять языков программирования ПЛК: Techno LD, Techno FBD, Techno ST, Techno IL и Techno SFC.

Часть 4. Руководства пользователя.

Часть 5. Коммуникации. Устанавливает технологии и методы обмена данными между ПЛК.

Часть 6. Функциональная безопасность. Описывает требования безопасности, предъявляемые к ПЛК, безопасность данных и управляющих программ.

Часть 7. Управление на основе нечеткой логики (англ. *fuzzy logic*). Описывает возможности расширения возможностей автоматизированных систем за счёт знаний, представленных в формате баз лингвистических правил.

Часть 8. Руководство по применению и реализации языков программирования. Включает в себя описание функций языков программирования и используемое программное обеспечение.

Часть 9. Интерфейс цифровой передачи данных для простых датчиков и исполнительных механизмов.

Часть 10. Форматы обмена данными на основе языка XML.

Наиболее интересной с точки зрения прикладного применения является третья часть, посвященная базовой архитектуре программного обеспечения и языкам программирования управляющих программ ПЛК. В стандарте описаны три графических и два текстовых языка программирования:

– Язык релейно-контактных схем (англ. *Ladder Diagram – LD*), который представляет программу в виде графической схемы описания релейной логики (см. рис. 1).

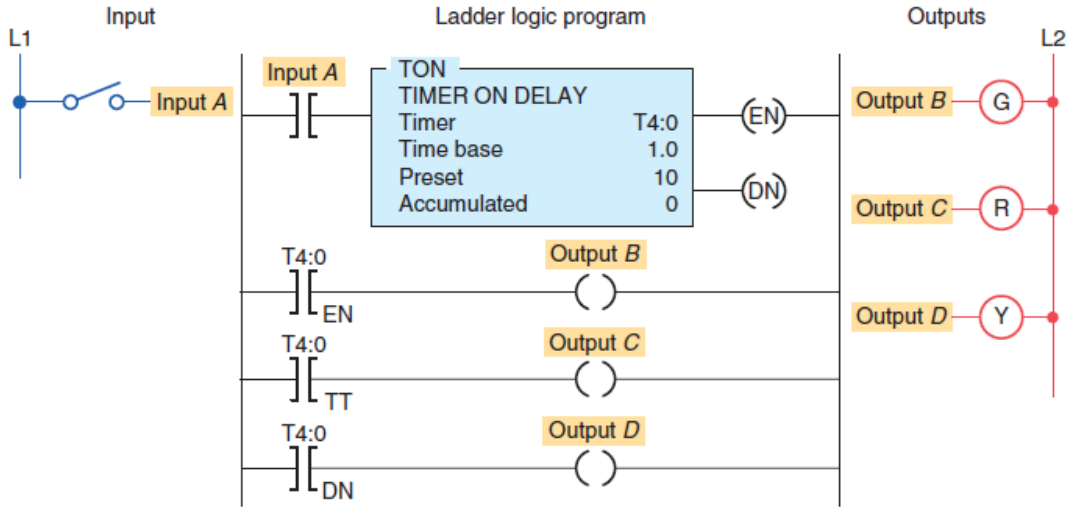


Рис. 1. Пример релейно-контактной схемы

– Функциональная блок-схема (англ. *Function Block Diagram – FBD*), представляющая собой графическое описание функции соответствия входных и выходных переменных (см. рис. 2).

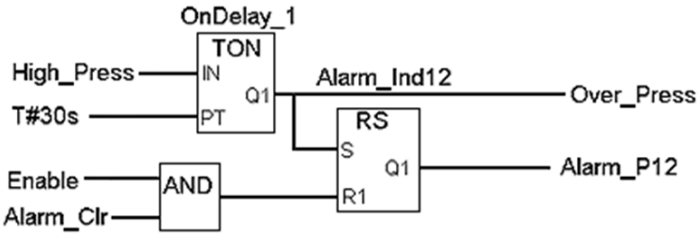


Рис. 2. Пример функциональной блок-схемы

– Структурированный текст (англ. *Structured Text – ST*) – текстовый язык программирования, синтаксически схожий с языком Pascal (см. рис. 3).
– Список инструкций (IL) – ассемблероподобный текстовый язык (исключен из стандарта третьей версии, вышедшей в феврале 2013 года).
– Последовательная функциональная диаграмма (англ. Sequential function chart – SFC), графический язык, который содержит элементы для описания как последовательных, так и параллельных процессов управления (см. рис. 4).

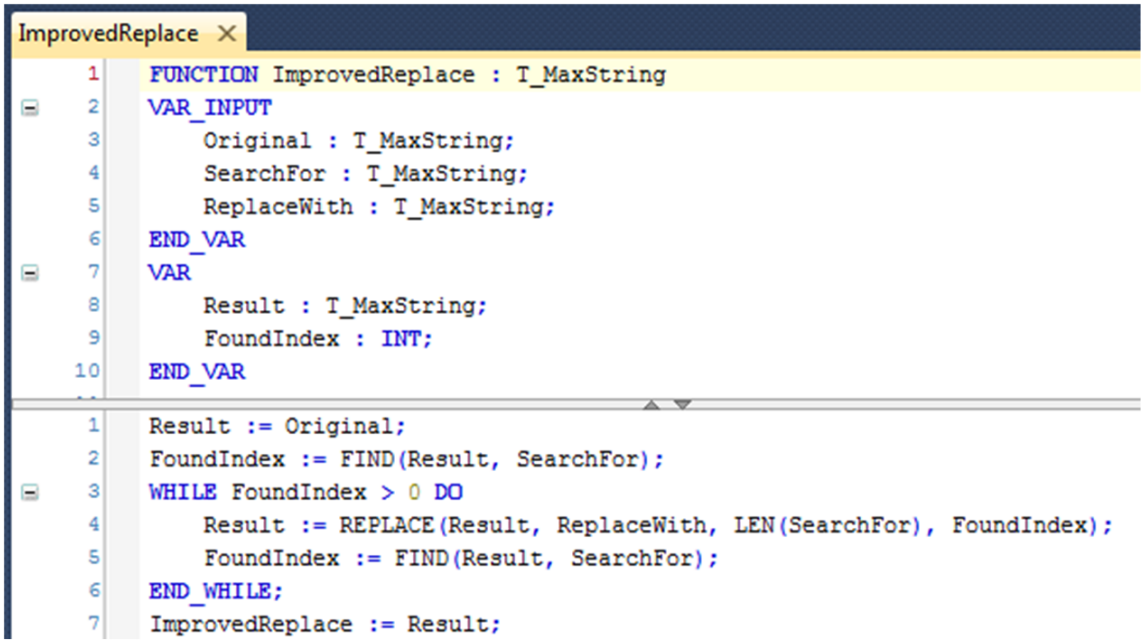


Рис. 3. Пример структурированного текста

Наличие стандарта программирования и четкого описания языков позволяет говорить о практически полной совместимости ПЛК на уровне программного кода. Программы, написанные для ПЛК одного производителя, могут быть успешно перенесены на контроллеры другого. При этом появляется возможность создания универсальных программных библиотек, которые могут реализовывать наиболее общие функции, необходимые для решения задач управления.

Как уже отмечалось ранее, работа ПЛК в наиболее эффективном режиме достигается при объединении нескольких устройств в сеть. На сегодняшний день существует большое количество специализированных программных протоколов, предназначенных для объединения ПЛК и периферийных устройств (датчиков, приводов, операторских панелей и т. д.) в промышленную сеть.

Наиболее популярными протоколами являются:

– Profinet
– Profibus (Siemens)
– EtherCat (Beckhoff)
– CAN (Bosch)
– Modbus (Schneider Electric)

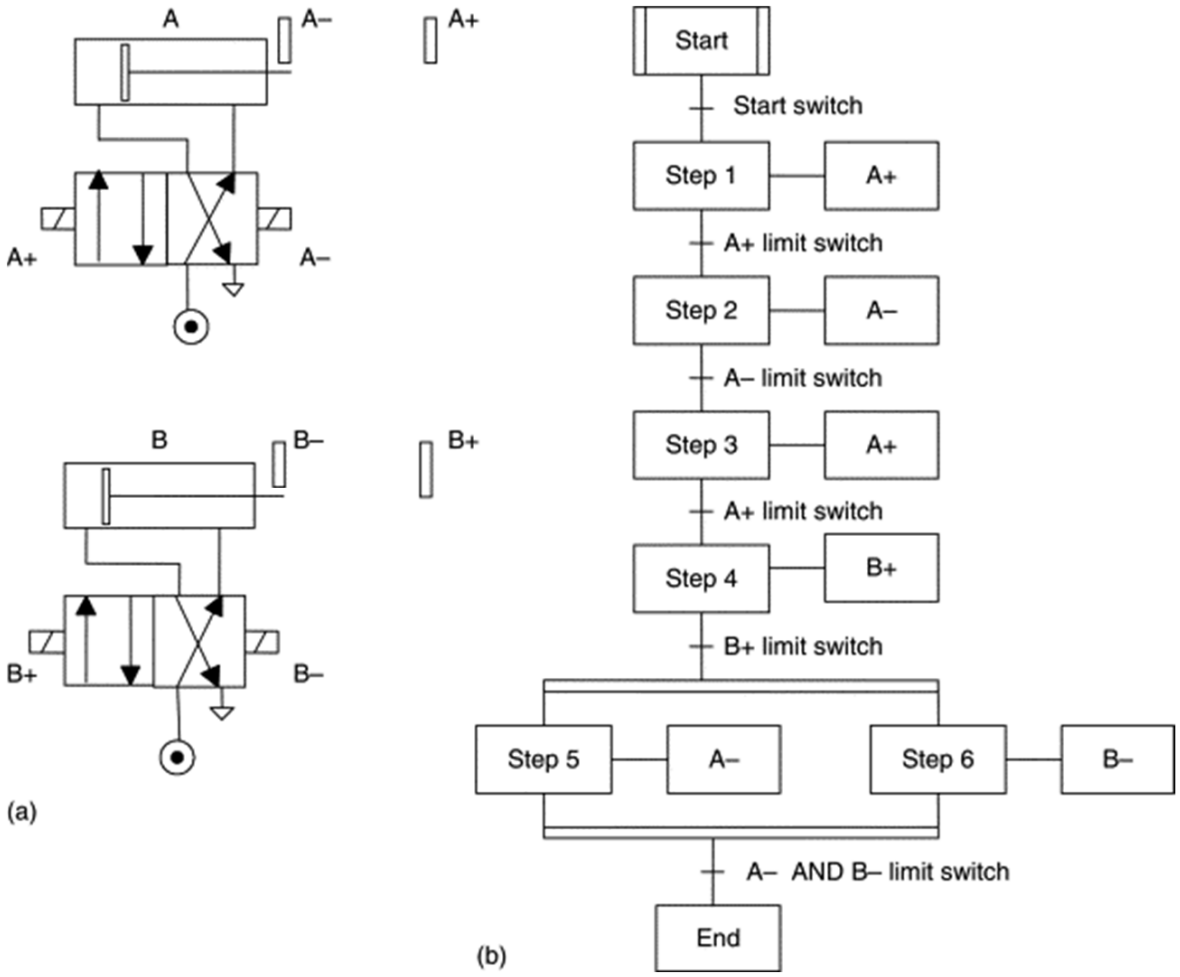


Рис. 4. Пример последовательной функциональной диаграммы

ProfiNet (Process Field Network)

ProfiNet (**Pro**cess **Fi**eld **Net**work) – открытый промышленный стандарт, разработанный ассоциацией Ethernet PROFIBUS & PROFINET International как решение для автоматизации производственного процесса. Среди задач, решаемых стандартом – широкий обмен данными между уровнями иерархии предприятия, использование IT-стандартов и поддержка проектирования в масштабе предприятия. Среди основных преимуществ стандарта – использование существующей инфраструктуры полевой сети Ethernet, включая каналы связи и сетевые компоненты, а также поддержка существующих механизмов обмена данными через сеть Ethernet в реальном времени (см. рис. 5).

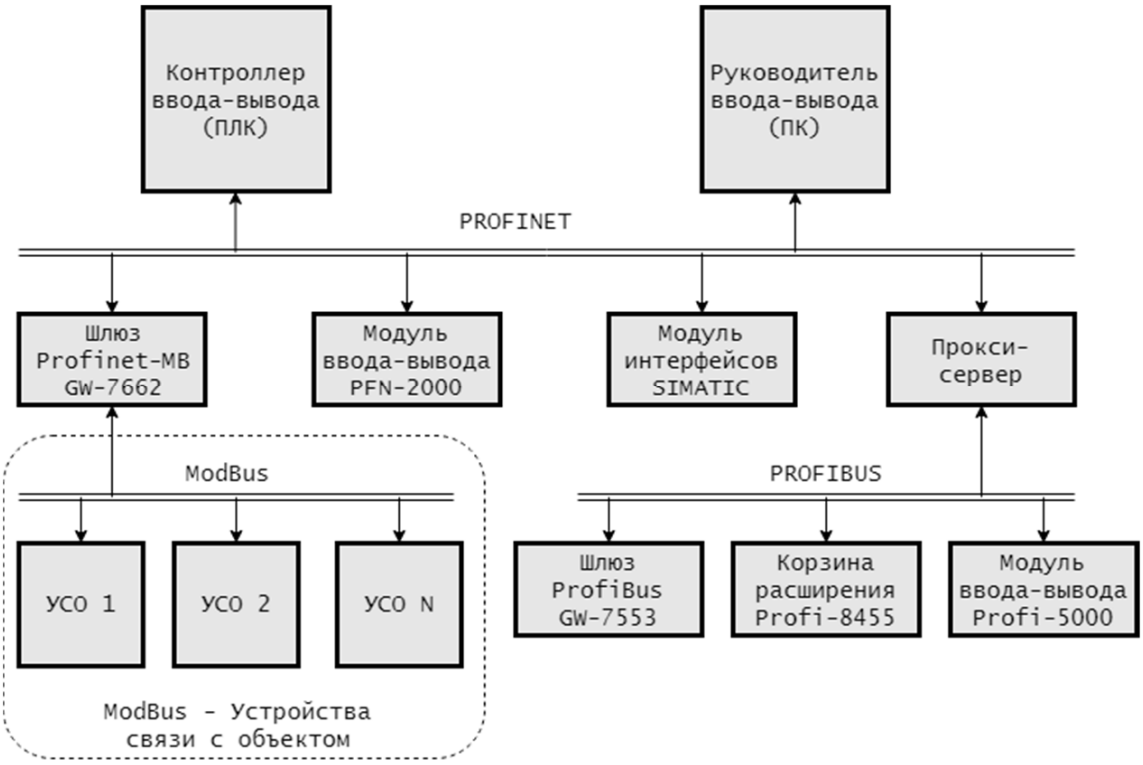


Рис. 5. Структура и состав ProfiNet

На данный момент разработано две спецификации стандарта ProfiNet: ProfiNet CBA (Component Based Automation) и ProfiNet IO. ProfiNet CBA — система, включающая в себя компоненты автоматизации, каждый из которых охватывает механические, электрические и IT-узлы и описывается файлом формата XML.Суть CBA состоит в представлении всей системы автоматизации как кластера автономных операционных подсистем с идентичными или слегка отличающимися внутренними структурами.

ProfiNet IO ориентирован на реализацию интерфейсов периферийных устройств. Он предназначен для осуществления организации быстрого обмена данными между ведущими (контроллерами) и ведомыми устройствами сети. Система включает в себя контроллер ввода-вывода, устройства ввода-вывода и руководителя ввода-вывода (ПК, используемый для настройки параметров и диагностики).

Существует три уровня производительности сети для разных типов приложений:

1. ProfiNet NRT – не поддерживает режим реального времени, поскольку цикл составляет 100 мс. Используется для задач автоматизации технологических процессов.
2. ProfiNet RT (Real-Time) – цикл 1–10 мс. Используется приложениями ProfiNet CBA и ProfiNet IO, а также задачами автоматизации предприятия.
3. ProfiNet IRT (Isochronous Real-Time) – цикл менее 1 мс. Используется для управления комплексными системами предприятия.

Таблица 1 – Характеристики ProfiNet

Параметр	**Значение**
Максимальное число устройств	255 с маской подсети 24
Максимальная скорость передачи	100 Мбит/сек, 1000 Мбит/сек
Кабель	Витая пара, оптоволокно
Максимальное расстояние, км	0,1 без повторителя

Подключение сети Profinet осуществляется при помощи модулей, выполняющих функции прокси-сервера. Монтаж сети в целом не отличается от монтажа Ethernet, возможны различные топологии сети и каналы связи. Каждый компонент сети Profinet имеет три адреса: MAC-адрес, IP-адрес и логическое имя устройства. Имеется поддержка динамической адресации через DCP и DHCP. Основные характеристики сети ProfiNet представлены в табл. 1.

Profibus (Process Field Bus)

Profibus (**Pro**cess **Fi**eld **Bus**) — открытая сеть промышленного назначения, разработанная компанией Siemens для использования в промышленных контроллерах семейства Simatic. Она поддерживается стандартами IEC 61158 и IEC 61784 и имеет широкую область применения в задачах автоматизации технологических процессов и предприятий (см. рис. 6).

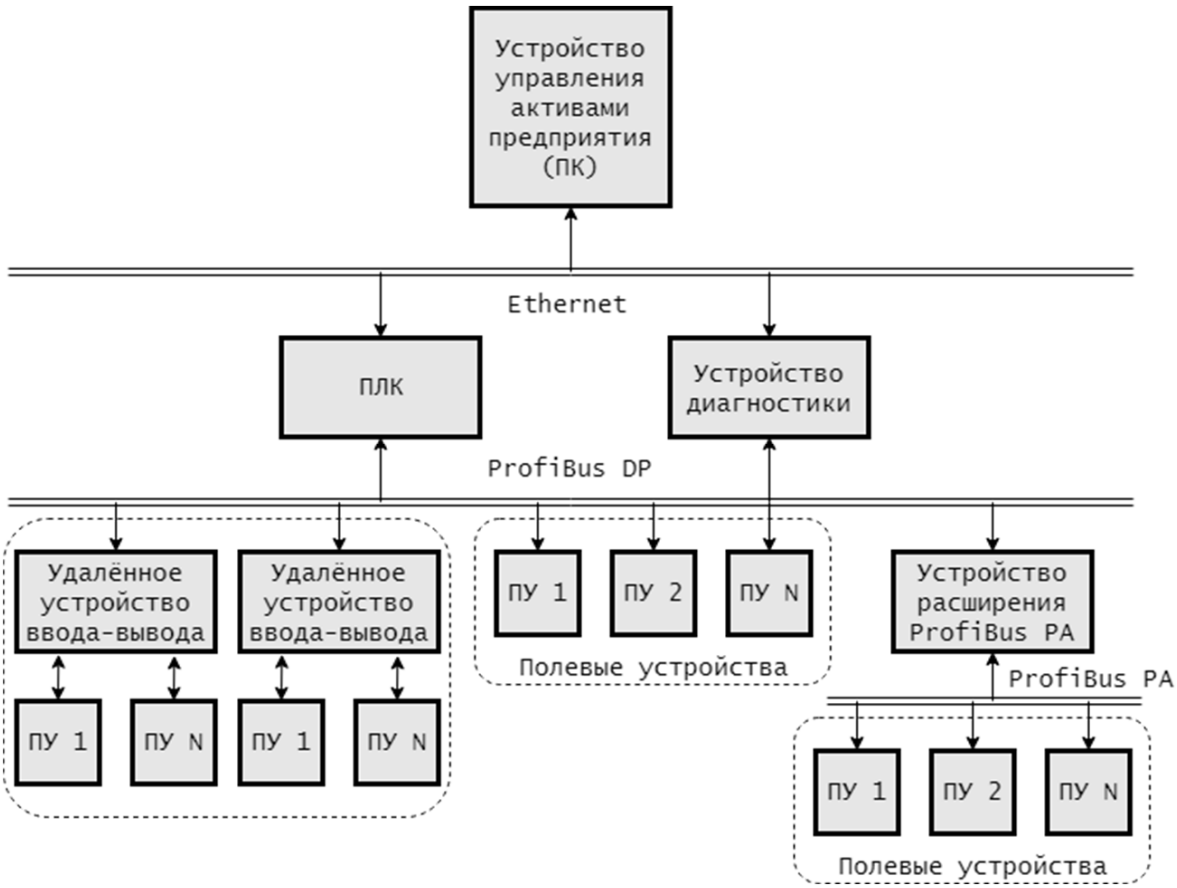


Рис. 6. Схема взаимодействия Profibus

Структура Profibus состоит из модулей, реализующих целый ряд технологий коммуникации, прикладных программ и инструментария. Сеть Profibus соответствует семиуровневой модели OSI и описывает три из них: первый, второй и седьмой.

В сети используются протоколы передачи данных RS485 и RS485-IS, поддерживается дистанционное программирование контроллеров, конфигурирование с использованием GSD-файлов. Также возможно применение технологии MBP (Manchester Coding Bus Powered) для синхронной передачи данных со скоростью 31,25 кбит/с.

Все сетевые узлы действуют в общей среде передачи данных. Доступ к шине разделён временными интервалами, используется эстафетный метод передачи данных. Активный узел получает доступ к сети на некоторый промежуток времени, в течение которого осуществляет все необходимые передачи с пассивными узлами по принципу «master-slave». Когда временное окно завершается, узел переходит в пассивное состояние, а статус активного переходит к следующему в очереди, который осуществляет свои передачи данных. Основные характеристики сети Profibus представлены в табл. 2.

Таблица 2 – Характеристики Profibus

Параметр	**Значение**
Максимальное число устройств	32 (до 254 с повторителем)
Максимальная скорость передачи	12 Мбит/сек
Кабель	Витая пара с медным экраном
Максимальное расстояние, км	1,2 (до 15, если применяется оптическое волокно)

Для передачи данных используются три протокола:

1. Profibus FMS (Fieldbus Message Specification), используется на полевом уровне для обеспечения коммуникации между «умными» сетевыми станциями.
2. Profibus DP (Decentralized Peripheral), используется для скоростного обмена данными между системами автоматизации и устройствами ввода-вывода. Поддерживается до 125 узлов, до 32 узлов на сегмент сети.
3. Profibus PA (Process Automation), используется для расширения Profibus DP, поддерживает подключение датчиков и приводов к общей линейной или кольцевой шине.

EtherCAT

Проприетарный коммуникационный протокол EtherCAT (Ethernet for Control Automation Technology) разработан компанией Beckhoff в 2003 году. Он является промышленной шиной, базирующейся на технологии Ethernet и используемой для управления различными системами

реального времени, станками и АСУ ТП. Протокол имеет высокое быстродействие за счёт высоких требований к отклику устройств (700 нс) и к технологии синхронизации.

