Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

А. В. Ковалев, Д. А. Литвинов

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ

ПРАКТИКУМ
по одноименной дисциплине для студентов специальности 1-53 80 01 «Автоматизация и управление технологическими процессами и производствами» дневной и заочной форм обучения

Гомель 2020

Библиотека ГГТУ им.П.О.Сухого

УДК 621.3.049.75(075.8)
ББК 32я73
К56

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 7 от 04.03.2019 г.*)

Рецензент: доц. каф. «Автоматизированный электропривод» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *В. А. Захаренко*

Ковалев, А. В.

К56 Программируемые логические контроллеры : практикум по одноим. дисциплине для студентов специальности 1-53 80 01 «Автоматизация и управление технологическими процессами и производствами» днев. и заоч. форм обучения / А. В. Ковалев, Д. А. Литвинов. – Гомель : ГГТУ им. П. О. Сухого, 2020. – 128 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Предназначен для получения и закрепления знаний, требуемых в рамках учебной программы по предмету «Программируемые логические контроллеры» на практических, лабораторных занятиях и при самостоятельной работе.

Для студентов специальности 1-53 80 01 «Автоматизация и управление технологическими процессами и производствами» дневной и заочной форм обучения.

УДК 621.3.049.75(075.8)
ББК 32я73

© Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», 2020

Библиотека ГГТУ им.П.О.Сухого

1. Внутренняя архитектура систем на базе ПЛК. Стандарты средств связи цифровых микропроцессорных систем управления с ПЛК

1.1.Типовая архитектура серийных ПЛК

Потребность в применении программируемых логических контроллеров (ПЛК) обозначилась в 60 – х годах прошлого века.

Термин ПЛК (Programmable Logic Controller, PLC) впоследствии был определен в стандартах EN 61131 (МЭК 61131). ПЛК – это унифицированная цифровая управляющая электронная система, специально разработанная для использования в производственных условиях. ПЛК постоянно контролирует состояние устройств ввода и принимает решения на основе пользовательской программы для управления состоянием выходных устройств.

Упрощенное представление состава и принципа действия ПЛК хорошо демонстрирует рисунок 1.1. Из него видно, что ПЛК имеет три основные секции:

- входную;
- выходную;
- центральную.

Рисунок 1.1 – Структура и функционал ПЛК

Библиотека ГГТУ им. П.О. Сухого

Центральная секция содержит центральный процессор (ЦП), память и систему коммуникаций. Она выполняет обработку данных, принимаемых от входной секции данных, и передает результаты обработки в выходную секцию. Следует сразу отметить, что в больших ПЛК, кроме ЦП, действующего в режиме «ведущий», могут быть дополнительные «ведомые» ПЛК со своими ЦП. В качестве ЦП небольшого ПЛК используются стандартные микропроцессоры (МП). Обычно 8- и 16-разрядные МП вполне справляются со всеми стандартными задачами. Но, как отмечено в МЭК 61131, выбор конкретного МП все же зависит от задач, возлагаемых на данный тип ПЛК.

Для многих технических задач применяемые средства автоматизации строились преимущественно на релейно – контактных элементах и за значительный период своего использования обнаружили целый ряд присущих им недостатков:

- для разработки, обслуживания и ремонта таких систем требовались значительные кадровые и экономические ресурсы, так как каждая отдельная схема создавалась под конкретную, и только под неё, задачу;
- переналадка схемы на решение другой задачи была невозможна без полной или кардинальной разборки её, и, если нужно, - с возможностью повторного использования компонентов – это трудоёмко и неудобно;
- затруднительно объединение в единую структуру фрагментов системы, территориально удалённых друг от друга;
- практически невозможно построить схему на реле, выдержав её в минимальных габаритах.

Только с появлением ПЛК, построенных на микропроцессорах, удалось сосредоточить в конструктивно очень компактном модуле сотни и даже тысячи «релейных» элементов, счётчиков, таймеров, пусть даже не существующих физически, а программно воспроизводимых. Это позволило создать гибко переналаживаемую структуру, способную выполнить любую из очень широкого круга задач.

Возможности программируемых логических контроллеров делают их практически незаменимыми для автоматизации насосных станций, компрессорных установок, котельных, конвейеров, для управления технологическими процессами в комплексе с датчиками самого различного вида, приводными устройствами, клапанами,

Библиотека ГГТУ им. П.О.Сухого

задвижками и т. д. по схеме взаимодействия с объектом, изображенной на рис. 1.2.

Именно поэтому первоначально сформулированной причиной появления ПЛК была названа задача вытеснения из управляющих устройств элементов релейно – контактной техники. Даже больше того, некоторые фирмы – производители (Schneider Electric, например) называют свою продукцию интеллектуальными реле.

Конечно, возможности ПЛК не безграничны. Некоторые из них обладают небольшим быстродействием, не всегда удобно сопрягаются по входным и выходным цепям. Они оперируют данными не в кодовой форме, а с их логическим или числоимпульсным представлением и т.д. Но, тем не менее, игнорирование их достоинств в настоящее время объяснимо только для тех, у кого принятие технических решений на уровне многолетней давности стало непререкаемым предпочтением.

Рисунок 1.2 - Схема взаимодействия ПЛК и объекта управления

Связь контроллера с окружающей средой осуществляется через устройства ввода и вывода. Через первые в контроллер вводятся сигналы о параметрах и характеристиках объекта управления, через вторые – выводятся управляющие воздействия на включение / выключение исполнительных устройств. Для прохождения сигналов

Библиотека ГГТУ им. П.О. Сухого

через порты ввода / вывода эти устройства реализуются не программно, они должны быть физически существующими.

Множество внутренних реле, счётчиков, таймеров, необходимых для составления и работы программы, физически не существуют, они моделируются центральным процессором (ЦП) контроллера. Именно благодаря этому удалось в ограниченном объёме «разместить» огромное количество блоков, каждый из которых предназначен для имитации выполнения некоторой вполне конкретной задачи. Более подробно состав и взаимодействие отдельных компонентов внутренней структуры ПЛК раскрываются на рис. 1.3.

Тот факт, что внутренняя структура ПЛК основывается на множестве программно моделируемых функционально законченных блоков, совершенно иначе ставит вопрос о технике программирования задач для ПЛК. В какой бы форме ни обращались данные внутри контроллера, очевидно, что среда программирования позволяет программисту работать без необходимости обращаться к Ассемблеру, а на некотором упрощенном языке. С точки зрения пользователя несущественными становятся углублённые представления о работе микропроцессорных устройств, о составе компонентов и объединении их в единую систему. Это значительно сглаживает требования к уровню квалификации пользователей и в немалой степени способствует росту привлекательности использования логических контроллеров.

Успешность применения ПЛК для задач управления зависит от того, насколько подробно и правильно в контроллер вводится информация о состоянии и поведении объекта. Чтобы грамотно встроить ПЛК в разрабатываемую систему управления (СУ), достаточно выполнить несколько очевидных, но, тем не менее, очень важных правил, основные из которых следующие:

- информация о наиболее важных параметрах и характеристиках объекта, определяющих особенности управления в данной задаче, должна быть введена в контроллер;

-форма представления сигналов должна быть такой, которую контроллер в состоянии правильно воспринять.

В связи с этим надо хотя бы в краткой форме дать характеристику наиболее часто встречающимся видам сигналов, тому, в какой форме они могут быть заданы, и какие схемы сопряжения контроллера с источниками сигналов при этом используются.

Библиотека ГГТУ им. П.О.Сухого

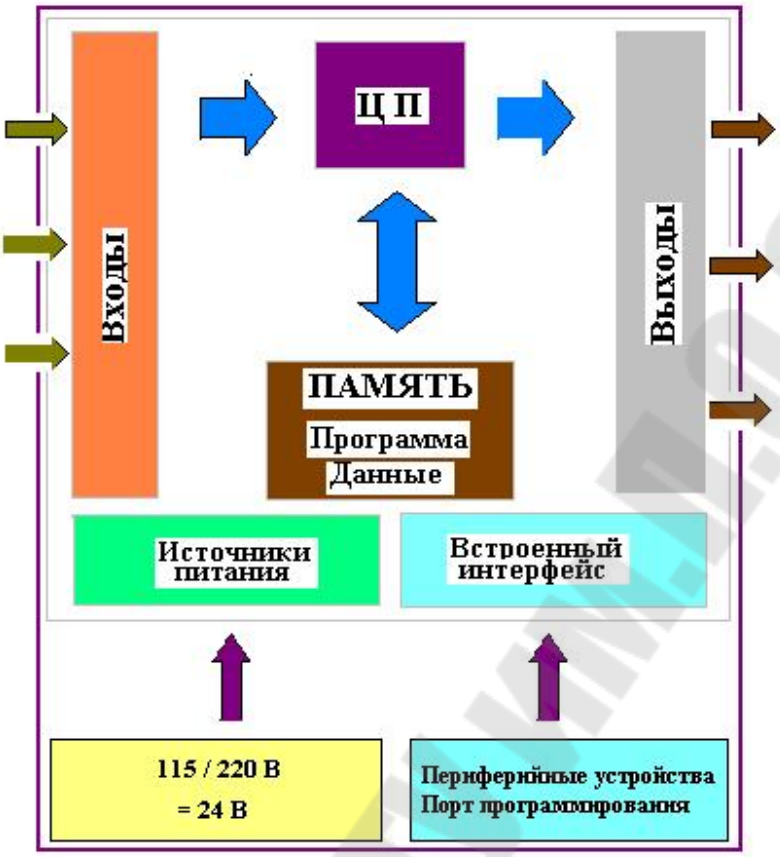


Рисунок 1.3 – Внутренняя структура типового ПЛК

Современные ПЛК, использующие инновационные технологии, далеко ушли от первых упрощенных реализаций промышленного контроллера, но заложенные в систему управления универсальные принципы были стандартизированы и успешно развиваются уже на базе новейших технологий.

Крупнейшими мировыми производителями ПЛК сегодня являются компании Siemens AG, Allen-Bradley, Rockwell Automation, Schneider Electric, Omron. Кроме них ПЛК выпускают и многие другие производители, включая российские компании ООО КОНТАР, Овен, Сегнетикс, Fastwel Групп, группа компаний Текон и другие.

По конструктивному исполнению ПЛК делят на моноблочные и модульные. В корпусе моноблочного ПЛК наряду с ЦП, памятью и блоком питания размещается фиксированный набор входов/выходов. В модульных ПЛК используют отдельно устанавливаемые модули входов/выходов. Согласно требованиям МЭК 61131, их тип и количество могут меняться в зависимости от поставленной задачи и

Библиотека ГГТУ им. П.О.Сухого

обновляться с течением времени. Подобные ПЛК могут действовать в режиме «ведущего» и расширяться «ведомыми» ПЛК через интерфейс Ethernet.

Моноблочные функционально завершенные ПЛК могут включать в себя небольшой дисплей и кнопки управления. Дисплей предназначен для отображения текущих рабочих параметров и вводимых с помощью кнопок команд рабочих программ и технологических установок. Более сложные ПЛК комбинируются из отдельных функциональных модулей, совместно закрепляемых на стандартной монтажной рейке. В зависимости от количества обслуживаемых входов и выходов, устанавливается необходимое количество модулей ввода и вывода.

1.2. Сопряжения цифровых и аналоговых устройств

Входные и выходные модули ПЛК - это соединения микропроцессора с реальным миром. Они могут классифицироваться по виду сигналов: дискретные или аналоговые.

Бинарные входы и выходы называют обычно дискретными. Они обрабатывают сигналы с кнопок, выключателей, датчиков положения (типа on-off).

Аналоговый или непрерывный сигнал - это уровень напряжения или тока, соответствующий некоторой технологической величине в каждый момент времени: температуре, давлению, расходу, положению, скорости, частоте. Аналоговые входы контроллеров имеют различные параметры и возможности: разрядность АЦП, диапазон входного сигнала, уровень шума и нелинейность, возможность автоматической калибровки, регулирование коэффициента усиления, фильтрация.

Существуют аналоговые входы, предназначенные для подключения термометров сопротивления и термопар, для которых требуется специальная аппаратная поддержка (трехточечное включение, источники Образцового тока, схемы компенсации холодного спая, схемы линеаризации и т. д.). Аналоговые сигналы в ПЛК обязательно преобразуются в цифровую, т. е. заведомо дискретную форму представления.

В системах ПЛК предусмотрены гальваническая развязка входов-выходов, защита по току и напряжению, зеркальные выходные каналы, сторожевой таймер задач и микропроцессорного ядра.

Библиотека ГГТУ им. П.О. Сухого

Гальваническая развязка входов-выходов ПЛК осуществляется с помощью оптоэлектронных пар или оптронов. Они состоят из светодиода и фототранзистора, объединённых в едином блоке. Ток, текущий на вход светодиода, вызывает свечение светодиода в видимой или ИК части спектра. Свет, попадая на фототранзистор, образует ток в выходной цепи. Рис.1.4.

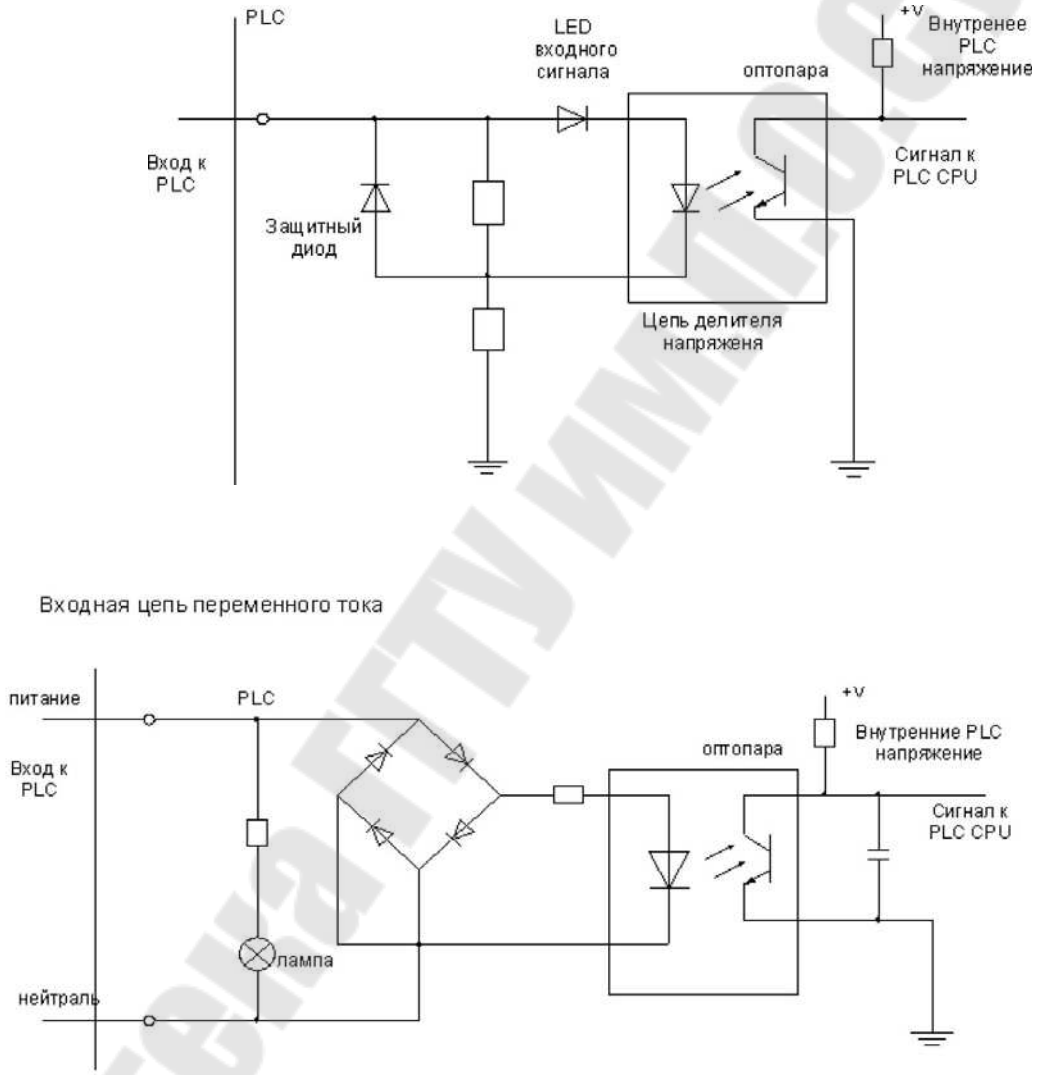


Рисунок 1.4 – Организация входных цепей ПЛК с опторазвязкой

Библиотека ГГТУ им. П.О. Сухого

1.3. Интерфейсы ПЛК

Под интерфейсом (interface) в практике построения микропроцессорных систем и ПЛК понимается совокупность информационно-логических, электрических и конструктивных требований, выполнение которых обеспечивает работоспособное, с оговоренными характеристиками сопряжение различных узлов (устройств) системы.

Информационно-логические требования к интерфейсу определяют структуру и состав линий и сигналов, способы кодирования и форматы данных, адресов, команд, протоколы обмена для различных режимов работы. Информационно-логические условия непосредственно влияют на основные характеристики интерфейса: пропускную способность, надежность обмена, аппаратные затраты.

Электрические требования к интерфейсу задают необходимые статические и динамические параметры сигналов на линиях интерфейса: уровни, длительности фронтов и самих сигналов, нагрузочные способности, уровни помех и т.п.

Конструктивные требования к интерфейсу указывают на тип соединительного элемента, распределение линий по контактам соединительного элемента, допустимую длину линий, геометрические размеры платы, каркаса и других конструктивных элементов.

В настоящее время не существует достаточно полной объективной классификации интерфейсов. Имеющиеся классификации основываются, как правило, на одном классификационном признаке или же строятся для одного класса интерфейсов. Определенным обобщением этих классификаций является стандарт на классификационные признаки интерфейсов (ГОСТ 26.016-81), включающий четыре признака классификации:
- способ соединения компонентов системы (магистральный, радиальный, цепочечный, смешанный);

При магистральном способе имеются коллективные шины, к которым подключены все устройства системы. Характерно, что сигналы шины доступны всем устройствам, но в каждый момент времени только два устройства могут обмениваться данными (1:1). Возможны также широковещательные операции (1:M). В системе с радиальной структурой имеется центральное устройство (контроллер или концентратор), связанный с каждым из абонентов индивидуальной группой однонаправленных линий.

Библиотека ГГТУ им. П.О. Сухого

При цепочечной структуре каждое устройство связано не более чем с двумя другими. Частным случаем цепочечной структуры является кольцевая.

Способ передачи информации (параллельный, последовательный, параллельно- последовательный).

Принцип обмена информацией (асинхронный, синхронный).

Режим обмена информацией (симплексный; полудуплексный; дуплексный и мультиплексный режим обмена).

Для случая связи двух абонентов в симплексном режиме лишь один из двух абонентов может инициировать в любой момент времени передачу информации по интерфейсу Для случая связи двух абонентов в полудуплексном режиме любой абонент может начать передачу информации другому, если линия связи интерфейса при этом оказывается свободной. Для случая связи двух абонентов в дуплексном режиме каждый абонент может начать передачу информации другому в произвольный момент времени. В случае связи нескольких абонентов в мультиплексном режиме в каждый момент времени связь может быть осуществлена между парой абонентов в любом, но единственном направлении от одного из абонентов к другому.
Указанные признаки позволяют характеризовать только определенные аспекты организации интерфейсов. Более полная характеристика и систематизация интерфейсов могут быть выполнены при условии классификации по нескольким совокупностям признаков:

- области распространения (функциональному назначению);
- логической и функциональной организации;
- физической реализации.

В соответствии с первой совокупностью признаков интерфейсы можно разделить на следующие основные классы:

- машинные (или системные);
- периферийного оборудования;
- мультимикропроцессорных систем;
-распределенных ВС (вычислительных локальных сетей, распределенных систем управления).

Интерфейсы периферийного оборудования выполняют функции сопряжения процессоров ПЛК, контроллеров с устройствами ввода-вывода (УВВ), измерительными приборами, исполнительными механизмами, аппаратурой передачи данных (АПД) и внешними

Библиотека ГГТУ им. П.О. Сухого

запоминающими устройствами (ВЗУ). Интерфейсы периферийного оборудования представляют самый большой класс систем сопряжения, что объясняется широкой номенклатурой и разнообразием периферийного оборудования. По своему функциональному назначению эти интерфейсы могут быть разделены на группы интерфейсов радиальной структуры (обеспечивающие схему сопряжения «точка-точка») и магистральной структуры (обеспечивающие схему «многоточечного» подключения).

Системы сопряжения первой группы составляют в основном так называемые малые интерфейсы, применяемые для сопряжения исполнительных механизмов ввода-вывода с контроллерами. К этим интерфейсам относятся: системы сопряжения с параллельной передачей информации, предназначенные для подключения стандартной периферии, системы сопряжения для подключения устройств, размещенных на большом удалении друг от друга.

Интерфейсы второй группы используются как самостоятельно, так и в качестве системотехнического дополнения, расширяющего функциональные возможности ЭВМ на уровне связи с объектом управления. К ним относятся магистральные интерфейсы программно-модульных систем типов IEC 625-1. Эти интерфейсы обеспечивают сопряжение программируемых контроллеров и ЭВМ с широким спектром цифровых измерительных приборов, преобразователей информации, генераторов, датчиков, пультов оператора.

В группу интерфейсов мультимикропроцессорных систем входят в основном внутриблочные, процессорно -независимые системы сопряжения. Характерным их отличием от обычных магистральных интерфейсов является техническая реализация функций селекции и координации, что позволяет подключать к ним один или несколько процессоров как обычные УВВ. Этот класс интерфейсов отличают высокая пропускная способность.

Данный класс систем сопряжения может быть разделен на две крупные группы в соответствии со структурой шин адреса и данных: с раздельными и мультиплексными шинами. Как правило, эти интерфейсы представляют собой внутриблочную систему сопряжения магистральной структуры с высокой пропускной способностью.

По конструктивному исполнению интерфейс могут быть разделены на четыре категории:

Библиотека ГГТУ им. П.О.Сухого

- межблочные, обеспечивающие взаимодействие компонентов на уровне прибора, автономного устройства, блока, стойки, шкафа;

- внутриблочные, обеспечивающие взаимодействие на уровне плат, субблоков;

- внутриплатные, обеспечивающие взаимосвязь между интегральными схемами на печатной плате;

- внутрикорпусные, обеспечивающие взаимодействие компонентов внутри микросхемм.

Межблочное сопряжение реализуется на уровне следующих конструктивных средств:

- коаксиального и оптоволоконного кабеля;
- многожильного плоского кабеля (шлейфа);
- многожильного кабеля на основе витой пары проводов.

Внутриблочное сопряжение печатных плат, субблоков выполняется печатным способом или накруткой витой парой проводов внутри блока, стойки, шкафа. Ряд интерфейсов может быть реализован комбинацией внутри-блочного и межблочного исполнений. Внутриплатное сопряжение реализуется печатным способом, внутрикорпусное — методами микроэлектронной технологии.

Для организации промышленных сетей используется множество интерфейсов и протоколов передачи данных, например Modbus, Ethernet, CAN, HART, PROFIBUS и пр. Они необходимы для передачи данных между датчиками, контроллерами и исполнительными механизмами (ИМ); калибровки датчиков; питания датчиков и ИМ; связи нижнего и верхнего уровней АСУ ТП. Протоколы разрабатываются с учетом особенностей производства и технических систем, обеспечивая надежное соединение и высокую точность передачи данных между различными устройствами. Наряду с надежностью работы в жестких условиях все более важными требованиями в системах АСУ ТП становятся функциональные возможности, гибкость в построении, простота интеграции и обслуживания, соответствие промышленным стандартам.

Наиболее распространённой системой классификации сетевых протоколов является теоретическая модель OSI (базовая эталонная модель взаимодействия открытых систем, англ. Open Systems Interconnection Basic Reference Model). Спецификация этой модели была окончательно принята в 1984 году Международной Организацией по Стандартизации (ISO). В соответствии с моделью

Библиотека ГГТУ им. П.О.Сухого

OSI протоколы делятся на 7 уровней, расположенных друг над другом, по своему назначению — от физического (формирование и распознавание электрических или других сигналов) до прикладного (API для передачи информации приложениями). Взаимодействие между уровнями может осуществляться, как вертикально, так и горизонтально (Рис. 1.5). В горизонтальном взаимодействии программам требуется общий протокол для обмена данными. В вертикальном – посредством интерфейсов.

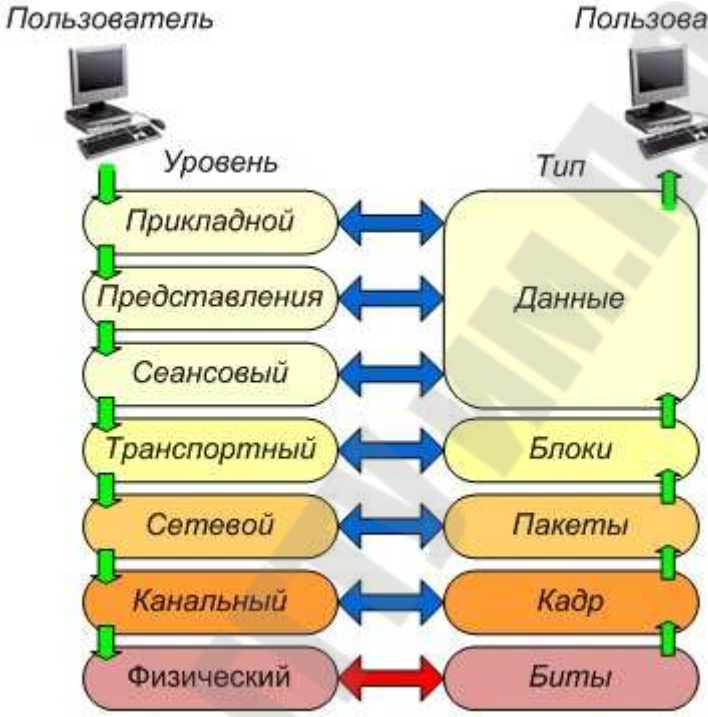


Рисунок 1.5 – Модель OSI

Прикладной уровень – уровень приложений (англ. Application layer). Обеспечивает взаимодействие сети и приложений пользователя, выходящих за рамки модели OSI. На этом уровне используются следующие протоколы: HTTP, gopher, Telnet, DNS, SMTP, SNMP, CMIP, FTP, TFTP, SSH, IRC, AIM, NFS, NNTP, NTP, SNTP, XMPP, FTAM, APPC, X.400, X.500, AFP, LDAP, SIP, ITMS, Modbus TCP, BACnet IP, IMAP, POP3, SMB, MFTP, BitTorrent, eD2k, PROFIBUS.

Представительский уровень (англ. Presentation layer) – уровень представления данных. На этом уровне может осуществляться преобразование протоколов и сжатие/распаковка или кодирование/декодирование данных, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально. Запросы приложений, полученные с уровня

Библиотека ГГТУ им. П.О. Сухого

приложений, он преобразует в формат для передачи по сети, а полученные из сети данные преобразует в формат, понятный приложениям. К этому уровню традиционно относят следующие протоколы: HTTP, ASN.1, XML-RPC, TDI, XDR, SNMP, FTP, Telnet, SMTP, NCP, AFP.

Сеансовый уровень (англ. Session layer) управляет созданием/завершением сеанса связи, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений. Синхронизация передачи обеспечивается помещением в поток данных контрольных точек, начиная с которых возобновляется процесс при нарушении взаимодействия. Используемые протоколы: ASP, ADSP, DLC, Named Pipes, NBT, NetBIOS, NWLink, Printer Access Protocol, Zone Information Protocol, SSL, TLS, SOCKS.

Транспортный уровень (англ. Transport layer) организует доставку данных без ошибок, потерь и дублирования в той последовательности, как они были переданы. Разделяет данные на фрагменты равной величины, объединяя короткие и разбивая длинные (размер фрагмента зависит от используемого протокола). Используемые протоколы: TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX, SCTP, DCCP, RTP, TFTP.

Сетевой уровень (англ. Network layer) определяет пути передачи данных. Отвечает за трансляцию логических адресов и имён в физические, за определение кратчайших маршрутов, коммутацию и маршрутизацию, за отслеживание неполадок и заторов в сети. Используемые протоколы: IP, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, ARP, RARP, DHCP, BootP, SKIP, RIP.

Канальный уровень (англ. Data link layer) предназначен для обеспечения взаимодействия сетей на физическом уровне. Полученные с физического уровня данные проверяет на ошибки, если нужно исправляет, упаковывает во фреймы, проверяет на целостность, и отправляет на сетевой уровень. Канальный уровень может взаимодействовать с одним или несколькими физическими уровнями. Спецификация IEEE 802 разделяет этот уровень на 2 подуровня — MAC (Media Access Control) регулирует доступ к разделяемой физической среде, LLC (Logical Link Control) обеспечивает обслуживание сетевого уровня. Используемые протоколы: STP, ARCnet, ATM, DTM, SLIP, SMDS, Ethernet, FDDI,

Библиотека ГГТУ им. П.О. Сухого

Frame Relay, LocalTalk, Token ring, StarLan, L2F, L2TP, PPTP, PPP, PPPoE, PROFIBUS.

Физический уровень (англ. Physical layer) предназначен непосредственно для передачи потока данных. Осуществляет передачу электрических или оптических сигналов в кабель или в радиоэфир и, соответственно, их приём и преобразование в биты данных в соответствии с методами кодирования цифровых сигналов. Используемые протоколы: RS-232, RS-422, RS-423, RS-449, RS-485, ITU-T, xDSL, ISDN, T1, E1, 10BASE-T, 10BASE2, 10BASE5, 100BASE-T, 1000BASE-T, 1000BASE-TX, 1000BASE-SX.

Многие протоколы упоминаются сразу на нескольких уровнях. Это говорит о недоработанности и отдаленности теоретической модели от реальных сетевых протоколов, поэтому привязка некоторых из них к уровням OSI является условной.

Компания ICPDAS для работы с протоколом HTTP предлагает следующее оборудование и программное обеспечение. Контроллеры серии XPAK, WinPAC, WinCon, LinPAC, ViewPAC работают под управлением операционных систем Windows и Linux, с встроенным HTTP-сервером. Программные пакеты InduSoft (SCADA), ISaGRAF, Web HMI, VXCOMM, MiniOS7 Studio, также используют HTTP-сервер для связи и взаимодействия с устройствами.

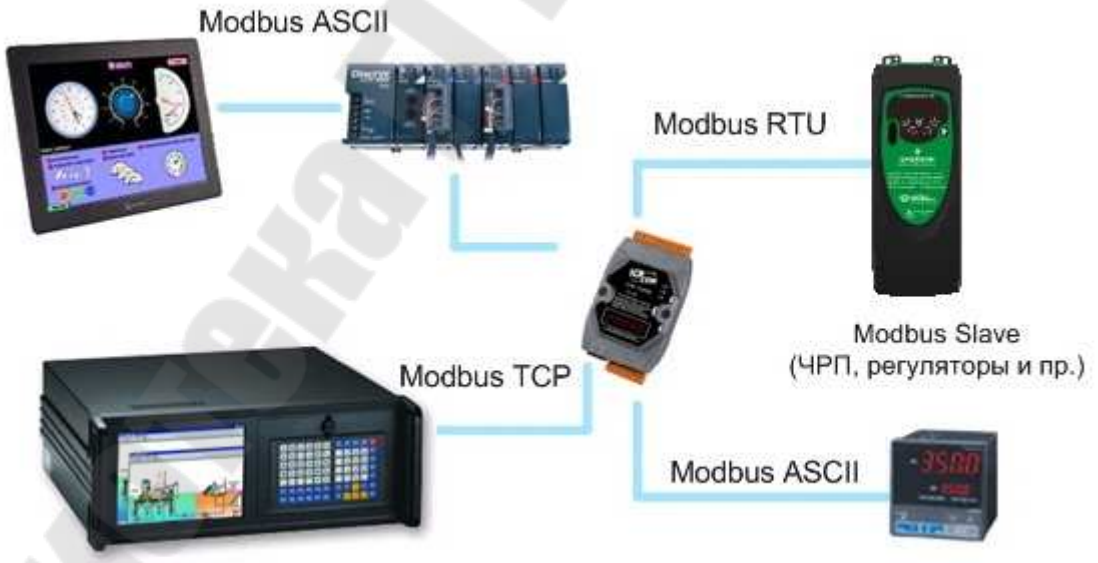


Рисунок 1.6 - Совместимость протоколов семейства Modbus

Для организации взаимодействия между элементами автоматизации в промышленных сетях передачи данных широко применяется коммуникационный протокол Modbus (См. Рис 1.6).

Существуют три основные реализации протокола Modbus, две для передачи данных по последовательным линиям связи, как медным EIA/TIA-232-E (RS-232), EIA-422, EIA/TIA-485-A (RS-485), так и оптическим и радио: Modbus RTU и Modbus ASCII, и для передачи данных по сетям Ethernet поверх TCP/IP: Modbus TCP.

Различие между протоколами Modbus ASCII и Modbus RTU заключается в способе кодирования символов. В режиме ASCII данные кодируются при помощи таблицы ASCII, где каждому символу соответствует два байта данных. В режиме RTU данные передаются в виде 8-ми разрядных двоичных символов, что обеспечивает более высокую скорость передачи данных. ASCII допускает задержку до 1 секунды в отличии от RTU, где сообщения должны быть непрерывны. Также режим ASCII имеет упрощенную систему декодирования и управления данными.

Протоколы семейства Modbus (Modbus ASCII, Modbus RTU и Modbus TCP/IP) используют один прикладной протокол, что позволяет обеспечить их совместимость. Максимальное количество сетевых узлов в сети Modbus – 31. Протяженность линий связи и скорость передачи данных зависит от физической реализации интерфейса. Элементы сети Modbus взаимодействуют, используя клиент-серверную модель, основанную на транзакциях, состоящих из запроса и ответа.

Обычно в сети есть только один клиент, так называемое, «главное» (англ. master) устройство, и несколько серверов — «подчиненных» (slaves) устройств. Главное устройство инициирует транзакции (передаёт запросы). Подчиненные устройства передают запрашиваемые главным устройством данные, или производят запрашиваемые действия. Главный может адресоваться индивидуально к подчиненному или инициировать передачу широковещательного сообщения для всех подчиненных устройств. Подчиненное устройство формирует сообщение и возвращает его в ответ на запрос, адресованный именно ему.

Области промышленного применения: организация связи датчиков и исполнительных механизмов с контроллером, связь контроллеров и управляющих компьютеров, связь с датчиками, контроллерами и корпоративными сетями, в SCADA системах.

Простота применения протоколов семейства Modbus в промышленности обусловило его широкое распространение. На

Библиотека ГГТУ им. П.О. Сухого

сегодняшний день, оборудование практически всех производителей поддерживает протоколы Modbus.

Традиционно протоколы семейства Modbus поддерживаются OPC серверами SCADA систем (Clear SCADA, компании Control Microsystems, InTouch Wonderware, TRACE MODE)для связи с элементами управления (контроллерами, ЧРП, регуляторами и др.).