Модель

Протокол использует модель «Master-Slave»[3]. Устройство-мастер инициирует отправление сообщения, а остальные участники изменяют его, добавляя туда данные или считывая их. Благодаря опросу всех подчиненных устройств с помощью одного сообщения достигается высокая скорость работы. MAC-адрес имеет только ведущее устройство, остальные устройства не адресуются.

Топология

Физическая топология сети EtherCAT может быть любой: звезда, дерево, шина и т. д. Однако за счёт использования полнодуплексного канала Ethernet логической топологией всегда является кольцо. Обычно устройства EtherCAT имеют два Ethernet-порта – входной и выходной. При подключении только одного порта устройство автоматически определяется как концевое и отправляет сообщение обратно в тот же порт, через который происходил приём (рис. 7 и 8).

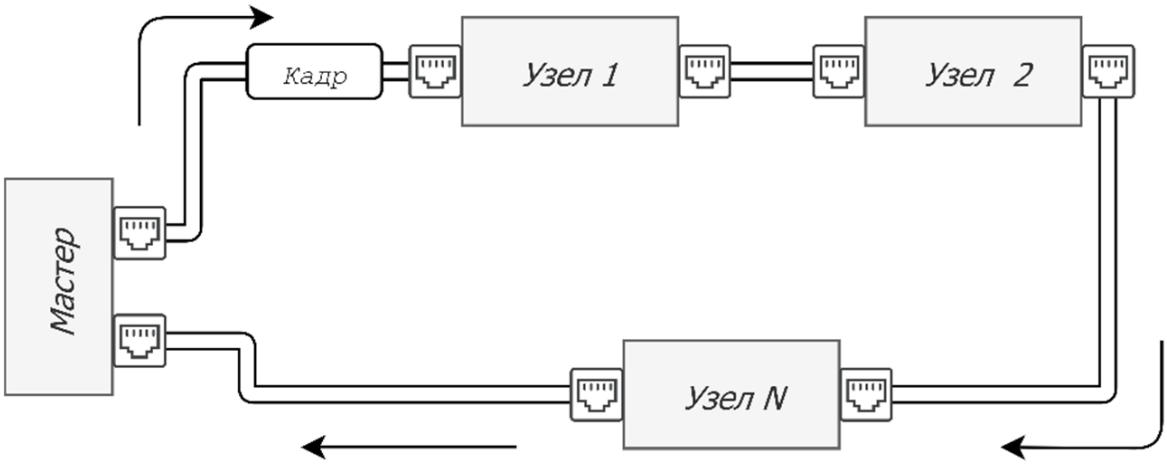


Рис. 7. Сеть EtherCAT с топологией типа «кольцо»

[3] англ. «Ведущий-Ведомый».

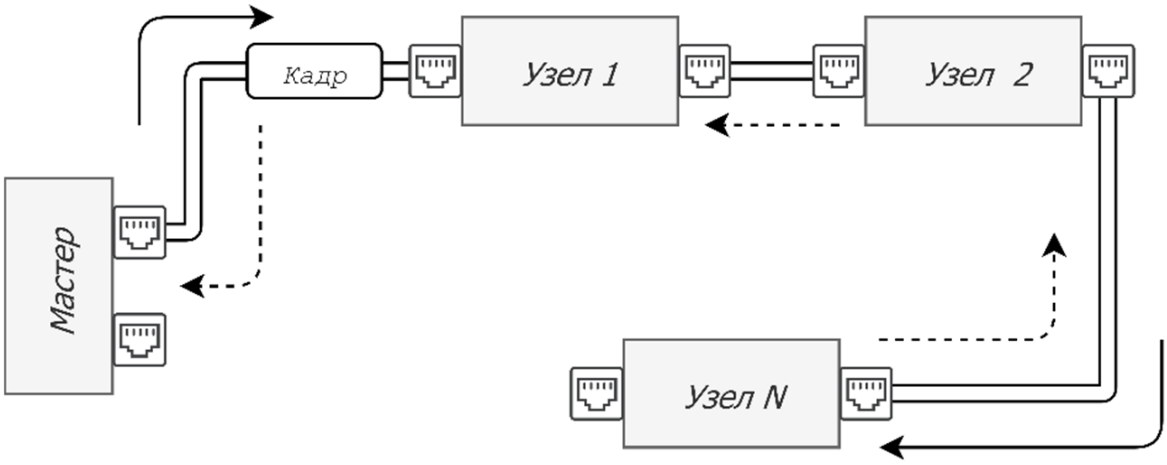


Рис. 8. Сеть EtherCAT с топологией типа «шина»

Физический уровень

На физическом уровне EtherCAT использует сети Ethernet. Для работы могут применяться уже существующие сети. Несмотря на то, что теоретически EtherCAT может работать параллельно с другими транспортными и прикладными протоколами, на практике специалисты стараются выделять отдельный сегмент. Характеристики протокола представлены в таблице 3.

Таблица 3 – Характеристики EtherCAT

Параметр	**Значение**
Максимальная скорость передачи данных	100 Мбит/с
Максимальная длина кабеля	100 м между двумя узлами
Максимальное число устройств	65535

Формат кадра

Кадр упаковывается в обыкновенный пакет Ethernet, поэтому размер его в соответствии со стандартом может варьироваться от 64 до 1522 байт, где 20 байт занимает заголовок Ethernet. Формат кадра представлен на рисунке 9.

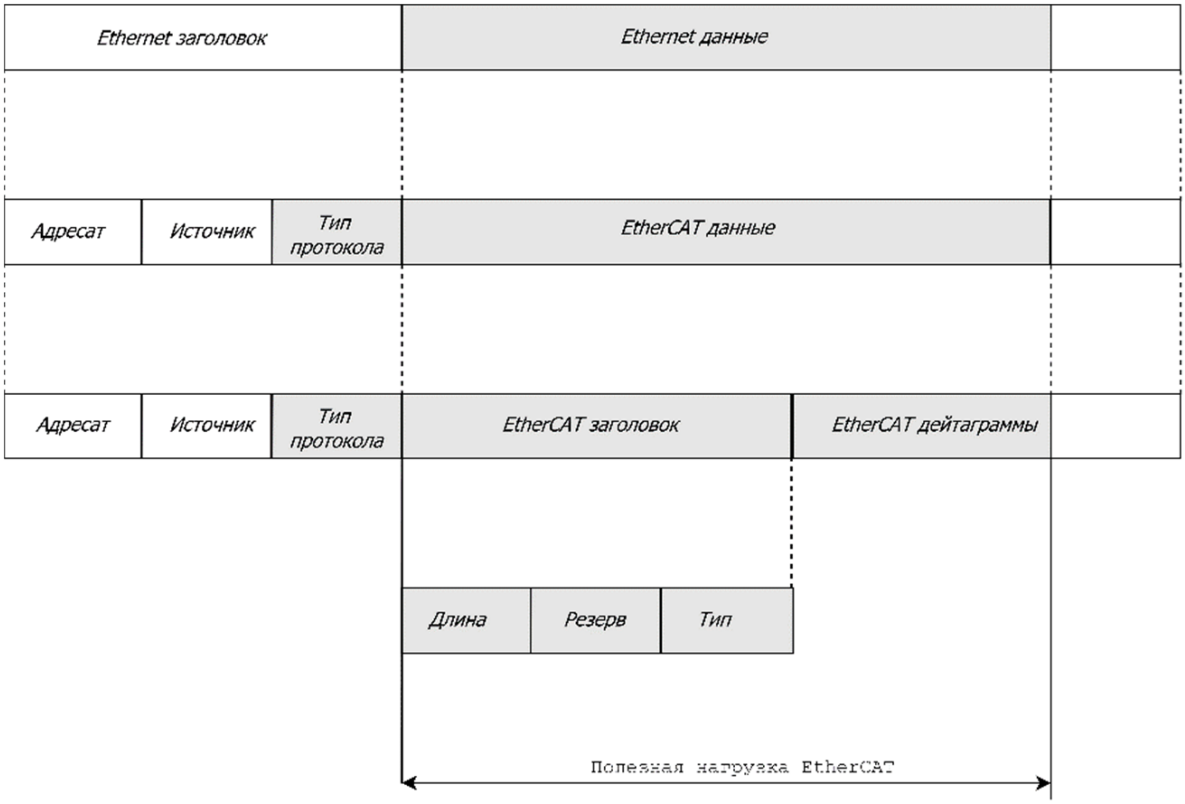


Рис. 9. Кадр EtherCAT

CAN

Коммуникационный протокол CAN (Controller Area Network) создан компанией Robert Bosch GmbH в середине 1980-х. Наибольшее применение он получил в автомобильной отрасли, а также в других областях промышленной автоматизации.

Топология

CAN сеть является сетью с общей средой передачи данных, это означает что все узлы получают все передаваемые в данный момент времени данные. Любой узел CAN представляет собой совокупность двух элементов – CAN-контроллера, который реализует протокол, и обработчика или процессора. Чаще всего CAN-сеть имеет топологию общей шины (см. рис. 10).

Для повышения надёжности на шине используются механизмы поиска и сигнализации ошибок, арбитраж, регулировка скорости, режим пониженного энергопотребления и разрешение коллизий.

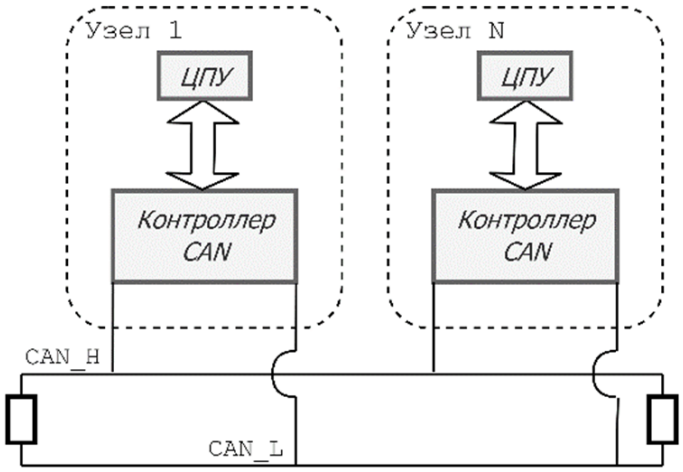


Рис. 10. Распространенная топология сети CAN

Физический уровень

Сам стандарт CAN не указывает конкретную реализацию физического уровня, однако наибольшее распространение получил стандарт ISO 11898. Согласно стандарту, шина является дифференциальной линией с импедансом 120 Ом, по краям которой устанавливаются терминальные резисторы. Дифференциальная линия состоит из двух каналов CAN HIGH и CAN LOW. Логический ноль передается, когда на канале CAN HIGH находится высокий уровень, а на канале CAN LOW — низкий. За логическую единицу принимается состояние, когда уровни обоих каналов одинаковы (см. рис. 11).

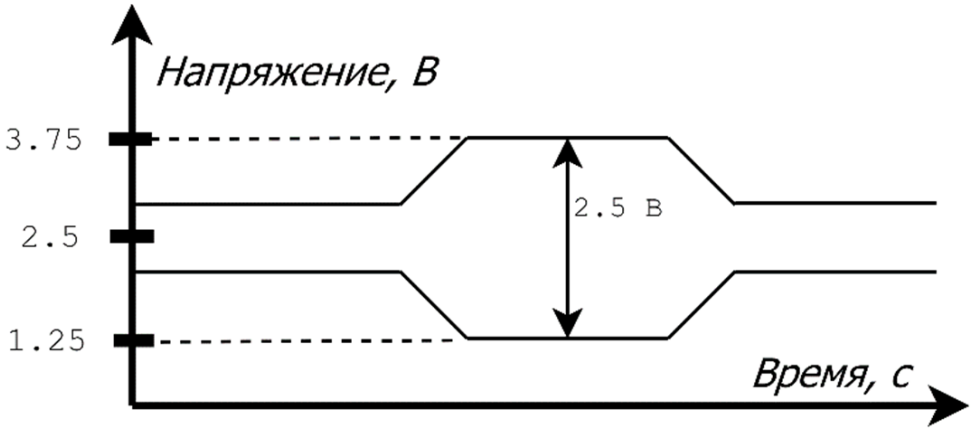


Рис. 11. Уровни сигнала шины CAN

Скорость передачи данных на шине напрямую зависит от длины шины. Таким образом, максимальная скорость в 1 Мбит/с достигается на шине длиной не более 40 м, а на шине длиной 6 км скорость может достигать максимум 10 Кбит/с (табл. 4).

Таблица 4 – Характеристика шины CAN

Параметр	Значение
Максимальная скорость передачи данных, Мбит/с	1
Максимальная длина кабеля, м	6000
Максимальное число устройств	64

Формат кадра

По протоколу CAN могут передаваться до 5 типов кадров (кадр данных, кадр удаленного запроса, кадр ошибки и кадр перегрузки). Основная полезная нагрузка передается в кадре данных, формат которого представлен на рисунке 12. Обычно при формировании пакета пользователю требуется только установить идентификатор, данные и длину сообщения.

Стоит отметить, что идентификатор не ассоциируется с конкретным узлом и не является его адресом. Идентификатор необходим для определения приоритета сообщения, что важно для систем, близким к масштабам реального времени. Более того, часто идентификатор используют для определения смысловой нагрузки пакета, например, типа передаваемой величины.

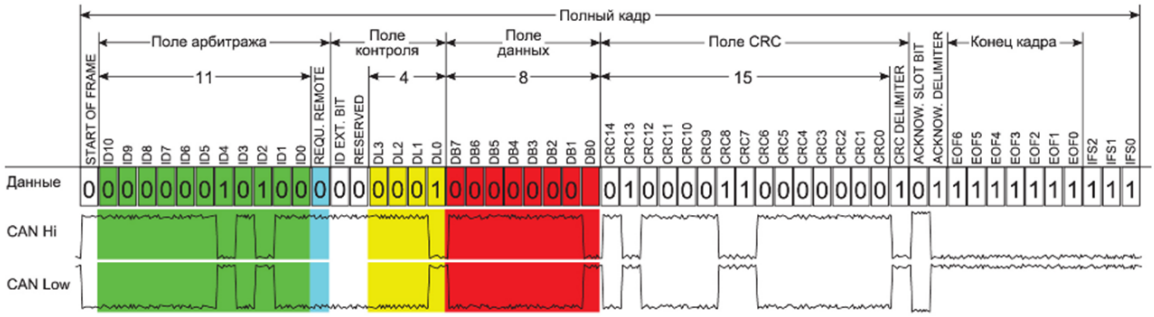


Рис 12. Формат кадра CAN

Необходимо отметить, что протокол CAN сам по себе не определяет явной адресации сообщений и узлов, а также того, как используется идентификатор в поле арбитража, для этого существуют протоколы более высокого уровня, например, CANopen, DeviceNet и CanKingdom.

Протокол Modbus

Коммуникационный протокол, который был создан в 1979 году компанией Modicon (сейчас принадлежит Schneider Electric). На сегодняшний день он является самым распространенным в мире протоколом промышленных сетей.

Модель

Протокол использует модель «*Master-Slave*». Следуя данной модели, мастер инициирует запросы к подчиненным устройствам по их адресам. Подчиненные устройства, в свою очередь, отвечают мастеру, однако не могут отправлять собственные запросы к другим устройствам на шине (рис. 13).

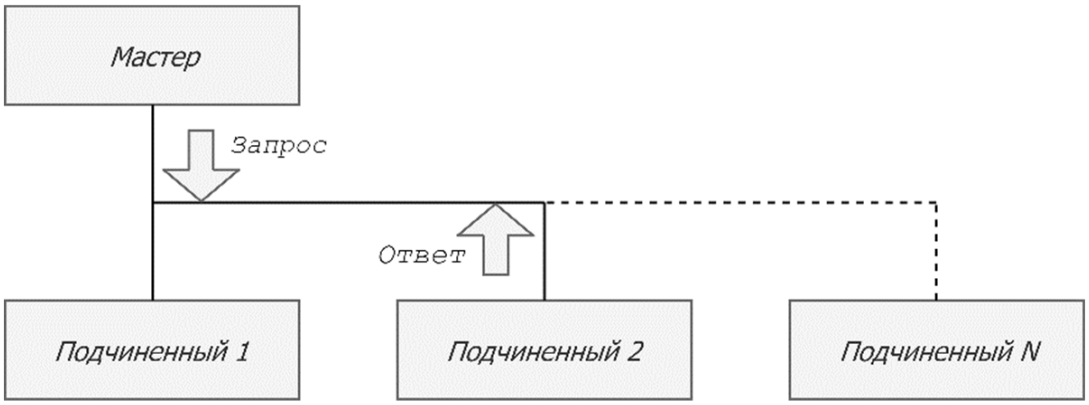


Рис. 13. Модель «Master-Slave»

Физический уровень

Шина Modbus может быть реализована поверх различных физических интерфейсов, к которым относятся:

– Универсальный асинхронный приёмопередатчик (RS-232, RS-485), использующий протоколы Modbus RTU, Modbus ASCII.
– Ethernet, использующий протокол Modbus TCP.

Реализация протокола на основе интерфейса Ethernet является более предпочтительной и позволяет достичь большей скорости, дальности, помехозащищённости и большего количества участников сети. Более того, такая сеть может быть развернута на основе существующих офисных сетей, что сокращает материальные затраты (табл. 5).

Таблица 5 – Сравнение характеристик Modbus поверх интерфейсов RS-485 и Ethernet

Параметр	RS-485	Ethernet
Максимальное число устройств	32 (254 с повторителем)	255 с маской 24, и более с другими масками
Максимальная длина кабеля, м	1200	100 без повторителя
Максимальная скорость передачи данных, Мбит/с	10	1000

На шине RS-485 на больших расстояниях возникает эффект длинных линий, и для ослабления искажений на концы шины требуется устанавливать терминальные резисторы (терминаторы). Обычно номинал резистора равен 120 Ом, однако номинал варьируется в зависимости от волнового сопротивления кабеля (см. рис. 14).

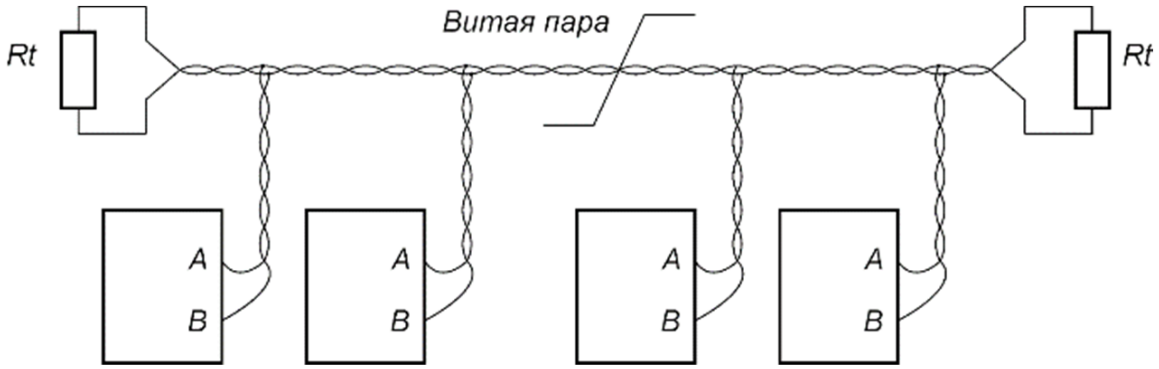


Рис. 14. Терминальный резистор

Регистры и функции

В основе запроса по протоколу Modbus лежат код функции и данные, которые надо считать или записать. Данные хранятся в регистрах памяти ввода/вывода, которые бывают нескольких видов (см. табл. 6).

Таблица 6 – Характеристики регистров Modbus

Регистр	**Размер**	**Адресация**[4]	**Тип доступа**	**Функции***
Input registers (Регистры ввода)	2 байта	Регистровая	Только чтение	0x04 — чтение группы регистров
Holding registers (Регистры хранения)	2 байта	Регистровая	Чтение и запись	03 — чтение группы регистров 06 — запись одного регистра 16 — запись группы регистров
Discrete Inputs (Дискретные входы)	1 бит	Битовая	Только чтение	02 — чтение группы входов
Coils (флаги)	1 бит	Битовая	Чтение и запись	01 — чтение группы флагов 05 — чтение одного флага 15 — запись группы флагов

Формат пакета

Пакет Modbus можно представить в виде двух вложенных частей – ADU (Application Data Unit) и PDU (Protocol Data Unit). ADU зависит от типа протокола. Ниже приведены пакеты ADU для протоколов Modbus TCP и RTU (см. табл. 7 и 8).

Таблица 7 – ADU для Modbus RTU

Адрес подчиненного устройства 1 байт	Данные 0-253 байт	Контрольная сумма CRC16 2 байта

[4] Функции и адресация могут зависеть от конкретного устройства.

Таблица 8 – ADU для Modbus TCP

ID транзакции 2 байта	ID протокола 2 байта	Длина пакета 2 байта	Адрес ведомого устройства 1 байт	Данные 0-253 байт

В поле данных ADU помещается пакет PDU следующего формата:

Код функции	Данные (зависят от функции)

Обзор программного и аппаратного обеспечения, используемого в лабораторном практикуме

Программируемый логический контроллер Modicon M251

Контроллер Modicon M251 представляет собой высокопроизводительную модульную систему автоматизации общего назначения (рис. 15). Особенностью данного контроллера является отсутствие встроенных портов ввода/вывода. Для расширения функциональности устройства, а также подключения к нему различных полевых устройств (электроприводов, датчиков, панелей оператора и т. д.) используются модули расширения, а также встроенные интерфейсы CANopen и Ethernet.

Рис. 15. ПЛК Modicon M251MESE

Сетевые возможности контроллера Modicon M251 включают в себя: FTP-клиент/сервер, веб-сервер, SQL-клиент, SNMP-клиент/сервер V1 и V2, OPC UA-сервер, SNTP-клиент и DNS-клиент. Это означает, что их можно легко интегрировать в системы управления для дистанционного мониторинга и удаленного обслуживания оборудования с помощью приложений для смартфонов, планшетов и ПК.

Основные технические характеристики ПЛК:

– Количество модулей ввода/вывода: 7 подключаемых непосредственно к ПЛК по протоколу TM3 и 14 удалённых.
– Напряжение питания: 20,4–28,8 В, номинальное – 24 В.

- Максимальное энергопотребление: 32.6–40.4 Вт при максимальном количестве внешних модулей ввода/вывода.
- Объем памяти для программ: 8 МБ.
- Объем оперативной памяти: 64 МБ.
- Объем постоянной памяти: 128 МБ для резервного копирования пользовательских программ и прошивок + разъем для карт памяти формата SD объемом до 32 ГБ.
- Время выполнения 1000 программных инструкций: 0,3 мс для периодических задач и задач по событию, 0,7 мс для остальных инструкций.
- Среднее время выполнения одной инструкции: 0,022 мкс.
- Структура программного приложения: 4 циклические задачи, 8 задач по событию и 1 свободная задача.
- Сетевые интерфейсы: 2 порта Ethernet 10BASE-T/100BASE-TX, USB, RS-485.
- Сетевые возможности: Modbus TCP клиент/сервер, DHCP клиент/сервер, DNS клиент, OPC UA сервер.
- Максимальное количество одновременных сетевых соединений: 8 Modbus «клиент/сервер», 4 FTP клиента.

Модуль расширения портов ввода/вывода TM3TM3

Модуль TM3TM3 представляет собой систему ввода/вывода для подключения двух аналоговых датчиков и формирования одного аналогового сигнала (рис. 16).

Рис. 16. Комбинированный модуль ввода/вывода TM3TM3

Основные технические характеристики модуля:

- Типы подключаемых датчиков:
  - По току: 0–20 мА и 4–20 мА.
  - По напряжению: 0–10 В и -10–10 В.
  - Термопары типов **J** (-200–1000°C), **K** (-200–1300°C), **R** (0–1760°C), **S** (0–1760°C), **B** (0–1820°C), **T** (-200–400°C), **N** (-200–1300°C), **E** (-200–800°C), **C** (0–2315°C).
  - Трёхпроводные термометры сопротивления: Ni 100/Ni 1000 (-60-180°C), Pt 100 (-200–850°C), Pt 1000 (-200–600°C).
- Выходной сигнал:
  - По току: 0–20 мА и 4–20 мА.
  - По напряжению: 0–10 В и -10–10 В.
- Разрешающая способность цифро-аналогового преобразователя: 12 бит (4096 точек).
- Разрешающая способность аналого-цифрового преобразователя: 16 бит или 15 бит + знаковый бит.
- Частота дискретизации цифро-аналогового преобразователя: 10 Гц.
- Частота дискретизации аналого-цифрового преобразователя 10 Гц или 100 Гц.

Частотный преобразователь Altivar ATV 630

Частотный преобразователь Altivar ATV 630 (рис. 17) предназначен для управления трёхфазными асинхронными двигателями. Данный частотный преобразователь питается от общей однофазной сети напряжением 230 В и частотой 50 Гц и способен генерировать трёхфазный управляющий сигнал напряжением 400 В и частотой от 0 до 500 Гц. Основными особенностями модели являются:

- Управление асинхронными двигателями мощностью до 750 Вт.
- Возможность управления направлением вращения двигателя.
- Возможность поддержания частоты питающего сигнала двигателя в диапазоне от 0 до 500 Гц с разрешающей способностью 0,1 Гц.
- Поддержание максимально достижимого момента вращения во всем диапазоне скоростей вращения.
- Управление со встроенной панели, через клеммную колодку или по протоколу Modbus TCP.
- Регулирование времени разгона и торможения двигателя в диапазоне от 0,01 до 9999 с.

– Система защиты от короткого замыкания, отклонений от номинального напряжения и тока, перегрева.

Рис. 17. Частотный преобразователь Altivar ATV630

Датчик температуры Pt100

Датчик температуры Pt1000 представляет собой терморезистор на основе платины, сопротивление которого при нормальных условиях равно 1000 Ом. При изменении температуры это сопротивление изменяется. Соответственно, если данное сопротивление включить в одно из плеч резистивного делителя и измерять напряжение в его центральной точке, можно получить текущее значение температуры. Следует отметить, что зависимость температуры от сопротивления близка к линейной; в модуль TM3TM3 записана калибровочная таблица, позволяющая сопоставить значения сопротивления и температуры.

Датчик влажности

Датчик влажности способен измерять значение относительной влажности в диапазоне 15–90 %. Датчик работает в токовой петле диапазона 4–20 мА. Значения снимаются с шунтирующего резистора номиналом 500 Ом, характеристика датчика линейная.

Лабораторная работа 1. Создание проекта в интегрированной среде разработки SoMachine 4.3

Цель работы: создать пустой проект в среде SoMachine 4.3 и произвести первоначальную настройку программной и аппаратной частей контроллера Modicon M251.

Задание по работе

1. Изучить теоретическую часть работы.
2. Создать проект в среде SoMachine 4.3.
3. Настроить параметры аналогового модуля ввода/вывода TM3TM3.
4. Настроить коммуникационные параметры контроллера.
5. Загрузить настройки во внутреннюю память контроллера.
6. Проверить доступность контроллера по сети.

Теоретическая часть

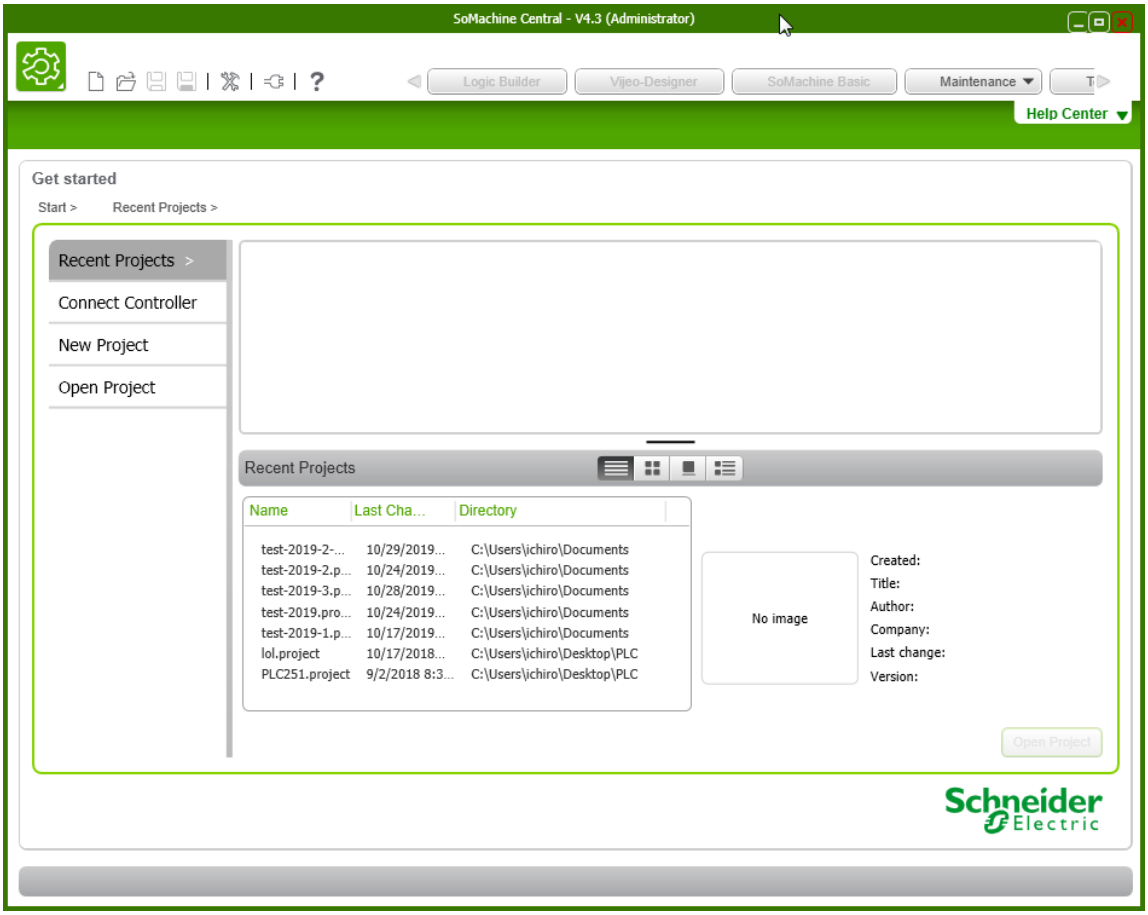
Разработка управляющих программ для ПЛК осуществляется в специализированной среде *SoMachine 4.3*. Данное программное обеспечение представляет собой адаптированную версию системы *Codesys 3* от компании *3S-Smart Software Solutions*. Оригинальная система дополнена расширениями, позволяющими работать с оборудованием *Schneider Electric* без необходимости дополнительных настроек. Все файлы с описанием конкретных моделей оборудования доступны непосредственно в дистрибутиве либо скачиваются из репозитария через встроенную систему обновления. Также *SoMachine 4.3* включает в себя все инструменты программирования: компилятор, отладчик, профайлер и загрузчик бинарных файлов в ПЛК.

Основным понятием в рассматриваемой системе программирования является проект. Проект является совокупностью описания оборудования, настроек и программного кода, организованного в виде дерева. При этом всё оборудование, включённое в дерево, конфигурируется совместно в рамках единого жизненного цикла проекта, что существенно упрощает управление версиями и загрузку бинарных файлов в память устройств.

SoMachine 4.3 выпускается под проприетарной лицензией, однако большинство модулей предоставляется бесплатно при условии регистрации на сайте. Для выполнения всех лабораторных работ, рассматриваемых в данном пособии, достаточно только свободно распространяемых компонентов, дополнительное платное программное обеспечение не требуется.

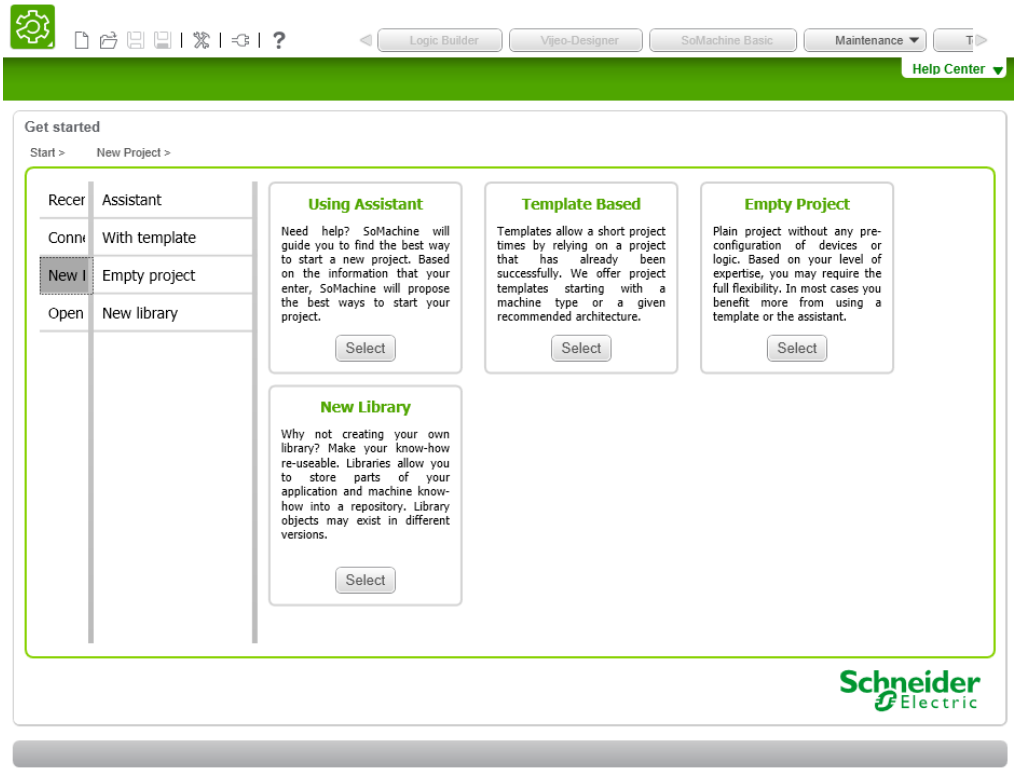
Экспериментальная часть

1. Работа с системой начинается с создания пустого проекта. Для его создания необходимо нажать кнопку *New Project* в окне, представленном ниже.

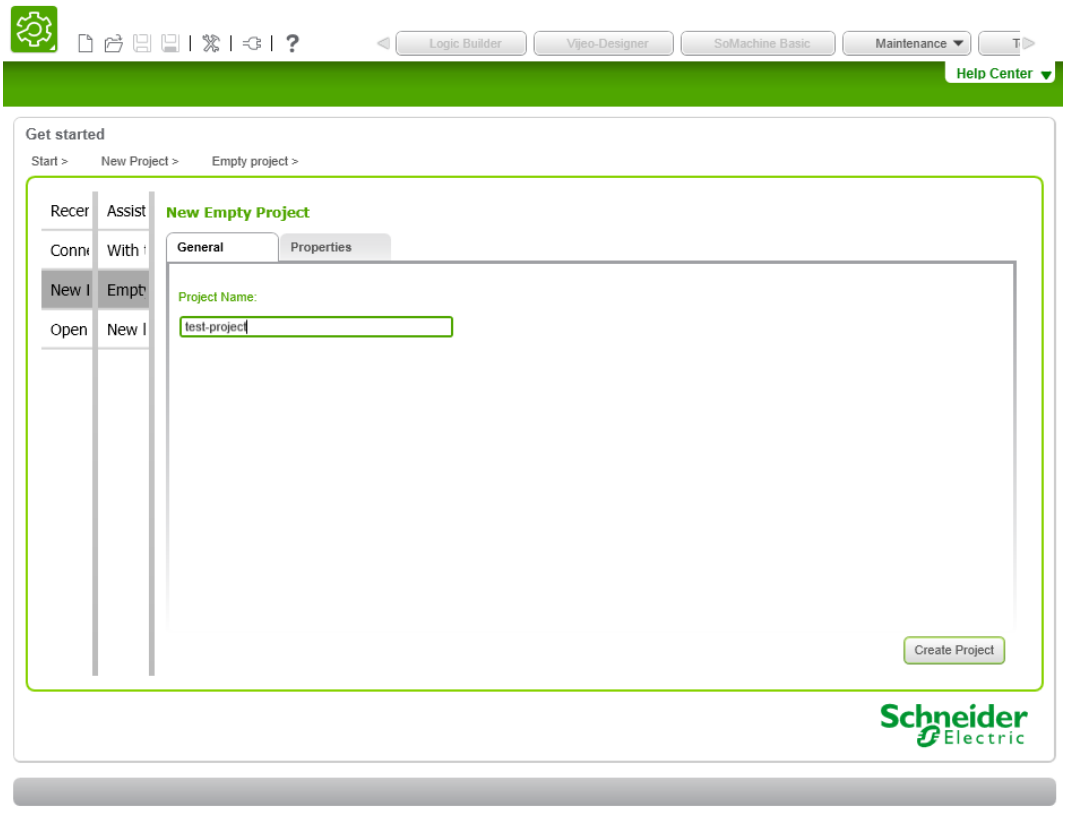


2. Система предложит на выбор несколько способов создания нового проекта:
   – С помощью ассистента (пункт меню Assistant).
   – По шаблону для решения конкретной производственной задачи (пункт меню *Template*).
   – Пустой проект (пункт меню *Empty Project*).
   – Создание пользовательской библиотеки, позволяющей автоматизировать процесс создания новых проектов для решения типовых задач (пункт меню *New Library*).

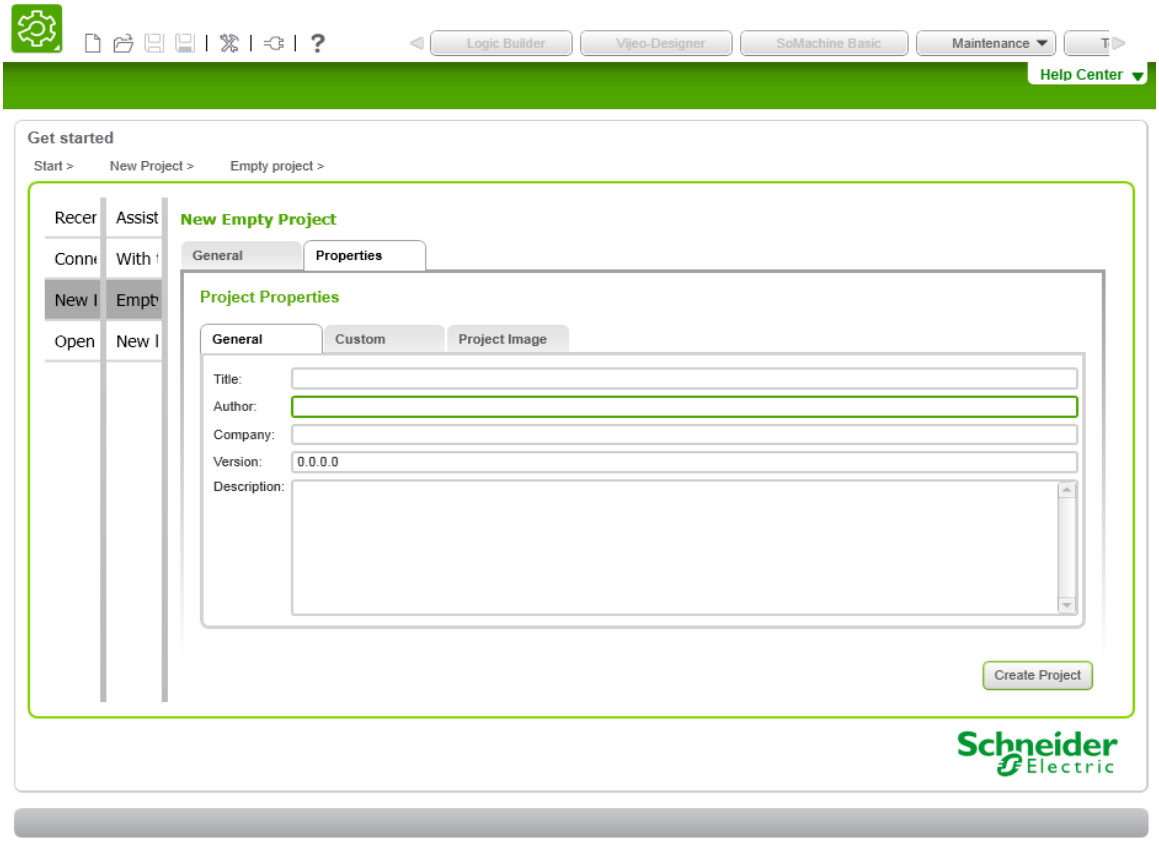
Необходимо выбрать пункт «Пустой проект» (*Empty Project)*.



3. В появившемся окне система предложит придумать название для будущего проекта (вкладка *General*). Название может быть произвольным, допускается использование кириллических символов.



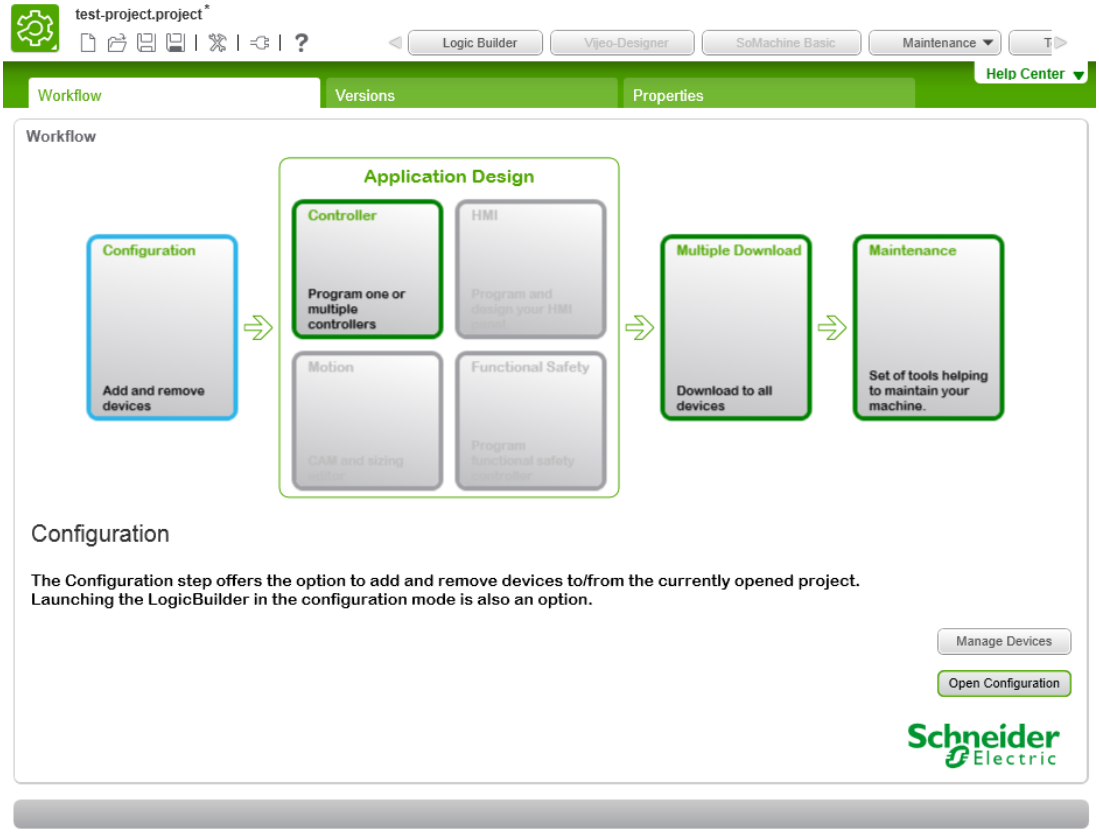
4. Далее на вкладке *Properties* необходимо заполнить поля «Название проекта» (*Title*) и «Автор» (*Author*). Остальные поля не заполняются. После заполнение основных параметров проекта необходимо нажать кнопку «Создать проект» (*Create Project*).