В Европе широкое распространение получила открытая промышленная сеть PROFIBUS (PROcess FIeld BUS). Изначально, прототип этой сети был разработан компанией Siemens для своих промышленных контроллеров.

PROFIBUS объединяет технологические и функциональные особенности последовательной связи полевого уровня. Она позволяет объединять разрозненные устройства автоматизации в единую систему на уровне датчиков и приводов. Сеть PROFIBUS основывается на нескольких стандартах и протоколах, использует обмен данными между ведущим и ведомыми устройствами (протоколы DP и PA) или между несколькими ведущими устройствами (протоколы FDL и FMS).

Сеть PROFIBUS можно ассоциировать с тремя уровнями модели OSI: физический, канальный и уровень приложений.

Единым протоколом для доступа к шине для всех версий PROFIBUS является реализованный на втором уровне модели OSI протокол PROFIBUS-FDL. Данный протокол использует процедуру доступа с помощью маркера (token). Так же, как и сети на базе протоколов Modbus, сеть PROFIBUS состоит из ведущих (master) и ведомых (slave) устройств. Ведущее устройство может управлять шиной. Когда у ведущего (master) устройства есть право доступа к шине, оно может передавать сообщения без удаленного запроса. Ведомые устройства – это обычные периферийные устройства, не имеют прав доступа к шине, то есть они могут только подтверждать принимаемые сообщения или передавать сообщения ведущему устройству по его запросу. В минимальной конфигурации сеть может состоять либо из двух ведущих, либо из одного ведущего и одного ведомого устройства. (См. Рис 1.7).

Библиотека ГГТУ им. П.О. Сухого

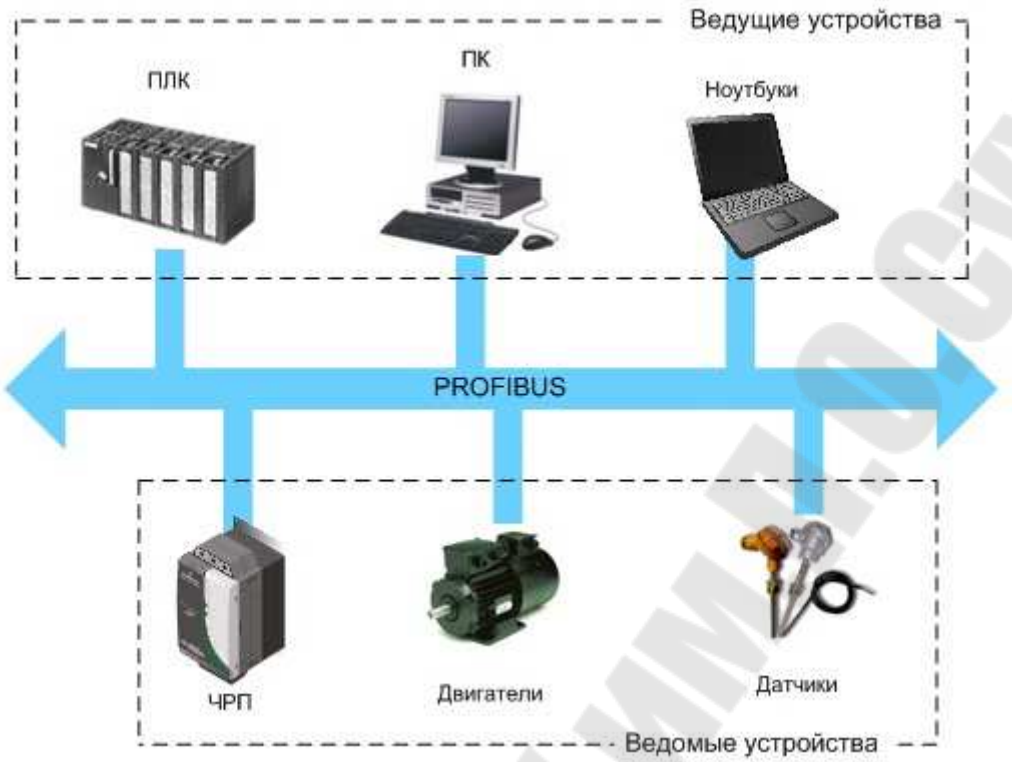


Рисунок 1.7- Сеть Profibus

Одни и те же каналы связи сети PROFIBUS допускают одновременное использование нескольких протоколов передачи данных. Рассмотрим каждый из них.

PROFIBUS DP (Decentralized Peripheral - Распределенная периферия) — протокол, ориентированный на обеспечение скоростного обмена данными между ведущими DP-устройствами и устройствами распределённого ввода-вывода. Протокол характеризуется минимальным временем реакции и высокой стойкостью к воздействию внешних электромагнитных полей. Оптимизирован для высокоскоростных и недорогих систем.

PROFIBUS PA (Process Automation - Автоматизация процесса) — протокол обмена данными с оборудованием полевого уровня, расположенным в обычных или взрывоопасных зонах. Протокол позволяет подключать датчики и приводы на одну линейную шину или кольцевую шину.

PROFIBUS FMS (Fieldbus Message Specification - Спецификация сообщений полевого уровня) - универсальный протокол для решения задач по обмену данными между интеллектуальными сетевыми устройствами (контроллерами, компьютерами/программаторами,

Библиотека ГГТУ им. П.О. Сухого

системами человеко-машинного интерфейса) на полевом уровне. Некоторый аналог промышленного Ethernet, обычно используется для высокоскоростной связи между контроллерами и компьютерами верхнего уровня.

Все протоколы используют одинаковые технологии передачи данных и общий метод доступа к шине, поэтому они могут функционировать на одной шине.

Положительные стороны: открытость, независимость от поставщика, распространенность.

Области промышленного применения: организация связи датчиков и исполнительных механизмов с контроллером, связь контроллеров и управляющих компьютеров, связь с датчиками, контроллерами и корпоративными сетями, в SCADA системах.

Основную массу оборудования использующего протокол PROFIBUS составляет оборудование компании SIEMENS. Но в последнее время этот протокол получил применение у большинства производителей. Во многом это обусловлено распространенностью систем управления на базе контроллеров Siemens.

Компания ICPDAS для реализации проектов на базе PROFIBUS предлагает ряд ведомых устройств: шлюзы PROFIBUS/Modbus серии GW, преобразователи PROFIBUS в RS-232/485/422 серии I-7000, модули и каркасы удаленного ввода/вывода PROFIBUS серии PROFI-8000. В настоящие время инженерами компании ICPDAS ведутся интенсивные разработки в области создания PROFIBUS ведущего устройства.

Задачи управления требуют непрерывного циклического контроля. В любых цифровых устройствах непрерывность достигается за счет применения дискретных алгоритмов, повторяющихся через достаточно малые промежутки времени. Таким образом, вычисления в ПЛК всегда повторяются циклически. Одна итерация, включающая замер, обсчет и выработку воздействия, называется рабочим циклом ПЛК. Выполняемые действия зависят от значения входов контроллера, предыдущего состояния и определяются пользовательской программой.

По включению питания ПЛК выполняет самотестирование и настройку аппаратных ресурсов, очистку оперативной памяти данных (ОЗУ), контроль целостности прикладной программы пользователя. Если прикладная программа сохранена в памяти, ПЛК переходит к основной работе, которая состоит из постоянного повторения

Библиотека ГГТУ им. П.О.Сухого

последовательности действий, входящих в рабочий цикл.Рабочий цикл PLC состоит из нескольких фаз:

- Начало цикла;
- Чтение состояния входов;
- Выполнение кода программы пользователя;
- Запись состояния выходов;
- Обслуживание аппаратных ресурсов ПЛК;
- Монитор системы исполнения;
- Контроль времени цикла;
- Переход на начало цикла.

Абсолютное большинство ПЛК работают по методу периодического опроса входных данных (сканирования). ПЛК опрашивает входы, выполняет пользовательскую программу и устанавливает необходимые значения выходов. Для математических систем характеристикой качества работы является правильность найденного решения. В системах реального времени помимо правильности решения определяющую роль играет время реакции. Логически верное решение, полученное с задержкой более допустимой, не является приемлемым.

Время реакции — это время с момента изменения состояния системы до момента выработки соответствующей реакции. Если изменение значений входов произошло непосредственно перед фазой чтения входов, то время реакции будет наименьшим и равным времени сканирования. Худший случай, когда изменение значений входов происходит сразу после фазы чтения входов. Тогда время реакции будет наибольшим, равным удвоенному времени сканирования минус время одного чтения входов. Иными словами, время реакции ПЛК не превышает удвоенного времени сканирования.

Помимо времени реакции ПЛК, существенное значение имеет время реакции датчиков и исполнительных механизмов, которое также необходимо учитывать при оценке общего времени реакции системы.

Существуют ПЛК, которые реализуют команды непосредственного доступа к аппаратуре входов и выходов, что позволяет обрабатывать и формировать отдельные сигналы с длительностью меньшей длительности рабочего цикла.

Для уменьшения времени реакции сканирующих контроллеров алгоритм программы разбивается на несколько задач с различным периодом исполнения. В наиболее развитых системах пользователь

Библиотека ГГТУ им.П.О.Сухого

имеет возможность создавать отдельные программы, исполняемые по прерыванию, помимо кода, исполняемого в рабочем цикле. Такая техника позволяет ПЛК существенно форсировать ограничение реакции временем сканирования при небольшом количестве входов, требующих сверхскоростной реакции.

Время цикла сканирования является базовым показателем быстродействия ПЛК.

Программа ПЛК может рассматриваться как постоянно бегущая замкнутая цепь. Инструкция пользователя считывается непрерывно и когда считывается последняя инструкция, операция начинается снова. Это называется сканированием программы, а период - временем сканирования. Время зависит от размера программы и скорости процессора.

Абсолютно одинаковые ПЛК могут выполнять совершенно разные функции. Причем для изменения алгоритма работы не требуется каких-либо переделок аппаратной части. Задачей прикладного программирования ПЛК является только реализация алгоритма управления конкретной машиной. Опрос входов и выходов контроллер осуществляет автоматически, вне зависимости от способа физического соединения. Эту работу выполняет системное программное обеспечение. В идеальном случае прикладной программист совершенно не интересуется, как подсоединены и где расположены датчики и исполнительные механизмы. Мало того, его работа не зависит от того, с каким контроллером и какой фирмы он работает. Благодаря стандартизации языков программирования прикладная программа оказывается переносимой. Это означает, что ее можно использовать в любом ПЛК, поддерживающем данный стандарт.

В настоящее время стандарт для ПЛК включает следующие части :

Часть 1. Общая информация;

Часть 2. Требования к оборудованию и тестам;

Часть 3. Языки программирования;

Часть 4. Руководства пользователя;

Часть 5. Спецификация сообщений;

Часть 6. Промышленные сети;

Часть 7. Программирование с нечеткой логикой;

Часть 8. Руководящие принципы применения и реализации языков ПЛК.

Библиотека ГГТУ им. П.О.Сухого

Если планируется использовать прерывания, то программы обработки прерываний, которые ставятся в соответствие прерывающим событиям, хранятся как часть основной программы. Однако они исполняются не как составная часть нормального цикла, а только тогда, когда происходит прерывающее событие (оно возможно в любом месте цикла). Прерывания обслуживаются, как правило, в последовательности их появления с учетом соответствующих приоритетов.

По прерываниям ПЛК запускает специальные программы обработки прерываний. Типы прерываний:

- Циклические прерывания по времени (например, каждые 5 секунд);
- Прерывание по дискретному входу (например, по сработке концевика);
- Прерывания по программным и коммуникационным ошибкам, превышению времени цикла, неисправностям модулей, обрывам контуров.

Подробную информацию о командах прерывания можно найти в документации к конкретному виду ПЛК.

Для нормального функционирования ПЛК обязательно должно выполняться условие: Тц < Тср. им, где Тср. им – время срабатывания исполнительных механизмов. Необходимость выполнения этого условия вызвано тем, что на входах ПЛК возможно появление помех, а, следовательно, и ошибочных сигналов о срабатывании того или иного датчика, что может привести к ошибочному формированию выходных сигналов ПЛК и к аварии на управляемом объекте. Если Тц < Тср. им, то исполнительный механизм не успевает включиться за один рабочий цикл ПЛК, а в следующем цикле, если помеха носит случайный характер, ошибочный управляющий сигнал выдан не будет. Рекомендуется опрос состояний входов производить не один раз, а n раз и только после n-кратного подтверждения состояния конкретного входа использовать эту информацию для последующей логической обработки.

К выходным функциям в первую очередь относятся функции включения-выключения (запуска-останова) исполнительных механизмов управляемого объекта. К ним также относятся функции пуска-сброса таймеров и счётчиков, включения-выключения внутренних операторов, которые используются как для запоминания импульсных входных сигналов, так и промежуточных состояний УП,

Библиотека ГГТУ им. П.О. Сухого

а также промежуточных результатов, получаемых при отработке участков программы. К выходным относятся также функции, связанные с организацией зон игнорирования участков программы, с организацией условных и безусловных переходов в программе с обнулением и без обнуления при этом указанных выше выходных функций.

Особенности функционирования ПЛК как управляющего устройства промышленного назначения Процесс управления промышленным оборудованием с помощью ПЛК сводится, в основном, к решению логических уравнений, при этом необходимо чётко представлять, какие функции, реализуемые ПЛК, следует считать выходными и какие состояния управляемого объекта, элементов системы управления и выходные сигналы ПЛК могут или должны рассматриваться в качестве аргументов указанных функций.

К выходным функциям в первую очередь относятся функции включения-выключения (запуск-останов) исполнительных механизмов управляемого объекта. К выходным функциям относятся также функции запуска-сброса таймеров и счётчиков, включения-выключения так называемых внутренних операторов, которые используются как для запоминания импульсных входных сигналов, так и промежуточных состояний управляющей программы, а также промежуточных результатов, получаемых при отработке участков программы. К выходным относятся также функции, связанные с организацией зон игнорирования участков программы, с организацией условных и безусловных переходов в программе с обнулением и без обнуления при этом указанных выше выходных функций.

Одной из важнейших особенностей функционирования ПЛК как управляющего устройства промышленного назначения является то, что в отличие от всех управляющих устройств предыдущих поколений в ПЛК один и тот же аргумент (состояние входа контроллера) может быть использован на различных этапах программы произвольное число раз (что равносильно релейно-контактному аппарату с бесконечным числом контактов).

Отмеченная особенность относится и к выходным функциям, для «размножения» которых не требуются традиционные блок-контакты пускателей либо их твердотельные аналоги. Подобным же образом при реализации того или иного алгоритма в ПЛК имеется возможность многократного использования в программе одного и

Библиотека ГГТУ им. П.О.Сухого

того же таймера, счётчика, а также обращения к любым промежуточным значениям таймеров и счётчиков.

Практически во всех ПЛК программно можно задать включение его выходов или внутренних операторов как с памятью (фиксацией состояния после исчезновения условий включения), так и без неё.

При реализации функций включения-выключения механизмов программным путём не имеет значения, каким сигналом осуществляется запуск механизма «1» или «0» в то время, как при реализации такой операции с помощью релейной аппаратуры необходимо каждый раз вводить дополнительное реле, выполняющее функцию инвертора.

В ПЛК используется как энергозависимая, так и энергонезависимая память. Очевидно, что для нормальной работы ПЛК текущие состояния выходов, таймеров, счётчиков и внутренних операторов запоминаются в оперативной памяти, которая может быть как энергозависимой, так и энергонезависимой (с подпиткой от аккумуляторов). Использование в данном случае энергозависимой памяти имеет как недостатки, так и достоинства. С одной стороны, отключение электропитания может приводить к потере значительного объёма информации, а с другой энергозависимая память в данном случае может эффективно выполнять роль так называемой «нулевой» защиты, предотвращающей прямое (без учёта заданной последовательности включения механизмов и всех необходимых блокировочных зависимостей) повторное включение механизмов при возобновлении подачи электропитания.

Функции таймера (реле времени) в ПЛК реализуются различными способами. Имеются модели ПЛК, в которых функции таймера реализуются аппаратным путём, при этом таймер включается последовательно с определённым выходом ПЛК. На таких таймерах временные уставки задаются с помощью регуляторов либо программных задатчиков (например, – программных переключателей барабанного типа) вручную. Такое решение является весьма удобным при наладке и настройке техно-логического оборудования, так как позволяет оперативно изменять уставки без изменения управляющей программы.

В общем же случае организация таймеров в ПЛК программным путём является более гибким и более универсальным решением, потому что в этом случае нет необходимости в жёсткой (монтажной) привязке таймера к конкретному выходу и имеется возможность

Библиотека ГГТУ им. П.О.Сухого

программного выбора типа выполняемой временной функции (задержка на включение, задержка на выключение и пр.). Кроме того, программно можно задавать и различные условия запуска и сброса таймера. Так, например, программно можно задать или не задать условие, при котором запущенный таймер сбрасывается в исходное состояние, если по каким-то причинам исчезли условия его запуска в период времени, соответствующий запрограммированной временной задержке.

По-разному выполняется и сброс таймера в исходное (стартовое) состояние. Во многих моделях ПЛК отсчёт времени заданной уставки осуществляется от уставки до нуля, поэтому в таких таймерах операция принудительного сброса смысла не имеет. В тех же ПЛК, где в таймерах отсчёт уставок выполняется от нуля к уставке, необходимо вводить команды принудительного обнуления таймеров.

Весьма существенной для ПЛК как управляющего устройства промышленного назначения является функция счёта количества событий, происходящих на управляемом объекте (например, – количества изготовленной продукции). Однако, счётчики в ПЛК могут устанавливаться и для подсчёта числа внутренних событий (например, – количества циклов отработки отдельных участков программы).

Счётчики, как и таймеры, могут быть реализованы и аппаратно, и программно, могут работать как суммирующие, вычитающие и реверсивные. Как правило, при использовании в ПЛК счётчиков все они обнуляются с помощью специальной команды «сброс счётчика».

В отличие от таймера (программная реализация которого осуществляется исключительно за счёт внутренних ресурсов ПЛК) для счёта числа внешних событий хотя бы один из входов контроллера должен быть задействован для приёма импульсов от датчика счёта, установленного на технологическом оборудовании.

В связи с тем, что счётчики в ПЛК выполняют подсчёт импульсов, поступающих в ПЛК из внешних электрических цепей, то существует достаточно серьёзная проблема обеспечения точности работы счётчиков. С этой целью приём единичного импульса в счётчик фиксируется не менее, чем по двум состояниям (импульс, пауза). Кроме того, в каналах счёта импульсов используется специальная схема (или её программный аналог) выделения одиночного импульса, позволяющая отстроиться от «дребезга» контактов и многих видов импульсных помех.

Библиотека ГГТУ им. П.О.Сухого

Указанные и многие другие схемные решения, повышающие достоверность передачи импульсов счёта от источника информации ко входам ПЛК, как правило, реализуются в месте их формирования (в датчиках, преобразователях), поэтому на входы ПЛК импульсы счёта поступают обычно уже в «отфильтрованном» виде. Тем не менее, в самих ПЛК также применяют различные программные способы, обеспечивающие повышение достоверности принимаемой информации и точности счёта, осуществляемого программно организованными счётчиками.

С целью повышения быстродействия, а также надёжности функционирования ПЛК в них применяется специальная операция – пропуск участка программы, который в этом случае называется «зоной игнорирования». Смысл организации зон игнорирования и возможность программного их пропуска состоит в следующем. При отработке отдельных частей управляющей программы (т.е. на определённых шагах технологического цикла) остальные части программы могут быть опущены (про-игнорированы), что и позволяет существенно сократить длительность цикла сканирования части управляющей программы, с которой активно взаимодействует процессор на данном шаге технологической циклограммы, что, естественно, приводит к повышению как быстродействия ПЛК, так и надёжности его функционирования.

Использование зон игнорирования позволяет упростить структуру программы. Во многих моделях ПЛК с целью повышения надёжности их функционирования ветвление программы с использованием условных и безусловных переходов эффективно заменяют неветвящейся управляющей программой с введением необходимого числа зон игнорирования.

При использовании ПЛК в промышленных объектах особое значение имеет автоматизация диагностирования неисправностей как в каналах ввода-вывода информации (в датчиках, исполнительных механизмах и линиях связи), так и в самом контроллере. И в этом случае в ПЛК не используется, какая бы то ни была дополнительная аппаратура, так как абсолютное большинство проблем, связанных с диагностикой и соответствующим аварийным отключением ПЛК и технологического оборудования, решается программным путём. Весьма часто объём памяти, занимаемой диагностическими программами и программами аварийного останова, значительно превышает объём памяти основной управляющей программы.

Библиотека ГГТУ им. П.О.Сухого

2. Контроллеры ОВЕН и программный комплекс CoDeSys

2.1. Основы программирования на стандартизированных языках МЭК (IEC) стандарта IEC61131-13

Стандарт МЭК 61131-3 устанавливает пять языков программирования ПЛК, три графических и два текстовых. Первоначально стандарт назывался IEC 1131-3 и был опубликован в 1993 г. но в 1997 г. МЭК (IEC) перешел на новую систему обозначений и в названии стандарта добавилась цифра "6". Продвижением стандарта занимается организация PLCopen (http://www.plcopen.org).

Основной целью стандарта было повышение скорости и качества разработки программ, а также создание языков программирования, ориентированных на технологов, обеспечение соответствия ПЛК идеологии открытых систем, исключение этапа дополнительного обучения при смене типа ПЛК.

Системы программирования, основанные на МЭК 61131-3, характеризуются следующими показателями:

- надежностью создаваемого программного обеспечения. Надежность обеспечивается тем, что программы для ПЛК создаются с помощью специально предназначенной для этого среды разработки, которая содержит все необходимые средства для написания, тестирования и отладки программ с помощью эмуляторов и реальных ПЛК, а также множество готовых фрагментов программного кода;
- возможностью простой модификации программы и наращивания ее функциональности;
- переносимостью проекта с одного ПЛК на другой;
- возможностью повторного использования отработанных фрагментов программы;
- простотой языка и ограничением количества его элементов.

Языки МЭК 61131-3 появились не как теоретическая разработка, а как результат анализа множества языков, уже используемых на практике и предлагаемых рынку производителями ПЛК. Стандарт устанавливает пять языков программирования со следующими названиями:

- структурированный текст (ST - Structured Text);
- последовательные функциональные схемы (SFC - "Sequential Function Chart");

Библиотека ГГТУ им. П.О.Сухого

- диаграммы функциональных блоков (FBD - Function Block Diagram);

- релейно-контактные схемы, или релейные диаграммы (LD - Ladder Diagram);

- список инструкций (IL - Instruction List).

Графическими языками являются SFC, FBD, LD. Языки IL и ST являются текстовыми.

В стандарт были введены несколько языков (а не один) для того, чтобы каждый пользователь мог применить наиболее понятный ему язык. Программисты чаще выбирают язык IL (похожий на ассемблер) или ST, похожий на язык высокого уровня Паскаль; специалисты, имеющие опыт работы с релейной логикой, выбирают язык LD, специалисты по системам автоматического управления (САУ) и схемотехники выбирают привычный для них язык FBD.

Выбор одного из пяти языков определятся не только предпочтениями пользователя, но и смыслом решаемой задачи. Если исходная задача формулируется в терминах последовательной обработки и передачи сигналов, то для нее проще и нагляднее использовать язык FBD. Если задача описывается как последовательность срабатываний некоторых ключей и реле, то для нее нагляднее всего будет язык LD. Для задач, которые изначально формулируются в виде сложного разветвленного алгоритма, удобнее будет язык ST.

Языки МЭК 61131-3 базируются на следующих принципах:

- вся программа разбивается на множество функциональных элементов - Program Organization Units (POU), каждый из которых может состоять из функций, функциональных блоков и программ. Любой элемент МЭК-программы может быть сконструирован иерархически из более простых элементов;

- стандарт требует строгой типизации данных. Указание типов данных позволяет легко обнаруживать большинство ошибок в программе до ее исполнения;

- имеются средства для исполнения разных фрагментов программы в разное время, с разной скоростью, а также параллельно. Например, один фрагмент программы может сканировать концевой датчик с частотой 100 раз в секунду, в то время как второй фрагмент будет сканировать датчик температуры с частотой один раз в 10 сек;

Библиотека ГГТУ им. П.О.Сухого

- для выполнение операций в определенной последовательности, которая задается моментами времени или событиями, используется специальный язык последовательных функциональных схем (SFC);
- стандарт поддерживает структуры для описания разнородных данных. Например, температуру подшипников насоса, давление и состояние "включено-выключено" можно описать с помощью единой структуры "Pomp" и передавать ее внутри программы как единый элемент данных;
- стандарт обеспечивает совместное использование всех пяти языков, поэтому для каждого фрагмента задачи может быть выбран любой, наиболее удобный, язык;
- программа, написанная для одного контроллера, может быть перенесена на любой контроллер, совместимый со стандартом МЭК 61131-3.

Любой ПЛК работает в циклическом режиме. Цикл начинается со сбора данных с модулей ввода, затем исполняется программа ПЛК и оканчивается цикл выводом данных в устройства вывода. Поэтому величина контроллерного цикла зависит от времени исполнения программы и быстродействия процессорного модуля.

Графический язык релейной логики впервые появился в виде электрических схем, которые состояли из контактов и обмоток электромагнитных реле (См. Рис 2.1). Такие схемы использовались в автоматике конвейеров для сборки автомобилей до эры микропроцессоров. Язык релейной логики был интуитивно понятен людям, слегка знакомым с электротехникой и поэтому оказался наиболее распространенным в промышленной автоматике. Обслуживающий персонал легко находил отказ в оборудовании, прослеживая путь сигнала по релейной диаграмме.

Однако язык LD проблематично использовать для реализации сложных алгоритмов, поскольку он не поддерживает подпрограммы, функции и другие средства структурирования программ с целью повышения качества программирования. Эти недостатки затрудняют многократное использование программных компонентов, что делает программу длинной и сложной для обслуживания.

Библиотека ГГТУ им. П.О. Сухого

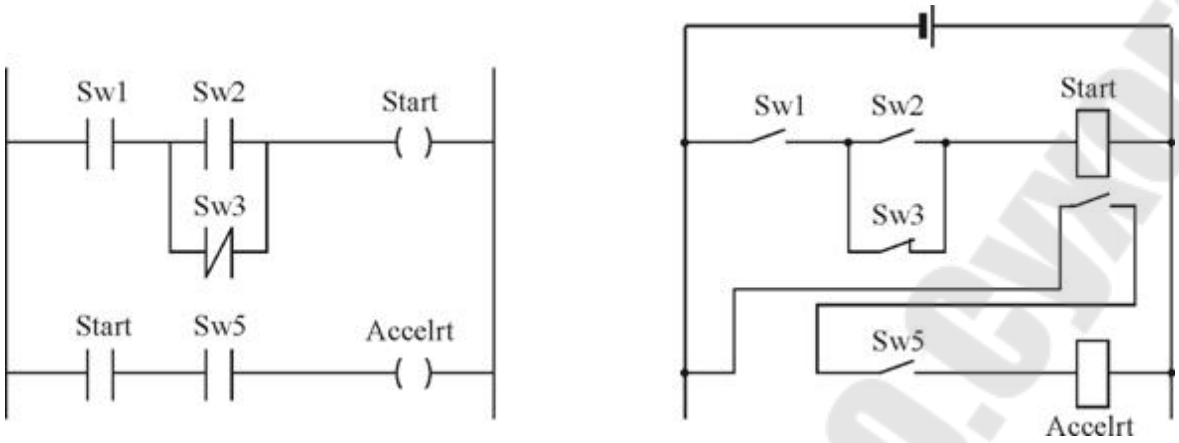


Рисунок 2.1 - Пример программы на языке LD (слева) и ее эквивалент в виде электрической цепи с реле и выключателями (справа)

Язык LD позволяет отображать пять категорий логического уравнения: аргумент, функцию, инверсию, логическое умножение и логическое сложение. Приняты обозначения: аргумента – замыкающим контактом; инверсии – размыкающим контактом; функции – нагрузкой релейной цепи; логического умножения и сложения – соответственно последовательным и параллельным соединением контактов электрической цепи.

Составление управляющей программы на языке LD осуществляется по заранее составленной или уже имеющейся принципиальной электрической схеме, которая представляет собой определённый набор электрических цепей. Технология программирования на этом языке сводится к привязке контролируемых входов и управляемых выходов ПЛК к конкретным контактам и обмоткам реле и пускателей, которым присваиваются соответствующие номера входов и выходов контроллера. Нумеруются также обмотки промежуточных реле и другие аппараты, рассматриваемые в ПЛК как внутренние переменные.

Для выполнения арифметических функций в язык LD были добавлены функциональные блоки, которые выполняли операции сложения, умножения, вычисления среднего и т.д. Сложные вычисления в этом языке невозможны. Недостатком является также то, что только маленькая часть программы умещается на мониторе компьютера или панели оператора при программировании.

Несмотря на указанные недостатки, язык LD относится к наиболее распространенным в мире, хотя используется для программирования только простых задач.

Библиотека ГГТУ им. П.О.Сухого

Идея применения ПЛК состоит в программировании и отработке заданного алгоритма функционирования технологического агрегата и исключает принципиальную необходимость предварительной разработки электрических схем в их традиционном представлении, а использование языка LD вынуждает всё же разработать релейный вариант схемы, привести её к некоторому нормализованному виду и лишь после этого приступить к собственно программированию. То есть имеет место не замена, а дополнение разработки электрических схем программированием.

Если учесть при этом, что разработка принципиальной электрической схемы всегда являлась наиболее сложным и наиболее трудоёмким этапом в проектировании систем управления технологическим оборудованием, становится очевидным, что технология программирования, основанная на применении LD, существенно снижает эффективность использования ПЛК как перспективного управляющего устройства. Кроме того, язык LD вряд ли можно рассматривать в качестве перспективного языка программирования, так как по мере перехода разработчиков систем управления на микроэлектронную элементную базу отпадёт необходимость в подготовке специалистов по релейным схемам, а обучение составлению принципиальных схем лишь для их последующего преобразования в управляющие программы представляется нецелесообразным.

Другой сложностью является то, что по мере роста объема программы, ее становится сложно читать и интерпретировать, если нет подробнейшей документации. Наконец, реализация полного процесса управления на языке релейно-контактных схем может быть чрезвычайно трудным.

FBD является графическим языком и наиболее удобен для программирования процессов прохождения сигналов через функциональные блоки. Язык FBD удобен для схемотехников, которые легко могут составить электрическую схему системы управления на "жесткой логике", но не имеют опыта программирования.

Функциональные блоки представляют собой фрагменты программ, написанных на IL, SFC или других языках, которые могут быть многократно использованы в разных частях программы и которым соответствует графическое изображение, принятое при

Библиотека ГГТУ им. П.О.Сухого

разработке функциональных схем электронных устройств, см. Рис. 2.2.

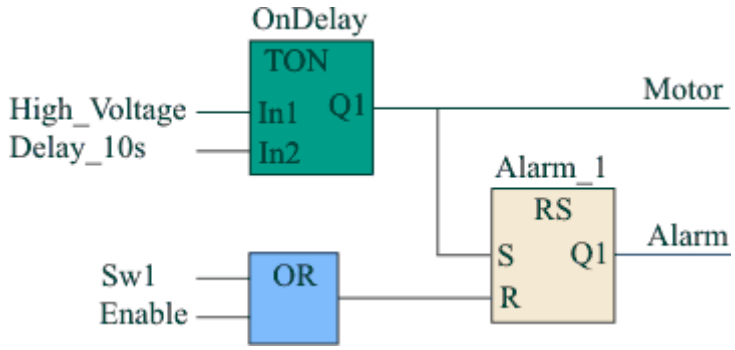


Рисунок 2.2 - Пример программы на языке FBD

Язык FBD может быть использован для программирования функций, функциональных блоков и программ, а также для описания шагов и переходов в языке SFC. Функциональные блоки инкапсулируют данные и методы, чем напоминают объектно-ориентированные языки программирования, но не поддерживают наследование и полиморфизм.

Типичным применением языка FBD является описание "жесткой логики" и замкнутых контуров систем управления. Язык функциональных блоков является удобным также для создания и пополнения библиотеки типовых функциональных блоков, которую можно многократно использовать при программирования задач промышленной автоматизации. К типовым блокам относятся блок таймера, ПИД-регулятора, блок секвенсора, триггера, генератора импульсов, фильтра, и т. п.

Многими аспектами этот графический язык напоминает электрическую схему даже больше, чем релейно-контактный язык. В функциональной блочной диаграмме блоки «соединены» вместе в последовательность, которую легко отслеживать. Этот язык использует такие же команды, как и релейно-контактный, но схема визуально более понятна пользователю, который не обладает специальными знаниями в релейной логике. Основным преимуществом этого языка является легкость отслеживания программы – просто двигайтесь по пути. Этот язык идеален для простых программ, состоящих из цифровых входов, таких как фотоэлектрические датчики, и выходов, таких как клапаны трубопроводов, и может использоваться в любых приложениях наряду с релейно-контактным языком или вместо него.

Библиотека ГГТУ им. П.О.Сухого

Однако, этот язык не идеален для больших программ, использующих специальные входы и выходы, а также функции. При использовании языка нужен большой объем экранного пространства, что делает программу нечитаемой при достижении определенного размера. Также при написании программы на языке функциональных блочных диаграмм требуется предварительная подготовка в виде прописывания алгоритма перед тем, как писать код, поскольку впоследствии будет достаточно сложно внести изменения.

Функциональные блоки являются не просто частью языка FBD, они применяются также для моделирования и проектирования систем автоматизации. Функциональные блоки могут быть использованы также для поддержания всего жизненного цикла системы, включая проектирование, изготовление, функционирование, валидацию и обслуживание. Описанию и применению функциональных блоков посвящены, помимо МЭК 61131-3, еще и стандарты МЭК 61499 и МЭК 61804.

Стандарт МЭК 61499 (См. Рис 2.3), состоящий из четырех частей, был опубликован в 2005 г. Он устанавливает обобщенную архитектуру функциональных блоков и предоставляет руководство для их применения в распределенных системах промышленной автоматизации. В таких системах программное обеспечение распределено между несколькими физическими устройствами (ПЛК) и несколькими функциональными блоками (ФБ), а промышленная сеть рассматривается как составная часть системы.

Особенностью ФБ в МЭК 61499 является возможность управления событиями и большая степень обобщения функциональных блоков. Стандарт МЭК 61499 может использоваться совместно с МЭК 61131-3 как средство описания базовых типов функциональных блоков для программирования ПЛК, а внутренне описание ФБ выполняется с помощью языков МЭК 61131-3.

Одной из существенных особенностей МЭК 61499 является ориентация на системы, в которых ФБ управляются событиями, в то время как традиционные системы автоматизации строятся обычно на базе тактирования или управления по временному расписанию. Событийное управление использовано потому, что в распределенных системах оно является более общим. Любая система с тактированием может быть представлена в виде системы с событийным управлением, но обратное не всегда верно.