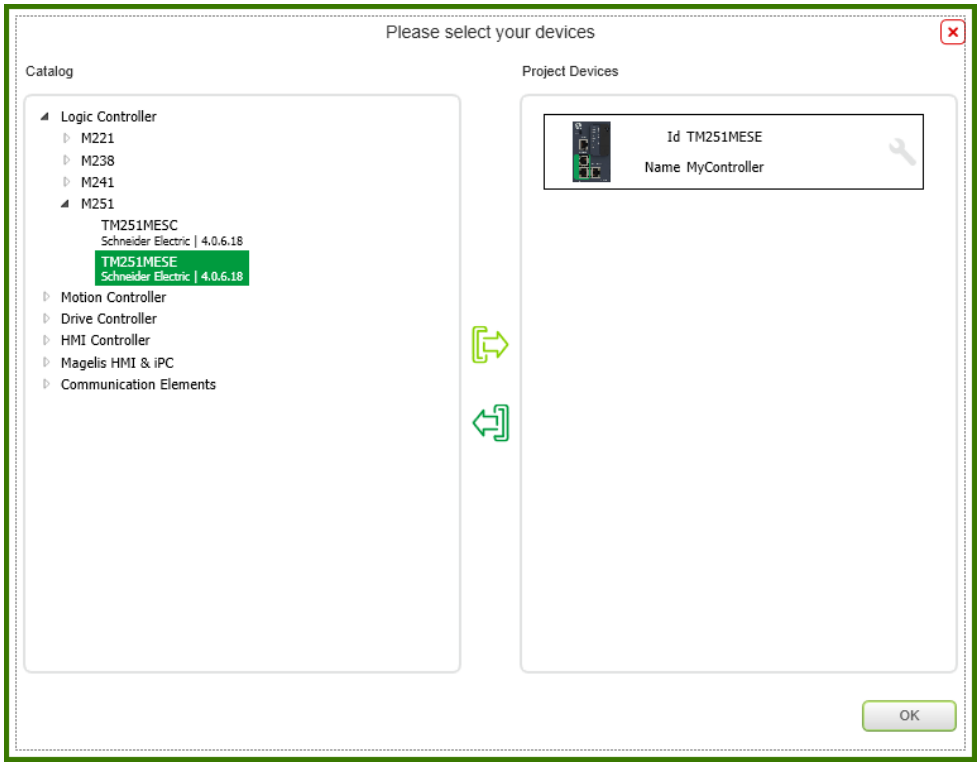
5. Откроется окно с диаграммой жизненного цикла проекта и текущей стадией работы. В системе заданы четыре стадии проекта:

– Конфигурация (пункт меню *Configuration*), на этом этапе производится добавление или удаление физических устройств в дереве проекта.
– Проектирование приложения (*Application Design*), включающая в себя подпункты «Контроллер» (*Controller*) для создания программ для одного или нескольких ПЛК; «Человеко-машинный интерфейс» (*HMI*) для разработки пользовательского интерфейса панелей оператора; «Управление движением» (*Motion*) для написания управляющих программ контроллеров движения; «Функциональная безопасность» (*Functional Safety*) для программирования специализированных контроллеров, обеспечивающих безопасность систем автоматизации производства.

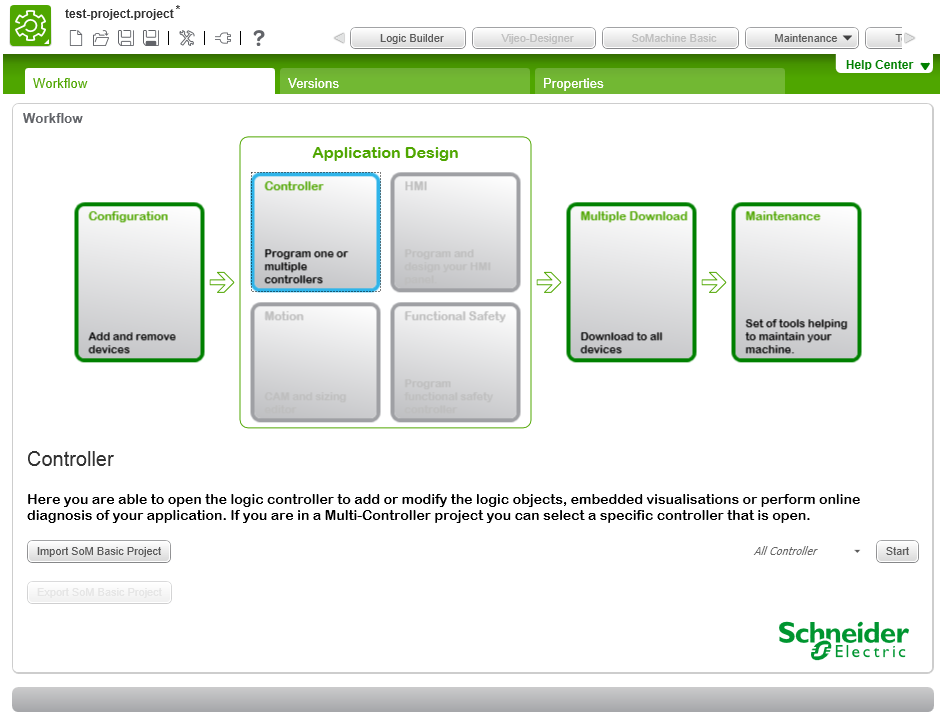
– Множественная загрузка (*Multiple Download*), на этом этапе осуществляется запись бинарных файлов разработанных программ в память ПЛК.
– Обслуживание (*Maintenance*), на этом этапе реализованы процедуры сервисного обслуживания оборудования, включенного в проект: резервное копирование внутреннего программного обеспечения оборудования (т. н. «прошивок»), обновление «прошивок» и другие подобные. На рисунке ниже в качестве текущего отмечен пункт жизненного цикла «Конфигурация», поэтому для дальнейшей работы необходимо нажать кнопку «Открыть конфигурацию» (*Open Configuration*). На этом этапе также необходимо отключить ПЛК Modicon M251 от цепи питания 24 В и подключить к ПК с помощью USB-кабеля. Вся дальнейшая настройка будет осуществляться по данному кабелю.



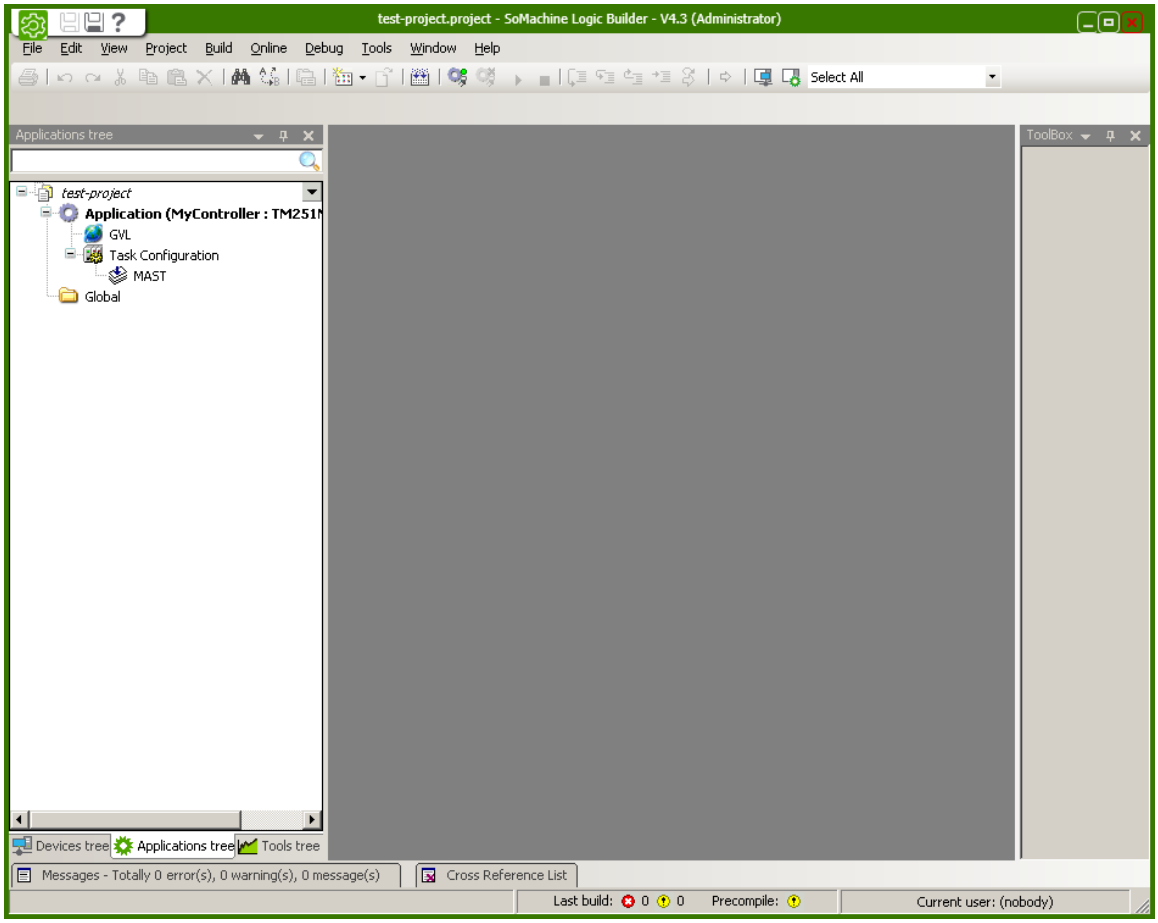
6. В появившемся окне слева будет отображено дерево всего оборудования, с которым можно работать в системе. В дереве необходимо раскрыть пункт «Логические контроллеры», а в нём подпункт M251, после чего выбрать модель M251MESE и нажать кнопку «Стрелка влево». Выбранный контроллер должен появиться на панели слева.



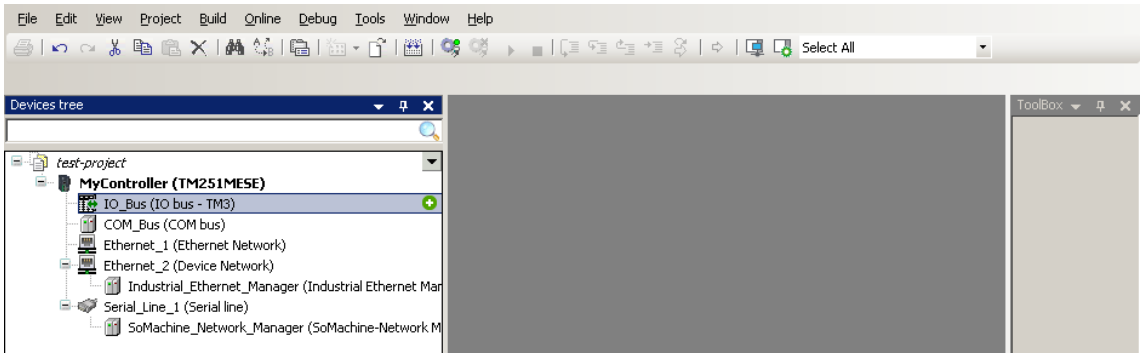
7. На следующем этапе необходимо подтвердить свои действия нажатием кнопки *Ok*, после чего опять откроется окно жизненного цикла проекта, где будет видно, что текущий этап изменился и можно приступать к созданию управляющей программы нажатием кнопки *Start*.



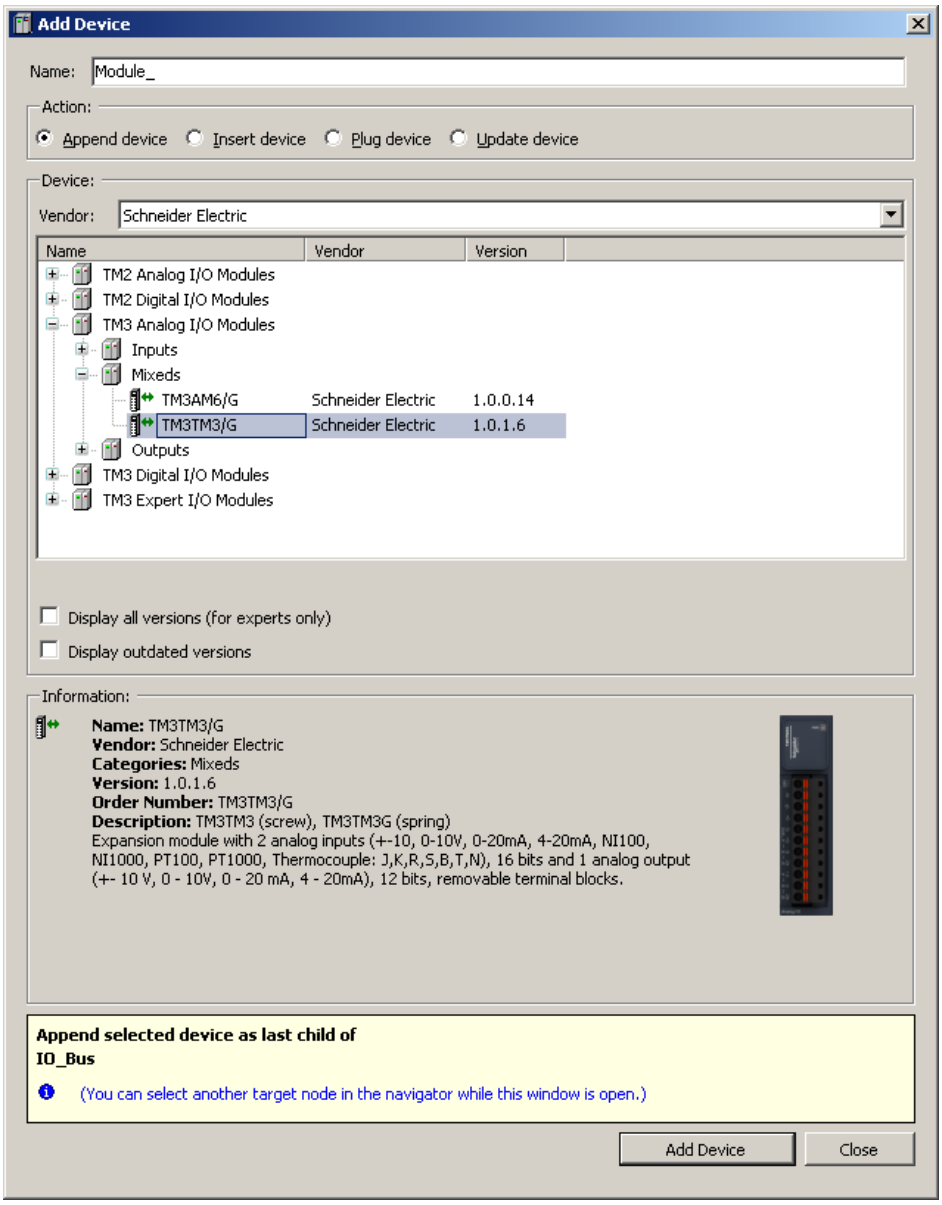
8. Будет открыто основное окно программы. Окно разделено на несколько панелей. Верхняя панель (панель инструментов) содержит иконки для быстрого доступа к основным функциям системы. Левая панель включает в себя три вкладки с основными иерархическими деревьями проекта: деревом физических устройств, деревом приложений и деревом вспомогательных инструментов проекта. Правая (основная) панель отображает инструменты работы с компонентами проекта: окна редактирования исходного кода, списки локальных и глобальных переменных, релейно-контактные схемы, диаграммы функциональных блоков, инструменты настройки периферийных устройств, диалоги настройки сетевых устройств и т. п. Нижняя панель является консолью, в которой отображаются сообщения системы, такие как ошибки предупреждения и информационные сообщения. Панели могут быть перегруппированы любым удобным для пользователя способом.



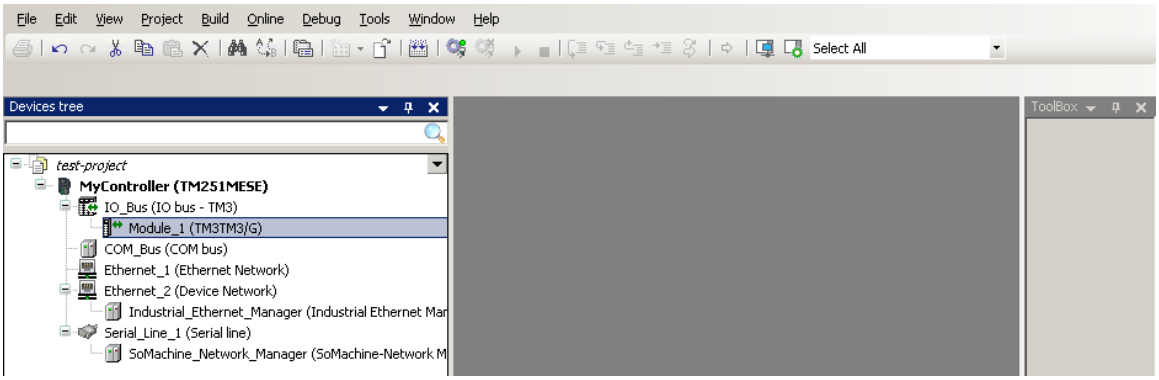
9. В левой панели необходимо перейти на вкладку *Application Tree* и выбрать пункт *IO_Bus (IO bus – TM3)*, после чего нажать на появившуюся в строке кнопку с зеленым символом «плюс».



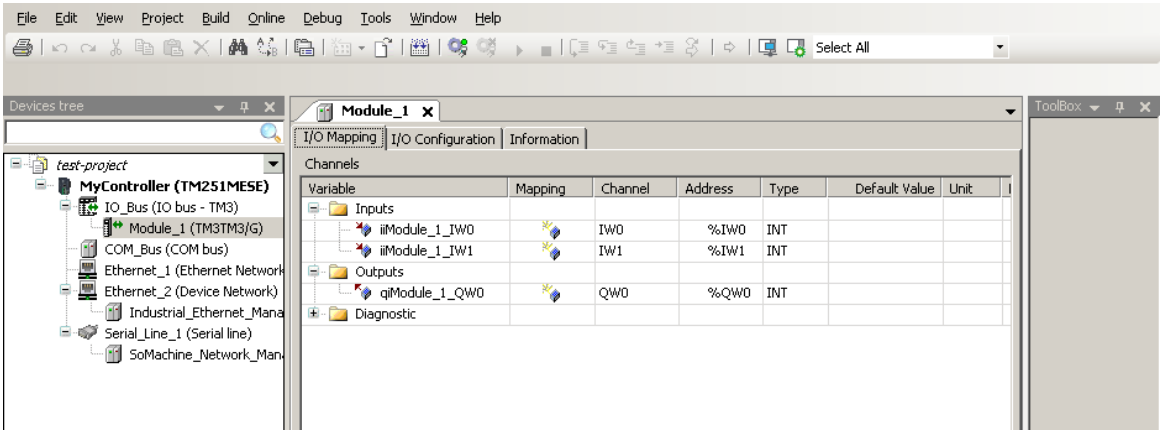
10. Откроется окно выбора периферийных модулей ввода-вывода, где необходимо выбрать модуль аналогового ввода-вывода TM3TM3/G, после чего нажать кнопку «Добавить устройство» (*Add Device*).



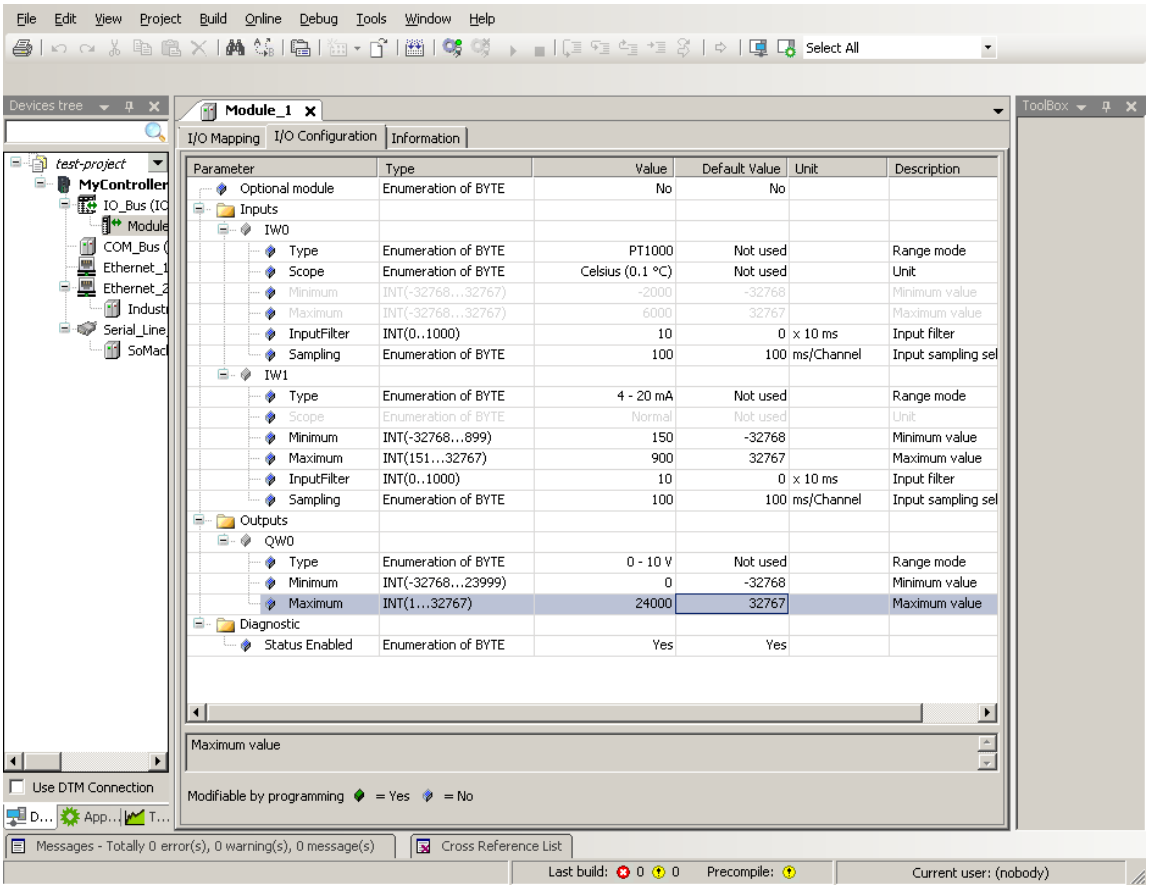
11. После того, как модуль появится в дереве устройств, необходимо дважды щелкнуть по нему левой клавишей мыши, чтобы активировать окно с настройками.



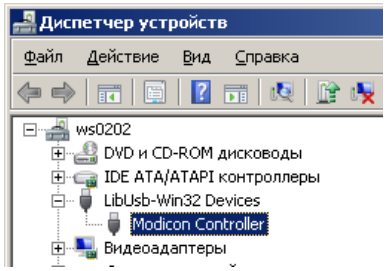
12. Открывшаяся панель будет содержать три вкладки: *I/O Mapping* (сопоставление физических портов модуля ввода/вывода с внутренними переменными системы); *I/O Configuration* (конфигурация портов ввода/вывода) и *Information* (сведения о модуле).



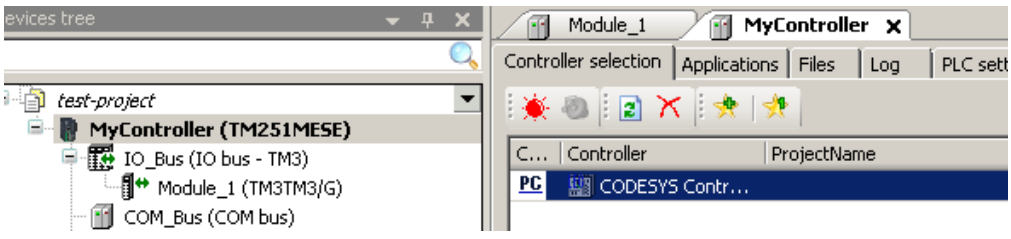
13. Необходимо перейти на вкладку конфигурации и заполнить параметры в соответствии с рисунком ниже. Необходимо отметить, что температура будет представлена в виде целого числа, представляющего вещественное число с точностью в один знак после запятой (например, число 258 будет означать 25,8 градуса Цельсия). То же самое задаётся для влажности, где целое число будет в диапазоне от 150 до 900. Значение для выходного порта (пункт Outputs) понадобится для выполнения следующей лабораторной работы. Для компенсации воздействия электрических шумов все входные параметры проходят фильтрацию с усреднением по 10 точкам, время выборки устанавливается равным 100 мс, что соответствует десяти измерениям в секунду.



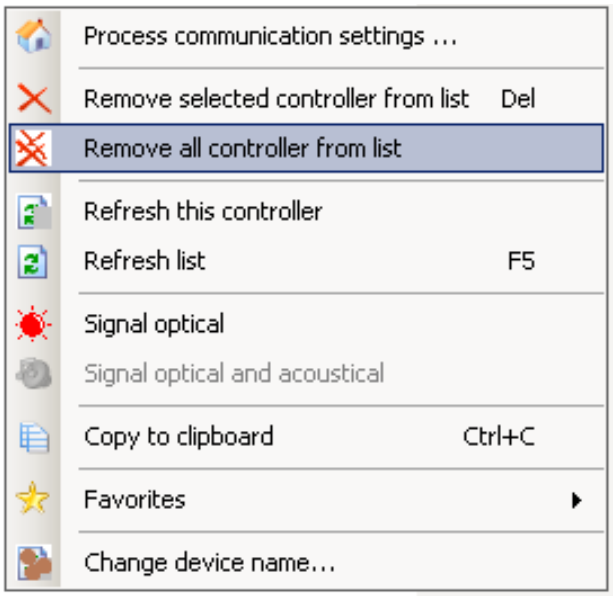
14. На следующем этапе необходимо убедиться, что персональный компьютер обнаружил ПЛК и может к нему подключиться. Для этого сперва необходимо проверить, есть ли устройство в диспетчере устройств Windows.



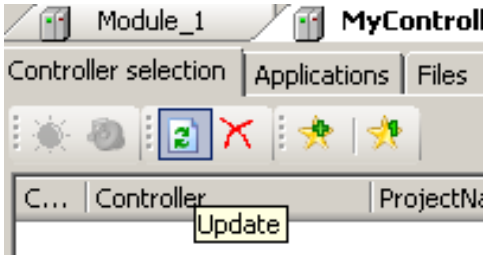
Затем в дереве устройств перейти в корень (*My Controller*) в центральной панели, нажать правую клавишу мыши.



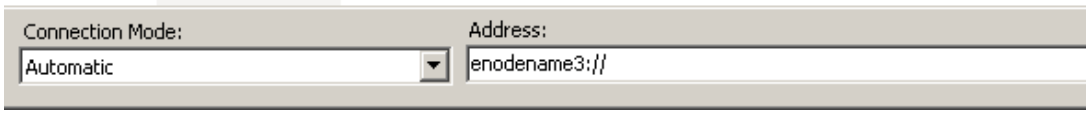
и в появившемся меню выбрать пункт *Remove all controller from list,*



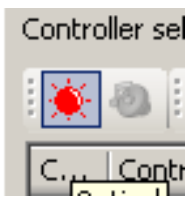
после чего нажать кнопку «Обновить».



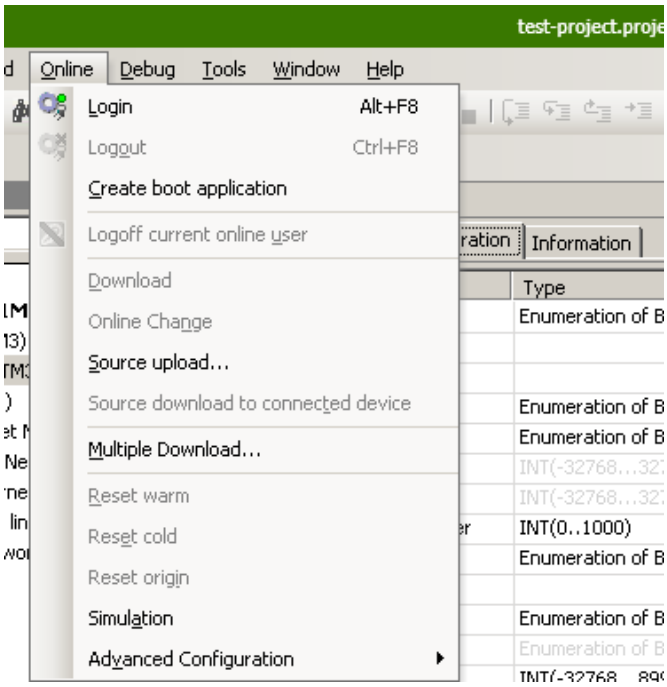
Контроллер должен появиться в списке со следующим адресом, отображаемым в нижней панели:



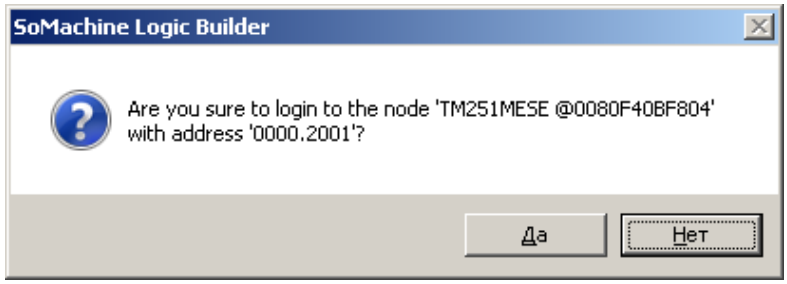
Далее необходимо нажать кнопку *Signal Optical* и убедиться, что контроллер начал попеременно включать и выключать все световые индикаторы на лицевой панели.



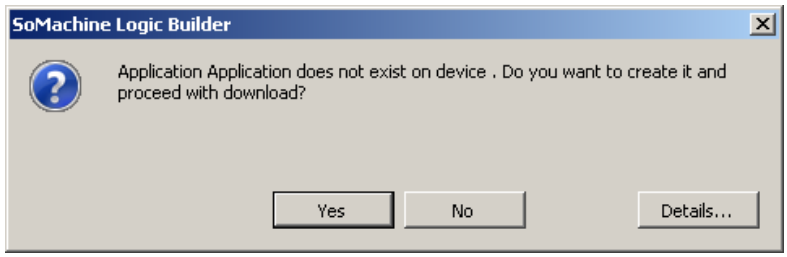
15. На следующем этапе необходимо подключиться к контроллеру, выбрав соответствующий пункт меню (*Login*), нажав на иконку в панели инструментов или воспользовавшись комбинацией клавиш *Alt+F8*.



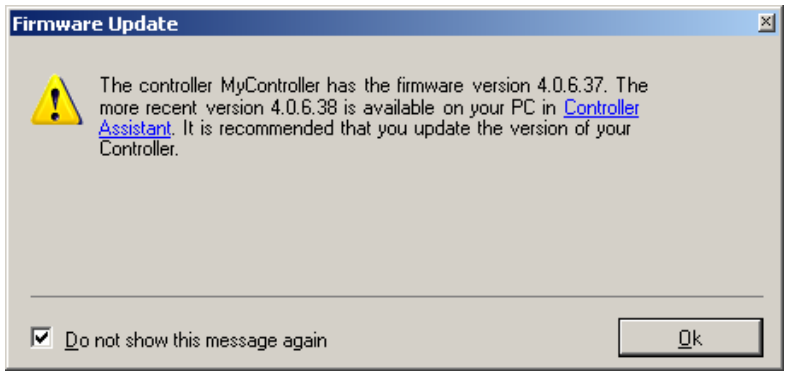
16. Система выдаст два предупреждения о безопасности, в первом случае необходимо использовать комбинацию клавиш *Alt+F*, во втором – нажать кнопку *Да*.



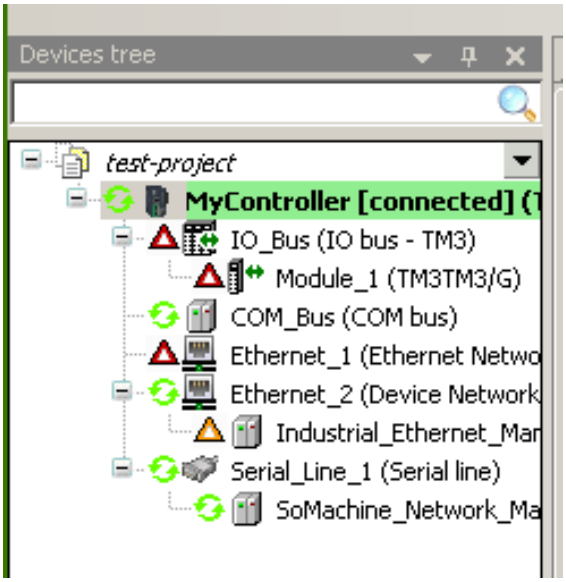
17. На следующем этапе система предложит создать новое пустое приложение с текущими настройками и загрузить его в ПЛК, в данном диалоговом окне необходимо нажать кнопку *Yes*.



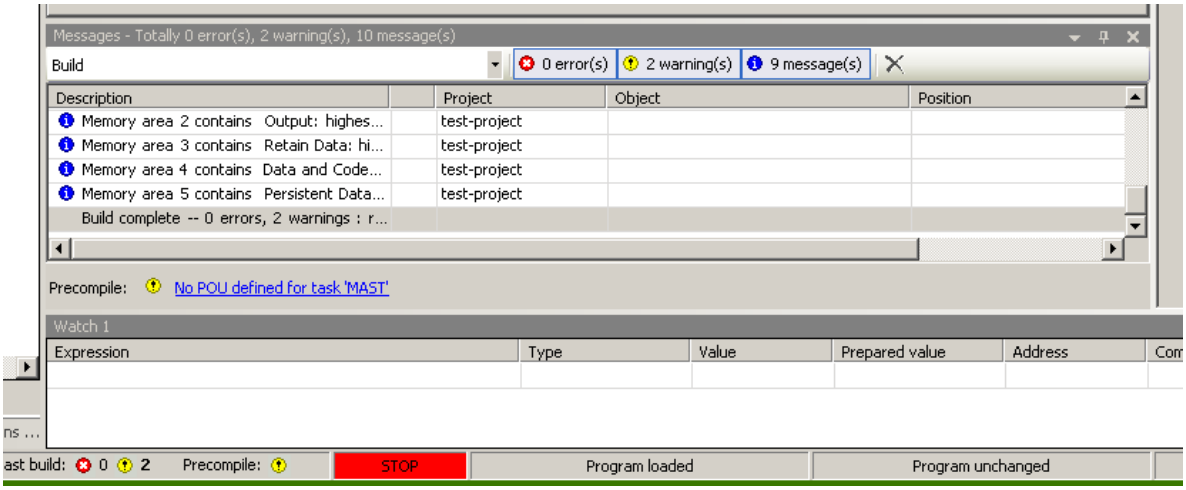
18. Далее система предупредит, что требуется обновить внутреннее программное обеспечение ПЛК. В данном диалоговом окне необходимо нажать кнопку *Ок*. Также рекомендуется установить флажок Do not show this message again, чтобы выключить подобные сообщения при следующих запусках. Стоит отметить, что данный этап не является обязательным, его отсутствие не повлияет на работоспособность проекта.



19. На данном этапе система должна установить соединение с ПЛК, о чём должны свидетельствовать появившиеся иконки рядом с каждым физическим компонентом проекта. Зеленые пиктограммы со стрелками означают, что данный модуль полностью настроен и работает, красные и оранжевые треугольники – модуль не настроен, либо настроен неправильно.



20. В нижней панели отображается статус проекта (наличие ошибок и предупреждений), а также текущее состояние ПЛК. На данном этапе контроллер находится в режиме «Останов» (*STOP*), то есть приложение на нём установлено, но не запущено.

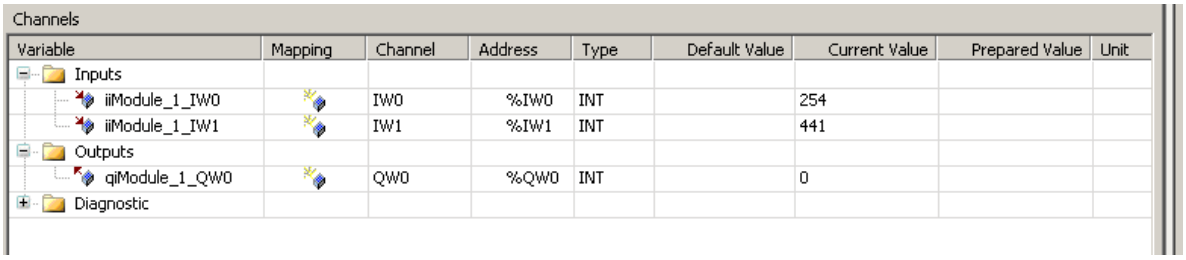


21. Для запуска приложения необходимо подключить ПЛК к источнику питания 24 В, после чего нажать на кнопку со стрелкой в панели инструментов.

22. Если ПЛК не подключен к сети 24 В, система выдаст следующее предупреждение:

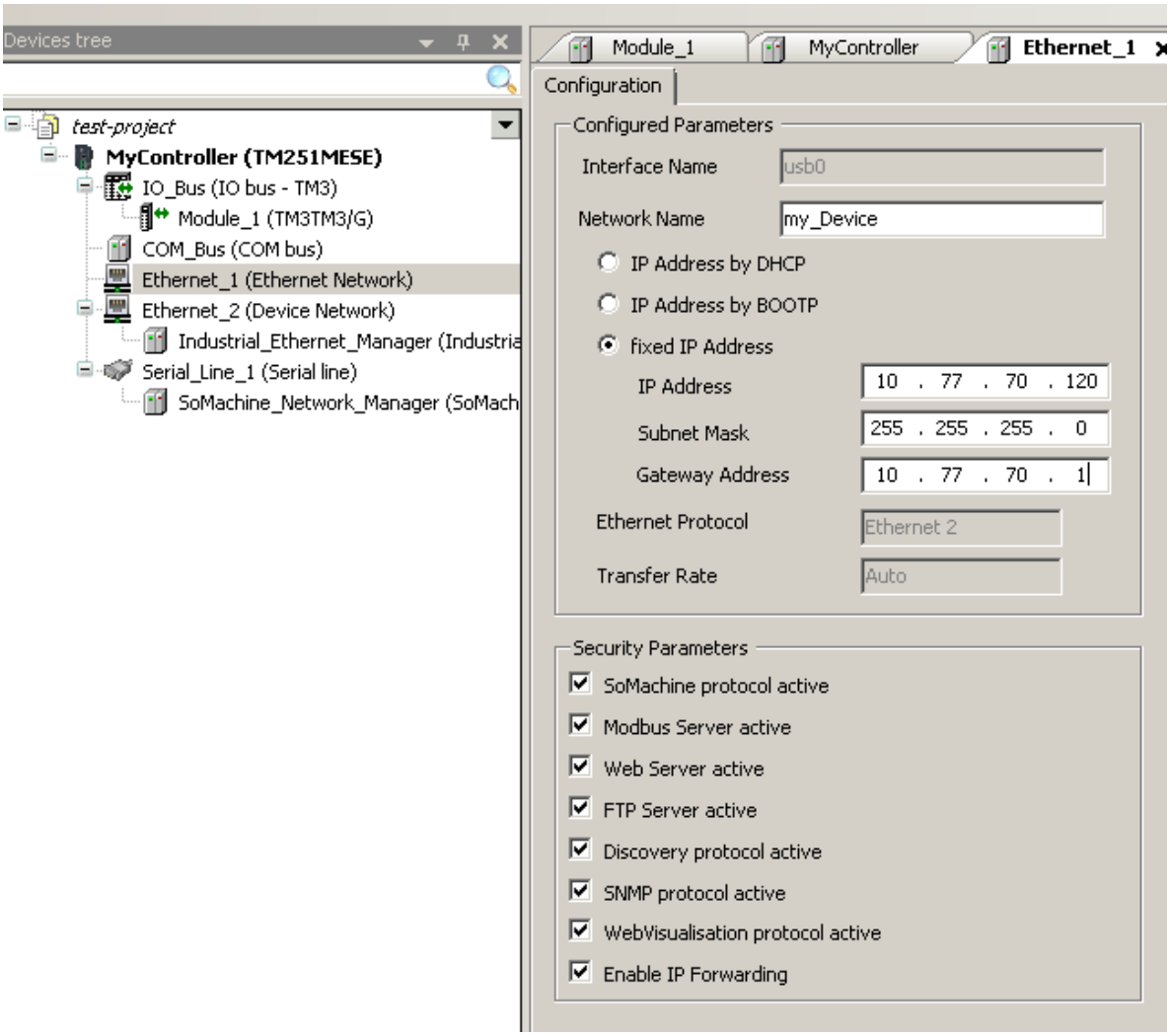
23. В противном случае в нижней панели будет отображен пункт «Работа» (*RUN*).

24. На вкладке *I/O Mapping* модуля *TM3TM3* появятся актуальные значения температуры и влажности.

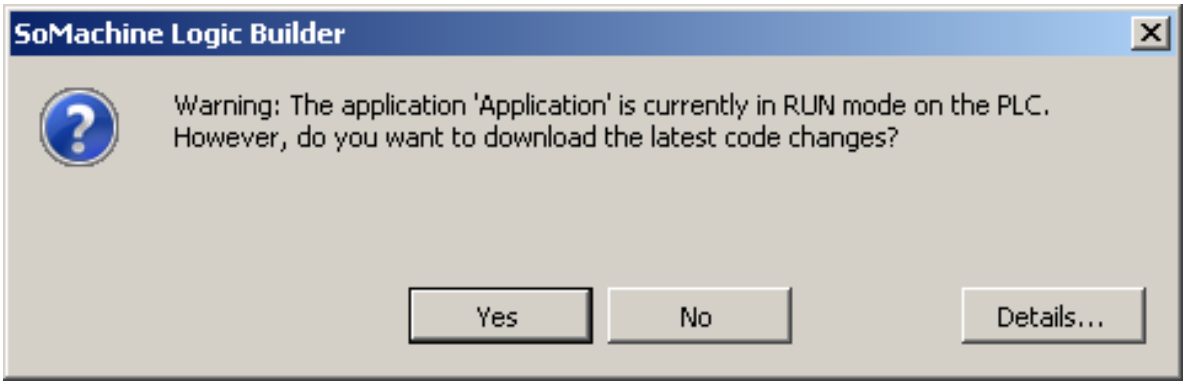


25. На следующем этапе необходимо настроить сетевые параметры контроллера. Для этого вначале необходимо отключиться от контроллера, нажав кнопку *Logout* на панели инструментов (переводить контроллер в состояние «Останов» необязательно),

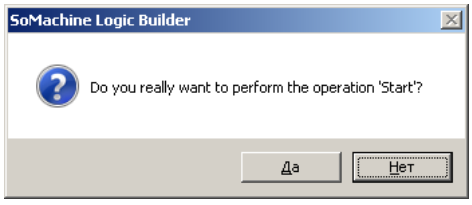
а затем в дереве устройств выбрать Ethernet_1 (Ethernet Network) и в появившемся окне заполнить следующие параметры:



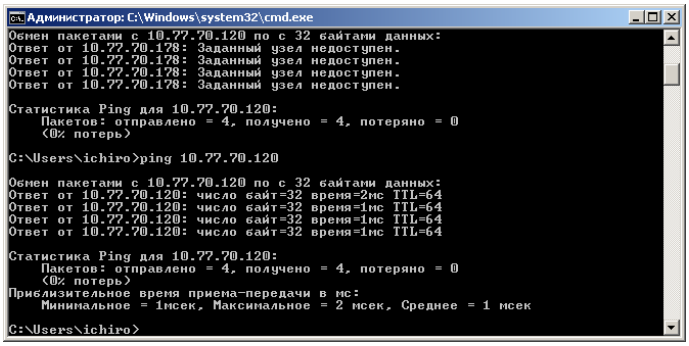
26. После этого следует опять подключиться к контроллеру. На вопрос системы о возможности заменить программу на работающем контроллере ответить *Yes*.



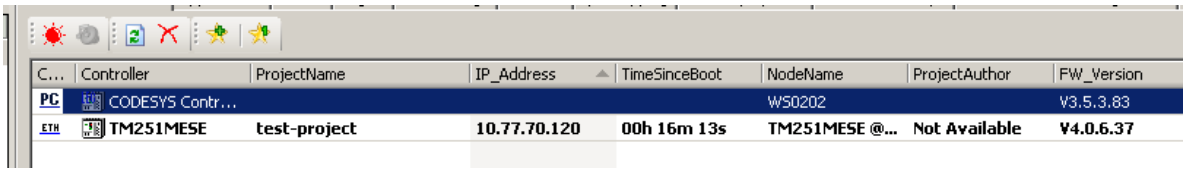
27. Затем перевести контроллер в состояние «Работа» так, как описано в пункте 21.



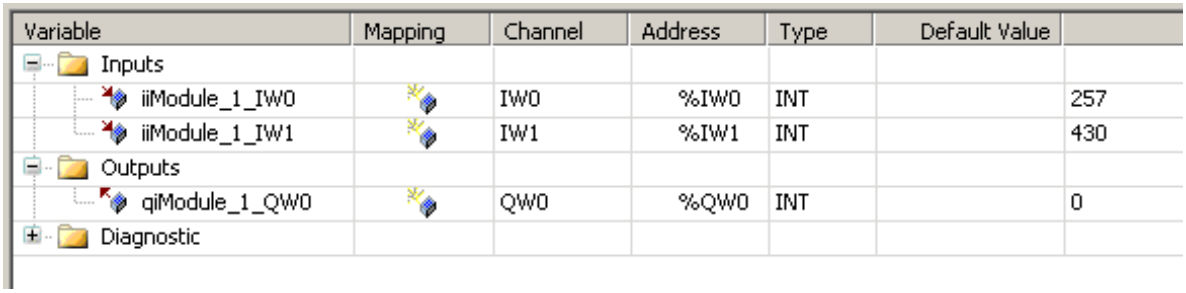
28. Проверить доступность контроллера по сети, выполнив команду *ping* в командной строке Windows (*cmd.exe*).



29. Проверить наличие контроллера в перечне устройств в дереве.



30. Убедиться, что значения температуры и влажности отображаются на вкладке *I/O Configuration*.