Библиотека ГГТУ им. П.О. Сухого

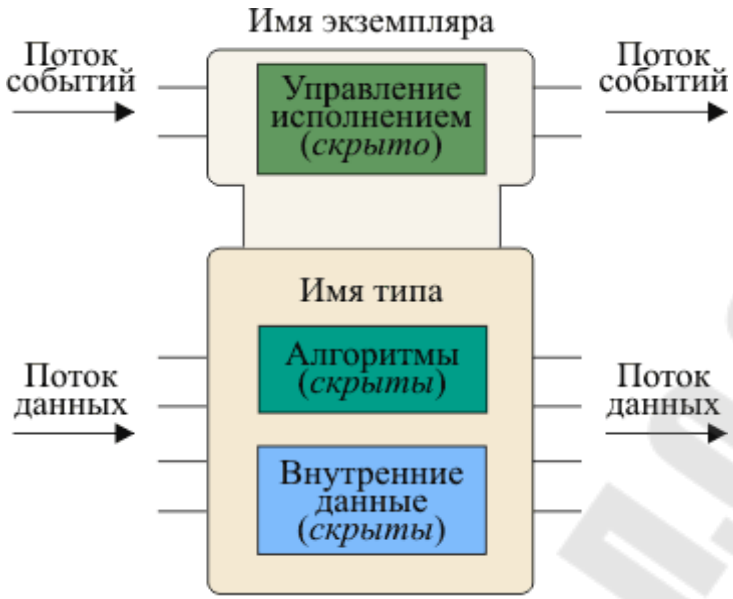


Рисунок 2.3 - Функциональный блок стандарта МЭК 61499

Функциональный блок характеризуется именем типа и именем экземпляра блока. Например, именем типа может быть "PID", а имен экземпляров может быть много: "PID1, PID 2, PID3, ...", по количеству ФБ, использованных в конкретной системе.

Каждый ФБ имеет множество входов и выходов для приема и передачи потока событий. Принятые события могут инициировать исполнение некоторых алгоритмов внутри блока, в результате чего могут вырабатываться события, которые передаются другим блокам системы.

ФБ имеет также множество входов, через которые поступает поток данных. Входящие данные отображаются во входные переменные, которые обрабатываются алгоритмами блока, после чего могут передаваться другим ФВ в виде выходящего потока данных. Блок может содержать также внутренние данные и соответствующие им внутренние переменные.

Каждый ФБ имеет свои функциональные характеристики, которые определяются комбинацией внутренних данных, состояний и алгоритмов, а также функциональными возможностями ресурсов устройства. Ресурс - это функциональный элемент, содержащийся в физическом устройстве и независимо управляющий его операциями, а также обеспечивающий различные сервисы для приложений, включая планирование и выполнение алгоритмов. Ресурс может быть создан, сконфигурирован, стартован, удален и т. д. без воздействия на другие ресурсы в устройстве. Функциями ресурса являются прием данных и

Библиотека ГГТУ им. П.О.Сухого

событий через входные интерфейсы, обработка и выдача их через выходные интерфейсы.

Третьим стандартом, развивающим представление о функциональных блоках, является МЭК 61804. Он содержит спецификацию (детализацию) требований к распределенным системам управления, построенным на основе функциональных блоков. МЭК 61804 конкретизирует абстрактные определения, данные в МЭК 61499. Он добавляет в МЭК 61499 описания параметров и функций, выполняемых функциональными блоками, которые могут быть реализованы в физических устройствах.

Стандарт определяет минимальный набор ФБ, который может быть необходим для индустриальных приложений. Набор состоит из двух частей: сложные ФБ (ПИД-регулятор, селектор для схем голосования, инкрементный сумматор, таймер, интегратор) и простые (вычисление тригонометрических функций, модуля, суммирования, усреднения, блоки арифметических операций, блоки Булевых функций и т. п.).

Одним из наиболее широко применяемых спецификаций стандарта МЭК 61804 является описание языка EDDL (Electronic Device Description Language), который является дальнейшим развитием методов генерации GSD файла в сетях Profibus и разрабатывался с поддержкой организации Fieldbus Foundation.

Описанию функциональных блоков для систем автоматизации зданий посвящен стандарт ISO 16484-3.

SFC называют языком программирования, хотя по сути это не язык, а вспомогательное средство для структурирования программ. Он предназначен специально для программирования последовательности выполнения действий системой управления, когда эти действия должны быть выполнены в заданные моменты времени или при наступлении некоторых событий. В его основе лежит представление системы управления с помощью понятий состояний и переходов между ними.

Язык SFC предназначен для описания системы управления на самом верхнем уровне абстракции, например (См. Рис 2.4), в терминах "Старт", "Наполнение автоклава", "Выполнение этапа №1", "Выполнение этапа №2", "Выгрузка из автоклава". Язык SFC может быть использован также для программирования отдельных функциональных блоков, если алгоритм их работы естественным образом описывается с помощью понятий состояний и переходов.

Библиотека ГГТУ им. П.О.Сухого

Например, алгоритм автоматического соединения модема с коммутируемой линией описывается состояниями "Включение", "Обнаружение тона", "Набор номер", "Идентификация сигнала" и переходами "Если длинный - то ждать 20 сек", "Если короткий - перейти в состояние "Набор Номера"" и т.д.

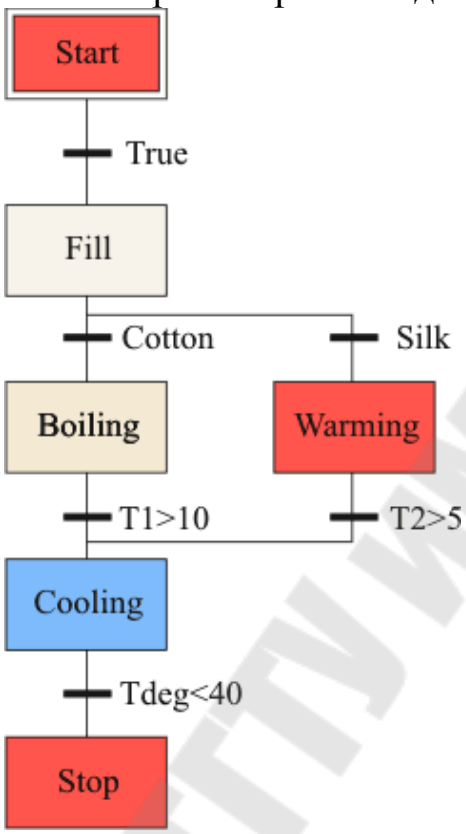


Рисунок 2.4 -Пример программы на языке SFC

Программа состоит из шагов и условий переходов. Шаги показываются на схеме прямоугольниками, условия переходов - жирной перечеркивающей линией. Программа выполняется сверху вниз. Начальный шаг на схеме показывается в виде двойного прямоугольника. Концепция SFC проста: шаг с внутренним кодом, написанным на любом языке программирования, активен до тех пор, пока не активен переход, следующий за ним. При активировании перехода, текущий шаг отключается, а следующий за переходом шаг становится активным. Переход также имеет код, проверяющий, что выполнены необходимые условия, позволяющие программе перейти к следующему шагу. Условия переходов записываются рядом с их обозначениями. Каждый шаг программы может представлять собой реализацию сложного алгоритма, написанного на одном из МЭК-языков.

Библиотека ГГТУ им.П.О.Сухого

Эту форму программирования легче всего использовать для приложений с повторяющимися многошаговыми процессами или последовательностью повторяющихся процессов. Примером может быть приложение, которое выбирает предмет в одной месте, проводит его по заданному пути и переносит в другое место. Поскольку обычно активен только один участок кода, и нужно следить только за одним переходом, проверка условий и управление процессом может быть достигнуто без больших сложностей. Язык очень подходит инженерам по обслуживанию, поскольку нагляден, и сегментация кода облегчает поиск неисправностей. Например, если механизм в программе перемещения предметов подходит к предмету, но не берет его, то инженер по обслуживанию или разработчик может найти в программе переход между шагом «переместиться к предмету» и шагом «взять предмет» и проверить, что мешает протеканию процесса.

К недостаткам языка относится то, что такой стиль программирования подходит не для всех приложений, поскольку структура, которая накладывается на программу, может ее излишне усложнить. Нужно потратить много времени на подготовку и планирование, прежде чем начать программировать, иначе функциональная диаграмма будет запутанной, и ее будет сложно отслеживать.

Дополнительные ресурсы, требующиеся для такого программирования, приводят к замедлению процесса написания программы по сравнению с другими языками. Наконец, нужно принять во внимание невозможность конвертирования в другие языки

2.2. Функционирование контроллеров ОВЕН

На рис. 2.5 Представлен внешний вид ПЛК ОВЕН. Остановимся на устройствах ввода/вывода. Они, в отличие, например, от ПИД-регуляторов, не только должны быть описаны в программе, но и существовать физически в виде клемм. Часть этих клемм может располагаться в головном модуле (CPU). Но большинство входных и выходных сигналов поступает на различные модули расширения, от которых передаётся по промышленным интерфейсам связи или внутренним информационным шинам ПЛК в головной модуль.

Библиотека ГГТУ им. П.О. Сухого

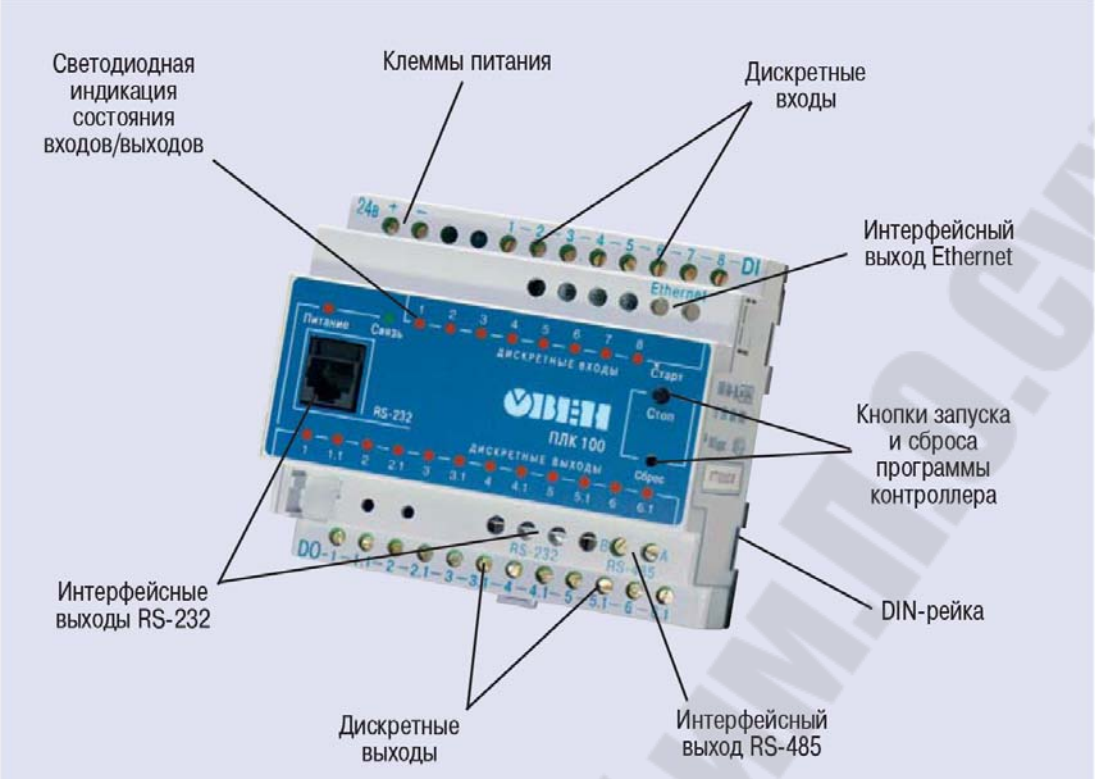


Рисунок 2.5 –ПЛК ОВЕН ПЛК-100 и его функционал

Производители и потребители выработали определённую классификацию контроллеров в зависимости от того, какое количество сигналов можно к ним подвести. Нано-контроллеры - самые маленькие, около 20 сигналов. Микро-контроллеры - уже больше, порядка 100 сигналов. Средние и большие программируемые контроллеры позволяют обрабатывать сотни и тысячи сигналов. ОВЕН ПЛК можно отнести к классу микро-ПЛК. Так, в нескольких тестовых проектах на нашем контроллере реализовано подключение около 100 датчиков и исполнительных устройств, таких как термометры сопротивления и термопары, датчики давления и расходомеры, лампы индикации и контакты пускателей, управляющих насосами, клапанами и ТЭНами.

Очевидно, что при большом количестве датчиков и исполнительных механизмов необходимо использовать дополнительные модули расширения. При подборе контроллера ОВЕН вам нужно оценить необходимое число каналов измерения и управления и выбрать тип лицензии («L» или «M»), которая определяет количество обрабатываемых сигналов. Лицензия «L» ограничивает объём памяти ввода/вывода, достаточной для обработки не более 70 аналоговых сигналов. Каждый аналоговый сигнал занимает объём, достаточный

Библиотека ГГТУ им. П.О. Сухого

для обработки 16 дискретных сигналов. При наличии лицензии «М» количество каналов ограничено лишь возможностями ОВЕН ПЛК. Такой контроллер производится на заказ и стоит дороже.

Язык IL напоминает ассемблер и используется для реализации функций, функциональных блоков и программ, а также шагов и переходов в языке SFC. Основным достоинством языка является простота его изучения. Наиболее часто язык IL используется в случаях, когда требуется получить оптимизированный код для реализации критических секций программы, а также для решения небольших задач с малым количеством разветвлений алгоритма.

В основе языка лежит понятие аккумулятора и переходов по меткам. Пример программы на зыке IL с комментариями приведен в листинге.

Листинг . Пример программы на языке IL

Метки	Операторы	Операнды	Комментарии
	LD	Voltage	(*Загрузить Voltage в аккумулятор*)
	GT	220	(*Если >220*)
	JMPCN	M1	(*Перейти к метке, если ">220" не верно*)
	LD	Current	(*Загрузить Current в аккумулятор*)
	SUB	10	(*Вычесть из аккумулятора 10 *)
	ST	Current	(*Присвоить Current значен. аккумулятора*)
M1:	LD	0	(*Загрузить в аккумулятор значение "0"*)
	ST	Out	(*Присвоить Out значение аккумулятора*)

Начинается программа с загрузки в аккумулятор значения переменной. Дальнейшие шаги программы состоят в извлечении содержимого аккумулятора и выполнении над ним ограниченного числа допустимых действий (их в языке всего 24).

Язык IL – это язык нижнего уровня, и как таковой работает в ПЛК быстрее графических языков. Этот язык также более компактен

Библиотека ГГТУ им. П.О. Сухого

и потребляет меньше памяти ПЛК. Метод построчного текстового ввода, поддерживаемый этим языком, также позволяет очень быстро вводить программу, при этом не требуется мышка или функциональные клавиши. Программы в современных системах автоматизации, написанные на этом языке, легче воспроизводить и редактировать на портативных устройствах, для чего не требуется дополнительного программного обеспечения или ноутбука.

Несмотря на преимущества данного языка, инженеры по обслуживанию его не очень любят. Возможно, это вызвано тем, что он менее нагляден, чем язык релейно-контактных схем, и поэтому труднее понять, что делает программа, и какие ошибки имеют место быть в ней. Аналогично релейно-контактной схеме по мере увеличения сложности ПЛК, в списке инструкций могут возникнуть сложности при вводе таких сложных функций, например, ПИД – регулирование. Это также относится и к сложным математическим расчетам. Список инструкций не очень подходит для таких форм структурного программирования, как диаграмма состояний или ступенчатая многозвенная схема, что ограничивает его полезность для реализации больших программ. Также спорным является факт, что преимущества скорости и компактности утрачивают свое значение по мере увеличения скорости работы современных ПЛК и большого объема доступной памяти.

Язык ST является текстовым языком высокого уровня и очень сильно напоминает Паскаль:

Листинг. Пример программы на языке ST

```
IF Voltage>220 THEN

  Current:=Current - 10;  (*Если V>220 В, то уменьшить ток на 10*)

ELSE

  Current:=50; Speed:= ON;(*Установить ток 50А и включить мотор*)

END_IF;
```


Библиотека ГГТУ им. П.О. Сухого

Язык ST имеет много отличий от языка Паскаль и разработан специально для программирования ПЛК. Он содержит множество конструкций для присвоения значений переменным, для вызова функций и функциональных блоков, для написания выражений условных переходов, выбора операторов, для построения итерационных процессов.

Этот язык предназначен в основном для выполнения сложных математических вычислений, описания сложных функций, функциональных блоков и программ. Этот язык лучше всего подходит для сложного программирования ПЛК, такого как, например, управление процессами в производстве пластмасс или химической промышленности. Тригонометрические функции, математические вычисления и анализ данных на этом языке можно реализовать легче, чем на языке релейно-контактных схемах или языке списка инструкций.

Циклы выбора и указатели (переменные, используемые для косвенной адресации) позволяют реализацию более компактных программ, чем могут быть созданы на языке релейно-контактных схем. Для написания программы на языке ST используется удобный текстовый редактор, который облегчает ввод комментариев в программу, а также позволяет использовать знаки абзацев и пробелы для выделения связанных участков кода. Это облегчает задачу структурирования комплексных программ. Текстовый, неграфический характер языка ST, похожего на язык IL, позволяет создавать программы, которые работают гораздо быстрее, чем программа созданные на языке LD. Дополнительным преимуществом языка ST является то, что он ближе других языков программирования подошел к достижению переносимости, обещанной стандартом IEC61131. Копирование и вставка в языке ST из редактора одного программного пакета в другой часто может быть выполнено с минимальными изменениями, освобождая программиста от аппаратной платформы. Окончательным преимуществом является то, что многие студенты инженерных специальностей лучше владеют компьютерными языками, чем основами электротехники, и поэтому лучше владеют языком ST, чем LD.

Недостаток языка ST заключается в том, что для многих старых специалистов в области программирования и отладки среда языка ST является чем-то незнакомым и неудобным. В определенном смысле, код и структура необходимые, чтобы сделать поддержку этого кода

Библиотека ГГТУ им. П.О. Сухого

удобной, снижают преимущества, связанные с компактностью программ. В результате основной тенденцией использования языка ST является его использование так сказать «за сценой». Например, IEC61131 позволяет программисту реализовать функции на одном языке, а затем использовать их в другом языке. Таким образом, программист, скорее всего, включит программу на языке ST внутрь команды, вызываемой на языке LD. Это не обязательно является недостатком, но программисту понадобится тщательно протестировать любой «скрытый» код, и удостовериться в отсутствии ошибок, поскольку у других, кто будет использовать этот код, возможно доступа к данному коду не будет.

2.3. Среда разработки CoDeSys

CoDeSys (**Co** ntroller **De**velopment **Sys**tem) представляет собой комплекс программ для проектирования прикладного программного обеспечения, отладки в режиме эмуляции и загрузки программы в ПЛК (См. Рис 2.6). Основными частями системы являются среда разработки программы и среда ее исполнения (CoDeSys SP), которая находится в ПЛК.

В CoDeSys входят графические и текстовые редакторы для всех пяти языков МЭК 61131-3. Этот комплекс полностью реализует требования стандарта и дополнительно вводит ряд оригинальных расширений, самым удобным из которых является объектно-ориентированные программирование. Однако расширениями языка можно не пользоваться, чтобы сохранить требования к совместимости языков, предъявляемое к открытым системам.

В одном проекте может быть использовано несколько контроллеров разных производителей. Каждый из них может программироваться как независимое устройство или с учетом их взаимодействия в промышленной сети. Проект состоит из нескольких приложений, распределенных по нескольким контроллерам. В одном ПЛК может существовать несколько независимых приложений.

Программа, написанная на языках МЭК, компилируется системой CoDeSys в машинный код, оптимизированный для заданной аппаратной платформы. Компилятор выдает диагностические сообщения как на этапе компиляции, так и на этапе ввода операторов языка.

Библиотека ГГТУ им. П.О. Сухого

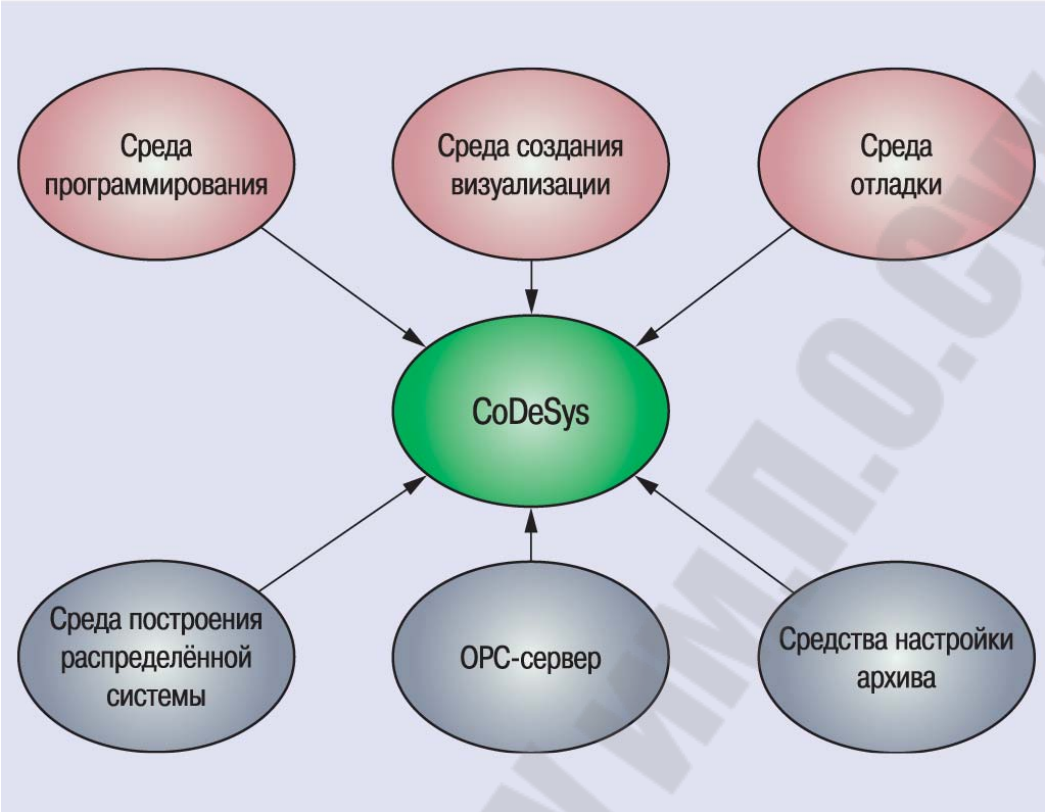


Рисунок 2.6 – Возможности среды разработки CoDeSys

Машинный код, сгенерированный компилятором CoDeSys, загружается в ПЛК, после чего разработчик имеет возможность использовать широкий набор функций для быстрой и эффективной отладки приложения. Текущие значения переменных видны непосредственно в редакторах программ. Программу можно выполнять по шагам или по контроллерным циклам. Можно задавать точки останова программы, просматривать стек вызовов, подготавливать связные наборы значений переменных и загружать их одной командой.

При отсутствии реального контроллера отладку программы можно выполнять с помощью встроенного программного эмулятора.

Система имеет также встроенный многоканальный программный трассировщик (графический самописец) значений переменных. Он позволяет наглядно представить динамически изменяющиеся данные проекта. Данные аккумулируются в памяти ПЛК и могут синхронизироваться с определенными событиями. Трассировщик

Библиотека ГГТУ им. П.О. Сухого

полезен не только при отладке, но и при анализе нештатных ситуаций в процессе эксплуатации оборудования.

После изменения программы во время отладки перекомпилируются только измененные части программы. Их можно подгружать в контроллер без остановки выполнения прикладной программы. Эта возможность системы называется "горячим обновлением" кода.

Программируемое устройство соединяется с CoDeSys через вспомогательный программный компонент – шлюз связи, который использует протокол TCP/IP. Шлюз работает на компьютере программиста или удаленно, например, через интернет или сеть Ethernet. Контроллер подключается компьютеру через любой последовательный канал или сеть. Добавив драйвер, изготовитель ПК может поддержать свой оригинальный протокол связи.

Общение ПЛК со SCADA осуществляется с помощью стандартного OPC сервера.

Для того, чтобы ПЛК можно было программировать с помощью CoDeSys, в контроллере должна быть установлена система исполнения. Установку системы выполняет изготовитель контроллера. Изготовитель обеспечивает также поддержку всех модулей ПЛК, поэтому конечный пользователь может сосредоточиться на разработке только прикладной программы.

Среда исполнения CoDeSys может функционировать в ПЛК под управлением различных операционных систем или вообще без них, в том числе на обычном персональном компьютере. Собственное ядро реального времени может устанавливать контроллерный цикл с точностью до нескольких микросекунд. Прикладная программа остается работоспособной даже при зависании ОС.

Помимо средств программирования, CoDeSys имеет встроенную систему визуализации, которая применяется для операторского управления, а также моделирования на этапе разработки. Визуализацию можно запустить на компьютере, графической панели ПЛК или встроенном в контроллер web-сервере.

Пользователь может самостоятельно расширять возможность CoDeSys путем создания библиотек программных модулей. Например, он может реализовать поддержку нестандартных интерфейсов.

Комплекс программирования CoDeSys построен по компонентной технологии Microsoft на базе автоматизации. Поэтому

Библиотека ГГТУ им. П.О.Сухого

изготовитель ПЛК может включить в комплекс свои собственные компоненты, от конфигуратора оригинальной сети до собственного языка программирования ПЛК.

Для систем, связанных с безопасностью, CoDeSys имеет библиотеку функциональных блоков PLCopen Safety, систему исполнения для оборудования с дублированием и специализированное расширение среды программирования. При внезапном отключении питания CoDeSys автоматически сохраняет значения переменных во флеш-памяти или в ОЗУ с батарейным питанием.

Компанией ITQ GmbH в 2011 г. было проведено исследование характеристик и распространенности программных инструментов в областях машиностроения и мобильных применений в Европе. По его результатам, CoDeSys и инструменты на его базе (Bosh Rexroth IndraWorks, Beckhoff TwinCAT и др) используют 36% компаний. Конкурирующие с CoDeSys универсальные инструменты совместно составили 7%.

Как продукт, CoDeSys ориентирован на изготовителей контроллеров. Разрабатывая новый контроллер, они устанавливают в него систему исполнения CoDeSys Control. Собирают из ее компонентов требуемую конфигурацию, добавляют собственные ноу-хау и специфические компоненты и получают собственное инструментальное ПО. Как правило, к пользователю CoDeSys попадает в коробке вместе с оборудованием. Ему нужно только установить систему и перейти к решению своих практических задач. Все коммерческие и технические вопросы, связанные с поддержкой ядра контроллера, всех типов его аппаратных модулей, библиотек, стеков и конфигураторов сетей его беспокоить не должны. Все это должно быть решено за него разработчиками ПЛК и CoDeSys совместно.

Среда программирования - это та часть, с которой непосредственно имеет дело пользователь (рис.2.7). Она функционирует на ПК и является основным компонентом комплекса. На выходе CoDeSys непосредственно дает быстрый машинный код. Поддержаны все распространенные семейства микропроцессоров от 16 до 64-разрядных.

Библиотека ГГТУ им. П.О.Сухого

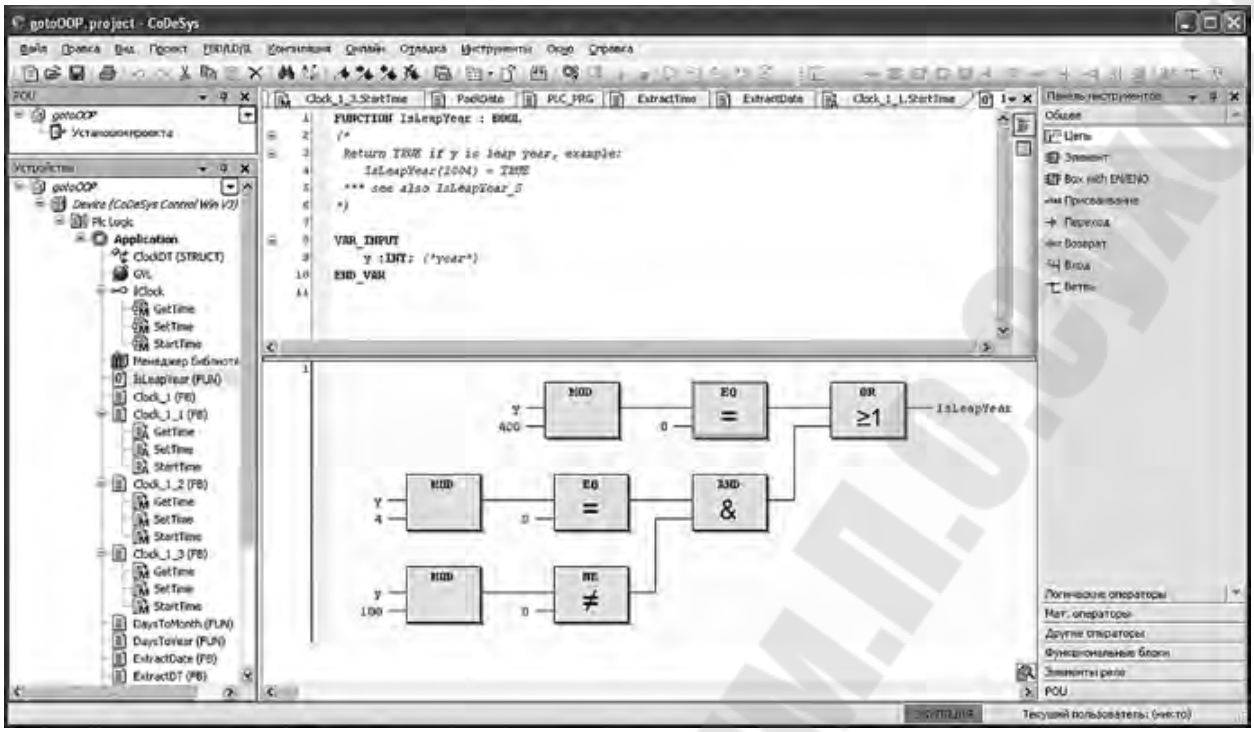


Рисунок. 2.7. Редактирование FBD диаграммы в CoDeSys

Среда программирования CoDeSys включает набор инструментов для подготовки и отладки программ, компиляторы, конфигураторы, редакторы визуализации и т.д. При необходимости функциональность системы дополняется опциональными компонентами. Проект CoDeSys можно хранить не только на диске ПК, но и в контроллере, если он имеет достаточный объем памяти, что позволяет избежать потери исходных текстов или путаницы в проектах. Для больших проектов предусмотрено использование системы контроля версий (SVN).

Для отладки пользователю не нужно открывать специальных отладочных окон или составлять каких-либо списков переменных. При подключении к ПЛК редакторы ввода программ "оживают". Непосредственно в них отображаются значения всех видимых на экране переменных. Причем в сложных выражениях видны все промежуточные результаты.

Из новшеств CoDeSys, добавленных за последний год, следует отметить странично-ориентированный FBD и поддержку языка Python для автоматизации работы в среде программирования. Обычно для таких целей используются пакетные файлы. Они удобны для примитивных задач, но не позволяют выполнять разные действия по условиям, разобрать XML файл, обработать результаты и отправить их по электронной почте. Использование Python снимает все

Библиотека ГГТУ им. П.О. Сухого

мыслимые ограничения. CoDeSys включает конфигураторы ввода/вывода с поддержкой полевых сетей Modbus, PROFIBUS, PROFINET, DeviceNet, CANopen, J1939, EtherCAT, SERCOS III, Ethernet IP и большое число сервисных модулей.

CoDeSys поставляется бесплатно. С сайта 3S-Smart Software Solutions доступен для загрузки полнофункциональный дистрибутив. В него входит интерфейс и интерактивная документация на русском языке.

CoDeSys Control - это часть, которая должна быть встроена ПЛК. Нередко возникает вопрос: "Если CoDeSys даёт на выходе машинный код, то зачем вообще нужна система исполнения?" Ответ кроется в стержневой идее технологии ПЛК. Программируя ПЛК, пользователь должен думать исключительно о сути прикладной задачи. Его не должны волновать организация памяти, процедуры опроса модулей ввода/вывода, способы синхронизации данных, функции сетевого обмена и связи с верхним уровнем, вызовы циклических и событийных задач, организация фиксации выходов при отладке на оборудовании и т. п. Так, для получения значения входа в своей программе, прикладной программист ПЛК выбирает переменную и задает в диалоговом окне единицы измерения, параметры фильтрации и другие параметры. Всю черновую работу за него должна выполнить система исполнения. Если программисту приходится думать о передаче байтов или вызове библиотечных функций для работы с вводом/выводом, то это не ПЛК? и говорить об удобстве и надежности прикладного программирования не приходится.

В общей сложности CoDeSys Control включает более 200 компонентов. Каждая "сборка" под конкретную модель ПЛК будет отличаться. Ее состав определяется возможностями аппаратуры и типом ПЛК. Включение абсолютно всех компонентов, на всякий случай, привело бы к неоправданному росту аппаратных ресурсов и стоимости. Например, включение функции "горячей" правки кода без остановки ПЛК удваивает требования к ОЗУ. Некоторые компоненты представлены в нескольких вариантах. Например, компонент "менеджер задач". Самый дешевый ПЛК может иметь единственный аппаратный таймер, "тикающий" каждые 10 мс, и не иметь ОС. Для него подойдет простой планировщик циклических задач без вытеснения. С ним не смогут работать некоторые другие компоненты, например, ЧПУ или стек CANopen, но они и не требуются в ПЛК такого уровня. Для ПЛК с мощным 32- или 64-битным процессором и

Библиотека ГГТУ им. П.О. Сухого

ОС РВ разумно включить наиболее совершенный "менеджер задач" с поддержкой событий, реального времени и нескольких приложений в одном устройстве. С каждым таким приложением можно работать как с независимым ПЛК: загружать, запускать, останавливать и отлаживать программы, не влияя на работу других приложений.

CoDeSys Control может функционировать под управлением любой ОС или даже без нее. Наиболее часто используют ОС VxWorks, Windows CE и Linux. Имеются адаптации под RT-OS32 (RTKernel), QNX, Nucleus, pSOS, OS9, TenAsys INtime. Изготовитель оборудования может самостоятельно адаптировать CoDeSys Control под другую ОС.

CoDeSys SP RTE представляет собой специальную систему исполнения для ОС семейства Windows со встроенным ядром жесткого реального времени. Она позволяет превратить обычный компьютер в быстродействующий ПЛК. Ввод/вывод подключается через полевые сети. SP RTE обеспечивает стабильность рабочего цикла МЭК программ в диапазоне микросекунд и работу контроллера при зависании ОС.

Среда CoDeSys оснащена встроенной системой визуализации и операторского управления. Непосредственно в CoDeSys можно построить графический интерфейс оператора или модели объекта, без использования внешних инструментов. Для интеграции с программой достаточно прописать в свойствах элементов соответствующие переменные. Не требуется создавать символьные файлы, настраивать связь или выполнять иные рутинные операции. Если работает CoDeSys, то работают и средства визуализации.

CoDeSys HMI часто называют SCADA-системой. Это не верно. Она не имеет столь мощных графических средств, не использует OPC, не имеет средств ведения суточных архивов и интеграции с БД, а также функций программирования. Но она обеспечивает управление в реальном времени и на порядок менее требователена к ресурсам. Весь интеллект системы сосредоточен в ПЛК, а HMI выполняет роль тонкого клиента отображения. Ее типичные применения - это встроенные пульты управления станками, погрузчиками, кранами, трамваями и подобными системами, где нужна быстрая гарантированная реакция и стоимость оборудования критична. Сервер данных (Data Server) позволяет собирать данные от нескольких контроллеров. При этом не обязательно, чтобы все они

Библиотека ГГТУ им. П.О.Сухого

программировались в CoDeSys. Сервер данных является частью системы исполнения.

Визуализация CoDeSys может параллельно работать на нескольких устройствах:

- CoDeSys WebVisu позволяет контролировать работу своей системы из любого места и в любое время через Internet. Web-сервер является компонентом системы исполнения;

- CoDeSys HMI - это отдельная утилита, предназначенная для операторского управления с отдельного компьютера локальной сети;

- CoDeSys TargetVisu - интегрированный компонент системы исполнения, предназначенный для создания панельных ПЛК. Применяется в локальных пультах управления.

В CoDeSys выделяется пакет библиотек элементов визуализации для различных прикладных областей с современным графическим представлением. Наиболее впечатляющим элементом можно назвать 3D редактор движений для SoftMotion.

CoDeSys SoftMotion - это встроенный в среду программирования и систему исполнения CoDeSys функциональный набор средств управления движением: от простых перемещений по одной оси до многоосевых ЧПУ. Поддерживается движение по лекалам (ECAM) и интерпретация программ в G-кодах (См. Рис 2.8). В среду программирования встроен текстовый и графический 3D редактор для задания траекторий и набор элементов визуализации стандартных узлов мехатроники. Установить SoftMotion можно на 32-битный ПЛК с математическим сопроцессором.

Комплекс Safety ориентирован на обеспечение безопасности там, где присутствует человек. CoDeSys Safety представляет собой комплекс инструментов, который позволяет разрабатывать контроллеры, удовлетворяющие требованиям стандарта IEC 61508 для оборудования систем безопасности Safety Integrity Levels 3 (SIL3). Он включает безопасную систему исполнения, безопасный компилятор, конфигураторы безопасных сетей, библиотеки PLCopen Safety и набор документов, включающий методику тестирования и сертификации. Эта технология существенно сложнее обычных ПЛК систем. Так, например, до запуска кода выполняется целый ряд специальных проверок. После загрузки машинного кода в контроллер и создания загрузочного образа код скачивается обратно в среду разработки, производится его декомпиляция и сравнение с исходным текстом. Безопасные контроллеры уровня SIL3 должны проходить

Библиотека ГГТУ им. П.О.Сухого

обязательную сертификацию. Это весьма сложный и дорогостоящий процесс. Применение CoDeSys Safety позволяет существенно упростить его.

```
N000 F100 E100 E-100
N010 G17 (activate 3-D mode)
N020 G01 Z0 (move to start-position)
N030 G51 D500 (enable smooth path)
N040 G01 Z200
N050 G01 X200 Y200
N060 G01 X400 Y0
N070 G01 Z0
N080 G01 Z0
N090 G01 Z200
N100 G01 X200 Y200
N110 G01 X0 Y0
N120 G01 Z0
N130 G50 (disable smooth path)
```

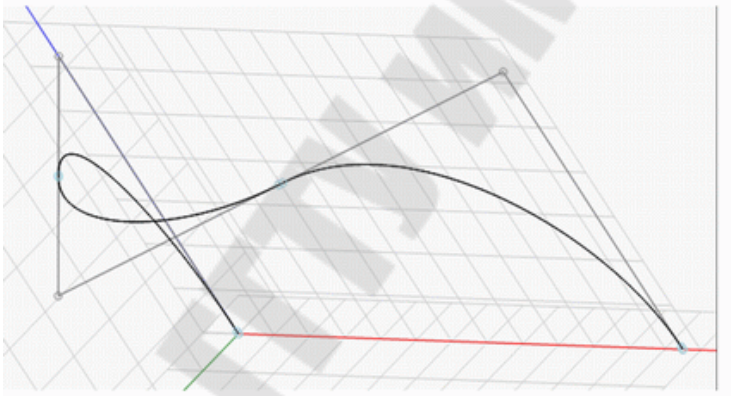


Рисунок 2.8 - G-код движения в CoDeSys и его визуальное представление

CoDeSys Professional Developer Edition - новый продукт комплекса CoDeSys. Он ориентирован на растущую группу пользователей, имеющих высшее образование и опыт работы с современными профессиональными системами программирования на языках высокого уровня для компьютеров. Его область - создание крупных, либо новых уникальных проектов, не имеющих аналогов. Профессиональная редакция среды разработки CoDeSys включает следующие компоненты: систему управления версиями проекта на базе Subversion (SVN), графические редакторы UML (диаграммы классов, состояний и деятельности) и статический анализатор кода. Все эти компоненты устанавливаются и интегрируются в среду программирования. Система контроля версий необходима в больших

Библиотека ГГТУ им. П.О.Сухого

проектах над которыми работает группа людей. При сохранении изменений в обычном файле проекта CoDeSys, они записываются поверх старой информации. Она теряется бесследно. При использовании SVN сохраняется вся история исправлений с указанием кто исправлял, когда и с какой целью. Если правка вызвала сбои, то всегда есть возможность вернуться к проверенной версии на любую дату. Кроме того, один проект могут открыть несколько людей со своих рабочих мест. Каждый человек может править параллельно 'свои' части. Контроль версий естественным образом интегрируется в среду программирования. Так, для всех языков программирования, включая графические, предусмотрены визуальные средства сравнения версий.

Интеграция UML стала следующим логическим шагом после реализации в CoDeSys ООП. Диаграмма классов дает визуальное представление зависимостей функциональных блоков, методов и интерфейсов, которые теперь могут редактироваться графически. Диаграммы состояний и диаграммы активности представляют собой новые языки для разработки прикладных проектов. Они позволяют описывать состояния и переходы сложных процессов. Оба высокоуровневых языка призваны упростить совместную работу программистов и инженеров технологов, ускорить построение структуры приложения и собственно программирование.

Статический анализатор кода осуществляет проверку исходного кода МЭК программ на соблюдение более чем 50 адаптивных правил. Он выявляет потенциально опасные места, способные вызвать ошибки и устранить их сразу, еще до фазы отладки и тестирования проекта. Это улучшает качество кода, ускоряет работу и позволяет избежать ошибок с самого начала работы.

Благодаря своим отличным функциональным возможностям, надежности и открытым интерфейсам, CoDeSys является одним из лидеров в области инструментов программирования ПЛК. Не случайно он выбран в качестве базового инструмента многими ведущими мировыми поставщиками аппаратных решений для промышленной автоматизации.

Библиотека ГГТУ им. П.О.Сухого

3. Программная реализация алгоритмов управления в системах автоматизации на базе программируемых логических контроллеров

3.1. Классификация аппаратных и программных средств микропроцессорных систем управления

Основные операции ПЛК соответствуют комбинационному управлению логическими схемами. Кроме того, современные ПЛК могут выполнять другие операции, например функции счетчика и интервального таймера, обрабатывать задержку сигналов и т. д. Основное преимущество ПЛК заключается в том, что одиночная компактная схема может заменить сотни реле. Другое преимущество — функции ПЛК реализуются программно, а не аппаратно, поэтому его поведение можно изменить с минимальными усилиями. С другой стороны, ПЛК могут быть медленнее, чем релейная аппаратная логика. Оптимальное решение для каждого конкретного приложения можно получить, применяя обе технологии в одной системе так, чтобы использовать преимущества каждой из них.

Первые ПЛК были сконструированы только для простых последовательностных операций с двоичными сигналами. Сегодня на рынке существуют сотни различных моделей ПЛК, которые различаются не только размером памяти и числом каналов ввода/ вывода (от нескольких десятков до нескольких сотен), но и выполняемыми функциями. Небольшие ПЛК предназначены в основном для замены реле и имеют некоторые -дополнительные функции счетчиков и таймеров. Более сложные ПЛК обрабатывают налоговые сигналы, производят математические операции и даже содержат контуры управления обратной связи, как ПИД-регуляторы.

Конструктивно ПЛК обычно приспособлены для работы в типовых промышленных условиях, с учетом уровней сигналов, термо- и влагостойкости, ненадежности источников питания, механических ударов и вибраций. ПЛК также содержат специальные интерфейсы для согласования и предварительной обработки различных типов и уровней сигналов. Типичный серийный ПЛК показан на рис. 3.1.

Библиотека ГГТУ им. П.О.Сухого

Рисунок 3.1 – Внешний вид ПЛК Direct logic 06

На рис. 3.2 показана основная структура программируемого логического контроллера. Сначала входные сигналы считываются в регистр буферной памяти. Эта функция всегда включается в системное программное обеспечение ПЛК и не требует явного программирования пользователем. Входной регистр может состоять и из одного бита, и из целого байта. В последнем случае один оператор считывания будет выдавать одновременно значения восьми входных бит.

Программа может выбрать входное значение из этого регистра и затем обработать отдельно или вместе с другими данными. Выработанный результат можно либо сохранить для дальнейшей обработки, либо направить на выход.

В системах промышленной автоматики ПЛК должны работать в режиме реального времени, т. е. быстро реагировать на внешние события. Ввод и обработка внешних сигналов осуществляется в ПЛК двумя способами — по опросу или по прерыванию.

Библиотека ГГТУ им. П.О. Сухого

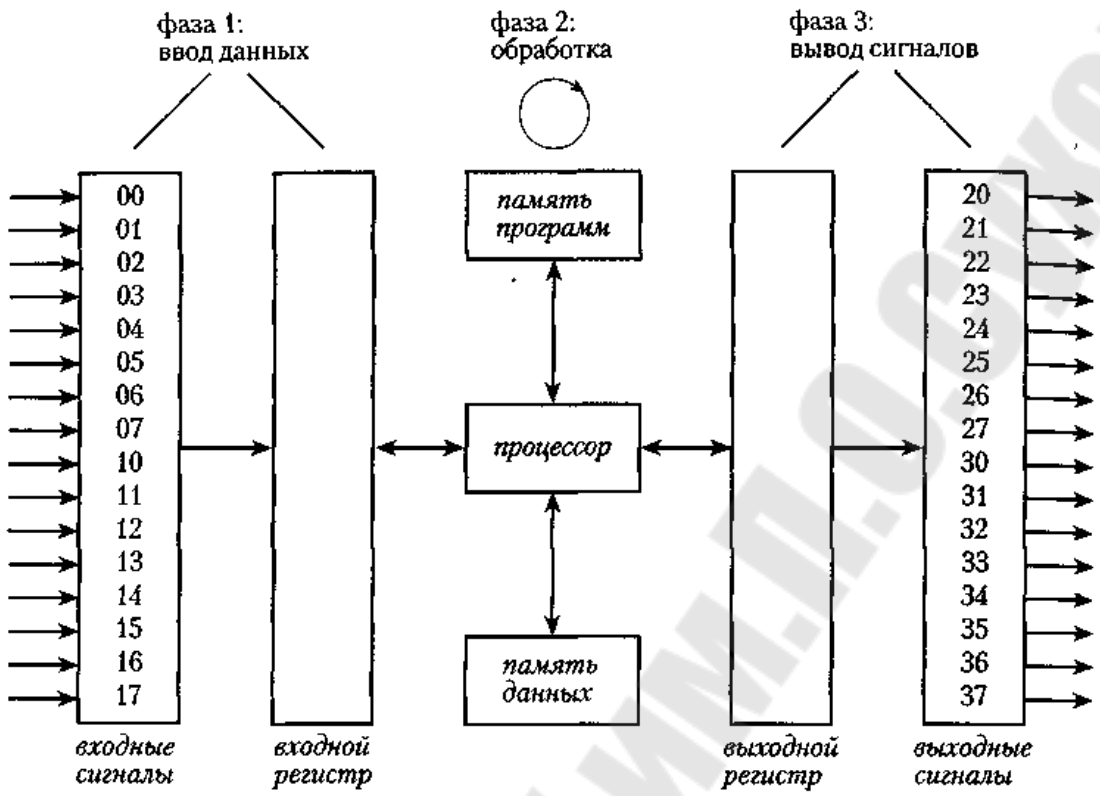


Рисунок 3.2 – Типовая структура ПЛК

Основной недостаток опроса — можно потерять некоторые внешние события, если ПЛК не обладает достаточным быстродействием, хотя такой подход проще для программирования. Управление по прерываниям сложнее для программирования, но риск пропустить какое-либо внешнее событие намного меньше. Управления по опросу обычно вполне достаточно для простых систем, а управление по прерыванию используется в сложных случаях.

Программирование ПЛК в основном представляет собой описание управляющих последовательностей. Функции ввода/вывода уже реализованы в базовом программном обеспечении ПЛК. Выполнение программы происходит в бесконечном цикле. Каждый полный цикл сканирования для малых ПЛК составляет примерно 15-30 мс, и это время приблизительно пропорционально размеру программы.

Скорость реакции ПЛК, очевидно, зависит от продолжительности цикла, поскольку во время исполнения программы процессор ПЛК не может считывать или выдавать какие-либо новые сигналы. Обычно это не очень серьезная проблема, так как большинство сигналов в промышленной автоматике изменяются

Библиотека ГГТУ им. П.О. Сухого

сравнительно медленно либо имеют относительно большую продолжительность.

Небольшого набора базовых машинных команд, как правило, достаточно для большинства задач последовательностного управления.

При выпуске средств автоматизации зачастую используют классический закрытый подход - тщательно охранять секреты технологии (несовместимая продукция) и привязать потребителей к своим изделиям. Но удовлетворить каждое конкретное пожелание индивидуального заказчика практически невозможно и дорого. Недовольство одной малосущественной деталью может привести к отказу от продукции данной фирмы вообще.

При производстве совместимой продукции (подчиненной требованиям открытого стандарта) производитель может развивать свои удачные решения и внедрять их даже в полностью захваченных областях рынка. Так начинающие коллективы получают шанс найти свое место среди промышленных гигантов. Тем самым расширяется и сам рынок. Выгоду открытого подхода наглядно доказала фирма IBM на примере своих ПК.

Международная Электротехническая Комиссия (МЭК) сделала стандарт, первый вариант которого был опубликован в 1982 году. Ввиду сложности он разбит на несколько частей.

Часть 1. Общая информация.

Часть 2. Требования к оборудованию и тестам.

Часть 3. Языки программирования.

Часть 4. Руководства пользователя.

Часть 5. Спецификация сообщений.

Часть 6. Полевые сети.

Часть 7. Программирование с нечеткой логикой.

Часть 8. Руководящие принципы применения и реализации языков ПЛК.

3.2. Принципы функционирования ПЛК

Для функционирования ПЛК как правило необходим следующий набор отладочных средств:

• *унифицированный механизм соединения с ПЛК*. (Отладка не зависит от способа соединения контроллера с отладчиком: эмуляция на ПК, подключение через COM порт или связь через Интернет;)

Библиотека ГГТУ им. П.О.Сухого

• *загрузка кода управляющей программы в оперативную память и электрически перепрограммируемую память ПЛК*;

• *автоматический контроль версий кода*. Проверка соответствия кода содержащегося в памяти ПЛК и кода полученного после текущей компиляции;

• *выполнение управляющей программы в режиме реального времени*;

• *режим останова*. Останов означает прекращение выполнения только кода управляющей программы. Все прочие фазы рабочего цикла выполняются. Способность наблюдать значения входов и управлять выходами ПЛК вручную сохраняется. В этом режиме можно проводить тестирование и настройку датчиков и механизмов объекта управления;

• *сброс ПЛК*. Может быть несколько видов сброса. В стандарте МЭК предусмотрено два вида сброса «горячий» и «холодный».

Первый включает перевод управляющей программы в исходное состояние и выполнение начальной инициализации переменных

Во втором виде сброса добавляется начальная инициализация переменных, размещенных в энергонезависимой области памяти.

В CoDeSys предусмотрен еще и «заводской» сброс (original), удаляющий пользовательскую программу и восстанавливающий состояние контроллера, в котором он поступает с завода изготовителя.

Кроме того, в ПЛК может произойти аппаратный сброс путем выключения питания или перезапуска микропроцессора. Система программирования должна адекватно реагировать в случае аппаратного сброса.

Детальная реакция на команды сброса определяется системой исполнения. Поэтому здесь возможны некоторые отличия для разных ПЛК, даже в одной среде программирования

• *мониторинг и изменение мгновенных значений всех переменных проекта, включая входы-выходы ПЛК.* Для удобства работы значения представляются в заданной пользователем системе счисления*;*

• *фиксацию переменных, включая входы-выходы.* Фиксированные переменные будут получать заданные значения в каждом рабочем цикле независимо от реального состояния ПЛК и действий управляющей программы. Данная функция позволяет имитировать элементарные внешние события в лабораторных условиях и избегать нежелательной работы исполнительных механизмов при отладке на

Библиотека ГГТУ им. П.О.Сухого

«живом» объекте управления. Неуправляемая работа механизмов может привести к поломке и представлять опасность для окружающих людей;

• *выполнение управляющей программы шагами по одному рабочему циклу.* Применяется при проверке логической правильности алгоритма;

• *просмотр последовательности вызовов компонентов в точке останова;*

• *графическую трассировку переменных.* Значения нужных переменных запоминаются в циклическом буфере и представляются на экране ПК в виде графиков. Запись значений можно выполнять в конце каждого рабочего цикла либо через заданные периоды времени. Трассировка запускается вручную или синхронизируется с заданным изменением значения определенной (триггерной) переменной;

• *визуализацию — анимационные картинки,* составленные из графических примитивов, связанных с переменными программы. Значение переменной может определять координаты, размер или цвет графического объекта. Графические объекты включают векторные геометрические фигуры или произвольные растровые изображения. Визуализация может содержать элементы обратной связи, например кнопки, ползунки и т. д. С помощью визуализации создается изображение, моделирующее объект управления или систему операторского управления.

Большинство микроконтроллеров имеют в своем составе модуль последовательного ввода/вывода, который часто называют контроллером последовательного ввода/вывода или контроллером последовательного интерфейса.

Задачи, которые решаются средствами этого модуля, можно разделить на три группы:

1) связь МКС с системой управления верхнего уровня, например, промышленным или офисным компьютером. Наиболее часто для этих целей используются интерфейсы RS-232 и RS-485, в последнее время все шире используется интерфейс USB;

2) связь с внешними по отношению к МК периферийными микросхемами МКС, а также датчиками физических величин с последовательным выходом. Для этих целей используются интерфейсы I2C, SPI, а также нестандартные протоколы обмена;

3) интерфейс связи с локальной сетью в мультимикроконтроллерных системах. В системах с небольшим

Библиотека ГГТУ им. П.О. Сухого

числом МК обычно используют сети на основе интерфейсов I2C, RS-422, RS-485.

С точки зрения организации обмена информацией все эти типы интерфейсов последовательной связи отличаются: режимами передачи данных (синхронный или асинхронный), форматами кадра (число бит в посылке), временными диаграммами сигналов на линиях.

При синхронном последовательном вводе/выводе передача отдельных бит данных синхронизируется с помощью одновременно передаваемого тактового сигнала.

При асинхронной передаче данных передается не тактовый сигнал, а старт-бит и стоп-бит, определяющие начало и завершение передачи данных.

Супервизором питания (Power Supervisory Circuits) (См. Рис 3.3) или *Монитором (Monitor)* называется электронная схема (далее — *SV),* предназначенная для выполнения функции начального сброса (или инициализации) микроконтроллера (микропроцессора) при включении питания или снижении напряжения питания ниже определенного значения в процессе работы. При этом длительность сигнала сброса должна быть больше определенной оговоренной величины. Полярность сигнала сброса должна соответствовать полярности входа сброса микроконтроллера (микропроцессора). Супервизор должен иметь дополнительный вход для сброса системы от дополнительной кнопки сброса. В некоторых случаях супервизоры имеют дополнительные входы, анализирующие другие напряжения питания. Кроме того, супервизоры могут формировать сигналы прерывания при различных ситуациях.

Охранным (сторожевым) таймером (WDT — WatchDog Timer) называется специализированная схема зашиты микроконтроллерной (микропроцессорной) системы от программных зависаний. Идея зашиты достаточно проста. Микроконтроллер периодически, не реже определенного фиксированного временно-*г* интервала подает импульс перезапуска таймера (далее импульс *WD)* на вход хемы зашиты. Импульс подается через дифференцирующую цепь для исключения потенциального влияния на схему. Схема представляет собой ждущий одно-вибратор с фиксированной задержкой выработки выходного импульса сброса (далее *RST)* и фиксированной длительностью этого импульса. Если период поступления импульсов *WD* не превышает величины фиксированной задержки,выходные импульсы RST не

Библиотека ГГТУ им. П.О.Сухого

вырабатываются, т. к. происходит постоянный перезапуск сторожевого таймера.

Микросхемы супервизоров питания выпускаются многими известными фирмами мира: *Analog Devices* , *Dallas Semiconductors*, *Electronic Technology*, *IMP*, *Linear Technology*, *MAXIM*, *Microchip*, *Sipex*, *Telcom*, *Texas Instruments*. Кроме основной функции — собственно супервизора питания, многие микросхемы оснащаются охранным таймером *WDT,* а также рядом дополнительных функций (См. Рис 3.4). В зависимости от количества встроенных функциональных узлов, микросхемы выпускаются в самых разнообразных корпусах.

Перечислим основные и возможные функции (свойства) микросхем супер визоров питания:

1) выработка сигнала*,RST при* включении питания;

2) выработка сигнала *RST* при снижении питания ниже определенного значения (допуска) в процессе работы для исключения возможности неправильное функционирования;

3) наличие выходов сигнала RST одной или обеих полярностей;

4) наличие входа для внешней кнопки сброса;

5) наличие компаратора раннего предупреждения (о снижении питания);

6) наличие встроенного сторожевого таймера *WDT;*

*7)*возможность программирования (подбором резисторного делителя) порогового напряжения (допуска) выработки сигнала *RST;*

8)возможность программирования (подбором конденсатора) длительность сигнала *RST;*

9)возможность программирования (подбором конденсатора) периода срабатывания *WDT;*

10)наличие схемы подключения резервного питания (аккумулятора);

11)наличие схемы контроля заряженности источника резервного питания;

12) наличие схемы подзарядки источника резервного питания;

13) наличие одного или нескольких входов и выходов выборки *(CS)* памяти блокирующих работу памяти при снижении напряжения питания ниже определенной величины (допуска);

14)наличие встроенной системы звуковой индикации снижения питания;

Библиотека ГГТУ им. П.О.Сухого

15)наличие встроенной светодиодной системы индикации снижения питания;

16)наличие ахода или входов для анализа дополнительных напряжений питания (например +12 В, -12 В, и т. д.);

17)наличие выхода или выходов сигналов прерывания при снижении питания;

18)наличие отдельного выхода от WDT или объединение этого выхода с выходами сигнала RST.

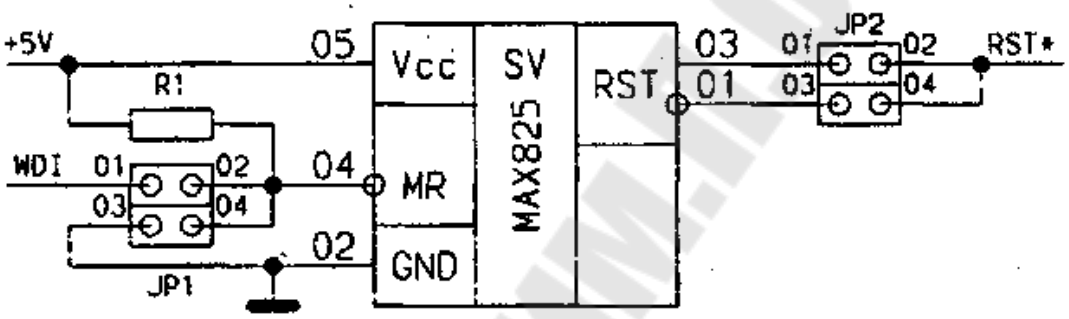


Рисунок 3.3 – Вариант реализации супервизора питания

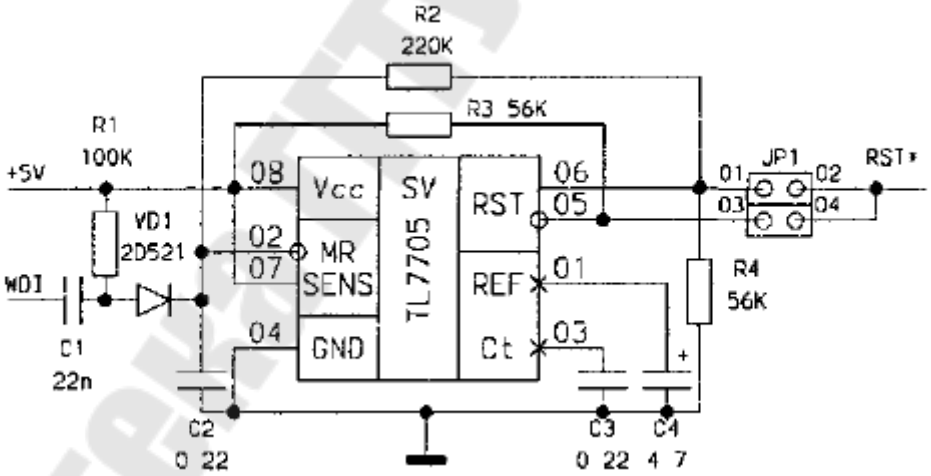


Рисунок 3.4 – Бюджетный вариант реализации супервизора питания с функцией WDT

Таймером реального времени (RTC— Real Time Clock) называется схема, предназначенная дня независимого (от процессора или контроллера, а также от наличия питания) подсчета текущего времени и ведения функций календаря. Таймеры реального времени (далее просто таймеры или *RTC)* выпускаются рядом фирм: *Dallas*

Библиотека ГГТУ им. П.О. Сухого

Semiconductors, *Thomson*, *Philips*, *Epson*, *Motorola*, *Unitrode (BenchMarq)* и некоторые другие.

Существует по крайней мере по два стандартных формата записи данных времени и календаря в *RTC*.

Первый, основной, формат записи данных времени обозначается как *STD (Standard Timer Data)* и содержит шесть байт (в некоторых 4-битных *RTC* — шесть младших полубайт), в которых записываются часы *(B),* минуты *(M)* и секунды *(S)* в виде *HH:MM:SS*.

Второй, расширенный, формат записи времени обозначается как *STD* + /r/t, где под байтами *hh* подразумевается значение десятков и единиц сотых долей секунды. Такой формат содержит 8 байт (полубайт) — *HH:MM:SS:hh*.

Первый, стандартный, формат записи данных календаря также обозначается как *STD* и содержит значения двух последних цифр года *(YY),* номер месяца *{MM),* номер дня *(DD)*. Этот формат содержит шесть байт — *YY-MM-DD*.

Второй, расширенный, формат записи данных календаря обозначается *Y2K* и содержит в отличие от предыдущего случая значения всех четырех цифр года. Формат содержит 8 байт данных календаря — *YYYY-MM-DD*.

Перечислим основные и возможные функции (свойства) микросхем *RTC*.

1. Основная функция — генерация кодов времени и календаря, для чего микросхема имеет встроенный высокостабильный кварцевый генератор, обеспечивающий, как правило, точность не хуже ± 1 минута за месяц, набор счетчиков и программно доступные регистры, в которых помешаются данные текущего времени и календаря.

2. Многие микросхемы *RTC* имеют функцию генерации прерывания в запрограммированное время *(Time of Day Alarm)*. Это прерывание генерируется раз в сутки. Обозначается эта функция буквой «А» (здесь и далее мы будем использовать обозначения, применяемые в документации фирмы *Dallas Sem iconductors)*.

3. Некоторые *RTC* имеют встроенный супервизор питания, который генерирует сигнал сброса *(Microprocessor Reset)* для микропроцессора (микроконтроллера) и удерживает его в состоянии сброса до тех пор, пока напряжение питания не достигло нормы. Функциональная возможность обозначается как *UR*. Это сделано для исключения возможности нарушения данных *RTC* во время пе-

Библиотека ГГТУ им. П.О. Сухого

реходных процессов питания и для блокировки возможности обращения процессора во время, когда питание не в норме.

4. В некоторых *RTC* имеется вход для внешнего сигнала *(Kickstart— KS),* который может вызвать генерацию выходного сигнала *RTC* для включения питания микропроцессорной системы.

5. Во многих микросхемах имеется выход *(Periodic — P),* который можно запрограммировать на генерацию импульсов в диапазоне периодов от 122 мкс до 500 мс.

6. Некоторые микросхемы имеют встроенный сторожевой таймер *(WDT).*

7. Многие микросхемы *RTC* имеют встроенную энергонезависимую оперативную память *(Random Access Memory — RAM),* причем размер этой памяти может изменяться от нескольких байт до 512 Кбайт. Это очень полезное свойство микросхем *RTC* позволяет при достаточном объеме такой памяти использовать ее в качестве *Flash-памяти* для хранения различного рода настроек.

8.Еще одно полезное свойство некоторых микросхем *RTC* заключается в возможности очистки всей оперативной памяти *(RAM Clear — RC),* для чего в этих микросхемах имеется специальный вход.

9.Встречаются *RTC* с выходом, индицирующем занятость микросхемы внутренними операциями *(Update in Progress — U).*

10.Встречаются таймеры с дополнительным прерыванием в запрограммированное время *(Wake-up — WU).*

11.Для некоторых применений может быть полезна еще одна особенность — возможность записи времени между событиями *(Event Recorder — ER).*

12.Очень полезное свойство *(Power Cycle Counter — PC)* — счетчик, который считает количество циклов, прошедших с момента включения питания, т. е. с момента начала работы системы.

13.Встречается также свойство *(Vcc Active Counter — VC)* — счетчик, который считает только, когда питание активно.

14.Выпускаются микросхемы *RTC* с напряжением питания 5 В или 3,3 В.

15.Микросхемы могут быть выполнены в различных корпусах *(DIP, SO),* модулях *(M)* и в виде слотов *(S),*

16.Выпускаются таймеры *RTC* с внешним резервным питанием (батарейкой), с встроенной литиевой батарейкой (как правило, обеспечивающей до 10 лет непрерывной работы) и со встроенными гнездами для сменных батарей.

Библиотека ГГТУ им. П.О.Сухого

17.Доступ к данным микросхемы может быть параллельными *(Parallel),* смешанным *(Muxed),* побайтным *(Bytemde),* а также с использованием последовательных интерфейсов *2-Wire, 3-Wire* и SPI.

18.Многие микросхемы *RTC* с параллельным интерфейсом поддерживают несколько спецификаций шин, например *Intel* и *Motorola.*

19.Выпускаются таймеры с байтовым и полубайтным доступом.

20.Естественно, что выпускаются микросхемы с нормальным и расширенным температурным диапазоном работы.

21.Некоторые микросхемы *RTC,* особенно содержащие большой объем энергонезависимой памяти, совместимы по корпусу и функциональному назначению выводов с некоторыми микросхемами *ROM (Read Only Memory).*

Кроме перечисленных основных и часто встречающихся дополнительных функций некоторые микросхемы имеют дополнительные специфические функции, например такие, как температурный контроль *(Temperature Control — TC),* температурный будильник *(Temperature Alarm — TA),* серийный номер *(Serial Number — SN),* встроенный аналого-цифровой преобразователь *(ADC)* и т. п.

В качестве примера использования средств ПЛК приведем часть общей программы управления штамповочной машиной (рис. 3.5). Для успешного решения поставленной задачи рекомендуется осознать принцип действия и последовательность операций на уровне эскизирования. При выходе ранее отштампованного изделия из штампа сигнал X0 (в синтаксисе он записывается как X000) имеет уровень ON. Это означает, что форма освободилась и готова для подготовки её к следующему циклу штамповки.

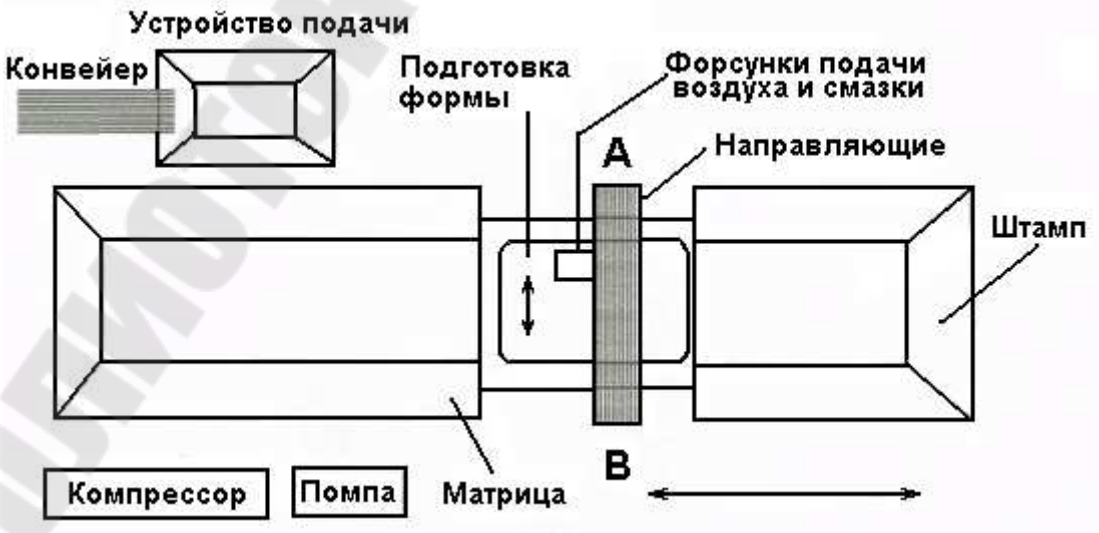


Рисунок 3.5 – Структурная схема штамповочной машины

Библиотека ГГТУ им. П.О.Сухого

Он начинается с выдвижения штампа из матрицы вправо, под действием команды Y2 при условии, что форсунки подачи воздуха (Y4) и масла (Y6), показанные на блок-схеме алгоритма (рис.3.6), не будут включены – это своего рода блокировка от ложного срабатывания.