Вопросы для самопроверки

1. В чем основное отличие ПЛК от персональных компьютеров?
2. Опишите основные характеристика протокола Modbus.
3. Назовите параметры аналогового модуля ввода/вывода TM3TM3.
4. Какие коммуникационные параметры контроллера необходимо настроить для работы по сети Ethernet?

Лабораторная работа 2. Реализация ПИД регулятора

Цель работы: разработать программу для управления скоростью вращения вентилятора постоянного тока на основе ПИД регулятора.

Задание по работе

1. Изучить теоретическую часть работы.
2. Реализовать модуль управления скоростью вентилятора постоянного тока с применением широтно-импульсной модуляции.
3. Реализовать модуль циклического опроса датчика температуры.
4. Реализовать модуль ПИД регуляции.

Теоретическая часть

Широтно-импульсная модуляция (ШИМ) – это метод снижения средней мощности, передаваемой источником электрического сигнала, путем циклического переключения данного источника. Среднее значение напряжения (и тока), подаваемого на нагрузку, контролируется путем размыкания и замыкания цепи между питанием и нагрузкой с высокой скоростью (рис. 18).

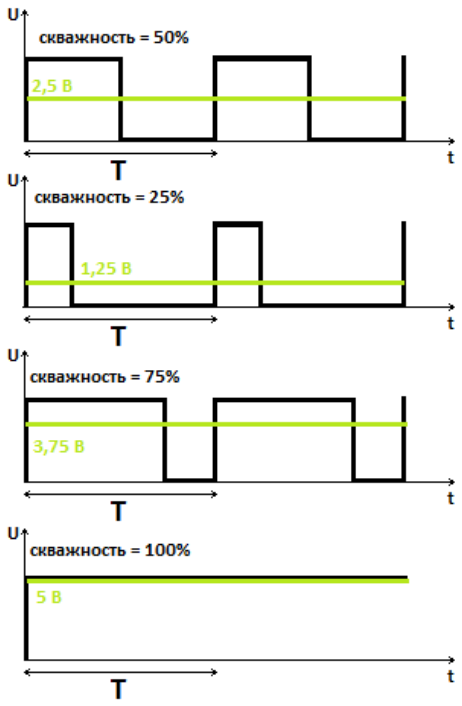


Рис. 18. Широтно-импульсная модуляция сигнала с напряжением 5 В

Чем дольше переключатель включен по сравнению с периодами выключения, тем выше общая мощность, передаваемая нагрузке. Больше всего ШИМ подходит для работы с инерционными нагрузками. К ним могут относиться двигатели, которые не столь сильно подвержены влиянию этого дискретного переключения, поскольку их роторы обладают большой массой, а следовательно, и инерцией.

Частота переключения ШИМ должна быть достаточно высокой, чтобы не влиять на нагрузку, то есть результирующий сигнал, воспринимаемый нагрузкой, должен быть как можно более плавным.

Скорость (или частота), с которой должен переключаться источник питания, может сильно различаться в зависимости от нагрузки и применения. Например, переключение индукционной электрической плиты должно выполняться несколько раз в минуту; частота переключения диммера электрической лампы составляет 120 Гц; частота переключения при управлении силовым электроприводом варьируется от единиц до десятков кГц; в усилителях звука и компьютерных источниках питания она может достигать сотен кГц.

Основным преимуществом ШИМ является крайне низкие потери мощности при коммутации. Когда источник сигнала выключен, ток в цепи практически отсутствует, а когда он включен и мощность передается на нагрузку, на источнике практически нет падения напряжения.

$$R_{tr} \to \infty \leftrightarrow P = \frac{U^2}{R} \to 0$$

$$R_{tr} \to 0 \leftrightarrow P = I^2 R \to 0$$

Потеря мощности, являющаяся произведением напряжения и тока, таким образом, в обоих случаях близка к нулю. ШИМ также хорошо работает с цифровыми элементами управления, которые благодаря своей дискретной природе могут легко изменять свой рабочий цикл.

Например, вольтамперная характеристика светоизлучающего диода сильно отличается от линейной, в то время как яркость диода напрямую зависит от тока. Таким образом, нельзя управлять яркостью светодиода за счёт изменения подаваемого на него напряжения. По сути, светодиод является дискретным источником света, который либо светит с определенной заранее известной яркостью (диод полностью открыт), либо почти не светит (диод закрыт). Однако с помощью ШИМ, меняя скважность управляющего сигнала, можно плавно изменять яркость данного светоизлучающего прибора (рис. 19).

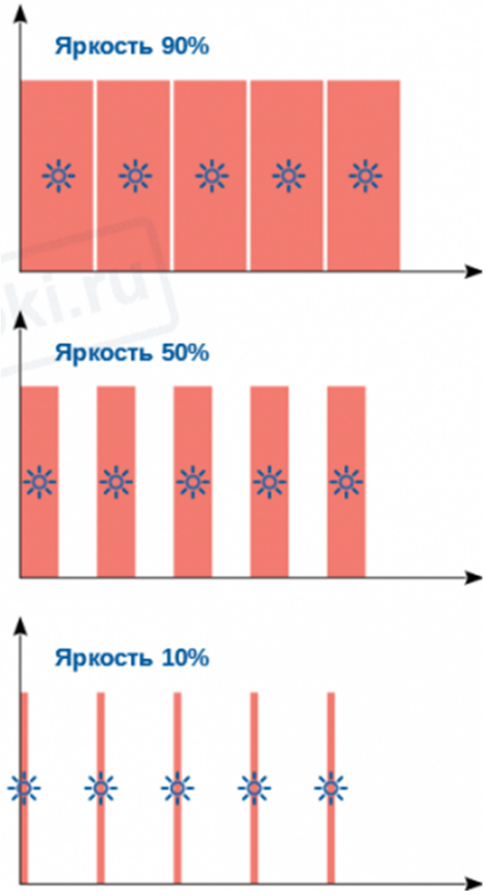


Рис. 19. Зависимость яркости светодиода от скважности управляющего сигнала

Пропорционально-интегрально-дифференциальный регулятор (ПИД регулятор) представляет собой способ автоматического управления, использующий обратную связь. ПИД-регуляция широко применяется в промышленных системах контроля и ряде других приложений, требующих непрерывно следящего воздействия.

ПИД-регулятор непрерывно вычисляет значение ошибки $e(t)$ как разницу между желаемой установкой (англ. *setpoint* – *SP*) и измеренной переменной процесса (англ. *process variable* – *PV*) и применяет коррекцию, основанную на пропорциональной, интегральной и производной компонентах (обозначенных П, И и Д соответственно). С практической точки зрения ПИД-регулятор автоматически применяет точную и отзывчивую коррекцию к функции управления.

Примером ПИД-регулятора из повседневной жизни является система круиз-контроля автомобиля. На практике ПИД-контроллер впервые был применен в начале 1920-х годов для создания автоматических систем рулевого управления судов. Затем он использовался для автоматического управления технологическими процессами в обрабатываю-

щей промышленности, где нашёл широкое применение в пневматических, а затем и электронных контроллерах. Сегодня концепция ПИД-регуляции используется в приложениях, требующих точного и оптимизированного автоматического управления.

В системе SoMachine 4.3 для реализации ПИД-регулятора используется функциональный блок PID, который реализует следующий закон регулирования:

$$Y = Y\_OFFSET + KP\left(e(t) + \frac{1}{TN}\int_0^{TN} e(t) + TV\frac{de(t)}{dt}\right)$$

где Y_OFFSET – стационарное значение, KP – коэффициент передачи, TN – постоянная интегрирования, TV – постоянная дифференцирования, $e(t)$ – сигнал ошибки (SET_POINT-ACTUAL).

Входы функционального блока:

Наименование	Тип	Описание
ACTUAL	REAL	Текущее значение контролируемой переменной
SET_POINT	REAL	Уставка
KP	REAL	Коэффициент передачи.
TN	REAL	Постоянная интегрирования, в секундах (т. е. «0.5» для 500 мс)
TV	REAL	Постоянная дифференцирования, в секундах (т. е. «0.5» для 500 мс)
Y_MANUAL	REAL	Определяет значение выхода Y, если MANUAL = TRUE
Y_OFFSET	REAL	Стационарное значение Y
Y_MIN, Y_MAX	REAL	Значение выхода Y ограничено Y_MIN и Y_MAX. При достижении Y границ ограничения, выход LIMITS_ACTVE, (BOOL) принимает значение TRUE. Ограничение работает только при Y_MIN < Y_MAX
MANUAL	BOOL	Значение TRUE, включает режим ручного регулирования по входу Y_MANUAL
RESET	BOOL	TRUE сбрасывает регулятор; в это время Y = Y_OFFSET

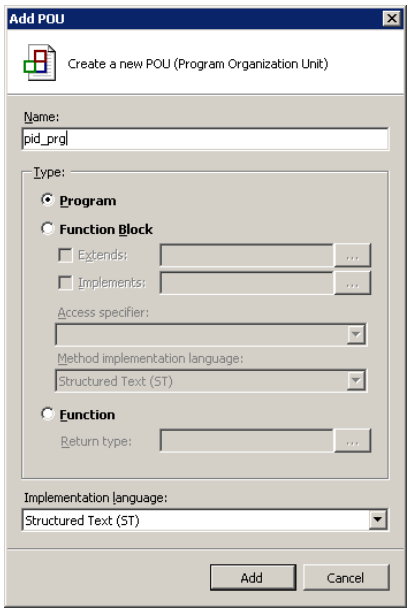
Входы функционального блока:

Наименование	Тип	Описание
Y	REAL	Выход регулятора
LIMITS_ACTIVE OVERFLOW	BOOL BOOL	TRUE означает что Y ограничивается пределами (Y_MIN, Y_MAX). TRUE – признак переполнения.

Неправильная настройка регулятора может вызвать неограниченный рост интегральной составляющей. Для обнаружения такой ситуации предназначен выход OVERFLOW. При переполнении он принимает значение TRUE, одновременно работа регулятора останавливается. Для повторного включения регулятора необходимо использовать рестарт.

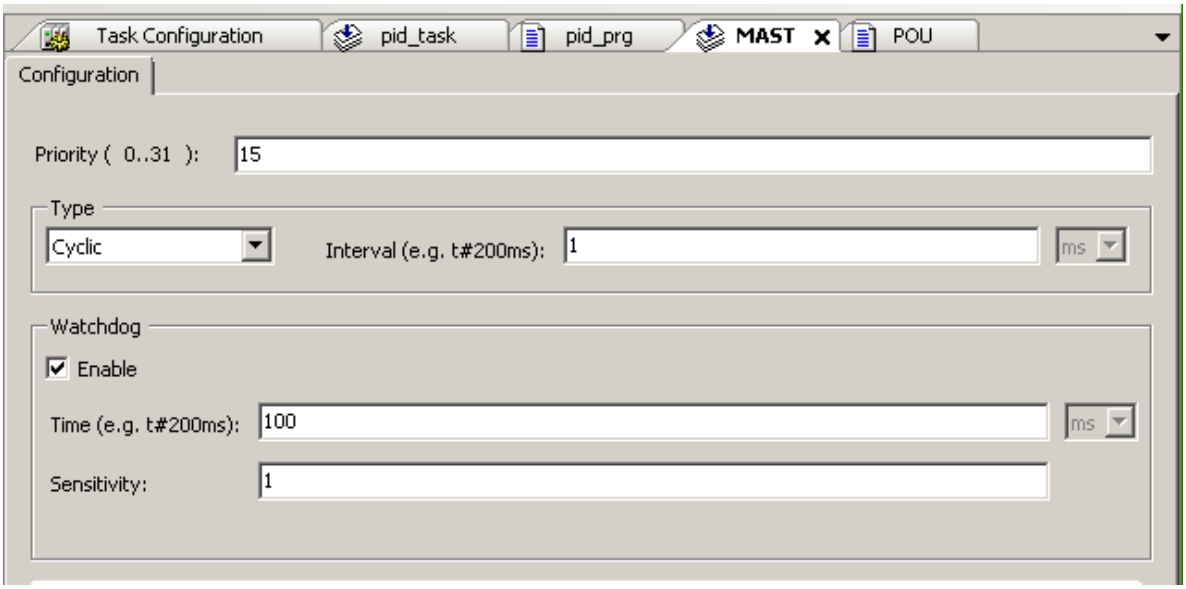
Экспериментальная часть

1. На первом этапе следует открыть проект из предыдущей лабораторной работы.
2. Далее необходимо открыть вкладку *Application Tree* левой панели и добавить в дерево новую программу с произвольным названием. Для этого надо нажать на кнопку «плюс», появляющуюся рядом с названием приложения при наведении на неё курсора мыши. В качестве языка программирования по умолчанию будет выбран *ST*. Теперь в дереве отображаются две программы (POU).

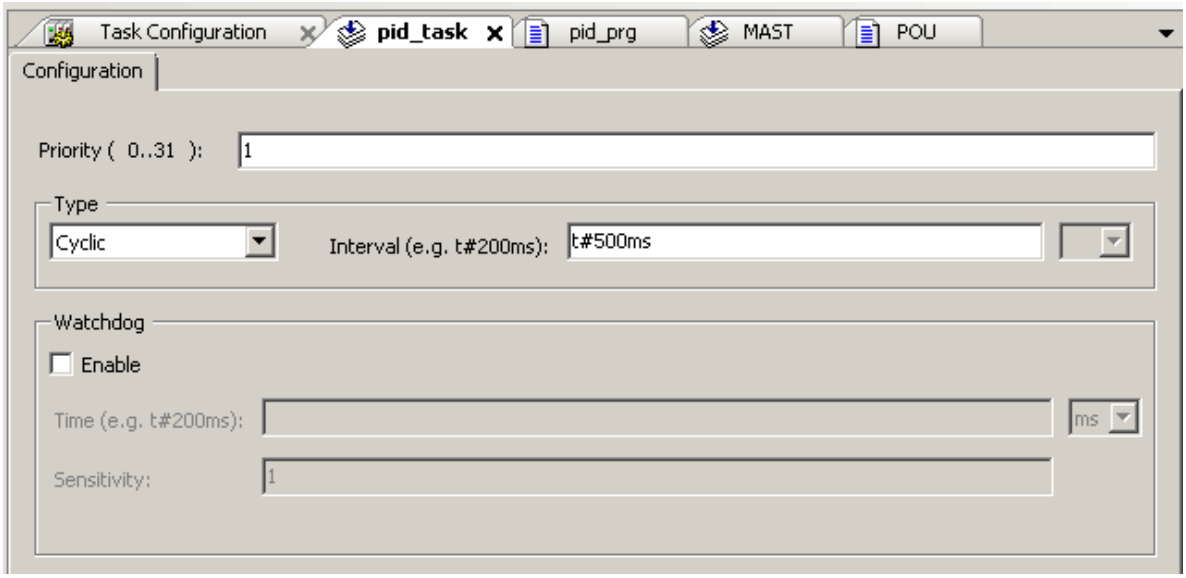


3. Далее необходимо прикрепить программы к задачам. Одна задача (MAST) создается в системе по умолчанию. Дважды щёлкнув по

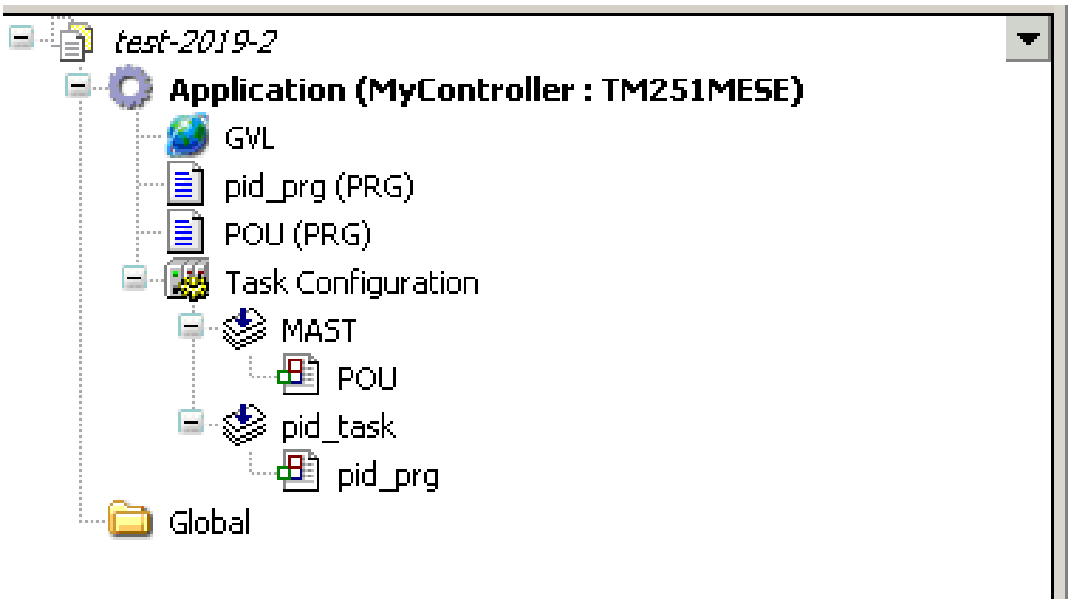
задаче левой кнопкой мыши, необходимо изменить ее параметры следующим образом:



Задача *MAST* является основной, теперь необходимо создать задачу для ПИД-регулятора. По аналогии с созданием *POU* следует навести курсор мыши на узел *Task Configuration* и, когда в строке появится кнопка «плюс», выбрать пункт *Task*, после чего создать циклическую задачу со следующими параметрами:



Для прикрепления программы к задаче необходимо выделить программу в дереве проекта, зажав левую кнопку мыши, а затем, не отпуская кнопки, перетащить на соответствующую задачу. В итоге дерево проекта должно выглядеть следующим образом:



4. Пункт *GVL* содержит глобальные переменные проекта, которые могут быть использованы во всех *POU* и служат одним из вариантов обмена данными между ними.

```
1  VAR_GLOBAL
2      counter_target: REAL := 20;
3  END_VAR
```

5. Откройте код основной программы (*POU*). В верхнее поле между служебными словами VAR и END_VAR записываются локальные переменные программы:

```
1  PROGRAM POU
2  VAR
3      loc_counter: INT := 0;
4  END_VAR
5
```

В нижнее окно помещается исходный код программы:

```
IF (loc_counter >= counter_target) THEN
    qiModule_1_QWO := 10000;
ELSE
    qiModule_1_QWO := 0;
END_IF
IF (loc_counter = 30) THEN
    loc_counter := 0;
ELSE
    loc_counter := loc_counter + 1;
END_IF
```

Алгоритм работы следующий: в программе определён локальный счётчик (*loc_counter*) со значением, по умолчанию равным нулю. На каждой итерации цикла программы счётчик увеличивается на единицу до достижения значения 30, после чего обнуляется. Таким образом, если вспомнить, что цикл исполнения основной задачи (*MAST*) равен 1 мс, то очевидно, что данное значение равняется периоду ШИМ, то есть 30 мс. При этом глобальная переменная *counter_target* задает скважность ШИМ. В данном примере изначальная скважность равна 30 %, то есть 2/3 времени на выход модуля TM3TM3 подается нулевой сигнал (0 В), а 1/3 – максимальный сигнал (10 В).

6. Перейдем к рассмотрению программы ПИД-регулятора.

```
PROGRAM pid_prg
VAR
    pid: PID;

    temp: REAL;
    init: BOOL := TRUE;


END_VAR
```

```
IF (init) THEN
    pid.RESET := TRUE;

    pid.SET_POINT := 268;
    pid.KP := 0.03;
    pid.TN := 0.5;
    pid.TV := 5;
    pid.Y_OFFSET := 0;
    pid.Y_MIN := -1;
    pid.Y_MAX := 30;

    pid.RESET := FALSE;
    init := FALSE;
END_IF

temp := INT_TO_REAL(iiModule_1_IW0);

pid(
    ACTUAL := temp
);

counter_target := pid.Y;
```

В программе определена локальная флаговая переменная *init*, которая имеет изначальное значение «логическая истина». При первом запуске программы выставляются коэффициенты уравнения ПИД-регулятора и граничные условия (данные значения подбираются экспериментально и могут отличаться в зависимости от объекта управления). После этого флаговая переменная обнуляется и, в соответствии с условием, данная часть кода не будет выполняться до следующего перезапуска ПЛК. Далее берется исходное значение температуры и приводится к вещественному типу. Следует обратить внимание, что для функционального блока *PID* важен только тип переменной, дополнительно делить значение на 10, чтобы получить точное значение температуры для правильной работы регулятора, не требуется. Далее температура (*temp*) передается на вход *ACTUAL* функционального блока *PID*, а полученное значение скважности (*pid.Y*) присваивается глобальной переменной *counter_target*. Данное измененное значение будет использовано в основной задаче для изменения выходного напряжения на блоке аналогового ввода/вывода.

7. На последнем этапе необходимо загрузить программу в ПЛК, запустить её и убедиться, что напряжение на выходе изменяется в зависимости от температуры.

Вопросы для самопроверки

1. Опишите принцип широтно-импульсной модуляции.
2. Что такое скважность сигнала?
3. Опишите принцип работы пропорционально-интегрально-дифференциального регулятора.
4. Что произойдет при быстром увеличении интегральной составляющей ПИД регулятора?

Лабораторная работа 3. Передача данных с ПЛК по протоколу Modbus TCP

Цель работы: разработать программу для ПЛК, которая передает в сеть данные с датчиков температуры и влажности по протоколу TCP.

Задание по работе

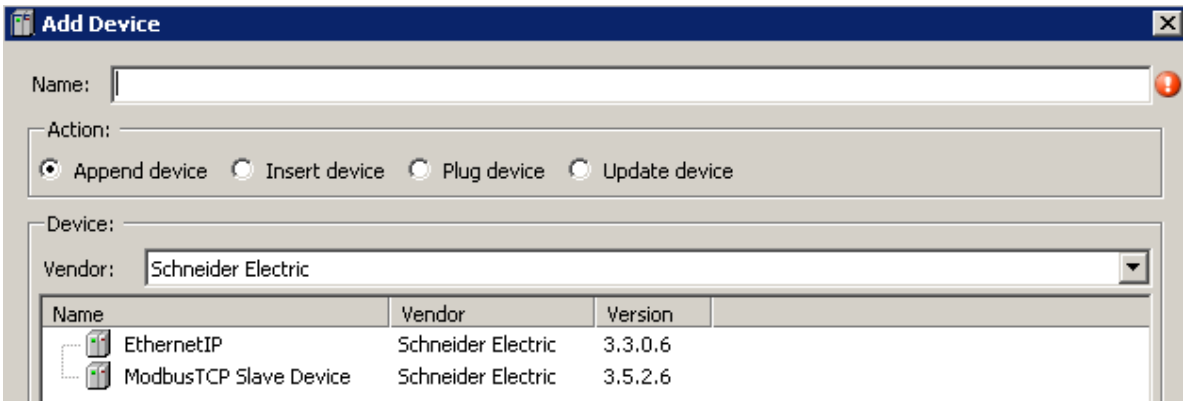
1. Изучить теоретическую часть работы.
2. Реализовать модуль передачи данных в сеть на языке ST.
3. Реализовать модуль получения, обработки и отображения данных на языке Python.