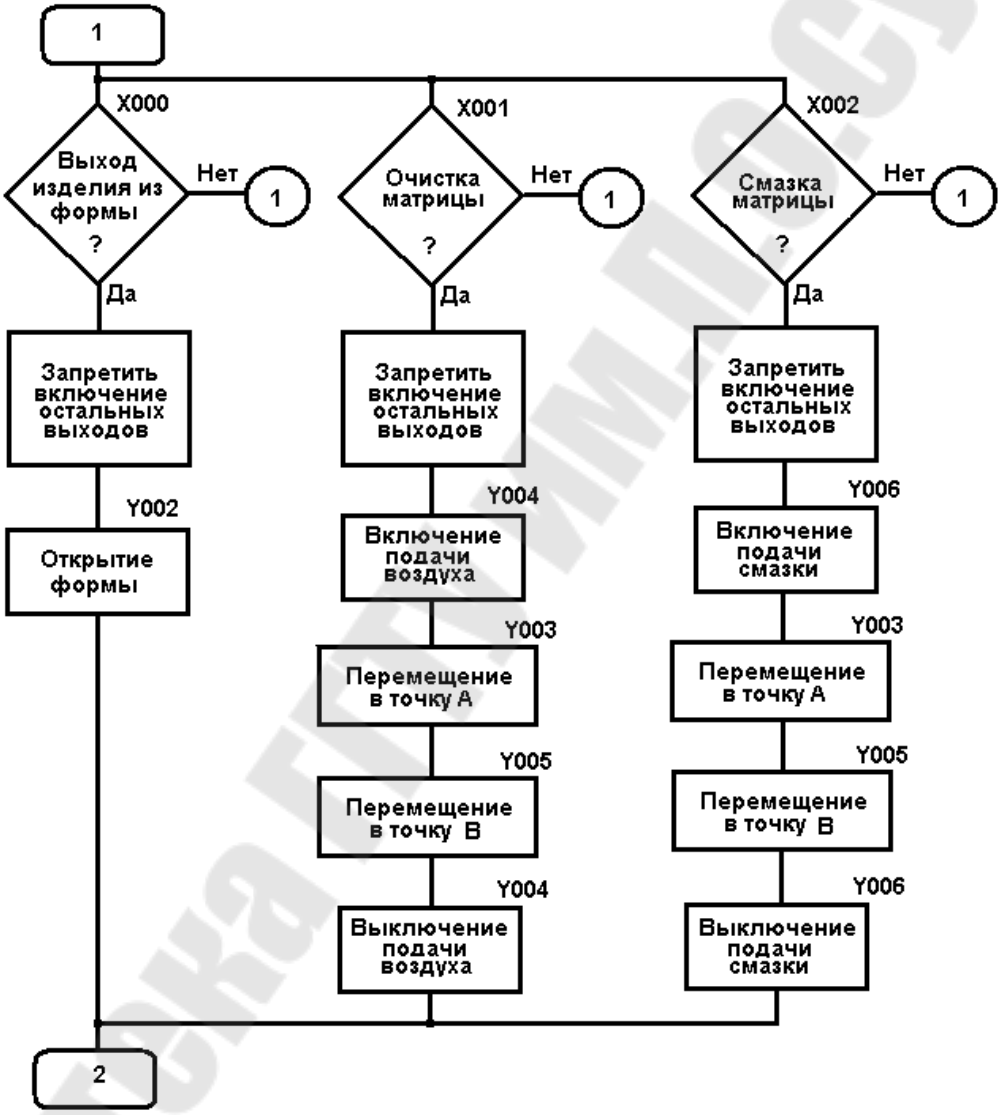


Рисунок 3.6 – Блок схема алгоритма работы штамповочной машины

Эта блокировка обеспечивается записью строки 1 программы (рис.3.8), и она находится в полном соответствии с отображением работы объекта на временных диаграммах процесса (рис. 3.7). Действительно, выходной сигнал на открытие формы Y2 будет выдан при наличии X0 и пассивном состоянии сигналов Y4 и Y6, что равносильно соответствию их отрицаний уровням ON. Этим

Библиотека ГГТУ им. П.О.Сухого

исчерпывается программное представление работы управляющей схемы по левой ветви алгоритма.

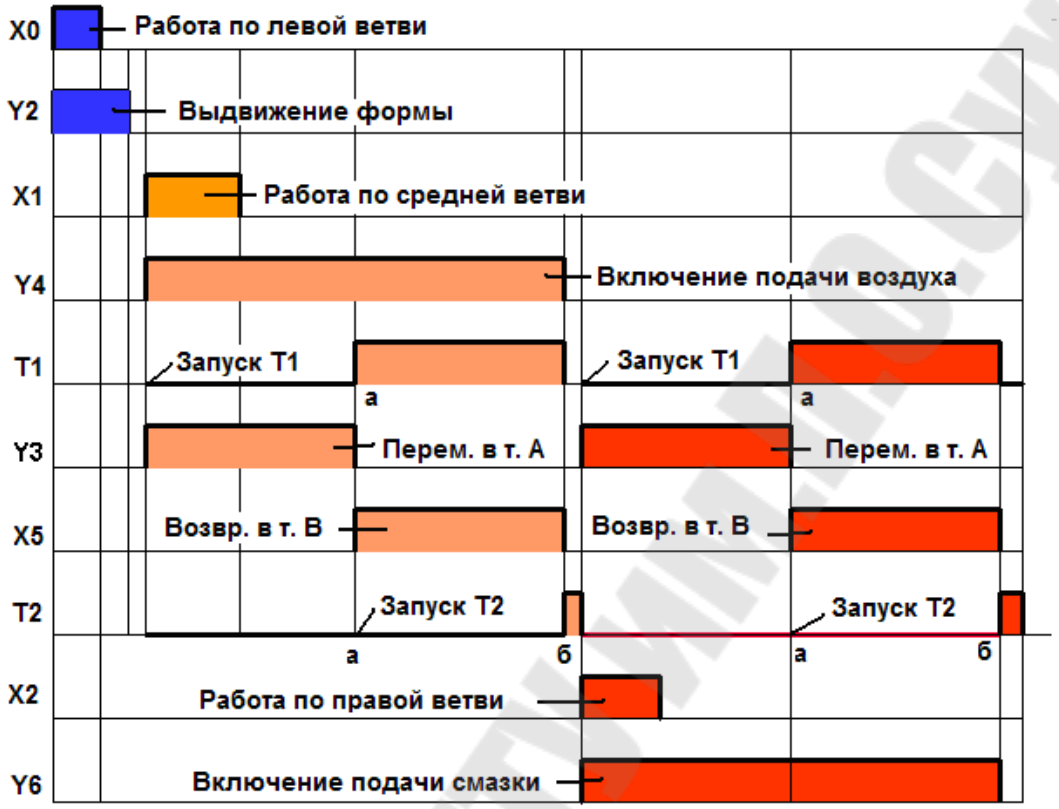


Рисунок 3.7 – Временные диаграммы работы узлов штамповочной машины

После выдвижения формы сигналом X1 активизируется отработка средней ветви алгоритма. Как видно из строки 2 программы, выход Y4 примет уровень ON, если не будут включены Y2 (выдвижение формы) и Y6 (включение подачи смазки). Цепь сразу встаёт на самоблокировку по Y4 последовательно с нормально замкнутым состоянием выдержки времени таймера T2. Назначение T2 как раз и состоит в том, чтобы после его срабатывания остановить выполнение программы по отработке средней, а как можно заметить по строке 3 алгоритма, и правой ветви тоже. Строки 2 и 3 функционально организованы одинаково: самоблокировка по своему выходному состоянию и запрет на включение «своей» ветви, если включена одна из соседних.

За время включённого состояния Y4 или Y6 каретка, несущая форсунки подачи воздуха и смазки, должна совершить два встречных движения: перемещение из исходной точки B в точку A за заданное время выдержки таймера T1 и вернуться за такое же время обратно в

Библиотека ГГТУ им. П.О. Сухого

точку В. Первое из этих движений под действием Y3 происходит на интервале, когда T1 отсчитывает свою выдержку времени, а второе – когда «свою» выдержку отсчитывает T2. В момент окончания выдержки времени T2 отработка рассмотренного фрагмента управления объектом завершается. Срабатывание T2 в строке 5 вызывает переход в OFF сигналов Y4 и Y6 (строки 2 и 3), что, в свою очередь, заставит выключиться сигналы Y3 и Y3 (строки 4 и 5) программы на рис. 3.8.

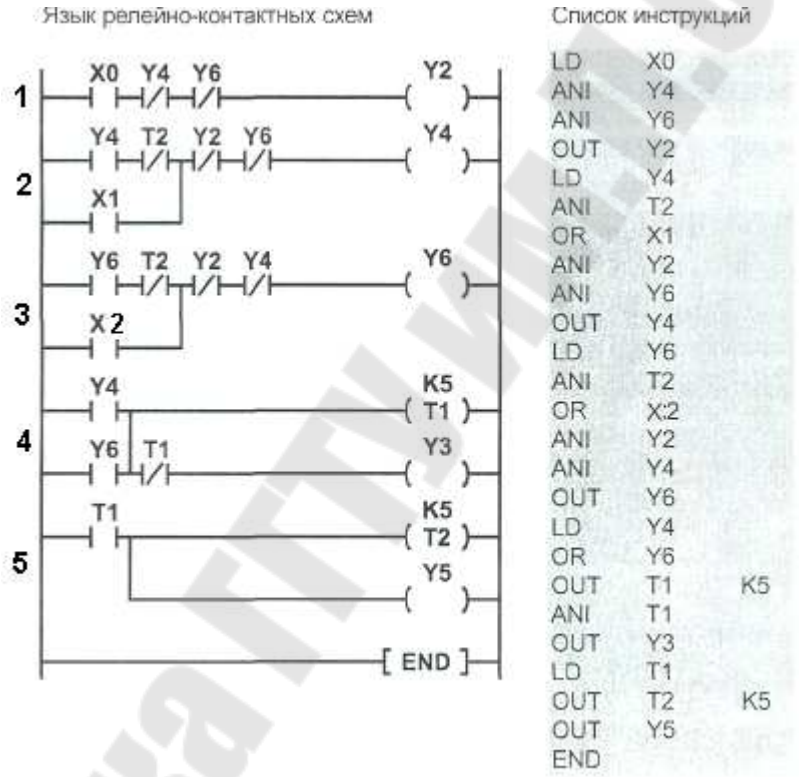


Рисунок 3.8 – Программная реализация алгоритма работы штамповочной машины

4. Контроллеры Siemens. САПР Quartus II фирмы Altera

4.1. Функционирование контроллеров Siemens

Регистры CPU Регистры CPU используются для адресации или обработки данных. Данные могут с помощью соответствующих команд быть обменены между областями памяти CPU и регистрами. CPU содержит следующие программно доступные регистры:

Аккумуляторы**:** Два (в S7-300) или четыре (в S7-400)

Библиотека ГГТУ им. П.О. Сухого

аккумулятора используются для арифметики, сравнений с байтами, словами или двойными словами.

Адресные регистры: Два адресных регистра используются как указатели для косвенной адресации памяти.

Регистры блоков данных: Регистры блоков данных содержат номера открытых блоков данных. Таким образом возможно, что открыты одновременно два DB: один DB с помощью регистра DB, другой как экземпляр DB с помощью регистра DI. Когда DB открыт, его длина (в байтах) автоматически загружается в связанный с ним регистр.

Слово статуса: Содержит различные биты, которые отражают результат или статус отдельных инструкций во время выполнения программы.

Области памяти Память S7-CPU может быть разделены на четыре области:

Загрузочная память используется, чтобы хранить программу пользователя без символов и комментариев. Загрузочная память может быть выполнена в виде RAM или FLASH EPROM.

Рабочая память (встроенная RAM) используется, чтобы хранить соответствующую часть S7- программы, необходимую для выполнения программы. Программа выполняется исключительно в рабочей памяти.

Область ввода - вывода разрешает прямой доступ ко входам и выходам, связанных с ней сигнальных модулей. Системная память (RAM) содержит области отображения входного и выходного процессов, меркеры, таймеры и счетчики. Кроме того, она содержит локальный стек, стек блоков и стек прерываний.

На рис. 4.1. показана типовая структура регистров и областей памяти контроллеров Siemens. Для создания программ управления необходимо знать назначение отдельных бит некоторых регистров (См. Рис.4.2).

Библиотека ГГТУ им. П.О. Сухого

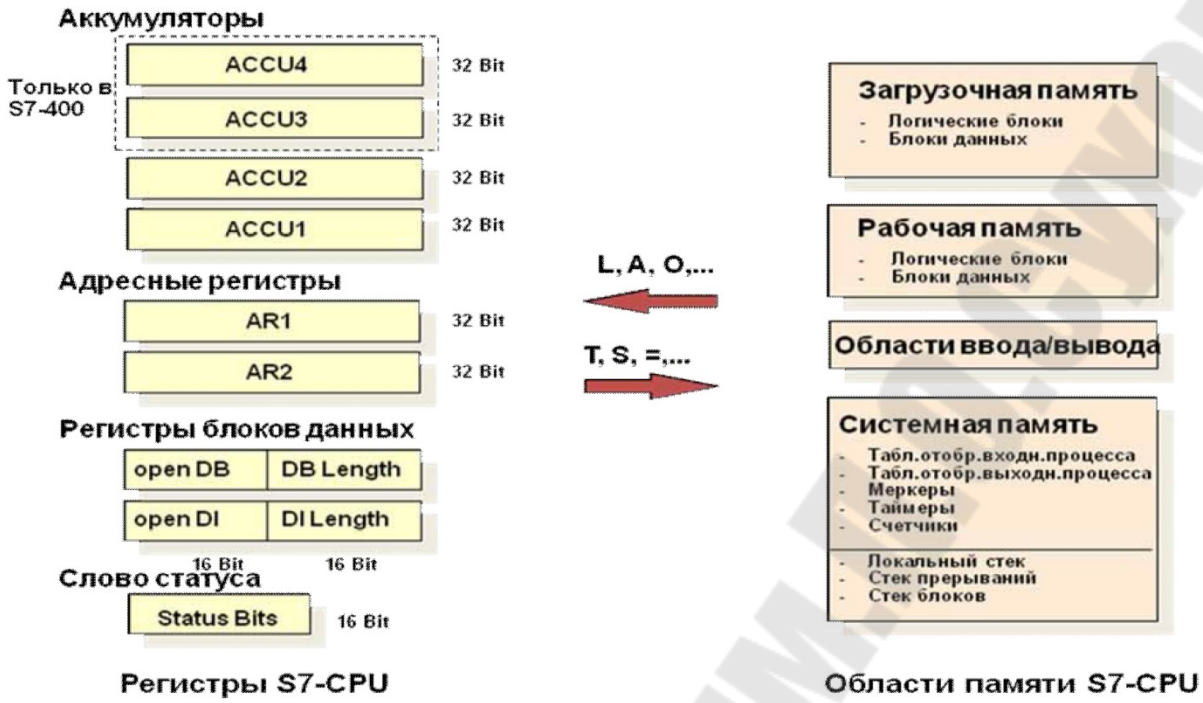


Рисунок 4.1 – Типовая структура элементов ПЛК Siemens

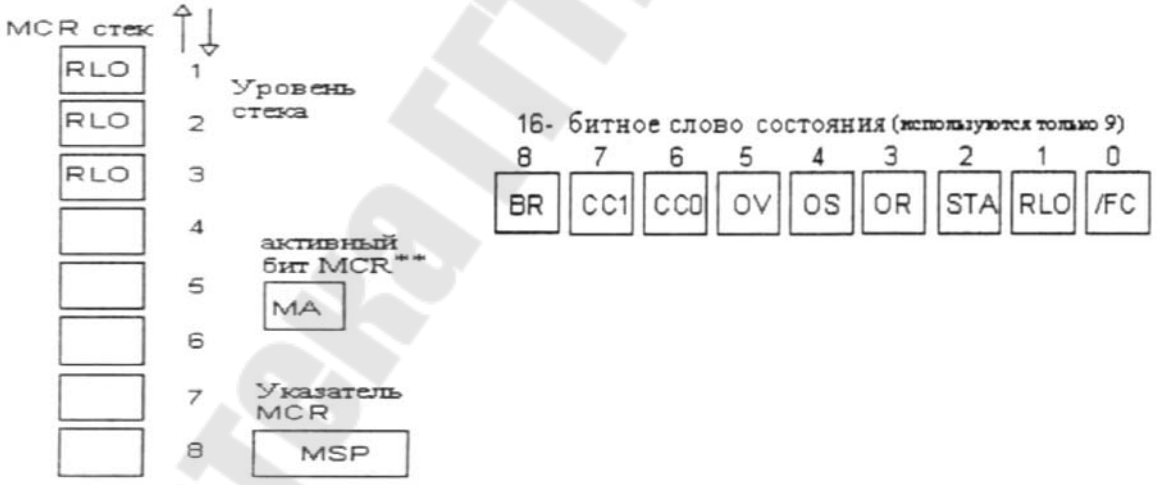


Рисунок 4.2 - Схематичное представление некоторых регистров процессора

В полную программу процесса входят:

- Операционная система**:** содержит общую часть всех инструкций

Библиотека ГГТУ им. П.О.Сухого

и соглашений для реализации внутренних функций (например, сохранение данных при сбросе напряжения питания, управление реакцией пользователя при прерывании и т. д.). Она расположена на так называемом EPROMe (Erasable Programmable Read Only Memory) и является фиксированной составной частью процессора. Как пользователь, вы не имеете возможности обращаться к операционной системе.

- Программа пользователя: содержит набор всех написанных пользователем инструкций и соглашений для обработки сигналов, с помощью которых производится управление установкой (процессом). Программа пользователя распределяется на блоки. Деление программы на блоки значительно проясняет структуру программы и подчеркивает программно-технические связи отдельных частей установки.

Блоком называется часть программы пользователя, ограниченная функционально и структурно или по целям использования.

Различают блоки, которые содержат

- инструкции для обработки сигналов;
- блоки, содержащие данные (блоки данных).

Блоки идентифицируются:

- типом блока (OB, PB, SB, FB, FX, DB,DX);
- номером блока (число от 0 до 255).

Инструкция языка STEP является наименьшей самостоятельной единицей программы пользователя. Она является предписанием для работы процессора. Инструкция может быть представлена в виде

- *Список команд (инструкций) - STL.* Представляет собой список команд подобно обычному языку Ассемблера.
- *Контактный план - LAD.* Управляющая программа записывается при помощи изображений элементов релейных контактных схем.
- *Функциональный план — FUP.* Для отображения программы используются схемы логических элементов.

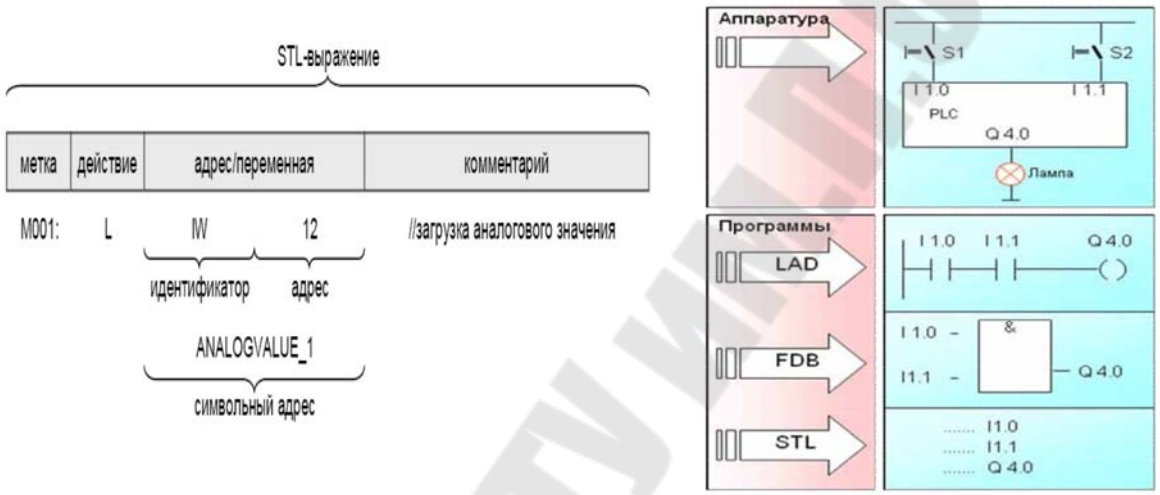
На рис. 4.3 представлены варианты программных реализаций для контроллера.

В контроллерах SIMATIC S7 существует несколько способов обработки управляющей программы:

1. Циклическая обработка**.** Состоит из повторных (периодически повторяющихся) обработок управляющей программы, которая начинается с вызова организационного блока OB1. В начале

Библиотека ГГТУ им. П.О.Сухого

цикла обработки программы ОС заполняет область отображения входов, сбрасывает таймер контроля длительности цикла, после этого вызывает для обработки блок ОВ1. В конце цикла обработки ОС переписывает в выходные модули значения из области отображения выходов, после чего начинается следующий цикл обработки. В блоке ОВ1 можно вызывать функции и функциональные блоки. После обработки вызванного блока управление передается блоку, из которого был произведен вызов данного блока.



а) в)

Рисунок 4.3 - Инструкции языка STEP 7
а)структура STL-выражения;
в) виды представления инструкций языка STEP 7

2. Циклическая обработка. Состоит из повторных (периодически повторяющихся) обработок управляющей программы, которая начинается с вызова организационного блока ОВ1. В начале цикла обработки программы ОС заполняет область отображения входов, сбрасывает таймер контроля длительности цикла, после этого вызывает для обработки блок ОВ1. В конце цикла обработки ОС переписывает в выходные модули значения из области отображения выходов, после чего начинается следующий цикл обработки. В блоке ОВ1 можно вызывать функции и функциональные блоки. После обработки вызванного блока управление передается блоку, из которого был произведен вызов данного блока.

3. Циклические прерывания. При управлении ТП всегда

Библиотека ГГТУ им. П.О.Сухого

существуют программы, которые должны обрабатываться через одинаковые, заранее заданные, промежутки времени. Для этих целей в контроллерах SIMATIC S7 существуют блоки обработки циклических прерываний. Промежуток времени, через который должен вызываться данный блок, задается программистом.

4. Прерывания по дате и времени. Существуют программы, которые должны выполниться один раз в определенный день и час или выполняться периодически. начиная с определенных даты и времени. Для этих целей в контроллерах SIMATIC S7 можно запрограммировать блоки прерываний по дате и времени.

5. Прерывания по задержке времени. Такие блоки вызываются по истечении определенного времени после возникновения какого-либо события.

6. Обработка включения питания. Часто при включении питания необходимо выполнить какие-либо однократные действия: первичную установку, инициализацию и т.д. Для этих целей предусмотрены блоки обработки включения питания.

7. Обработка ошибок. Такие блоки выполняются в случае возникновения аппаратных или программных ошибок.

Основными утилитами пакета STEP 7, которые доступны из папки SIMATIC - STEP 7, являются:

- SIMATIC Manager;
- LAD, STL, FDB - Programming S7;
- Memory Card Parameter Assignment;
- NetPro - Configuring Networks;
- PID Control Parameter Assignment;
- S7 SCL - Programming S7 Blocks;
- S7-GRAPH - Programming Sequential Control System;
- S7-PDIAG - Configuring Process Diagnostic;
- S7-PLCSIM Simulating Modules;
- Setting the PG-PC Interface;
- Configure SIMATIC Workspace.

Основной программой STEP 7 является **SIMATIC Manager**, который позволяет производить основные операции с проектом, такие как создание, сохранение, открытие, а также управлять работой проекта, запускать различные утилиты, связывать их между собой и т.д.

Программа LAD, STL, FDB - Programming S7 Blocks - редактор, позволяющий программировать блоки, основываясь на одном из трех

Библиотека ГГТУ им. П.О. Сухого

представлений языка программирования. Язык LAD - Ladder Diagram (контактный план) - использует представление программы в виде коммутационной схемы, состоящей из переключателей, линий связи, ключей и т.п. STL - Statement List (список операторов) - язык, подобный ассемблеру. FDB - Function Block Diagram - функциональная схема, основанная на логических элементах, триггерах и т.п.

Утилита **Memory Card Parameter Assignment** позволяет сохранять пользовательскую программу в память EPROM (электрически программируемая постоянная память), используя программатор или, в случае персональной ЭВМ, на внешнее устройство.

Программа **NetPro** - Configuring Networks позволяет конфигурировать промышленные сети, такие как MPI, PROFIBUS или Industrial Ethernet.

Утилита **PID Control Parameter Assignment** позволяет автоматизировать процедуру расчета и настройки параметров ПИД-регуляторов, используемых в системах управления.

С базовым пакетом обычно поставляются специальные утилиты, позволяющие проводить создание программ различными способами, такими как: написание программ на языке программирования высокого уровня SCL, который похож на паскаль, с помощью программы S7 SCL; графическая разработка программ в виде последовательности шагов и переходов между ними посредством утилиты S7-GRAPH. Могут также поставляться дополнительные пакеты.

S7-PDIAG - Configuring Process Diagnostic - это программа, используемая для диагностики проектов.

Утилита **S7-PLCSIM** Simulating Modules предназначена для программной имитации работы контроллера, что позволяет разрабатывать проекты и проверять и отлаживать работу программ без подключения реального оборудования.

Программа **Setting the PG-PC Interface** применяется для установки параметров локальных станций, подключенных к многоточечному интерфейсу MPI.

Configure SIMATIC Workspace позволяет конфигурировать проекты, создаваемые с использованием нескольких терминалов.

SIMATIC Manager - это графический интерфейс для редактирования объектов S7 (проектов, файлов пользовательских

Библиотека ГГТУ им. П.О.Сухого

программ, блоков, оборудования станций и инструментов). Основное окно утилиты показано на рис. 4.4.

Основными элементами панели главного меню программы SIMATIC Manager являются разделы File, PLC, View, Options, Window и Help, содержание которых зависит от текущего окна.

На панели инструментов вынесены наиболее часто используемые кнопки. Вначале рассмотрим структуру проекта в SIMATIC Manager, которая показа на рис.4.5.

Данные хранятся в проекте в виде объектов. Объекты в проекте размещаются в древовидной структуре, которая показана в левой части рис.4.5. Она подобна структуре, используемой в Windows Explorer. Различаются только иконки объектов. Содержимое правой части окна SIMATIC Manager зависит от выбранного в левой части объекта.

На самом верхнем уровне, который называется S7_Pro1, расположен сам проект. Каждый проект представляет базу, в которой хранятся все относящиеся к нему данные. Элементами проекта являются сети и их элементы - станции и другие узлы. В данном примере проект S7_Pro1 содержит многоточечный интерфейс MP1(1), к которому подключена одна станция SIMATIC 300 Station.

На втором уровне, который показан на рисунке 4.6, находятся станции, которые являются исходными объектами для конфигурирования аппаратуры. Здесь хранится информация о конфигурации аппаратуры и параметрах модулей. На рис.4.6 уровень станций содержит один элемент – SIMATIC 300 Station, который в свою очередь содержит контроллер CPU316-2DP(1). Другое оборудование можно просматривать утилитой Hardware.

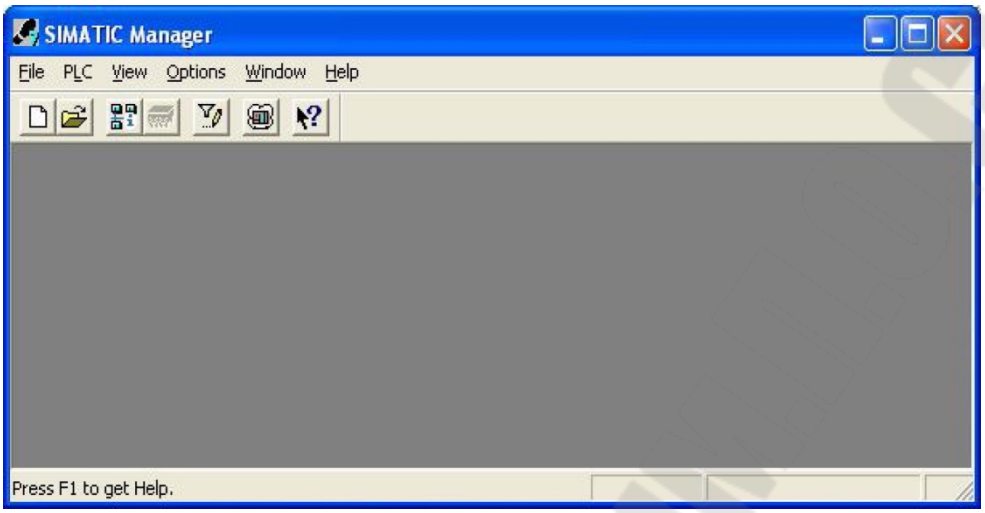
В свою очередь, процессор CPU316-2DP(1) содержит пользовательские программы, в данном случае S7 Program(1), которые могут быть написаны в виде блоков Blocks или исходных кодов Sources. Последующие уровни зависят от содержимого предыдущих.

На рис. 4.7 показан один из примеров уровня Blocks Основными блоками, которые используются в STEP 7, являются: а) организационный блок, например OB1, который является основной циклически исполняемой программой; б) функция, например FC1, применяемая для замены типовых или часто повторяющихся блоков; в) функциональный блок, например FB1, в отличие от функции имеет отдельную память в глобальном пространстве, называемую блоком

Библиотека ГГТУ им. П.О. Сухого

данных, за счет чего функциональный блок может сохранять свои переменные в общем адресном пространстве;

г) блоки данных, например DB1, наличие которых обусловлено гарвардской архитектурой контроллеров.



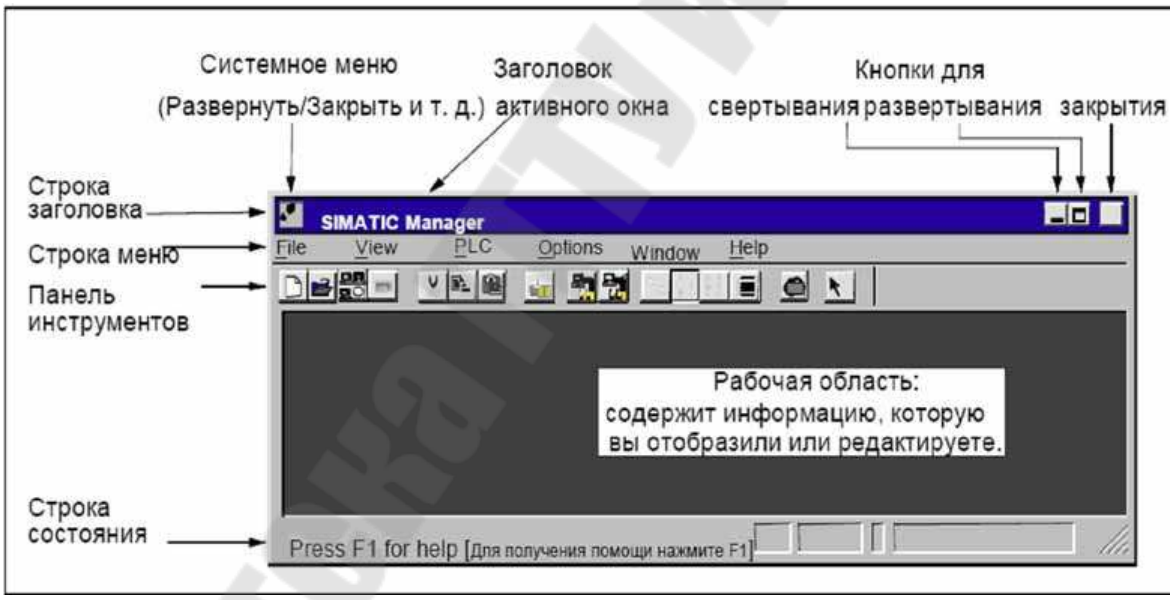


Рисунок 4.4 – Основное окно утилиты SIMATIC Manager

Рассмотрим основные этапы создания проекта с помощью мастера «New Project Wizard», который находится в разделе «File» главного меню утилиты SIMATIC Manager. Создание проекта состоит из четырех шагов, которые демонстрируются на рис.4.8.

Последовательно надо определить тип проекта, элементную базу, типы используемых блоков и дать название проекту.

Библиотека ГГТУ им. П.О. Сухого

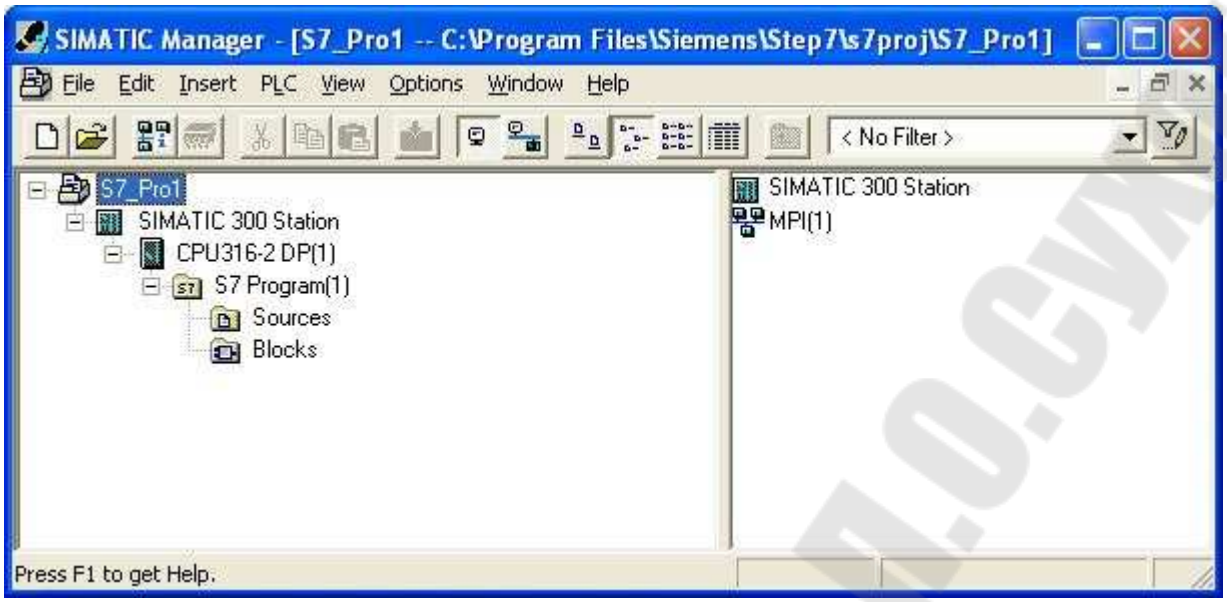


Рисунок 4.5 – Структура проекта в SIMATIC Manager

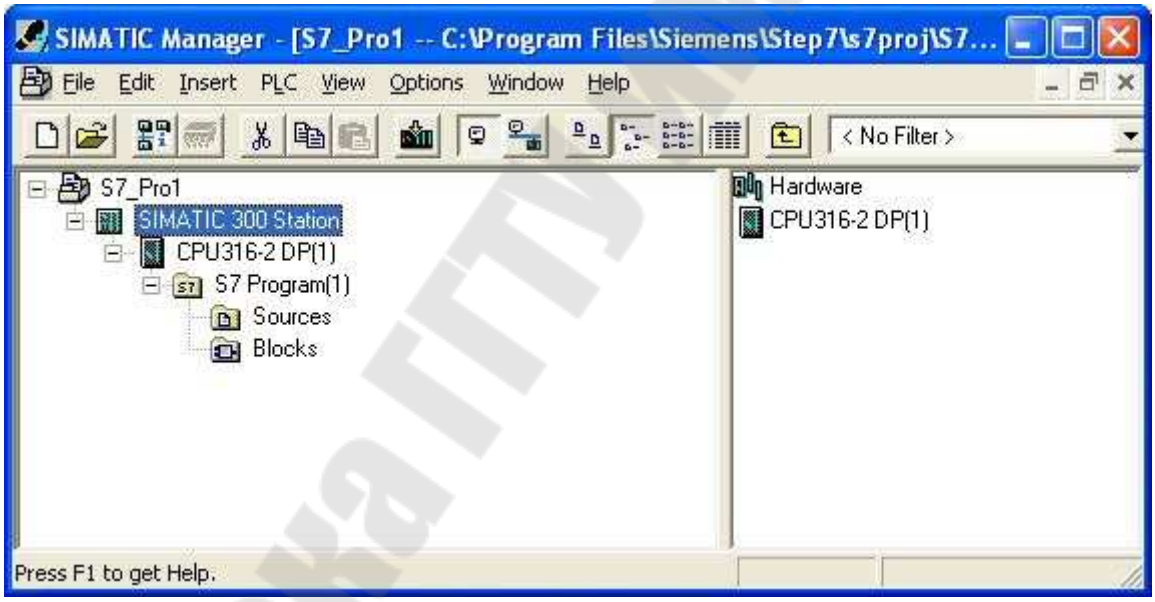


Рисунок 4.6 – Структура проекта во втором уровне SIMATIC Manager

Кроме того, имеется возможность установить язык программирования, наиболее удобный для пользователя – STL (список операторов), LAD (контактный план) или FBD (функциональный оператор).

Добавление новых элементов в проект осуществляется через меню «Insert».

Конфигурирование аппаратных средств проекта осуществляется посредством утилиты «Hardware Configuration». Чтобы запустить указанную программу, необходимо перейти на уровень станций, и

Библиотека ГГТУ им. П.О.Сухого

двойным щелчком нажать кнопку «Hardware», в результате чего появится окно, показанное на рис.4.9.

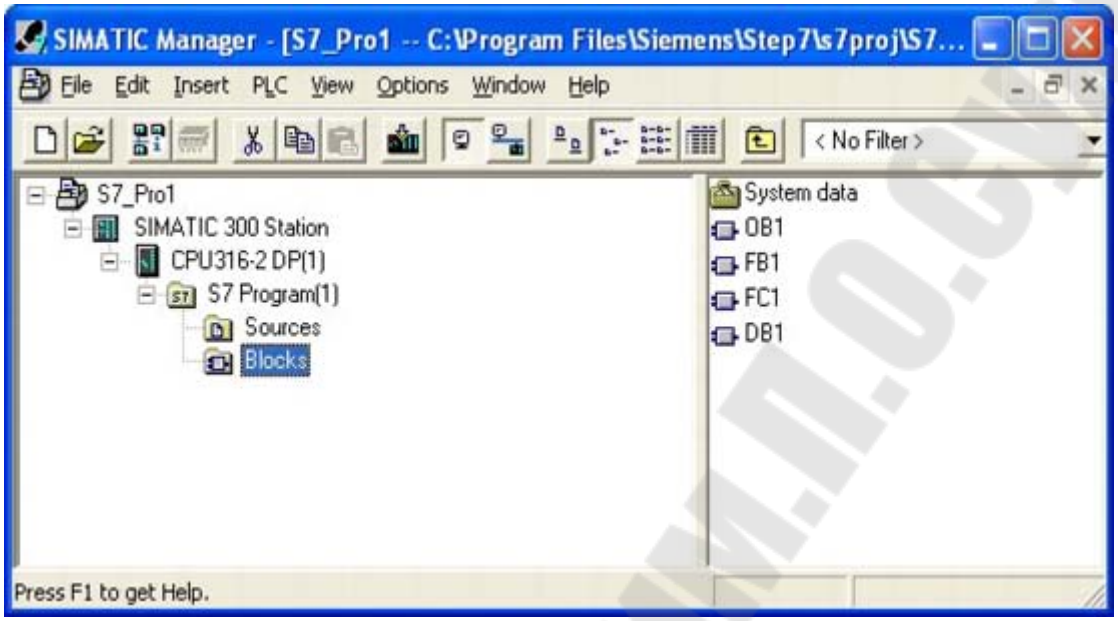


Рисунок 4.7 – Структура проекта на уровне блоков SIMATIC Manager

В верхней части показаны стойки с отдельными слотами. Они расположены на шинах. В левой нижней части находится таблица с адресами входов-выходов, различных блоков и контроллеров. В правой части окна расположена библиотека элементов, из которых можно собирать стойки.

Создание аппаратной части начинается с добавления стойки (Rack), которая находится в соответствующем каталоге. Например, при создании станции SIMATIC 300 необходимо открыть каталог элементов SIMATIC 300 и из папки Rack-300 добавить элемент Rail. Добавление можно производить либо двойным щелчком, либо перетаскиванием по технологии «drag & drop».

Если требуется установить блок питания, то необходимо вставлять его в слот 1 стойки. Соответствующий модуль станции SIMATIC 300 находится в группе PS-300.

Библиотека ГГТУ им. П.О. Сухого

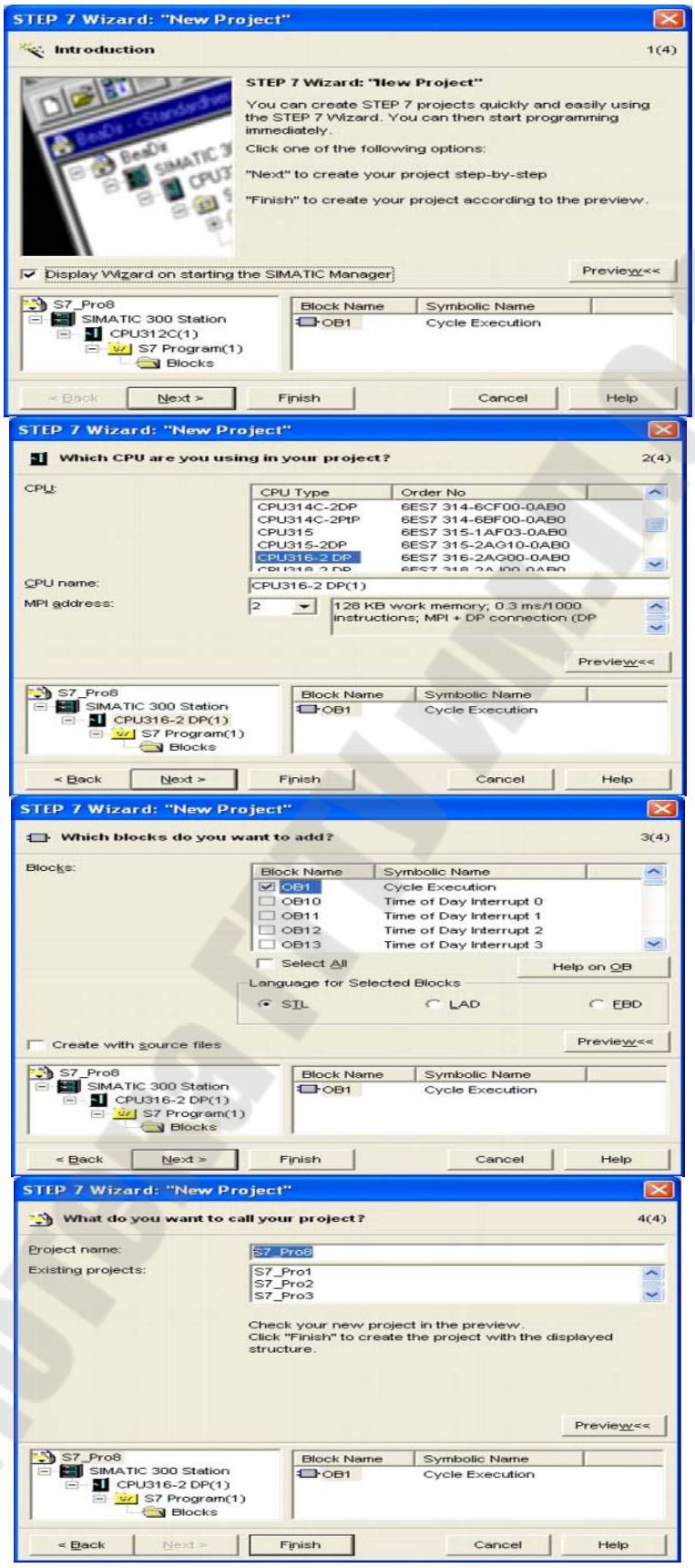


Рисунок 4.8 – Этапы создания проекта SIMATIC Manager

Библиотека ГГТУ им. П.О.Сухого

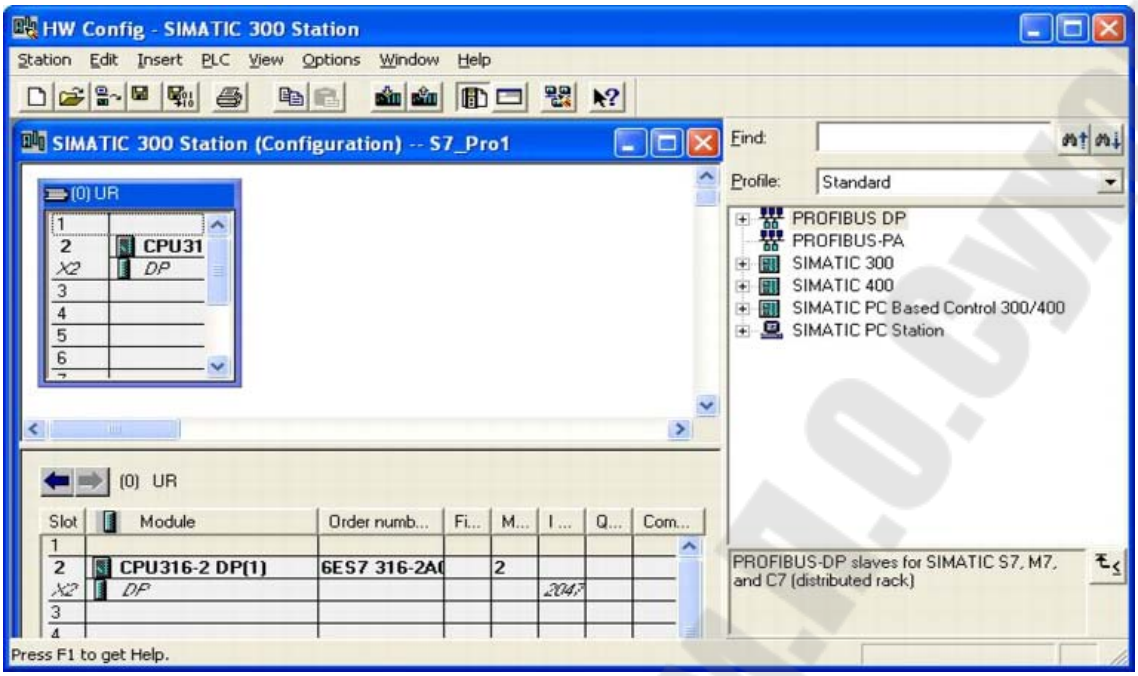


Рисунок 4.9 – Окно конфигурации аппаратной части

CPU контроллера можно найти в каталоге CPU-300, он вставляется в слот 2.

S7-300 слот 3 зарезервирован для интерфейсного модуля IM, необходимого для многоуровневых конфигураций, поэтому на рис.4.9. этот слот пустой.

Если эта позиция должна быть резервирована для последующей фактической установки интерфейсного модуля, то необходимо вставить в фактическую конфигурацию холостой модуль DM370 DUMMY из каталога SM-300\Special-300.

Начиная с четвертого слота, можно вставлять сигнальные модули. Можно добавить на выбор до 8 сигнальных блоков (SM), коммуникационных процессоров (CP) или функциональных модулей (FM). Необходимо отыскивать нужный модуль в папке и вставлять его, выбирая слот в стойке.

В стандартной конфигурации в стойку может входить процессор, блок питания и модули ввода и вывода, которые бывают аналоговые или дискретные.

Чтобы просмотреть адресное пространство, образованное модулями стойки, необходимо войти в меню «View»->«Address Overview», в результате чего появится окно, показанное на рис. 4.10. В данном окне отражаются те блоки, которые имеют входы или выходы, в данном случае модуль дискретного ввода DI32xDC24V и

Библиотека ГГТУ им. П.О.Сухого

блок дискретного вывода DI4xNAMUR.

В первом столбце Type указывается тип адресного пространства: I – для входов, Q – для выходов.

Во втором и третьем столбцах Addr. from и Addr. to указывается диапазон адресов в байтах, который занимает данное устройство. В данном случае модуль дискретного ввода DI32xDC24V имеет 32 входа, поэтому он занимает 4 байта с номерами от 0 до 3. Следующие байт с номером 4 занимает блок дискретного вывода DI4xNAMUR.

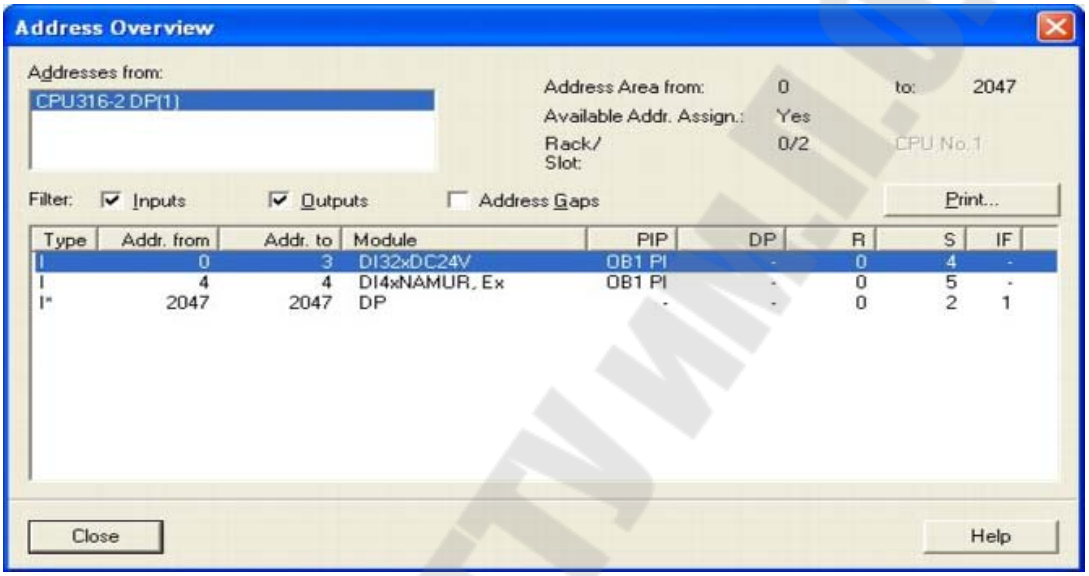


Рисунок 4.10 – Окно конфигурации входов и адресов

В следующих двух столбцах указываются названия блоков и организационный блок, который осуществляет опрос входов и назначение выходов. В данном случае оба модуля принадлежат одному блоку OB1.

Столбец R отображает номер стойки, а столбец S – слота для модуля. В данном случае модуль DI32xDC24V занимает слот 4, а модуль DI4xNAMUR – слот 5.

Столбец DP используется для системы распределенных выходов, а IF в тех случаях, когда используется специальный интерфейсный модуль при программировании системы на C++.

Чтобы получить доступ к свойствам блока достаточно открыть его пиктограмму с помощью двойного щелчка мыши. Основные параметры сосредоточены в контроллере, поэтому рассмотрим его свойства. Свойства контроллера отображаются в окне, которое содержит девять раскрывающихся вкладок. Вкладка «General»,

Библиотека ГГТУ им. П.О.Сухого

показанная на рисунке 26, содержит информацию о типе модуля, его название и, если он программируемый, MPI адрес. Чтобы назначить адрес многоточечного интерфейса, например в случае создания многоконтроллерной конфигурации, достаточно нажать кнопку «Properties, в которой имеется возможность задать параметр «Adress».

При программировании блоков в STEP 7 можно применять три языка программирования: LAD (контактный план), STL (список операторов), FBD (функциональный план).

Отличия языков показаны на рис. 4.11, где демонстрируется логическая функция И. Язык LAD удобен для инженеров-электриков, STL – для программистов, а FBD – для инженеров-схемотехников. Отметим, что переход от одного языка к другому в SIMATIC Manager может осуществляться автоматически.

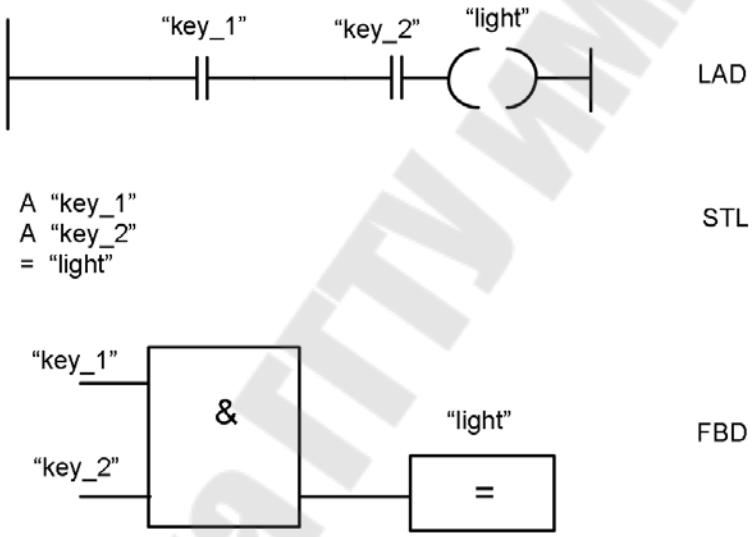


Рисунок 4.11 – Функция И в STEP 7

Можно писать программы или сегменты на LAD или FBD и затем автоматически преобразовывать сегменты программы в STL. Однако результат этого преобразования не всегда является наиболее эффективным решением для STL (программа, созданная непосредственно на STL, может быть короче).

Обратное преобразование из STL в LAD или FBD не всегда возможно. Сегменты программы, которые не могут быть преобразованы, остаются в STL. При преобразованиях никакие сегменты программы не теряются (См. Рис. 4.12).

Библиотека ГГТУ им. П.О. Сухого

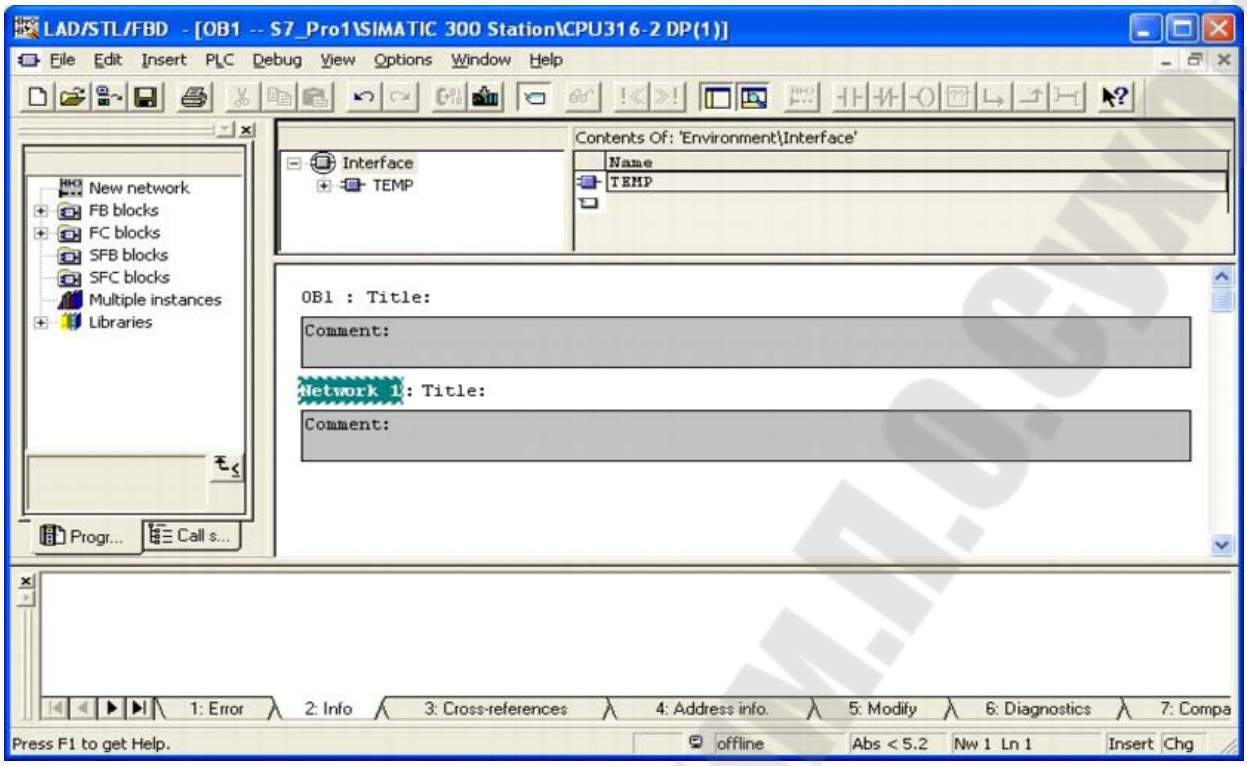


Рисунок 4.12 – Окно редактора LAD/STL/FBD

При редактировании блоков в FBD или LAD часто используемые элементы представлены кнопками в панели инструментов. Можно щелкать на них мышью, чтобы установить эти элементы на выбранную в программе позицию. Другие элементы можно вставить в программу из списка в любую позицию перетаскиванием или в выделенную позицию – дважды щелкнув на элементе из списка. Соединять элементы можно посредством мыши, захватывая выход и перетаскивая его к нужному выходу.

Для добавления нового сегмента достаточно нажать на кнопку «New Network» в панели инструментов, после текущего сегмента добавляется новый сегмент.

При программировании на STL нужно знать инструкции для записи программы. Можно получить информацию о синтаксисе и функциональном назначении через подсказку: «Help» -> «Help on STL». В справке доступна следующая информация: «Statement List Instructions» – описывает все инструкции, которые имеются в этом языке программирования; «Working with Statement List» (работа с списком команд) – описывает список команд и основы синтаксиса, ввод и наблюдение констант, типы блоков, контакты и состояния сигнала. При программировании на STL окно элементов содержит только список существующих блоков, которые могут быть вызваны

из текущего блока. Сегменты вставляются в программу так же, как в редакторе LAD/FBD.

После редактирования блока его необходимо сохранить, что возможно только в случае отсутствия синтаксических ошибок.

Часто удобно писать программы в виде отдельных функций. В этом случае основная программа, например блок OB1, будет содержать список вызовов. Например, на языке STL:

CALL FC 1

CALL FC 2

Чтобы загрузить блоки в контроллер, нужно выделить их в окне и воспользоваться меню «PLC» -> «Download». При этом нужно, чтобы был предварительно подключен контроллер или его программный эмулятор.

Проверку программной части без подключения реального оборудования можно проводить с помощью дополнительного пакета S7-PLCSIM. После того как проект готов, симулятор можно вызвать из главного окна SIMATIC Manager. Для этого в меню «Options» необходимо выбрать пункт «Simulate Modules», что приведет к запуску S7-PLCSIM, основное окно которого показано на рис.4.13.

С помощью значков, расположенных на панели инструментов симулятора S7-PLCSIM, можно добавлять для просмотра различные блоки и элементы контроллера:

а) IB – входная переменная;

б) QB – выходная переменная;

в) MB – биты памяти;

г) T – таймер;

д) C – счетчик;

е) Variable – переменная;

ж) Stacks – стек логических операций;

з) ACCUs – аккумуляторы и слово состояния;

и) Block Regs – блок регистров.

В блоках а) – е) можно вводить свои адреса. Для того, чтобы можно было использовать символьную адресацию, нужно войти в меню «Tools» -> «Options» -> «Attach Symbols», в результате чего появится окно. В этом окне нужно в разделе «Entry Point» указать вид блока, например проект или библиотека, имя проекта, в проекте выйти на уровень S7 Program и выбрать значок с именем Symbols.

Библиотека ГГТУ им. П.О.Сухого

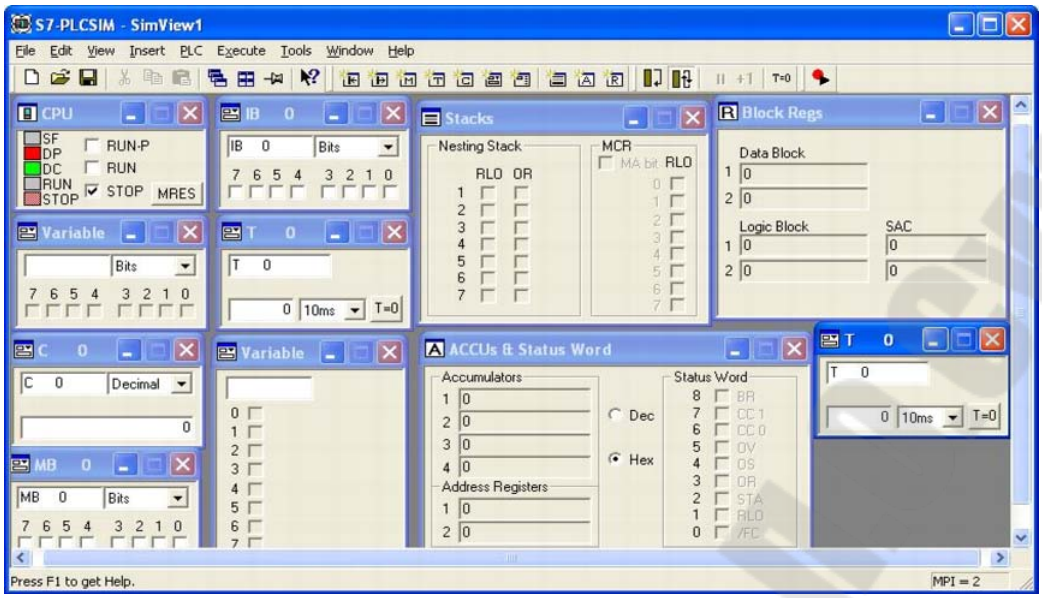


Рисунок 4.13 - Симулятор PLCSIM

Прежде чем проверять работу программы ее необходимо загрузить в контроллер. Это можно сделать либо из основного окна SIMATIC Manager, либо из редактора LAD/STL/FBD. В первом случае нужно выделить необходимые блоки, выбрать пункт меню «PLC» -> «Download». Во втором случае также используется меню «PLC» -> «Download», но загружается только текущий открытый блок. После этого нужно перейти в окно S7-PLCSIM и убедиться, что в его меню «PLC» установлен флаг «Power on». При загрузке блоком процессор симулятора должен находиться в режиме STOP.

Для того, чтобы запустить программу на выполнение, достаточно установить флажок RUN (циклическое выполнение) или RUN-P (однократное выполнение). При этом можно мышкой менять входы и смотреть как изменяются выходы, отлаживая программу, записанную в контроллер.

Существует много способов планирования проекта автоматизации. Основная последовательность действий проиллюстрирована на рис.4.14.

Библиотека ГГТУ им. П.О.Сухого

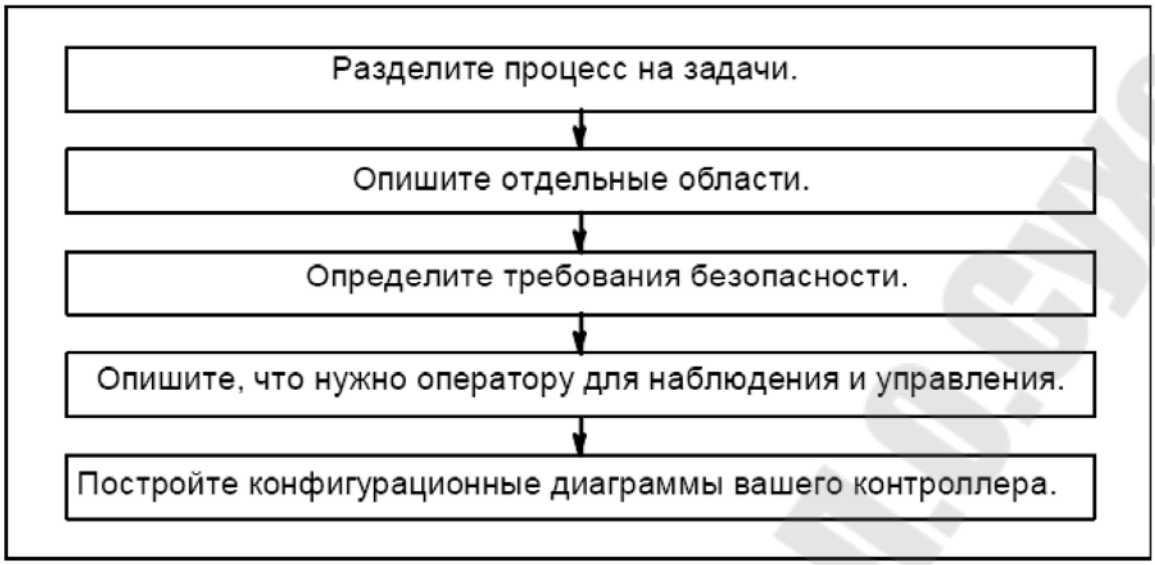


Рисунок 4.14 – Этапы подготовки проекта автоматизации технологического процесса

Деление процесса на задачи и области. Процесс автоматизации состоит из ряда отдельных задач. Путем выделения групп связанных задач внутри процесса и последующего разбиения этих групп на более мелкие задачи может быть определен даже самый сложный процесс.

Определение областей процесса. После определения процесса, подлежащего управлению, разделите процесс на связанные группы областей. Так как каждая группа разделена на более мелкие задачи, то задачи, необходимые для управления этой частью процесса, становятся менее сложными.

Описание отдельных функциональных областей. Описывая каждую область и задачу внутри Вашего процесса, Вы не только определяете функционирование каждой области, но и различные элементы, управляющие этой областью. Они включают в себя: электрические, механические и логические входы и выходы для каждой задачи, блокировки и зависимости между отдельными задачами

Список входов, выходов и входов/выходов. Сделав физическое описание каждого устройства, подлежащего управлению, нарисуйте диаграммы входов и выходов для каждого устройства или группы задач.

Определение требований безопасности. Определите, какие дополнительные элементы необходимы для обеспечения безопасности процесса - на основе юридических требований и

Библиотека ГГТУ им. П.О. Сухого

корпоративной политики в области охраны здоровья и безопасности. В свое описание Вам следует также включить все воздействия, которые элементы безопасности оказывают на области Вашего процесса.

Описание требуемых для оператора устройств отображения и управления. Каждый процесс требует интерфейса с оператором, который обеспечивает вмешательство человека в процесс. Часть спецификации проекта включает в себя проект пульта оператора.

Составление конфигурационной диаграммы принять решение относительно типа управляющего оборудования, требующегося для проекта. Принимая решение о том. какие модули Вы хотите использовать, Вы также определяете структуру программируемого контроллера. Составьте конфигурационную диаграмму, определяющую следующие аспекты:

- тип CPU;
- количество и тип модулей ввода/вывода;
- конфигурация физических входов и выходов.

После выполнения всех выше изложенных этапов приступают к составлению программы в среде программирования.

4.2.Функционирование контроллеров Altera

Пакет Quartus II представляет собой автоматизированную систему сквозного проектирования цифровых устройств на кристаллах ПЛИС фирмы Altera. Он предоставляет пользователю широкие возможности по вводу описаний проекта, логическому синтезу, компиляции проекта, программированию ПЛИС, функциональному и временному моделированию, временному анализу и анализу потребляемой мощности проекта, реализации внутрисистемной отладки.

В Quartus II используется удобный графический интерфейс и простая в применении справочная система, содержащая всю необходимую для выполнения проектирования информацию. Также пакет позволяет использовать командную строку для выполнения каждого этапа проектирования. Причем, в зависимости от предпочтений пользователя, графический интерфейс или командная строка могут использоваться как для выполнения отдельных этапов, так и для всего проекта в целом.

Библиотека ГГТУ им. П.О. Сухого

Пакет Quartus II интегрирует в себе большое количество программных модулей, предназначенных для выполнения различных этапов проектирования. Задание параметров и выполнение типовых команд выполняется в отдельных модулях одинаково, что значительно облегчает работу пользователя. Редакторы исходных файлов проекта (графический, текстовый, редактор символов, содержимого модулей памяти, временных диаграмм, конечных автоматов) используют одинаковые подходы и приёмы, а также похожие оконные формы, применяемые при создании и редактировании исходных файлов с описанием модулей проектируемого устройства.

В одном иерархическом проекте можно сочетать использование различных типов описания исходных файлов модулей проекта, подбирая наиболее подходящий тип, для каждого модуля.

В состав стандартной библиотеки Quartus II входит большое количество базовых элементов, включая мегафункции и макрофункции. Составной частью мегафункций являются операционные устройства, созданные по стандарту библиотеки параметризируемых модулей (LPM – library of parameterized modules).

Значительная часть мегафункций разработана фирмой Altera. Они описаны на языке низкого уровня и оптимизированы для применения в ПЛИС компании Altera. Остальная часть разработана компаниями партнерами. Применение мегафункций в проектах пользователя значительно расширит возможности проектирования и ускорит выполнение проекта.

Пакет Quartus II содержит средство ***SOPC*** (System on programmable chip) ***Builder***, предназначенное для проектирования реализуемых на кристалле процессорных систем [4]. С помощью этого средства можно легко создавать конфигурируемые процессорные ядра, реализовывать на кристалле различные контроллеры, а также значительную часть периферийного оборудования.

Под термином «проект» в Quartus II понимается набор файлов, связанных с проектируемым устройством, и набор соответствующих библиотек.

Файлы могут быть двух типов – логические и вспомогательные. Логические файлы описывают поведение или структуру отдельных модулей проектируемого устройства. К ним относятся файлы с текстовым описанием на языках описания аппаратуры (HDL,

Библиотека ГГТУ им. П.О.Сухого

Hardware Description Language), файлы с графическим представлением схем, файлы с представлением отдельных модулей в виде конечных автоматов. Вспомогательные файлы содержат дополнительную информацию о проектируемом устройстве. Большинство вспомогательных файлов не содержит описания логики проекта. Некоторые из них автоматически создаются приложением Quartus II, некоторые вводятся пользователем. Примерами вспомогательных файлов являются файлы установок и назначений (**.qsf**), символьные файлы (**.bsf**), файлы отчетов (**.rpt**).