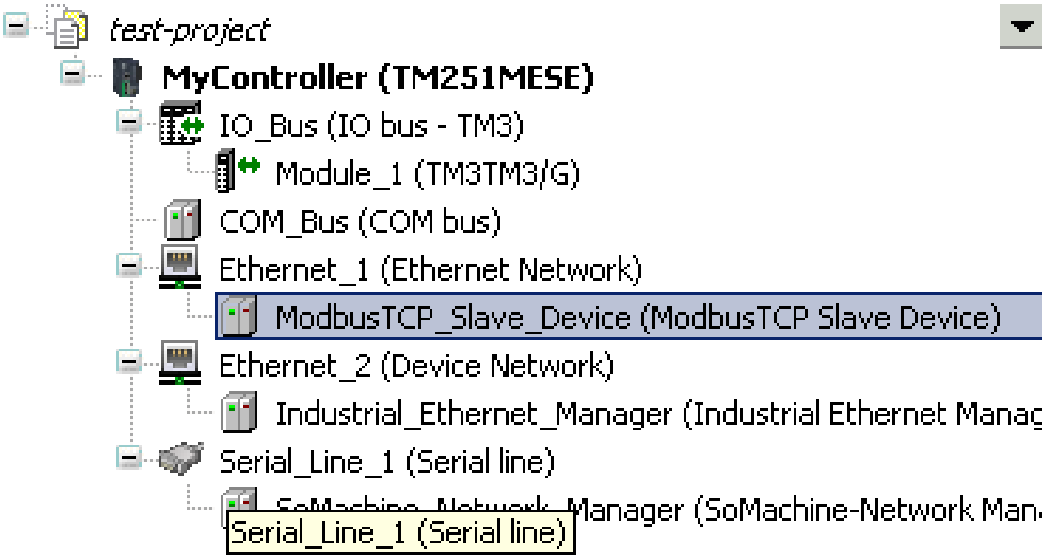
Теоретическая часть

Modbus TCP является вариантом протокола Modbus. Данный протокол является открытым и не зависит от производителя используемого аппаратного обеспечения. Modbus TCP предназначен для контроля и управления оборудованием в системах несложной промышленной автоматизации. В частности, он позволяет реализовать обмен сообщениями Modbus в локальных и глобальных вычислительных сетях с использованием стека протоколов TCP/IP, который в настоящее время все чаще используется для подключения к общей промышленной сети ПЛК, модулей ввода-вывода и «шлюзов». В целом все функции протокола Modbus TCP аналогичны функциям протокола Modbus, описанным во введении.

Экспериментальная часть

1. Создайте пустой проект.
2. В дереве проекта на вкладке *Device Tree* в узле *Ethernet_1* создайте объект *ModbusTCP_Slave_Device*.

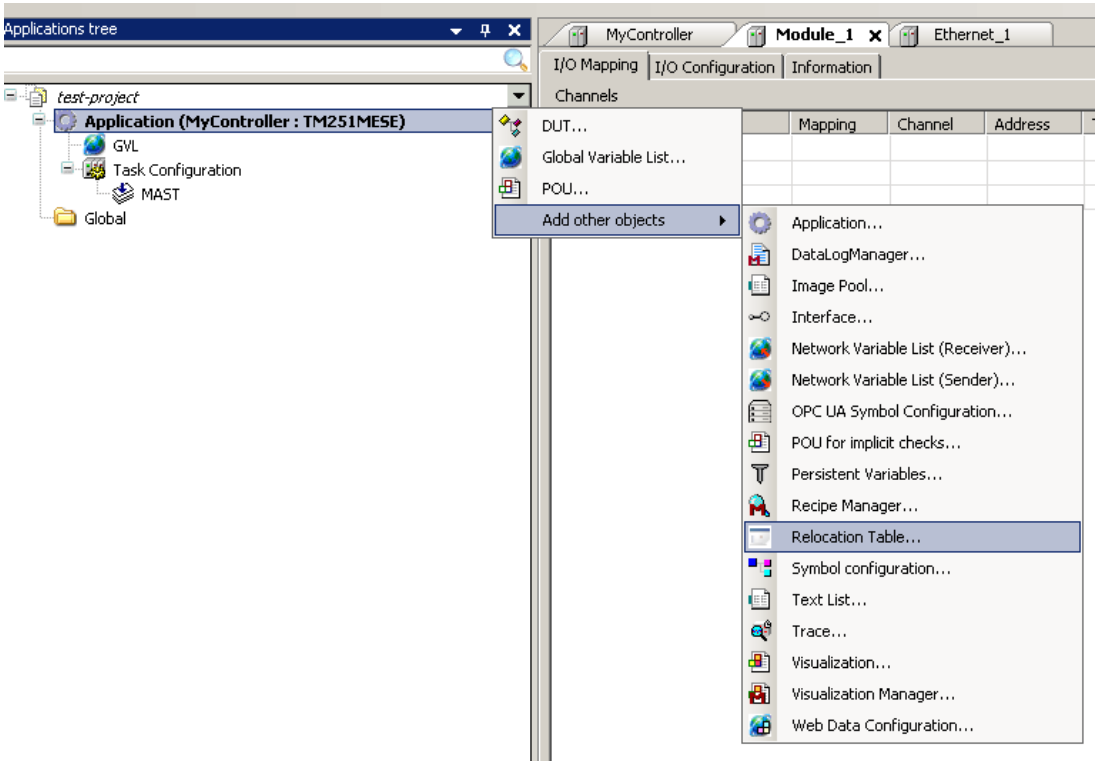




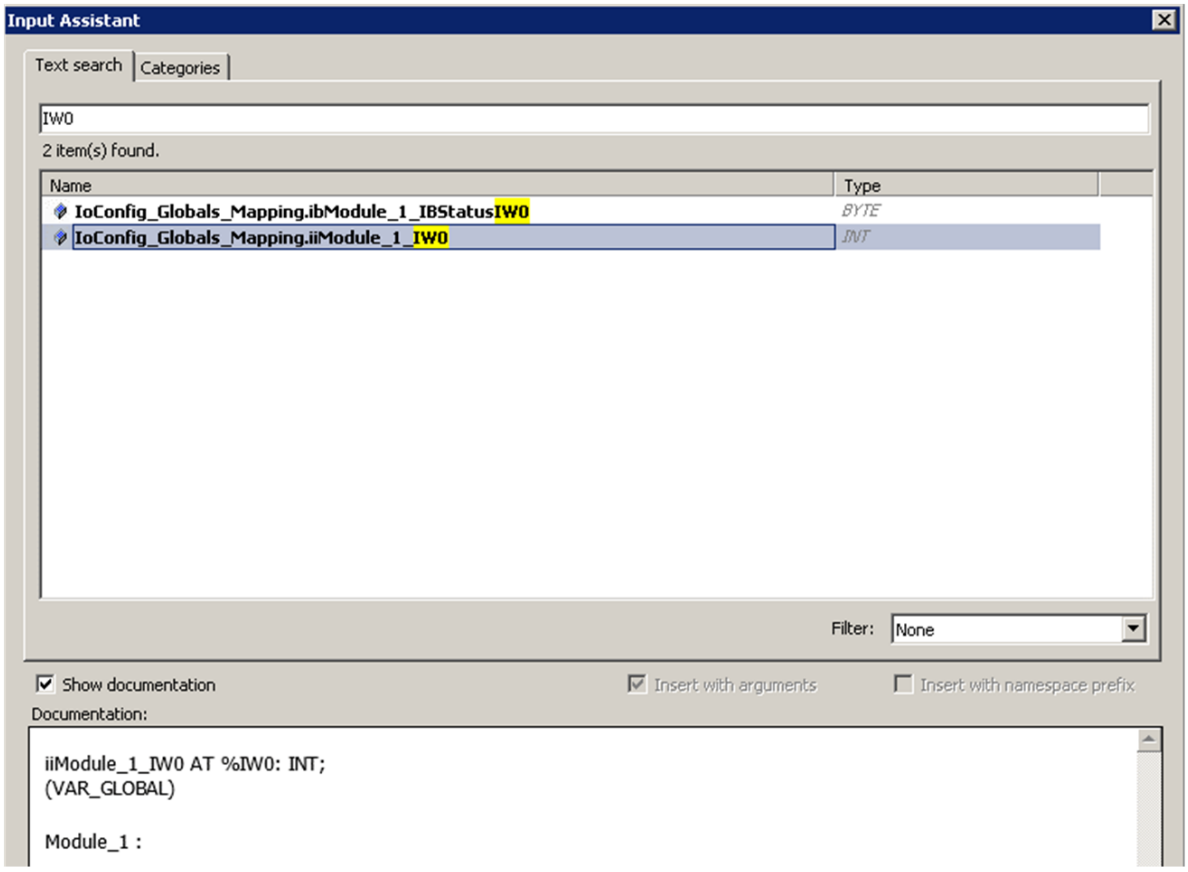
3. Создайте новую глобальную переменную целого типа в узле *GVL* на вкладке *Application Tree*.

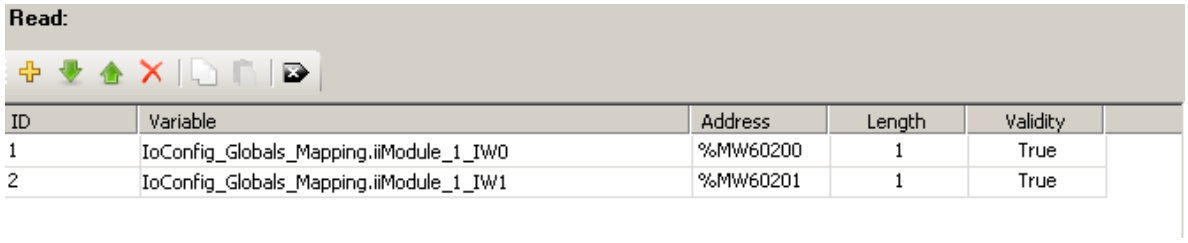
```
1  VAR_GLOBAL
2      foo: INT;
3  END_VAR
```

4. На вкладке *Tools Tree* в корне создайте объект *Relocation Table*.



5. В таблице *Read* создайте две новые пустые переменные Modbus. Для создания новой переменной нажмите на кнопку с символом «плюс».
6. Дважды щелкните левой кнопкой мыши в ячейке *Variable* и в открывшемся диалоге найдите переменные для температуры и влажности из предыдущей работы.





7. Создайте пустую переменную в таблице Write и ассоциируйте её с глобальной переменной из пункта 3.

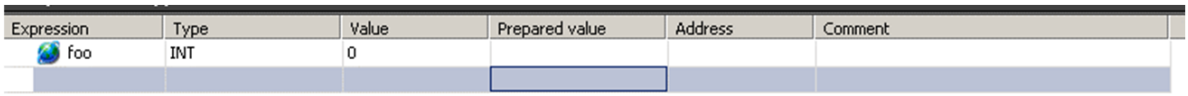
8. Загрузите программу в ПЛК и запустите её.
9. Создайте пустой Python-скрипт и поместите в него следующий код:

```
import time
import random
from pyModbusTCP import utils
from pyModbusTCP.client import ModbusClient


def read_int16(addr):
    regs = c.read_holding_registers(reg_addr=addr, reg_nb=1)
    hex_uint = utils.get_list_2comp(regs, 16)
    return hex_uint[0]

c = ModbusClient(host='10.77.70.120', port=502, auto_open=True)

while True:
    if c.is_open():
        temp = read_int16(60200) / 10
        hum = read_float(60201) / 10
        if(c.write_single_register(62200,random.randint(0,10000))):
            print("ok")

        print(temp, hum)
    else:
        c.open()
    time.sleep(1)
```

Данный скрипт устанавливает соединение с ПЛК по протоколу ModbusTCP, после чего начинает бесконечный циклический опрос регистров, содержащих значения температуры и влажности, а также запись случайных значений в регистр, ассоциированный с глобальной переменной проекта.

10. Запустите скрипт на исполнение и убедитесь, что данные температуры и влажности отображаются в консоли.

11. Перейдите в SoMachine 4.3, откройте вкладку Application Tree и убедитесь, что значение глобальной переменной в узле GVL изменяется каждую секунду.



Вопросы для самопроверки

1. В чем заключается отличие протоколов Modbus и Modbus TCP?
2. Сколько ведущих устройств может быть в сети Modbus?
3. Какое максимальное количество устройств может быть в сети Modbus?
4. Какой порт используется по умолчанию при подключении к ведомым устройствам по протоколу Modbus TCP?

Лабораторная работа 4. Управление частотным преобразователем

Цель работы: разработать программу для управления скоростью и направлением вращения асинхронного двигателя с помощью частотного преобразователя.

Задание по работе

1. Изучить теоретическую часть работы.
2. Изучить документацию на частотный преобразователь.
3. Реализовать модуль для управления частотным преобразователем по протоколу Modbus TCP.

Теоретическая часть

Частотный преобразователь – это электронное или электромеханическое устройство, которое преобразует переменный ток (AC) одной частоты в переменный ток другой частоты. Устройство также может изменять напряжение, но это является второстепенным по отношению к его основной цели, поскольку преобразование напряжения переменного тока гораздо проще реализовать, нежели преобразование частоты.

Традиционно подобные устройства представляли собой электромеханические машины, называемые мотор-генераторными установками. Также использовались устройства с ртутными дуговыми выпрямителями или электронными вакуумными трубками. С появлением полупроводниковой электроники стало возможным создавать полностью твердотельные преобразователи частоты.

Эти устройства обычно состоят из выпрямителя (вырабатывающего постоянный ток), который затем инвертируется для выработки переменного тока желаемой частоты. В инверторе могут быть использованы обычные тиристоры, IGCT[5] или IGBT[6]. Если требуется преобразование напряжения, трансформатор обычно включается во входную или выходную схему переменного тока, и этот трансформатор также может обеспечивать гальваническую развязку между входной и выходной цепями.

---

[5] от англ. *Integrated Gate-Commutated Thyristor* – коммутируемый тиристор с интегрированным управлением.

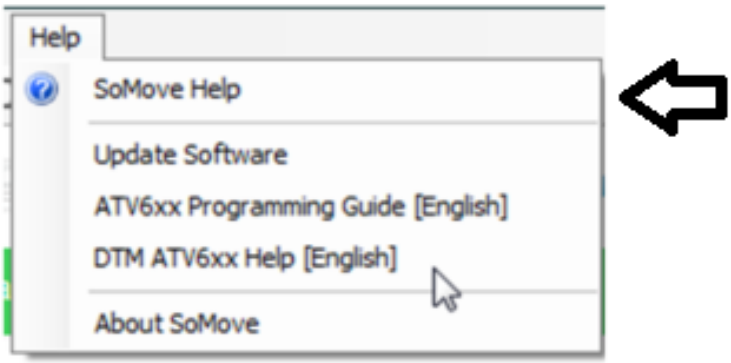
[6] от англ. *Insulated-Gate Bipolar Transistor* – биполярный транзистор с изолированным затвором.

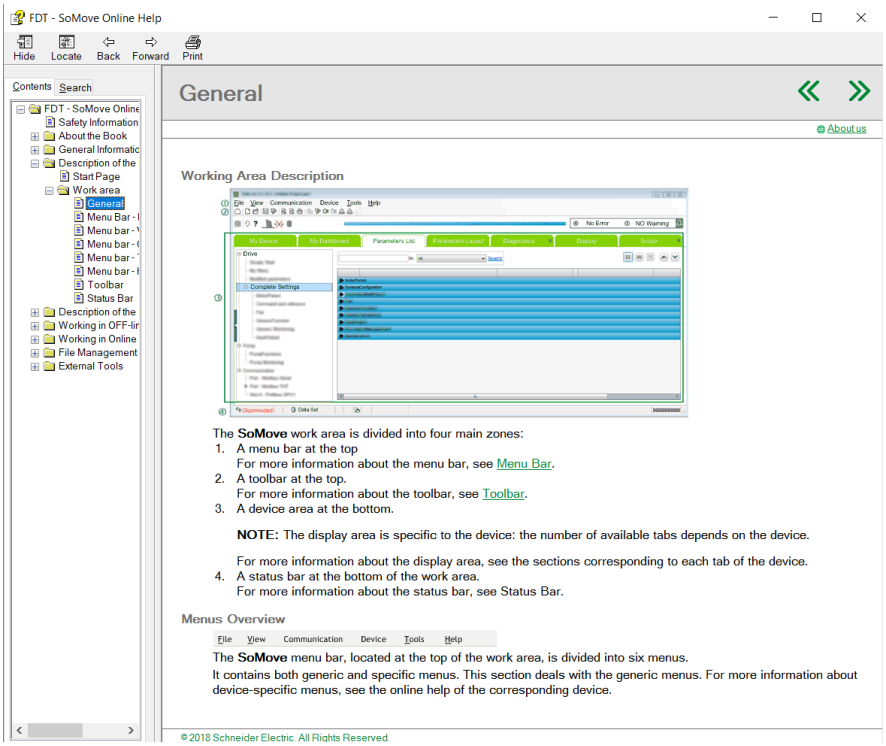
Для улучшения защиты частотного преобразователя от кратковременных отключений питающей сети в схему постоянного тока также может быть добавлена аккумуляторная батарея.

Как известно, скорость вращения асинхронного двигателя зависит от частоты электросети и не может быть изменена за счёт изменения питающего напряжения. Поэтому наиболее частым применением частотных преобразователей является управления трехфазными асинхронными двигателями. Современные частотные преобразователи являются сложными электронными устройствами, в которых реализована обратная связь и т. н. бездатчиковое векторное управление с адаптивным наблюдателем скорости и непосредственной коррекцией электрического угла. Данная система позволяет поддерживать постоянную заданную скорость вращения, а также постоянный момент практически во всем диапазоне регулирования.

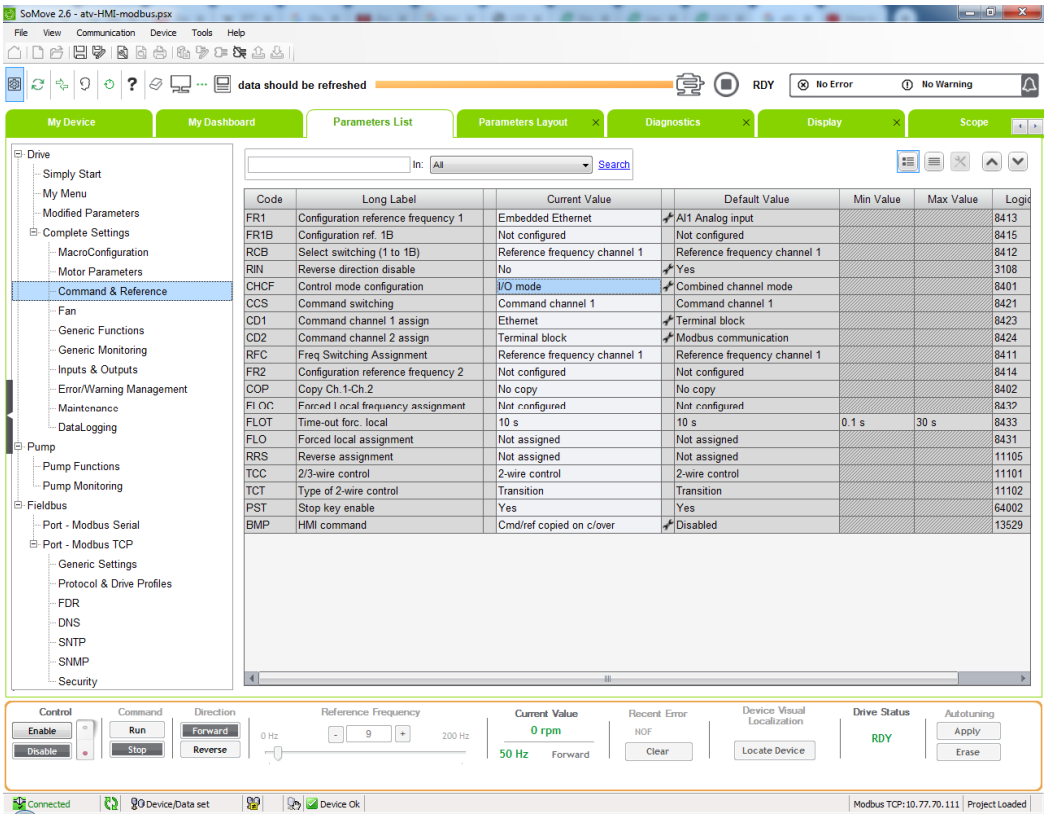
Экспериментальная часть

1. Ознакомьтесь с документацией на частотный преобразователь Altivar ATV 630, размещенной по адресу:
   `https://www.se.com/ww/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/294000/FA294964/en_US/ATV6xx_Programming_Manual_EN_EAV64318_03.pdf`
2. Ознакомьтесь с таблицей регистров Modbus частотного преобразователя, размещенной по адресу:
   `https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=ATV600_Communication_parameters_EAV64332_V2.5.xlsx&p_Doc_Ref=EAV64332`
3. Ознакомьтесь с документацией программы конфигурации силовых приводов SoMove. Для доступа к документации необходимо зайти в программу и нажать клавишу F1 или открыв пункт главного меню Help → Somove Help.