Как правило, проект содержит несколько логических файлов, образующих иерархическое описание создаваемого устройства. Один из файлов представляет описание устройства на верхнем уровне иерархии. Он является модулем верхнего уровня.

Если описание устройства содержит единственный логический файл, то по умолчанию он является модулем верхнего уровня описания.

Каждый проект размещается в отдельной папке и имеет своё собственное имя, которое назначается пользователем при создании нового проекта. Автоматически при создании нового проекта создаются файлы с расширением **.qpf** (quartus project file) и **.qsf** (quartus settings file). Первый из этих файлов содержит номер версии Quartus II, используемой для создания проекта, дату создания и название активной версии проекта.

Процедура проектирования устройств на ПЛИС включает в себя следующие этапы :

1. Ввод проекта. На этом этапе разработчик вводит описание проекта и его частей. Проект или его части могут быть описаны традиционным способом в виде схемы, содержащей отдельные элементы, соединенные между собой цепями связи. Для создания и последующего редактирования таких описаний в пакете Quartus II используется графический редактор. Объектом его работы являются файлы с расширением **.bdf.**

Графическое представление проекта в пакете Quartus II может создаваться, как в базисе библиотечных элементов, так и в базисе графических символов проектировщика. В качестве компонентов проекта Quartus II позволяет использовать IP (Intellectual Properties) ядра, представляющие собой единицы интеллектуальной собственности. Эти блоки являются полностью синтезируемыми, перемещаемыми и могут по желанию проектировщика располагаться

Библиотека ГГТУ им. П.О. Сухого

в разных частях кристалла. Примерами IP ядер являются процессорные ядра NIOS II и большой набор различных контроллеров периферийных устройств.

Главным достоинством графического способа ввода проекта является его традиционность и наглядность, связанная с привычностью разработчиков к восприятию изображений схем.

В настоящее время все большую популярность приобретают языки описания аппаратуры (HDL). Они допускают описание проектируемого устройства с точки зрения, как его поведения, так и структуры. Такие возможности позволяют представлять проект в форме текстового описания алгоритмов функционирования его модулей в сочетании с текстовым описанием межмодульных соединений для сложных проектов. Для создания и последующего редактирования текстовых описаний частей проекта в Quartus II используется текстовый редактор. Допустимыми являются языки VHDL, Verilog, AHDL (Altera HDL), System Verilog. Соответствующие текстовые файлы имеют расширения **.vhd**, **.v**, **.tdf**, **.sv**.

Достоинствами текстового описания проекта являются его компактность и относительная простота автоматизации любых преобразований, включая процесс создания описания, однозначность понимания и возможность переноса проектов в другие САПР.

Quartus II допускает использование в проектах отдельных компонентов, созданных в системах автоматизации проектирования сторонних производителей. Такими компонентами могут быть и IP ядра. В этом случае, файл описания, как правило, представляет собой таблицу соединений конфигурируемых логических блоков и таблиц соответствия, выполненную до этапа размещения и разводки элементов. Соответствующие файлы имеют расширение **.edf**.

2. Компиляция проекта. Компиляция представляет собой процесс преобразования описания проекта в его структурную реализацию на выбранном кристалле ПЛИС. Компиляции может подвергаться как весь проект, так и отдельные его фрагменты. В Quartus II компиляция всегда выполняется для модуля верхнего уровня (top level). Поэтому для компиляции отдельного компонента схемы необходимо предварительно объявить его модулем верхнего уровня. Компиляция включает выполнение нескольких этапов.

3. Анализ и синтез. Составной частью процесса компиляции проекта является этап анализа и синтеза. Соответствующий модуль

Библиотека ГГТУ им. П.О.Сухого

компилятора Quartus II строит базу данных проекта, которая объединяет все файлы описания проекта в единое целое с учетом иерархического представления проекта. Созданная база данных проекта в дальнейшем будет обновляться другими модулями компилятора до тех пор, пока не будет получен полностью оптимизированный проект. После своего создания база данных содержит только таблицу соединений проекта (netlist). После завершения полной компиляции – полностью оптимизированный, смонтированный проект, который используется для создания файлов, применяемых для временного моделирования, временного анализа, анализа потребляемой мощности и программирования кристалла.

Модуль анализа и синтеза Quartus II выявляет синтаксические ошибки в проекте. Он проверяет логическую завершенность проекта, то есть возможность объединения файлов описания проекта в единое целое, и возможность реализации проекта на выбранном кристалле ПЛИС. Он также преобразует конструкции используемого языка HDL в их аппаратную реализацию на ресурсах кристалла таких, как функциональные преобразователи (lut), триггеры, защелки, логические элементы, блоки встроенной памяти, встроенные умножители. Рис. 4.15. демонстрирует реализацию этапа анализа и синтеза. Исходными файлами для выполнения этапа являются файлы с описанием модулей проекта на языках HDL **(.vhd**, **.v**, **.tdf)** и файлы со схемным представлением **.bdf**. На выходе получаются файлы отчета **(.rpt**, **.htm)** и созданная база данных **(.rdb**).

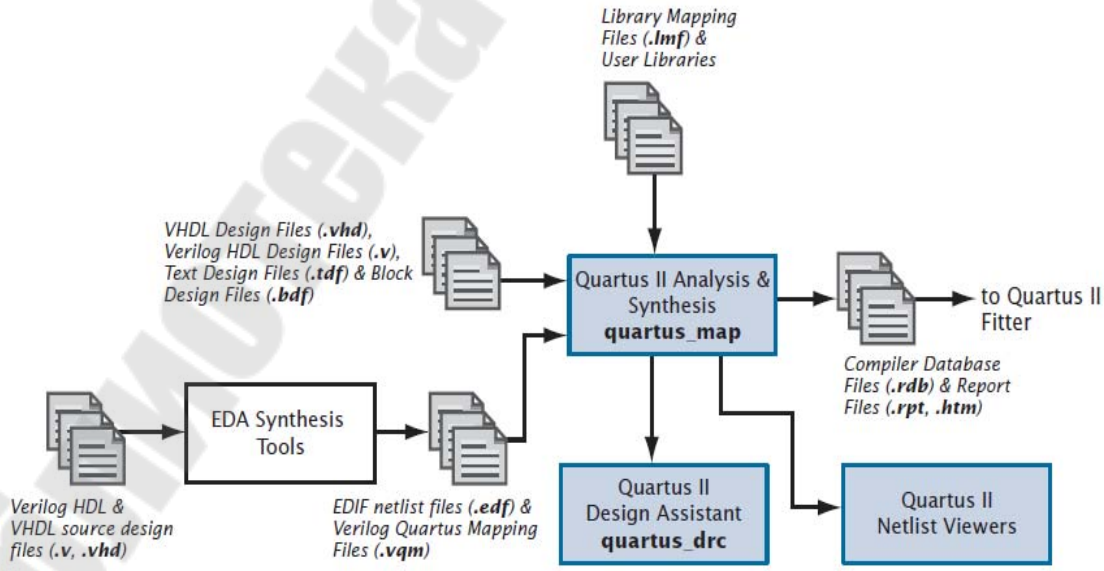


Рисунок 4.15 – Этапы анализа и синтеза проекта

Библиотека ГГТУ им. П.О. Сухого

Пакет Quartus II допускает использование средств синтеза сторонних производителей САПР. В этом случае, созданные проектировщиком файлы описания проекта на языке VHDL или Verilog, должны быть преобразованы средствами синтеза в файл соединений в формате EDIF (**.edf**) или файл Verilog Quartus Mapping File (**.vqm**), которые затем могут быть использованы в пакете Quartus II (см. рис. 4.15).

4. Функциональное моделирование проекта. После завершения этапа синтеза проекта может быть выполнена верификация описания проекта. В основе верификации описания проекта лежит моделирование его работы при имитации различных внешних воздействий. Если при моделировании не учитываются задержки распространения сигналов, то такое моделирование называется функциональным. Существует два подхода к генерации внешних, относительно проекта, воздействий. Первый подход заключается в формировании воздействий путем задания последовательности входных сигналов в редакторе временных диаграмм. Второй подход состоит в написании специальной тестирующей программы с помощью одного из языков HDL. При использовании второго подхода тестируемый объект представляется как структурный компонент, соединенный с одной стороны с генератором тестовых воздействий, с другой стороны с анализатором реакций. Пакет Quartus II поддерживает реализацию обоих подходов. Верификация может быть выполнена как для отдельных частей проекта, так и для проекта в целом. Для выполнения этапа верификации также могут привлекаться средства моделирования, разработанные сторонними производителями САПР.

5. Размещение и трассировка. Модуль компилятора Quartus II , реализующий этот этап проектирования, называется ***Fitter***. Он использует базу данных, созданную на предыдущем этапе модулем анализа и синтеза компилятора. Модуль ***Fitter*** осуществляет монтаж проекта в структуру выбранного кристалла программируемой логики. То есть, полученная на этапе синтеза модель полного представления проекта в техническом базисе кристалла отображается на внутренние ресурсы ПЛИС, которыми являются конфигурируемые логические блоки, блоки встроенной памяти, встроенные умножители и устанавливаются соответствующие соединения с помощью ресурсов трассировки кристалла. Модуль размещения и трассировки

Библиотека ГГТУ им. П.О. Сухого

подбирает для каждой логической функции подходящее место на кристалле, с точки зрения уменьшения времени распространения сигнала, выполняет соответствующие соединения и назначения контактов ввода-вывода. Рис. 4.16. демонстрирует реализацию этапа размещения и трассировки. Исходными файлами для выполнения этапа являются файлы с расширением **.cdb**, созданные после выполнения синтеза компилятором и файлы с установками, имеющие расширение **.qsf**. На выходе получаются файлы отчета о компиляции **(.rpt**, **.htm**) и обновленная база данных.

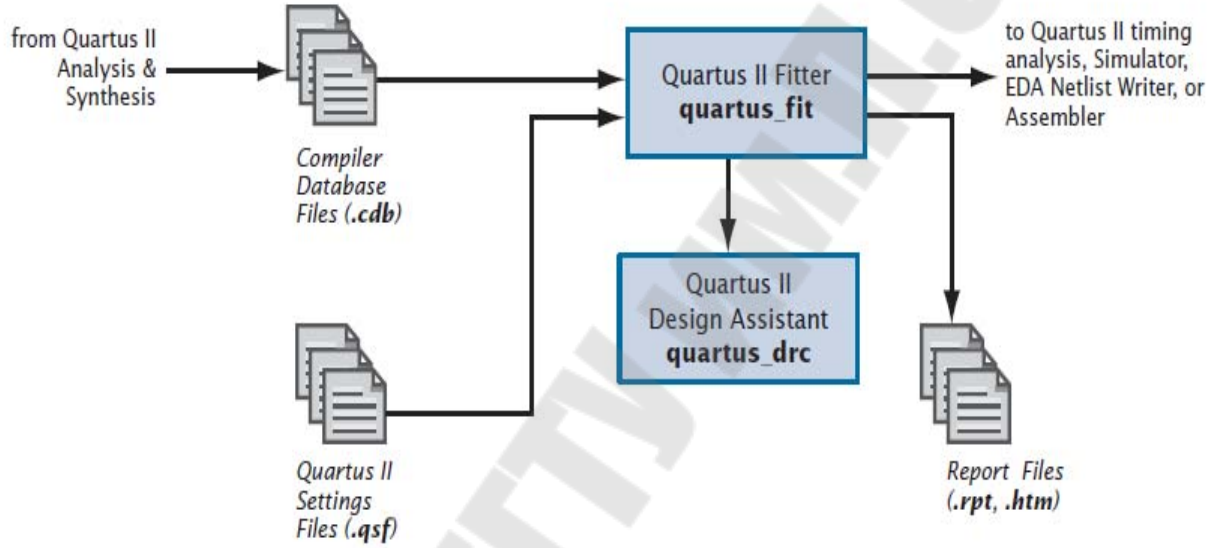


Рисунок 4.16 – Этапы размещения и трассировки проекта

При выполнении этапа размещения и трассировки пользователь может задать свои собственные назначения. Модуль размещения и трассировки реализует их, а затем выполняет оптимизацию оставшейся части проекта. После выполнения монтажа, пользователь может просмотреть результаты размещения и трассировки с помощью специального средства ***Chip Planner*** и, в случае необходимости, изменить некоторые назначения.

Результатом компиляции проекта в пакете Quartus II является загрузочный файл, т.е. конфигурационная информация для выбранной микросхемы ПЛИС или для загрузочного ПЗУ этой интегральной схемы. Также создается файл отчета, содержащий информацию, как о процессе компиляции, так и о его результатах.

В пакете Quartus II имеется специальное средство ***Resource Property Editor***, называемое редактором топологии, с помощью которого можно вручную вмешаться в процесс компиляции и

Библиотека ГГТУ им. П.О. Сухого

изменить структуру проекта на кристалле, например, с целью повышения производительности проектируемого устройства.

6. Временной анализ. Проверка соответствия реализованного проекта требованиям быстродействия.

7. Анализ потребляемой мощности. Проверка соответствия реализованного проекта требованиям по потребляемой мощности.

8. Временное моделирование на вентильном уровне. Осуществляется проверка правильности функционирования проекта после выполнения этапов синтеза, размещения и трассировки.

9.Программирование ПЛИС. Выполняется загрузка конфигурационной информации в программируемый кристалл, посредством чего реализуется пользовательский проект.

10.Тестирование и отладка проекта в составе системы. Проводятся комплексные испытания реализованного проекта, в результате чего принимается решение о соответствии проекта техническому заданию и завершении проектирования. В противном случае принимается решение о доработке проекта.

5. Контроллеры Omron и Mitsubishi

5.1. Функционирование контроллеров Omron

Линейка ПЛК Omron на рынке представлена тремя сериями:

- ПЛК до 320 точек ввода/вывода – это линейка компактных ПЛК со встроенным блоком питания и встроенными входами/выходами, простирающаяся от простого нерасширяемого контроллера CPM1A на 10 точек ввода/вывода до мощного CP1H. Снабженные обширным набором команд для эффективного программирования, эти контроллеры подходят для управления небольшими установками и машинами. Встроенные в них высокоскоростные счетчики и импульсные выходы позволяют легко создавать устройства простого позиционирования или управления скоростью;
- ПЛК до 2500 точек ввода/вывода – это концепция виртуальной "стойки", впервые введенная компанией Omron в 90-х годах, получила свое развитие в нашем ПЛК серии CJ1. Эта модульная система отличается широким набором взаимозаменяемых ЦПУ, сетевых модулей и модулей управления перемещениями. Помимо хороших характеристик управления, она также обладает открытой системой связи и поддерживает интерфейсы Ethernet, DeviceNet,

Библиотека ГГТУ им. П.О.Сухого

PROFIBUS-DP и CAN, благодаря чему она легко доступна для любого интеллектуального сетевого устройства;

- ПЛК для стоечного монтажа характеризуются наличием модулей скоростного высокоточного измерения и сбора аналоговых данных, модулей многоосного управления перемещениями по непрерывной траектории и программируемых модулей связи. Это обстоятельство по праву позволяет назвать CS1 универсальным программируемым контроллером, предназначенным для решения широкого круга задач повышенной сложности. В состав семейства CS1 также входят двухпроцессорные устройства с функцией контроля цикла, предназначенные для построения отказоустойчивых систем для ответственных объектов.

ПЛК Omron серий CJ1 и CS1 PLC построены на основе общей архитектуры. Серия модульных ПЛК CJ1 обладает компактными размерами и значительными возможностями расширения, в то время как ПЛК серии CS1, предназначенные для установки в стойку, предлагают широкий диапазон специализированных модулей управления и функции двойного резервирования. Таким образом, обе серии отлично дополняют друг друга.

Объединяют обе серии надежность и высокая скорость управления. Новые ЦПУ версии 3 дополнены возможностью включения в определямые пользователем функциональные блоки структурированного текста, отвечающего требованиям стандарта IEC 61131-3, что расширяет возможности многократного использования кода и повышает эффективность программирования. Теперь эта функция стандартна для всех ЦПУ, а установка дополнительной памяти или приобретение дополнительного программного обеспечения не требуется. Кроме того, эти нововведения не ухудшают качества управления.

Поскольку новые ЦПУ сохраняют совместимость с предыдущими моделями, переход от использования старых моделей к новой архитектуре производится без внесения каких-либо изменений в программы. Кроме того, непрерывно расширяемая компанией Omron библиотека отлаженных функциональных блоков еще более упрощает составление программ для новых систем. При подключении к ПЛК регуляторов температуры, микропроцессорных датчиков, контроллеров динамического управления и систем технического зрения настройка этих устройств осуществляется автоматически. Это значительно сокращает время, затрачиваемое на разработку, ввод в

Библиотека ГГТУ им. П.О. Сухого

эксплуатацию и техобслуживание. Прозрачная коммуникационная архитектура платформы Omron Smart Platform служит основой полностью интегрированной среды разработки приложений для всех компонентов -от датчика до исполнительного устройства.

В то время, как большинство изготовителей ПЛК полностью полагается на применение в микропрограммном обеспечении функциональных блоков, компания Omron разработала базовый компонент, который значительно сокращает количество дополнительных операций при управлении данными в функциональных блоках (См. Рис. 5.1).

Каждый функциональный блок пределяется только один раз, однако каждый вызов функционального блока из главной программы требует создания нового экземпляра функционального блока и загрузки соответствующих параметров и данных ввода/вывода. После выполнения программы обработанные данные должны быть возвращены, а соответствующее состояние необходимо сохранить до следующего обращения к данному экземпляру функционального блока. Многоязычное ядро управления Omron обеспечивает автономное управление обменом данными с функциональным блоком. В результате эффективность программирования повышается без ухудшения производительности. (См. Рис. 5.2, 5.3).

Рисунок 5.1 – Пример использования функционального блока

Библиотека ГГТУ им. П.О. Сухого

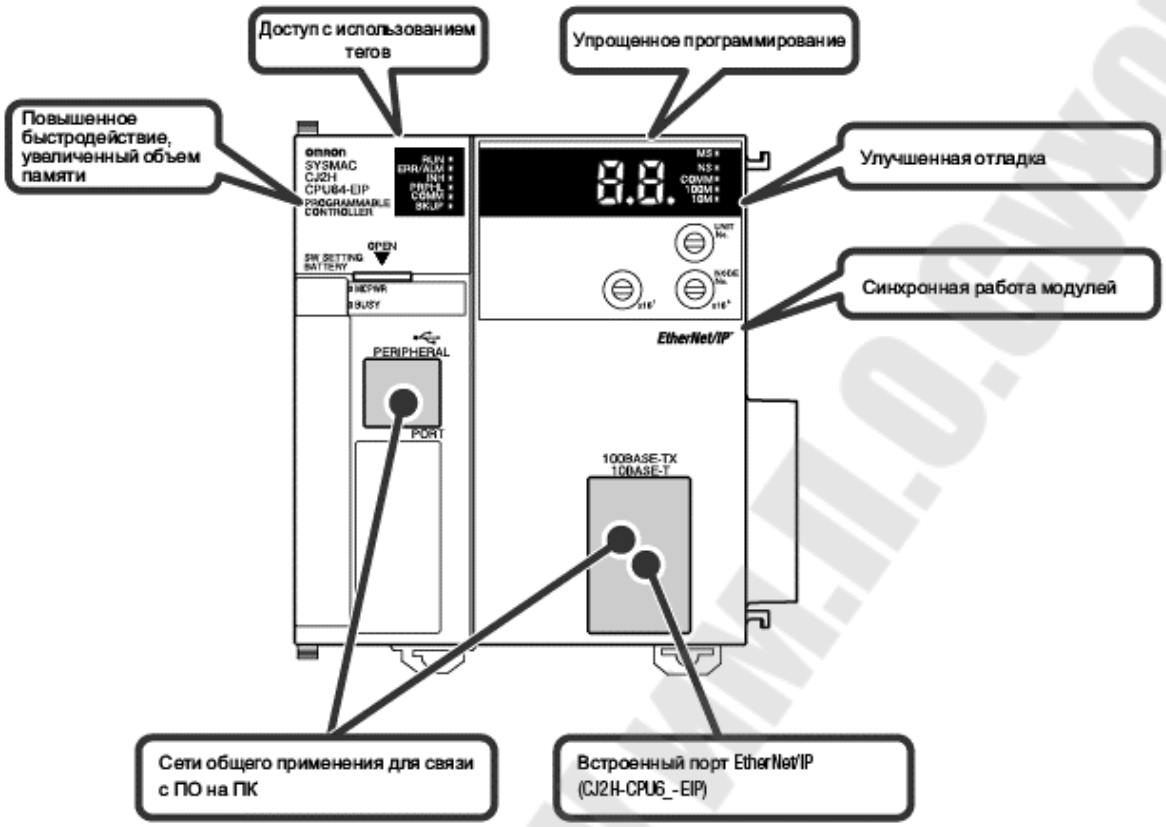


Рисунок 5.2 – Расположение и назначение элементов модуля CJ2H

Программирование контроллера производится на ПЭВМ с помощью пакета программирования CX-Programmer. Его запуск осуществляется через соответствующий ярлык на рабочем столе или по пути меню «Пуск\Программы\Omron\CX-Programmer». Для того чтобы создать новый файл объекта, необходимо выбрать в главном окне программы пункт New в меню File. При этом должно появиться окно (рис.5.4а), в котором необходимо задать нужное имя контроллера (поле «Device Name»), тип контроллера CPM2* (поле «Device Type»), а также тип связи с контроллером SYSMAC WAY (поле «Network Type»). В закладке «Driver» меню «Network Setting» (рис.5.4б) (поле Network Type» кнопка «Settings») необходимо выбрать номер порта, через который осуществляется связь с контроллером. Выбрав нужные параметры, следует нажать кнопку «OK» для подтверждения выбора или «Cancel» – для отмены. После этого вид программы должен измениться – появилось окно проекта и окно редактирования программы (рис.5.5), представляющее чистое поле. В последующем можно изменить параметры проекта, нажав дважды правой кнопкой мыши в окне проекта (левое поле на экране)

на имя контроллера. Команды выбираются из главного меню программы (См. Рис. 5.6).

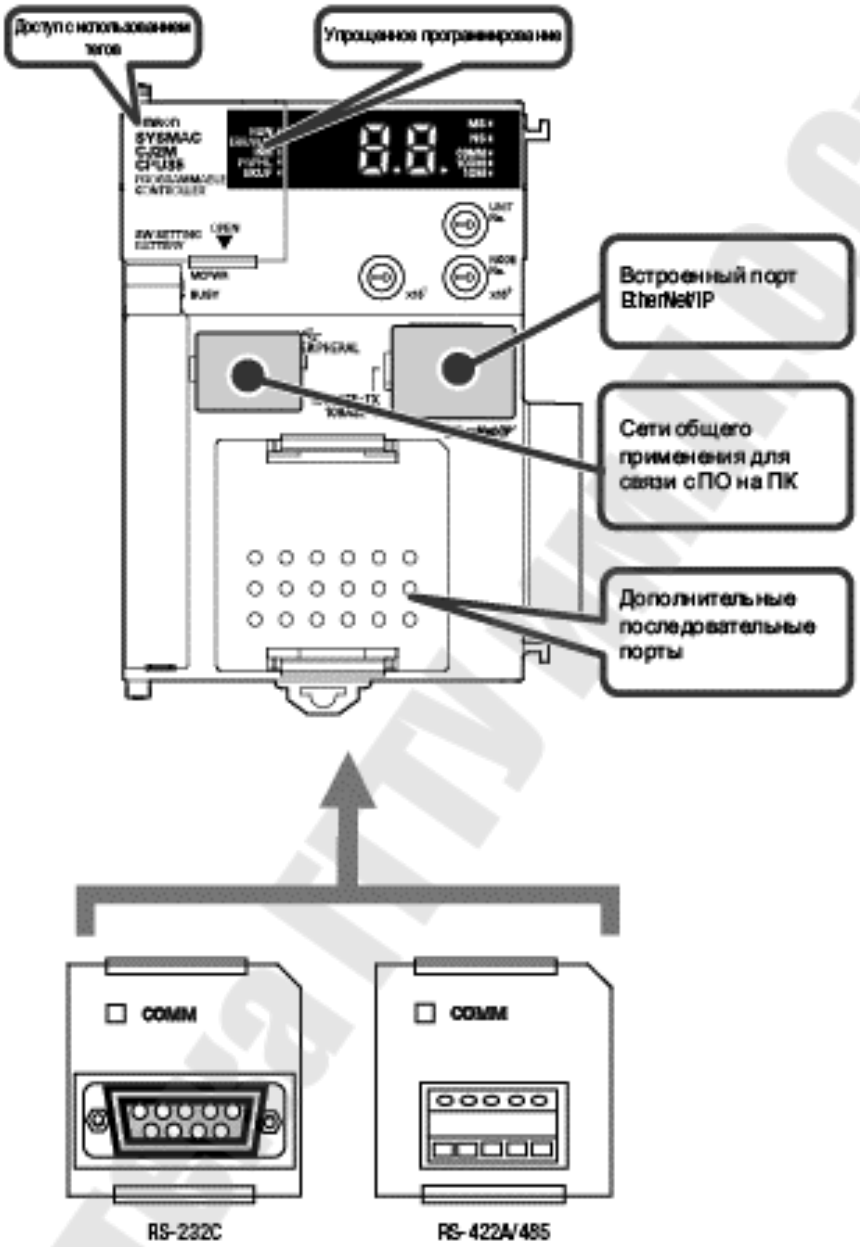


Рисунок 5.3 – Расположение и назначение элементов модуля CJ2M

В рабочей области программы находится окно проекта (рис.5.7). В нем показано иерархическое дерево проекта.

Оно содержит следующие ветви:

- Symbols – таблица глобальных переменных контроллера. Здесь хранится список глобальных переменных используемых ПЛК, их тип, адрес в памяти и описание;

Библиотека ГГТУ им. П.О. Сухого

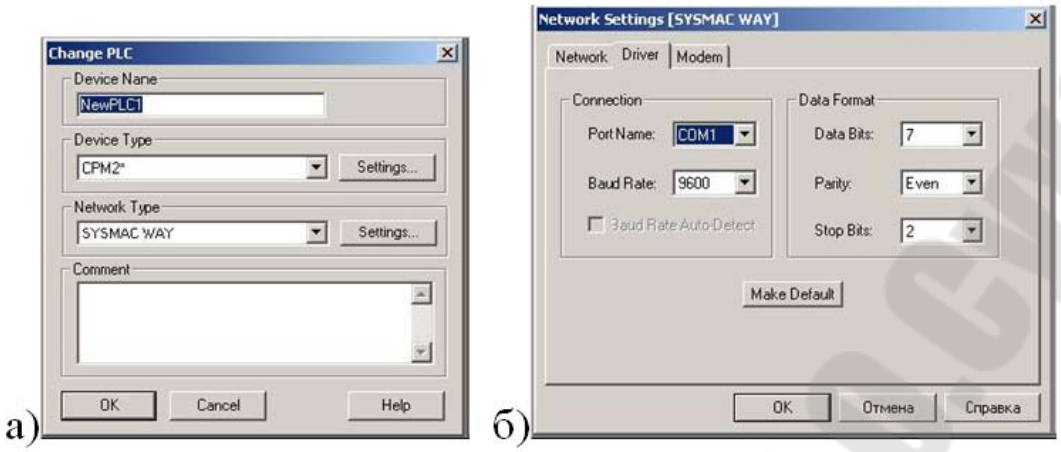


Рисунок 5.4 - Окна выборов параметров контроллера

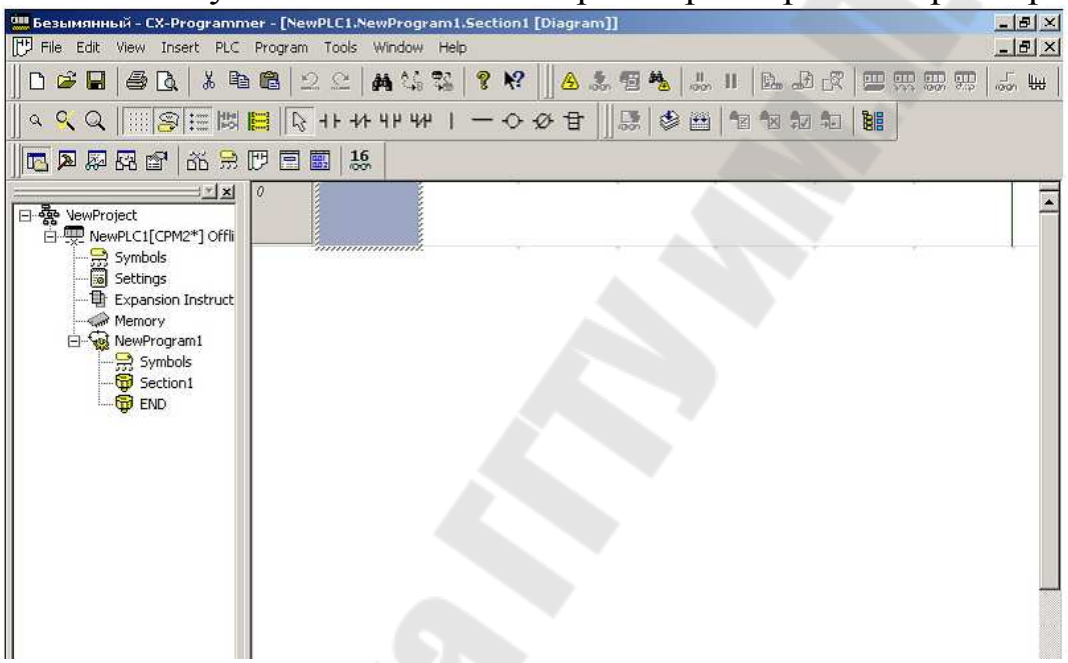


Рисунок 5.5 - Главное окно программы

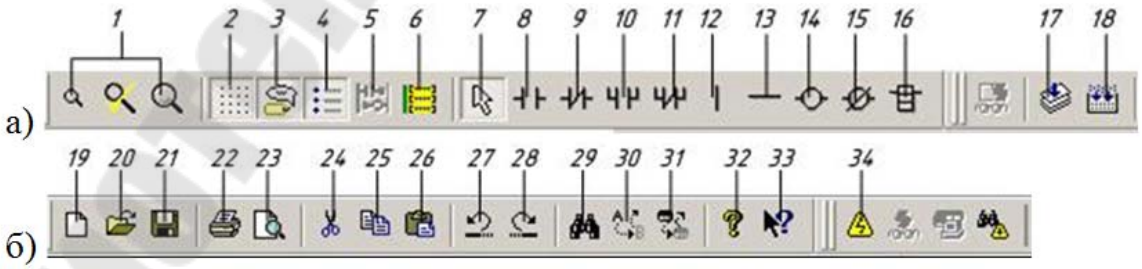


Рисунок 5.6 - Главное меню программы

В табл.5.1 приведено краткое описание элементов панелей инструментов.

Таблица 5.1. Описание элементов панелей инструментов

Номер	Описание

Библиотека ГГТУ им. П.О.Сухого

элемента	
1	Выбор масштаба
2	Показать сетку
3	Показать подписи к элементам программы
4	Показать подписи к командным линиям
5	Наблюдение за командными линиями
6	Показать комментарии
7	Указатель
8	Нормально замкнутый контакт
9	Нормально разомкнутый контакт
10	Нормально замкнутый контакт ИЛИ
11	Нормально разомкнутый контакт ИЛИ
12	Вертикальная соединительная линия
13	Горизонтальная соединительная линия
14	Выход
15	Выход с инверсией
16	Инструкция
17	Компиляция программы
18	Компиляция программы в ПЛК
19	Создать новый проект
20	Открыть проект
21	Сохранить проект
22	Печать программы
23	Предварительный просмотр
24	Вырезать (с запоминанием)
25	Копировать
26	Вставить
27	Отменить операцию
28	Повторно выполнить операцию
29	Поиск элемента программы
30	Изменить адреса в программе
31	Изменить все адреса в программе
32	Справка о программе
33	Справка
34	Связь с ПЛК

- IO Table – таблица доступных входов выходов ПЛК. Отображается информация о модулях ввода-вывода, подключенных к базовой панели

Библиотека ГГТУ им. П.О.Сухого

ПЛК;

- Settings – здесь отображаются настройки ПЛК такие, как режим работы контроллера, время циклов контроллера, настройка работы с удаленными модулями по сети и т.п.

- Memory – таблица распределения памяти ПЛК.

Далее следуют ветви относящиеся к конкретной программе проекта:

- Symbols – переменные используемые в программе, их тип, адрес, описание;

- Section1.. – перечень секций составляющих программу;

- END – обязательная секция означающая конец программы.

Для того чтобы начать программирование в дереве проекта необходимо выбрать пункт «Section1», при этом активизируется окно редактирования программы представляющее собой поле, ограниченное левой и правой шинами, а также панель инструментов.

При вводе команд «Нормально замкнутый контакт», «Нормально разомкнутый контакт», «Нормально замкнутый контакт ИЛИ», «Нормально разомкнутый контакт ИЛИ», «Выход» или «Выход с инверсией» появляется окно (например, как на рис. 5.7), в котором необходимо ввести адрес новой переменной и задать соответствующие параметры. Далее нужно нажать «OK» для подтверждения или «Cancel» – для отмены. Далее появляется окно, в котором можно ввести комментарий к выбранному контакту или выходу. После этого необходимо опять нажать «OK» для подтверждения своих действий или «Cancel» – для их отмены.

Вид окна проекта после компиляции представлен на рисунке 5.8.