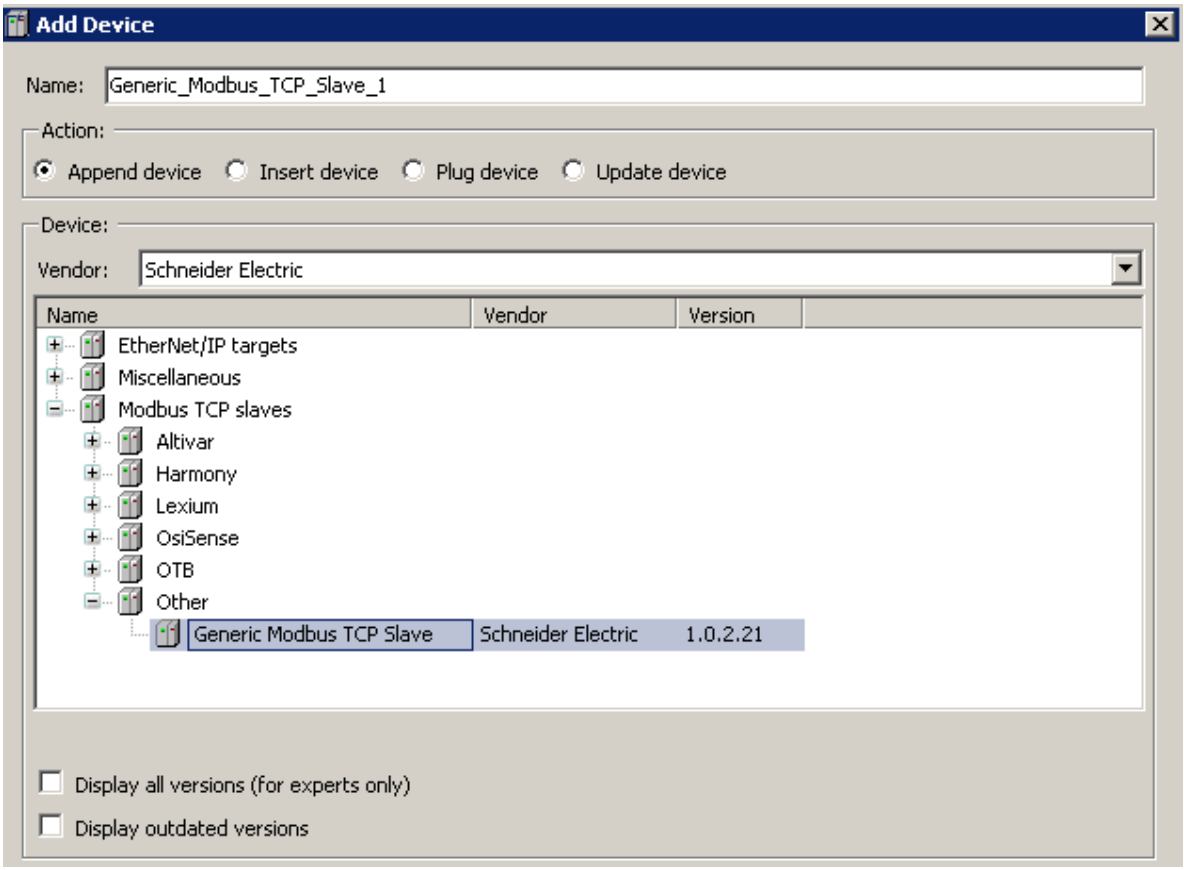


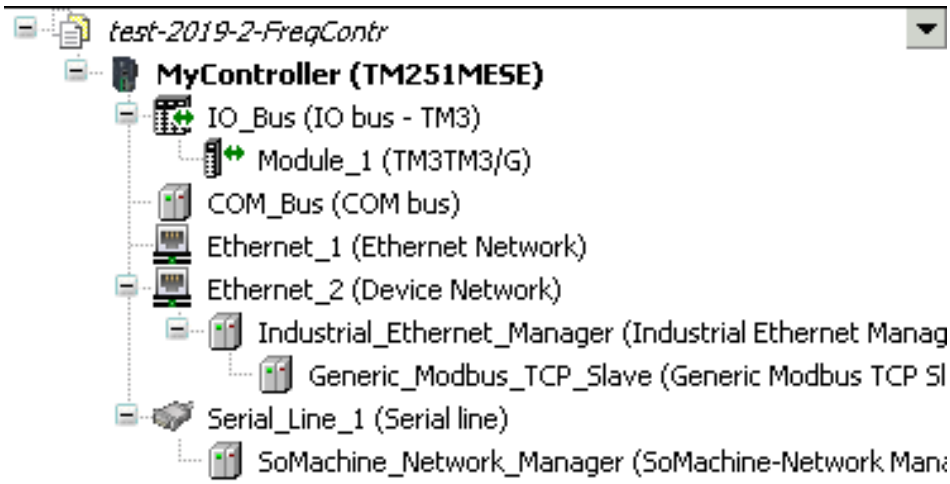


4. Запустите программу SoMove и убедитесь, что параметры соединения с частотным преобразователем соответствуют представленным на рисунке:

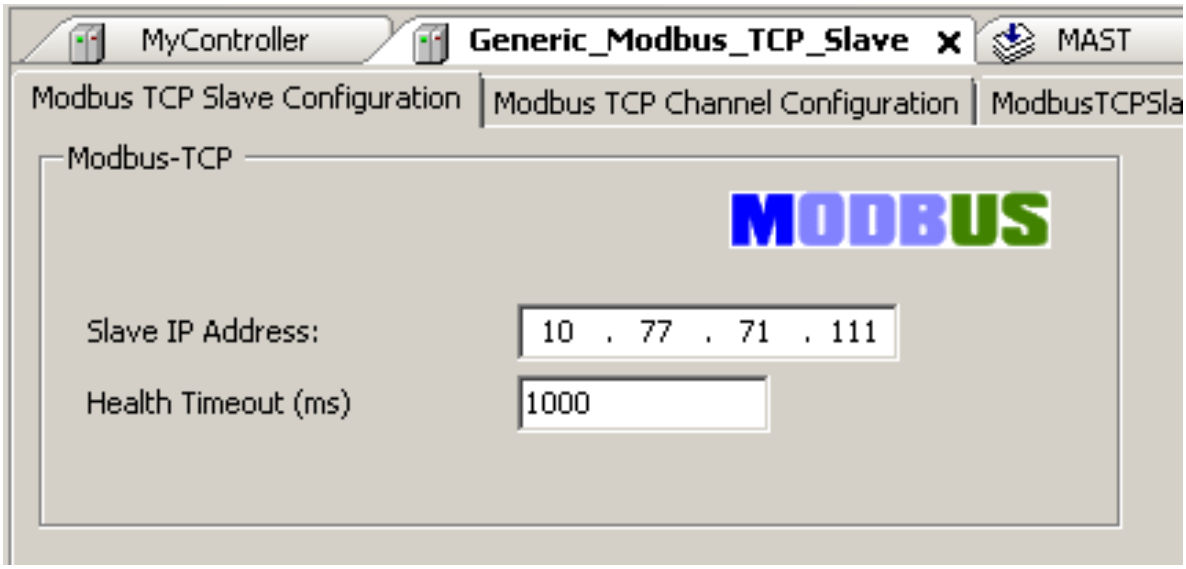


5. Откройте в программе SoMachine 4.3 проект из лабораторной работы 2. В дереве устройств добавьте устройство *Generic_ModbusTCP_Slave* к узлу *Ethernet_2*.



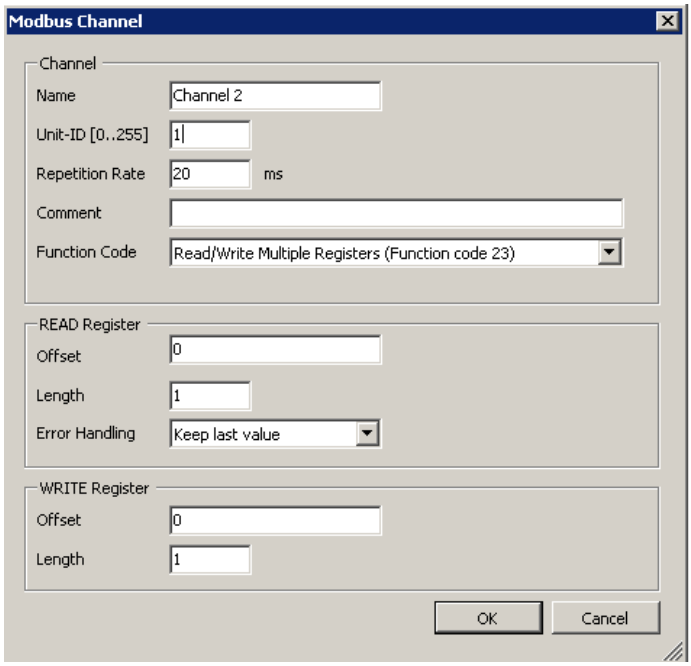


6. Задайте IP адрес частотного преобразователя. Обратите внимание, что устройство должно быть подключено ко второму порту Ethernet на ПЛК.



7. На вкладке *Modbus TCP Channel Configuration* нажмите кнопку *Add Channel*:

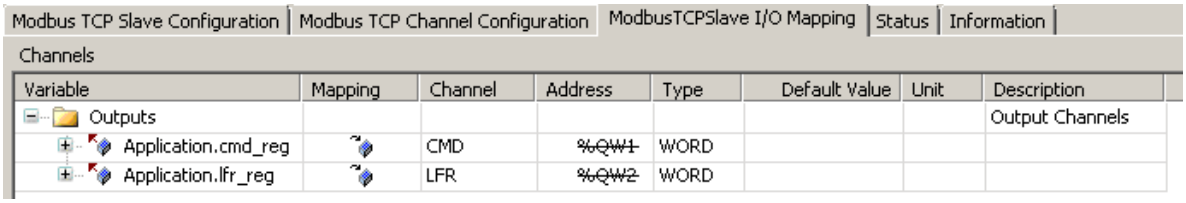
8. В открывшемся диалоговом окне задайте два канала управления для включения частотного преобразователя (регистр CMD, адрес 8501) и частоты (регистр LFR, адрес 8501). Адреса и описание регистров можно найти в таблице из пункта 2 настоящей лабораторной работы.



9. Для описанных выше регистров в узле GVL создайте две новые глобальные переменные:

```
VAR_GLOBAL
    counter_target: REAL := 20;

    cmd_reg: WORD := 0;
    lfr_reg: INT := 0;

END_VAR
```

10. После этого ассоциируйте их с каналами управления на вкладке *ModbusTCPSlave I/O Mapping*:



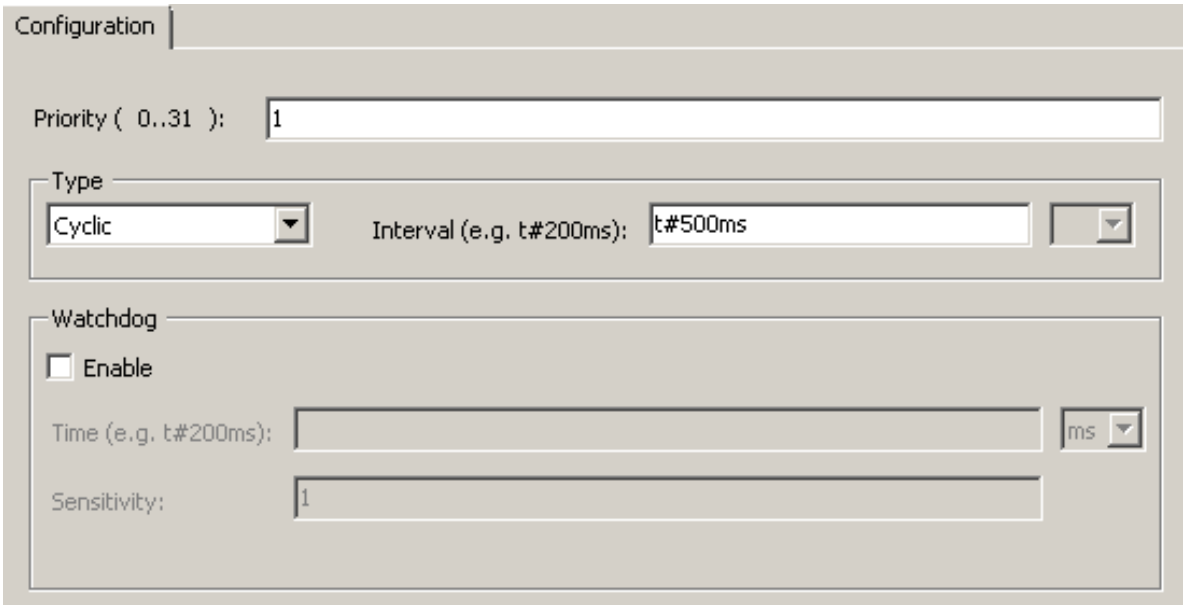
11. На вкладке Application *Tree* добавьте новую программу с произвольным именем. Создайте следующую локальную переменную:

```
PROGRAM atv_prg
VAR
    count : INT := 0;
END_VAR
```

и допишите текст программы:

```
lfr_reg := count * 50;
cmd_reg := 1;

IF (count = 10) THEN
    count := 0;
ELSE
    count := count + 1;
END_IF
```

12. Создайте *циклическую* задачу и ассоциируйте ее с программой управления скоростью частотного преобразователя:




```
1   lfr_reg := count * 10 ;
2   cmd_reg := 1;
3
4   IF (count = 10) THEN
5       count := 0;
6   ELSE
7       count := count + 1;
8   END_IF
```

13. Программа работает следующим образом. Для включения частотного преобразователя необходимо в нулевой бит регистра CMD записать значение 1, то есть в нашем случае просто присвоить значение 1 всей переменной. Изначальная частота вращения двигателя равна 0 (переменная *ifr_reg*).
На каждом цикле программы мы берем локальный счётчик, умножаем его значение на 10, а затем записываем в регистр частоты, после чего инкрементируем счётчик, а при достижении значения 10 обнуляем его. Таким образом частота плавно увеличивается от 0 до 100 Гц.
При этом скорость вращения двигателя будет зависеть от его

паспортных характеристик, например, если на шильдике двигателя написано 50 Гц, 1400 об/мин., частота будет меняться в диапазоне от 0 до 2800 об/мин.

14. Перепишите программу таким образом, чтобы частота вращения двигателя зависела от показаний датчика влажности, эмулируя таким образом простейшую климатическую систему, которая включает вентилятор, если в помещении становится слишком влажно.

Вопросы для самопроверки

1. Объясните принцип работы частотного преобразователя.
2. Какие два основные регистра Modbus используются для управления частотным преобразователем,
3. Почему в ПЛК, как правило, используются два интерфейса Ethernet?

Список литературы

1. Смирнов, Ю.А. Технические средства автоматизации и управления : учебное пособие / Ю.А. Смирнов. — 2-е изд., стер. — Санкт-Петербург : Лань, 2018. — 456 с. — ISBN 978-5-8114-2376-7. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/109629 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

2. Рег, Д. Промышленная электроника : учебник / Д. Рег. — Москва : ДМК Пресс, 2011. — 1136 с. — ISBN 978-5-94074-478-8. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/891 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

3. Гаврилов, А.Н. Средства и системы управления технологическими процессами : учебное пособие / А.Н. Гаврилов, Ю.В. Пятаков. — 3-е изд., стер. — Санкт-Петербург : Лань, 2019. — 376 с. — ISBN 978-5-8114-4584-4. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/122190 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

4. Бишоп, О. Электронные схемы и системы : учебное пособие / О. Бишоп ; перевод с английского А.Н. Рабодзей. — Москва : ДМК Пресс, 2016. — 576 с. — ISBN 978-5-97060-172-3. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/93262 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

5. Аппаратные и программные средства встраиваемых систем : учебное пособие / А.О. Ключев, Д.Р. Ковязина, П.В. Кустарев, ПлатуновА.Е.. — Санкт-Петербург : НИУ ИТМО, 2010. — 290 с. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/40708 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

6. Смирнов, Ю.А. Основы микроэлектроники и микропроцессорной техники : учебное пособие / Ю.А. Смирнов, С.В. Соколов, Е.В. Титов. — 2-е изд., испр. — Санкт-Петербург : Лань, 2013. — 496 с. — ISBN 978-5-8114-1379-9. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/12948 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

7. Музипов, Х.Н. Программно-технические комплексы автоматизированных систем управления : учебное пособие / Х.Н. Музипов. — Санкт-Петербург : Лань, 2018. — 164 с. — ISBN 978-5-8114-3133-5. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/108458 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

8. Интерфейсы периферийных устройств : учебное пособие / А.О. Ключев, Д.Р. Ковязина, Е.В. Петров, А.Е. Платунов. — Санкт-Петербург : НИУ ИТМО, 2010. — 290 с. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/43548 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

9. Вороненко, В.П. Проектирование машиностроительного производства : учебник / В.П. Вороненко, М.С. Чепчуров, А.Г. Схиртладзе ; под редакцией В.П. Вороненко. — 2-е изд., стер. — Санкт-Петербург : Лань, 2019. — 416 с. — ISBN 978-5-8114-4519-6. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/121984 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

10. Бойков, В.И. Интегрированные системы проектирования и управления / В.И. Бойков, Г.И. Болтунов, О.К. Мансурова. — Санкт-Петербург : НИУ ИТМО, 2010. — 163 с. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/40736 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

11. Технологии создания интеллектуальных устройств, подключенных к интернет : учебное пособие / А.В. Приемышев, В.Н. Крутов, В.А. Треяль, О.А. Коршакова. — 2-е изд., стер. — Санкт-Петербург : Лань, 2018. — 100 с. — ISBN 978-5-8114-2310-1. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/103911 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

12. Ли, П. Архитектура интернета вещей / П. Ли ; перевод с английского М.А. Райтман. — Москва : ДМК Пресс, 2019. — 454 с. — ISBN 978-5-97060-672-8. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: https://e.lanbook.com/book/112923 (дата обращения: 10.11.2019). — Режим доступа: для авториз. пользователей.

13. Широтно-импульсная модуляция [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 102198508, сохранённая в 18:46 UTC 16 сентября 2019 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2019. — Режим доступа: https://ru.wikipedia.org/?oldid=102198508.

14. ПИД-регулятор [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 104077899, сохранённая в 11:32 UTC 22 декабря 2019 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2019. — Режим доступа: https://ru.wikipedia.org/?oldid=104077899.

15. Частотный преобразователь (электропривод) [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 100312529, сохранённая в 13:02 UTC 9 июня 2019 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2019. — Режим доступа: https://ru.wikipedia.org/?oldid=100312529.

16. Скважность [Электронный ресурс] : Материал из Википедии — свободной энциклопедии : Версия 103037954, сохранённая в 12:31 UTC 30 октября 2019 / Авторы Википедии // Википедия, свободная энциклопедия. — Электрон. дан. — Сан-Франциско: Фонд Викимедиа, 2019. — Режим доступа: https://ru.wikipedia.org/?oldid=103037954

Приложение. Пример оформления отчета по лабораторной работе

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет Систем Управления и Робототехники

Дисциплина: _____

Отчет

по лабораторной работе № ___
Полное название лабораторной работы

Студент: _____
Группа: _____
Преподаватель: _____

Санкт-Петербург
20__ г.

Цель работы: в данном разделе должна быть сформулирована цель работы. Допускается копирование цели работы из методических указаний.

Задачи, решаемые в работе:

1. Задача 1.
2. Задача 2.
3. Задача N.

Теоретическая часть

Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать цель и перечень задач, решаемых в работе. Во время защиты лабораторной работы студенту следует ясно и чётко изложить цель работы и основные решённые задачи, а также ответить на дополнительные вопросы, заданные преподавателем в процессе защиты отчёта.

Теоретическая часть отчёта по лабораторной работе должна включать в себя всю необходимую информацию о предметной области, к которой относятся описание работы программных и аппаратных компонентах, используемых при ее выполнении, выдержки из технической документации и т. п.

Данный раздел не является обязательным, однако рекомендуется оставлять его в отчете, так как он может быть использован как план ответа при защите лабораторной работы. В случае, если теоретическая части отчета по лабораторной работе содержит листинги программного кода, таблицы или рисунки, то они должны быть оформлены по правилам, описанным далее в этом разделе.

Часть отчета, описывающая экспериментальную или практическую часть лабораторной работы, является обязательной и не может быть опущена. Данная часть отражается персональные результаты студента, полученные им при выполнении лабораторной работы. Этими результатами являются листинги написанного программного кода, таблицы и изображения, полученные в процессе решения сформулированных задач. Исходные данные, использованные студентом для практической проверки реализованных в ходе практической работы программных компонентов, также должны быть представлены в экспериментальной части. Данные по выполненной лабораторной работе можно предоставить отдельно, в том числе в виде ссылки на интернет-ресурс. При изучении ранее выполненных работ студентам рекомендуется самостоятельно переписать всю программу в том виде, в каком она была написана, с целью лучшего понимания алгоритма программы. Если студент не завершил цикл выполнения программы, должно быть проведено ее

окончательное редактирование. В экспериментальную часть отчета по лабораторной работе необходимо включить пояснения и комментарии к написанному программному коду (если написание программного кода является одной из задач лабораторной работы).

Выводы должны быть кратко изложены в виде списка и отражать наиболее важные аспекты лабораторной работы. В конце отчета студент должен привести список использованной при подготовке отчета и процитированной литературы (в том числе ссылки на стандарты, руководства пользователя и прочую техническую документацию). Текст отчета по лабораторной работе может являться планом ответа при защите лабораторной работы. Прямое чтение текста не допускается.

Экспериментальная часть

Правила оформления текста

Отчет должен быть представлен в формате PDF. Поля текста могут быть выбраны произвольно, например, 25 мм по всем сторонам печатного листа. Рекомендуется при оформлении отчета использовать шрифт чёрного цвета с засечками (например, DejaVu Serif) высотой 14 пунктов с полуторным интервалом между строками. Обязательным требованием при оформлении текста отчёта по лабораторной работе является его выравнивание по ширине страницы и использование переносов.

На рисунках в отчете могут быть представлены блок-схемы, графики, изображения аппаратных средств разработки, а также прочая графическая информация. Рисунки должны быть чётки и легкочитаемыми. Если рисунок является снимком экрана, то настоятельно рекомендуется сохранять исходное в форматы без сжатия (например, PNG) или копировать снимок экрана из буфера обмена непосредственно в программу, в которой редактируется текст отчета (например, LibreOffice). Графики и диаграммы рекомендуется создавать в векторных форматах. Ниже представлен пример оформления рисунка.

Рис. 20. Пример оформления рисунка

Листинги программного кода должны оформляться как скриншоты окна редактора, либо непосредственно в тексте отчета. В обоих случаях должны быть соблюдены следующие правила:

1. Необходимо использовать моноширинные шрифты.
2. Строки программы должны быть пронумерованы, нумерация должна начинаться с единицы.
3. Должна быть включена «подсветка» синтаксических конструкций используемого языка программирования.

Исходные коды должны быть снабжены комментариями. На рисунки и листинги обязательно должны присутствовать ссылки в тексте. Например "*На рисунке 1 изображен ...* " или *"Исходный код программы представлен в листинге 2 ...*". Листинги, таблицы и рисунки должны быть снабжены подписями, отражающими их содержание.

Выводы

В выводах необходимо кратко пояснить полученные студентом в ходе выполнения лабораторной работы результаты и следующие из них выводы, а также определить, достигнута ли сформулированная цель.

Крылова Анастасия Андреевна
Шорохов Сергей Александрович
Афанасьев Максим Яковлевич

Основы проектирования систем автоматизации технологических процессов с применением ПЛК

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверский пр., 49