Библиотека ГГТУ им. П.О. Сухого

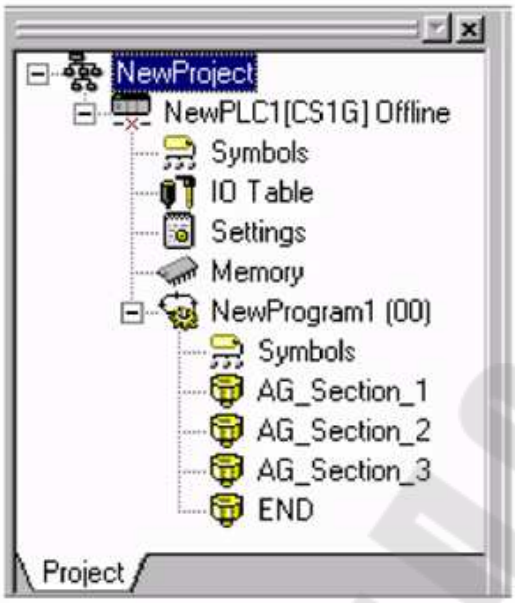


Рисунок 5.7 – Окно проекта

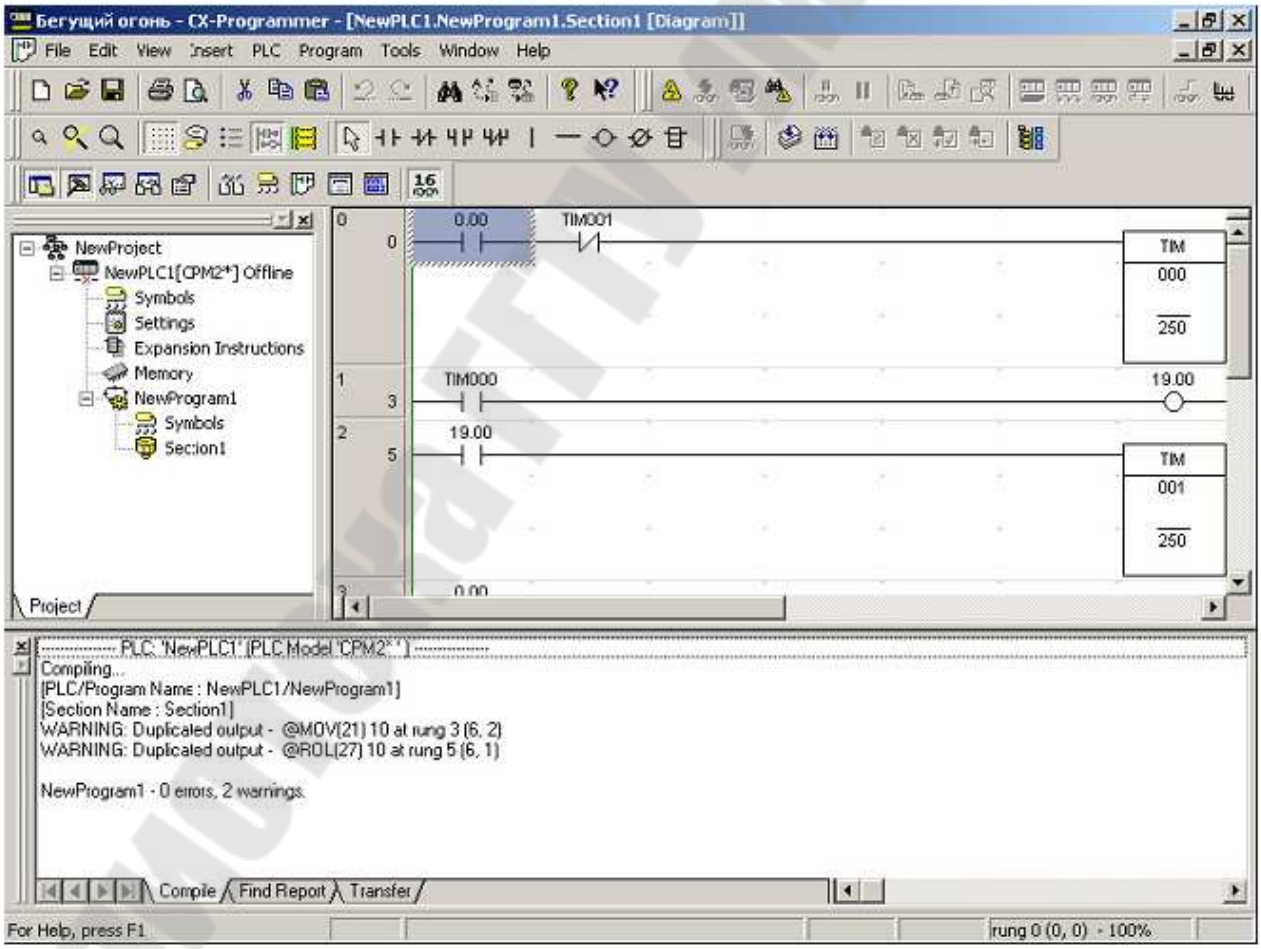


Рисунок 5.8 – Вид окна проекта после компиляции программы

Библиотека ГГТУ им. П.О.Сухого

5.2. Функционирование контроллеров Mitsubishi

Серия MELSEC iQ-R является ядром автоматизированной среды следующего поколения. В этой серии Mitsubishi Electric реализовала концепцию автоматизации, создающую явную добавочную стоимость при одновременном уменьшении общих эксплуатационных расходов.

Серия MELSEC iQ-R (См. Рис. 5.9) разработана с целью устранения узких мест по семи ключевым параметрам: производительность, инжиниринг, техобслуживание, качество, возможности коммуникации, безопасность и совместимость.

При решении этой задачи одновременно преследовались три цели: уменьшение общих эксплуатационных расходов, повышение надежности и дальнейшее применение уже имеющихся производственных средств.

Рисунок 5.9 – Внешний вид контроллеров Mitsubishi iQ-R

Особенность серии iQ-R – возможность установки аппаратного ключа безопасности в ЦП, без которого модуль ЦП не запустится. Данные в этом ключе хранятся в зашифрованном виде и не могут быть скопированы посторонними. Более того, можно задать доверенные IP-адреса, что повлечет отказ в доступе с несанкционированных устройств и снизит риск взлома или изменения программы ПЛК несанкционированным персоналом. Все это – дополнительно к функции идентификации пользователя.

Серия MELSEC iQ-F (См. Рис. 5.10) является дальнейшим развитием успешной серии F и отличается усовершенствованной

Библиотека ГГТУ им. П.О. Сухого

высокоскоростной шиной, расширенным перечнем встроенных функций, поддержкой перспективной сети SSCNETIII/H и улучшенной средой разработки. Для программирования и параметрирования используется программное обеспечение разработчика GXWorks3.

Многофункциональный базовый модуль с источником питания, ЦПУ и каналами ввода/вывода. Встроенные возможности включают высокоскоростные счетчики, выходы позиционирования, Ethernet и слот для SD-карты.

Рисунок 5.10 – Внешний вид контроллеров Mitsubishi iQ-F

Базируясь на своей предшественнице - серии AnSH, концепция управления MELSEC System Q позволяет выбрать наилучшее сочетание из модулей центральных процессоров, модулей коммуникации, специальных модулей, модулей ввода/вывода и объединить их на едином базовом шасси. Это позволяет быстро сконфигурировать для прикладной задачи индивидуальную систему.

Имеется возможность составить уникальную систему System Q (См. Рис. 5.11), содержащую до 4 различных процессорных модулей. Это могут быть базовые и высокомощные модули контроллерных процессоров, специальный контроллер для управления движением, процессоры для аналогового регулирования и даже компьютерные процессоры (промышленный персональный компьютер). Таким образом, пользователь располагает выбором из большого числа концепций управления и программирования, а также языков программирования – и все это на единой платформе.

Библиотека ГГТУ им. П.О.Сухого

Большой выбор сетевых блоков, центральных процессоров, модулей ввода/вывода, специальных и коммуникационных модулей делает MELSEC System Q одной из самых гибких модульных систем автоматизации в мире.

Кроме того, для повышения производительности вашей установки System Q позволяет через ETHERNET непосредственно обращаться к базам данных SQL.

Рисунок 5.11 – Внешний вид контроллеров Mitsubishi System Q

GX Developer поддерживает все контроллеры MELSEC: от компактных контроллеров MELSEC серии FX до модульных систем, включая MELSEC System Q. Данное программное обеспечение отличается простой структурой и очень быстрой осваиваемостью.

GX Developer поддерживает список инструкций (IL) MELSEC, язык релейных диаграмм (LD) MELSEC и язык последовательных функциональных схем (SFC) MELSEC. При программировании можно в любое время переключаться между IL и LD. Можно программировать функциональные модули (MELSEC серий QnA/QnAS/System Q), а для MELSEC System Q имеется множество утилит для параметрирования специальных модулей. Необходимость в программировании специального модуля отпадает - отныне он только параметрируется.

Для параметрирования сетей и аппаратуры MELSEC в вашем распоряжении мощные редакторы и возможности диагностики. Особое внимание было уделено поддержке при вводе в эксплуатацию. Имеются многочисленные возможности тестирования и мониторинга (См. Рис. 5.12).

Моделирующая программа GX представляет собой средство автономного моделирования, с помощью которого вы уже перед вводом в эксплуатацию можете проверить все важные функции вашей программы. С помощью моделирующей программы GX вы можете

Библиотека ГГТУ им. П.О.Сухого

также имитировать все свои операнды и предварительно выбирать реакции вашего приложения, благодаря чему возможно реалистичное тестирование.

Достоинства использования программного обеспечения Mitsubishi:

- Стандартное программное обеспечение для программирования всех контроллеров MELSEC;
- Удобное ведение пользователя под Microsoft Windows;
- Язык релейных диаграмм (LD), список инструкций (IL) или язык последовательных функциональных схем (SFC);
- Переключение во время работы;
- Мощные функции контроля и тестирования;
- Автономное моделирование программируемых контроллеров всех типов.

Рисунок 5.12 – Структура взаимодействия аппаратного и программного обеспечения ПЛК Mitsubishi

Библиотека ГГТУ им. П.О. Сухого

Для запуска MELSOFT Navigator (См. Рис. 5.13) или одного из приложений для программирования выберите соответствующий продукт в стартовом меню Windows:

- MELSOFT Navigator: [MELSOFT Application][MELSOFT IQ Works] [MELSOFT Navigator] ;
- GX Works2:[MELSOFT Application] [GX Works2] [GX Works2];
- MT Developer2:[MELSOFT Application][MTWorks2][MT Developer2];
- GT Deslgner3:[MELSOFTApplication] [GTWorksJ] [GTDeslgner3].

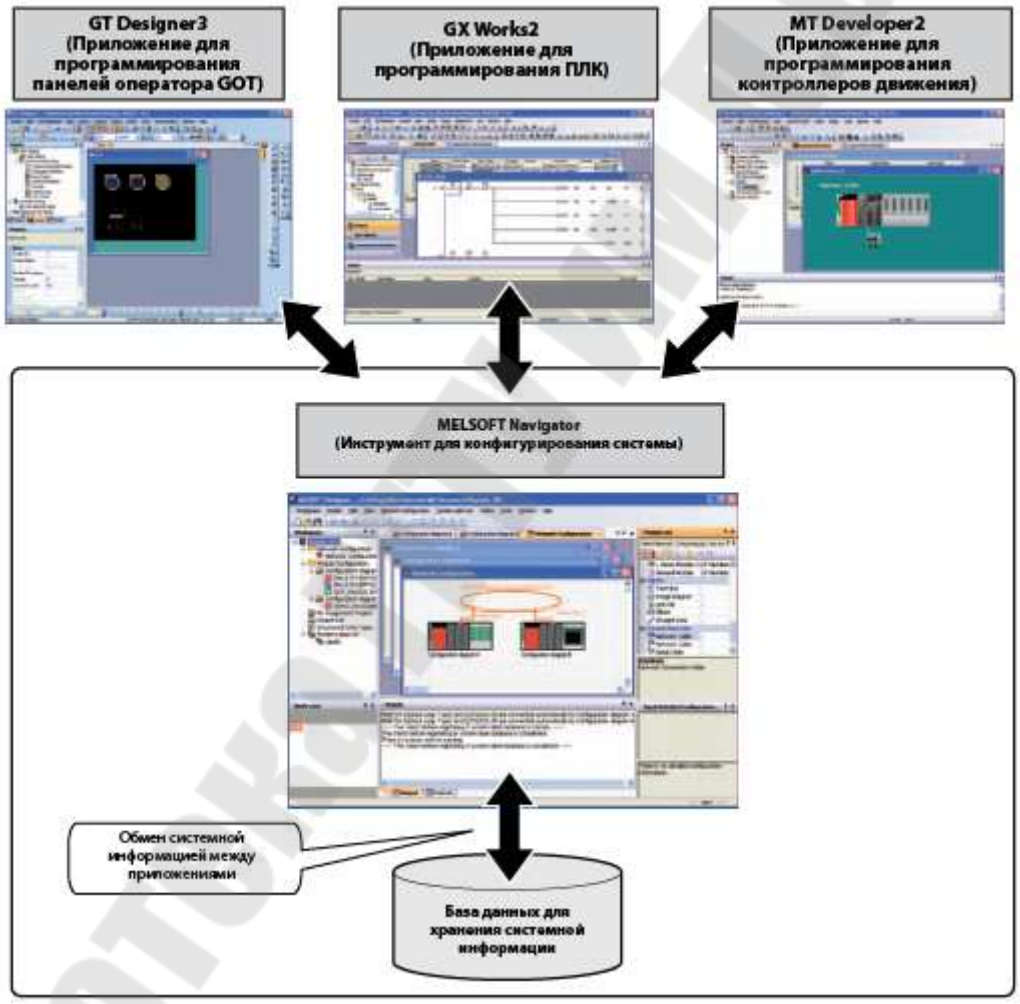


Рисунок 5.13 – Управление системой с помощью MELSOFT NAVIGATOR

Компоненты реальной среды, относящиеся к определенному проекту, управляются в реальном времени в режиме графического

Библиотека ГГТУ им. П.О.Сухого

представления. Изображение компонентов соответствует реальной аппаратной конфигурации всей системы, причем отдельные компоненты можно представлять в графическом виде и связывать с определенным проектом.

Вы можете управлять одновременно данными нескольких проектов (например, проекта ПЛК. контроллера движения и/или графической панели оператора) посредством рабочей области.

Созданные или измененные данные можно также синхронизировать с помощью списка проектов.

Вы можете выполнять настройку значений параметров, например, назначение входов/выходов, или настройку сетевых параметров, не открывая соответствующие проекты (См. Рис. 5.14).

Например, если требуется сконфигурировать несколько многопроцессорных систем, достаточно передать сконфигурированные параметры проекта № 1 в проект№2.

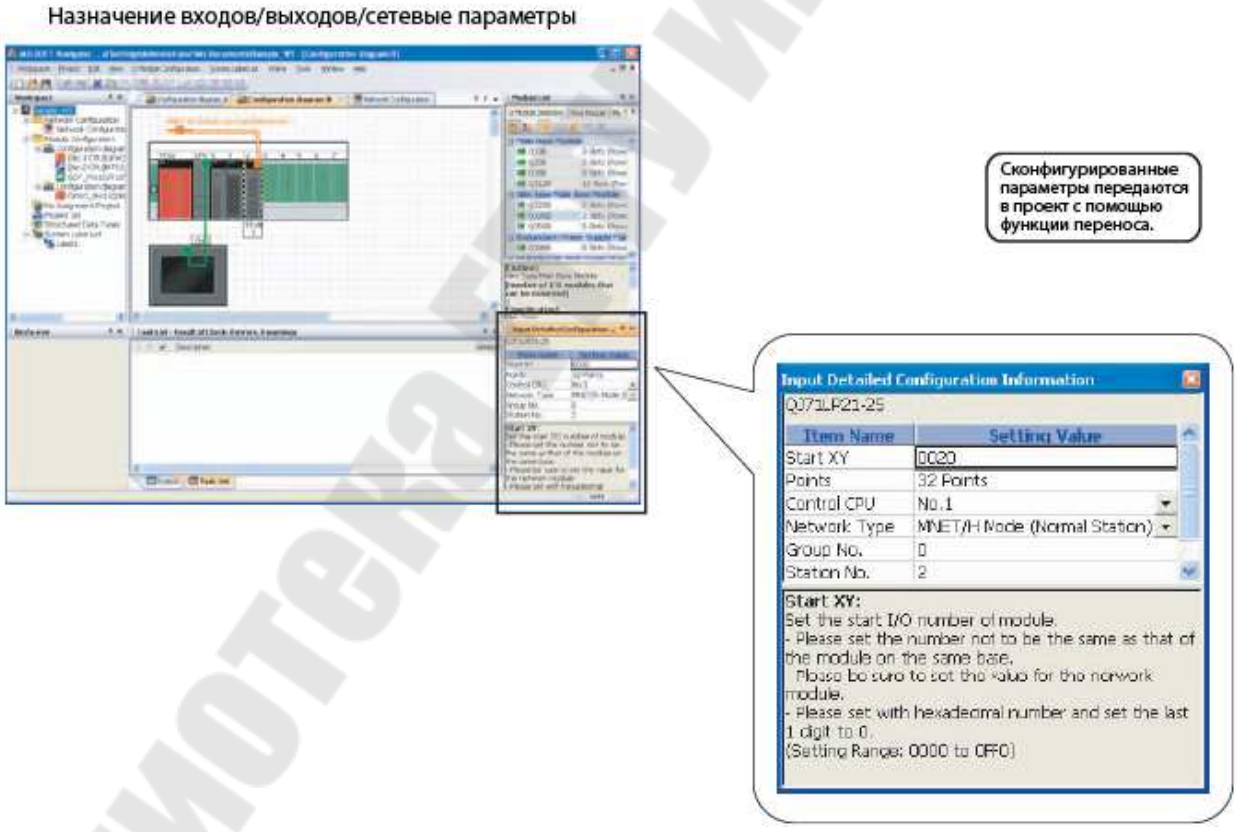


Рисунок 5.14 – Назначение входов/выходов и настройка сетевых параметров

Библиотека БГУИР

Системные метки используются в качестве общесистемных в каждом проекте в рабочей области (См. Рис. 5.15). Это означает, что они доступны для всех устройств, включенных в схему конфигурации сети или модулей. Тем самым повышается эффективность программирования, так как это позволяет объявлять операнды в проекте ПЛК или контроллера движения в качестве системных меток. Эти системные метки можно также использовать в других проектах, без их дополнительного нового объявления в соответствующем проекте.

Поскольку настройки операндов изменяются на каждом этапе работы, применение всех настроек и изменение системных меток для всех проектов выполняется автоматически за один раз.

Рисунок 5.15 – Программирование с использованием системных меток

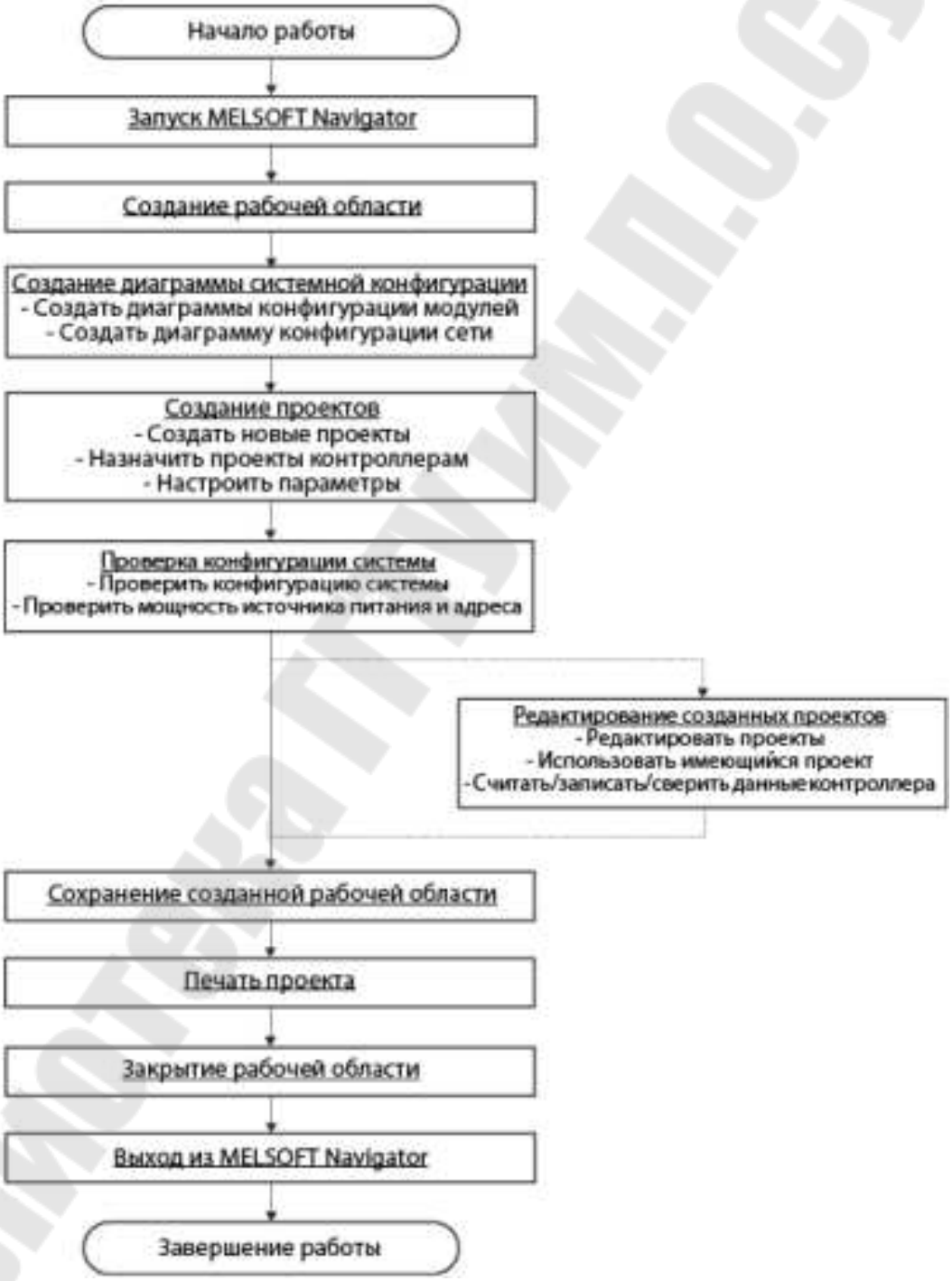
Чтобы использовать системные метки в среде MELSOFTiQ Works, поддерживающей платформу iQ, вы можете вставлять в качестве системных меток операнды, зарегистрированные в MELSOFT Navigator, непосредственно в проекты (нисходящее структурирование), или регистрировать в качестве системных меток уже определенные в отдельных проектах глобальные переменные (восходящее структурирование).

На следующей блок-схеме (Рис. 5.16) показаны основные этапы работы с MELSOFT Navigator.

1. Контроллеры Direct Logic. Функционирование контроллеров DL06

Библиотека ГГТУ им. П.О. Сухого

Микроконтроллеры серии DL06 комбинируют в себе мощные возможности и компактные размеры. Серия DL06 предлагает расширяемый ввод/вывод, высокоскоростной счетчик, арифметику с плавающей точкой, ПИД-регулирование, ряд коммуникационных возможностей, LCD панель и многое другое.



Библиотека ГГТУ им. П.О. Сухого

Рисунок 5.16 – Этапы создания проекта

Для программирования микроконтроллеров доступны два языка: RLL (Relay Ladder Logic – язык релейной логики) и RLLPLUS. RLLPLUS объединяет стандартный язык релейной логики с возможностями стадийного программирования (StageTM). Оба языка одинаково хорошо поддерживаются как ручным программатором, так и пакетом программирования DirectSoftTM.

Все микроконтроллеры DL06 имеют встроенный порт для программирования с использованием ручного программатора (D2-HPP), такой же программатор используется с сериями DL05, DL105 и DL205. Ручной программатор может быть использован для создания, изменения или отладки прикладной программы. Для программирования DL06 нам понадобится D2-HPP со встроенным программным обеспечением версии 2.2 или выше.

Большая часть соединений, индикаторов и обозначений расположены на передней панели микро ПЛК DL06. Порты связи находятся на передней стороне контроллера, так же как слоты для дополнительных модулей и переключатель выбора режимов. См. рисунок 6.1.

Выходные разъемы и разъемы питания принимают внешнее питание AC(L) - AC(N), логическую землю, заземление корпуса на обозначенных контактах. Оставшиеся клеммы для подключения общих проводов C0-C3 и выходов с Y0 до Y17. Клеммы 16 выходов пронумерованны в восьмеричной системе счисления, от Y0 до Y7 и от Y10 до Y17. В моделях с выходами постоянного тока крайняя правая клемма обеспечивает питание для выходных цепей. Входные разъемы обеспечивают подключение входов X0 и X23 и общих проводов C0-C4.

Перед дальнейшим изучением схем подключения мы должны иметь полное понимание понятий «приемник» и «источник» тока. Эти термины часто используются при описании дискретных входных и выходных цепей постоянного тока. Цель данного раздела — облегчить понимание этих понятий. Сначала

Библиотека ГГТУ им. П.О. Сухого

приводится краткое определение, за которым следуют практические приложения.

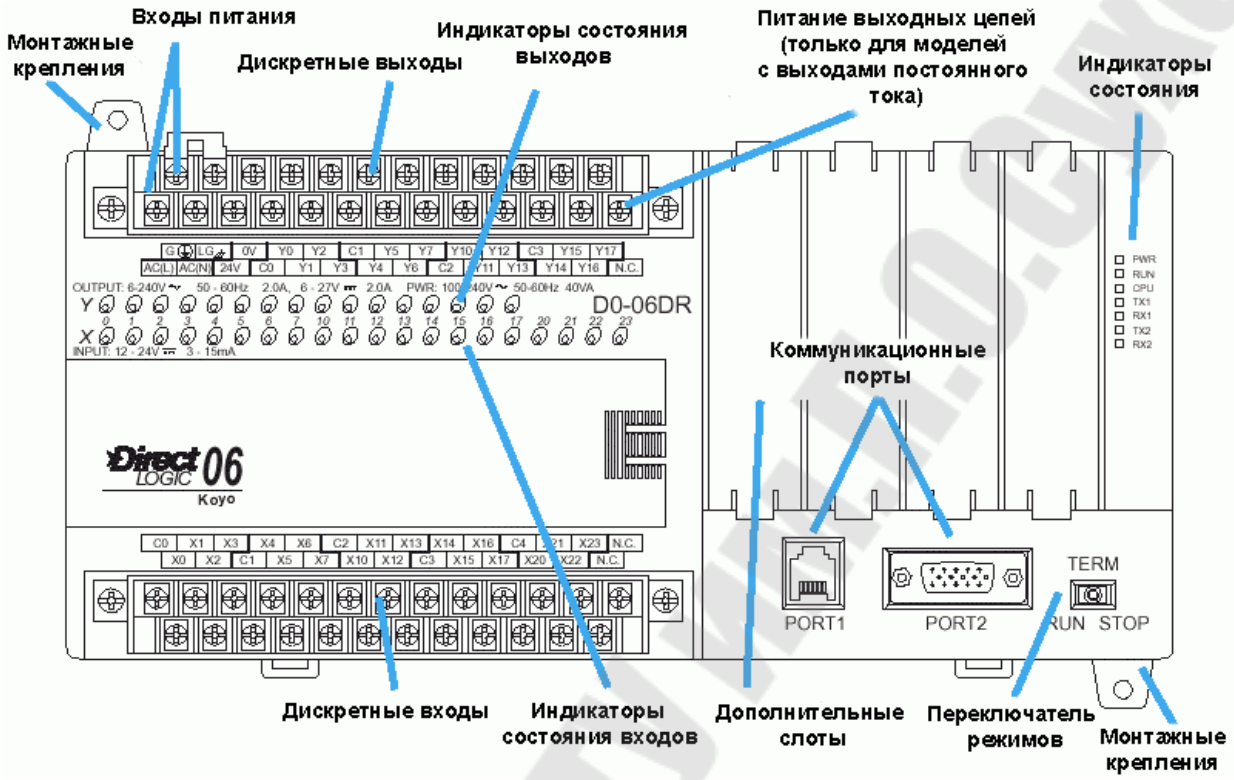


Рисунок 6.1 – Передняя панель DL06

Приемник = предусматривает замыкание цепи на землю (-)

Источник = предусматривает замыкание цепи на источник питания (+)

Обратим внимание на полярность. *Понятия «приемник» и «источник» относятся только к цепям постоянного тока.* Входные и выходные точки, которые являются приемниками либо источниками, могут проводить ток только в одном направлении. Это означает, что можно подсоединить внешний источник питания и полевые устройства к точке Ввода/Вывода с обратным направлением тока, и эти цепи не будут работать. Таким образом, мы можем приступить к монтажу только при понимании понятий «Приемник» и «Источник». Например, на рисунке 6.2 справа показан вход — «приемник». Для правильного подсоединения внешнего источника питания, вы должны осуществить это соединение таким образом, чтобы вход обеспечивал проводимость к заземлению(-).

Библиотека ГГТУ им. П.О. Сухого

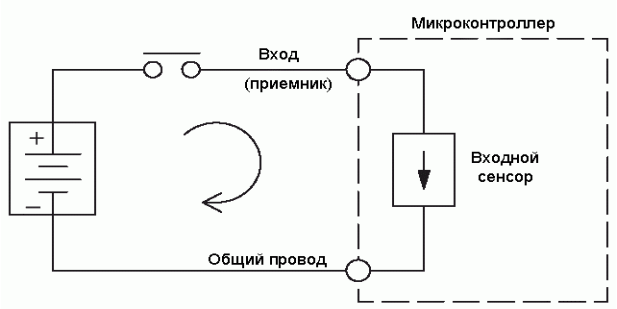


Рисунок 6.2 – Схема подключения источника информации ко входу ПЛК

Начнем с входной клеммы ПЛК, продолжаем через цепи считывания входа ПЛК и закончим на общем полюсе, соединим источник питания (-) с общим полюсом. После добавления выключателя между источником питания (+) и входом мы замкнем цепь. Если замкнуть выключатель, ток потечет в направлении, указанном стрелками. Применяя указанный принцип построения цепи, получим показанные ниже четыре возможные схемы входного/выходного приемника/источника.

У микроконтроллеров DL06 входы могут быть приемником или источником и выходы или приемником или источником. Любая пара схем ввода/вывода, показанных ниже, возможна с одной из моделей DL06.

Для того чтобы цепи Ввода/Вывода работали, необходимо, чтобы ток входил в одну клемму и выходил через другую. Поэтому с каждым дискретным Вводом/Выводом связаны, по крайней мере, две клеммы. На рисунке 6.3 справа клемма для Входа или Выхода предназначена для питающего провода тока. Еще одна клемма необходима для обратного провода (к источнику питания).

Практически все дискретные входы или выходы на ПЛК совместно используют путь возврата с другими точками входа/выхода. На рисунке 6.4 справа показана группа (или блок из 4 входов, которые совместно используют обратный общий провод. В этом случае четыре входа требуют только пять клемм вместо восьми.

Библиотека ГГТУ им. П.О.Сухого

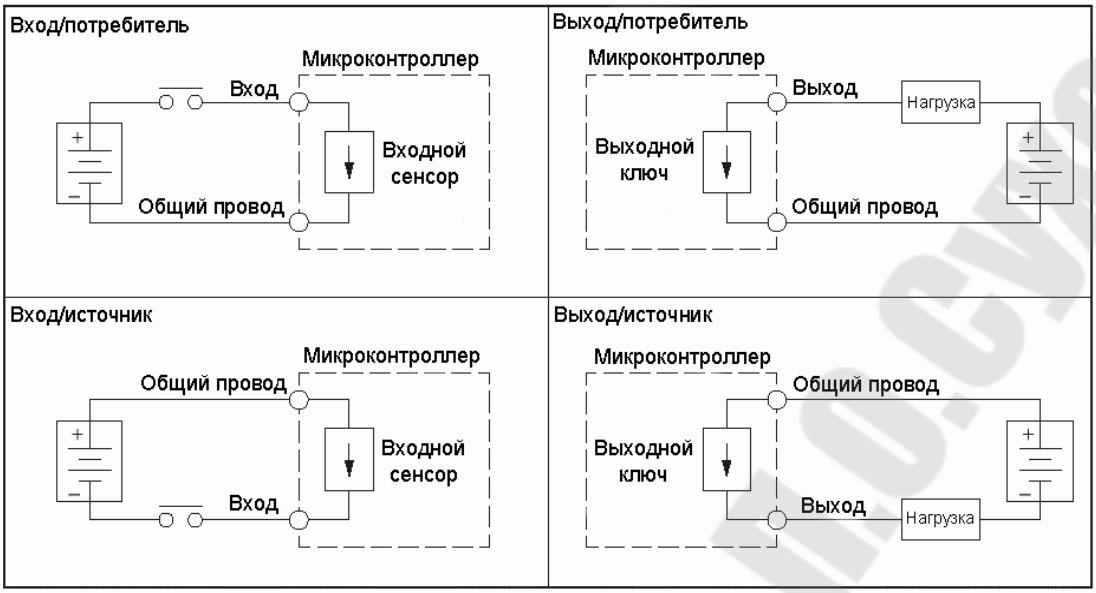


Рисунок 6.3 – Варианты схем подключения входов/выходов ПЛК

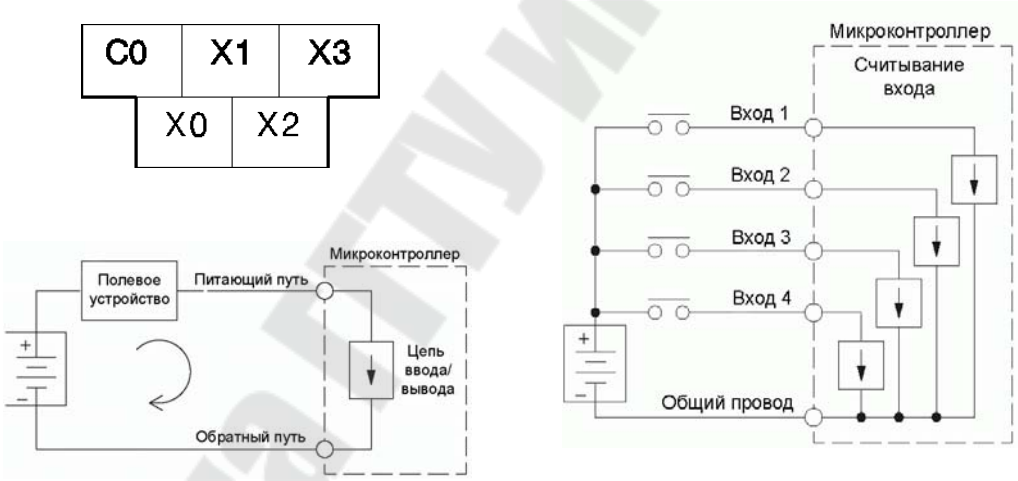


Рисунок 6.4 – Вариант схемы с объединением общего провода источников

Большинство входных и выходных цепей DL06 объединены в блоки, которые совместно используют обратный провод. Хорошим указателем такой группы Ввода/Вывода являются монтажные метки. Общие блоки отделены жирной линией. Более тонкая линия отделяет входы связанные с тем общим проводом. На рисунке справа X0, X1, X2 и X3 совместно используют общий провод C0, расположенный слева от X1.

Библиотека БГТУ им. Н.О. Сухого

Приведенная ниже маркировка показывает наличие пяти групп по 4 входа и четырех групп по 4 выхода. Для каждой группы существует один общий канал. (См. Рис. 6.5).

G ⏚	LG ⏚	0V	Y0	Y2	C1	Y5	Y7	Y10	Y12	C3	Y15	Y17	
AC(L)	AC(N)	24V	C0	Y1	Y3	Y4	Y6	C2	Y11	Y13	Y14	Y16	N.C.

C0	X1	X3	X4	X6	C2	X11	X13	X14	X16	C4	X21	X23	N.C.
X0	X2	C1	X5	X7	X10	X12	C3	X15	X17	X20	X22	N.C.	

Рисунок 6.5 – Группировка входов/выходов DL06

Для того чтобы начать редактировать программу, нужно открыть DirectSOFT 5 (См. Рис. 6.6).

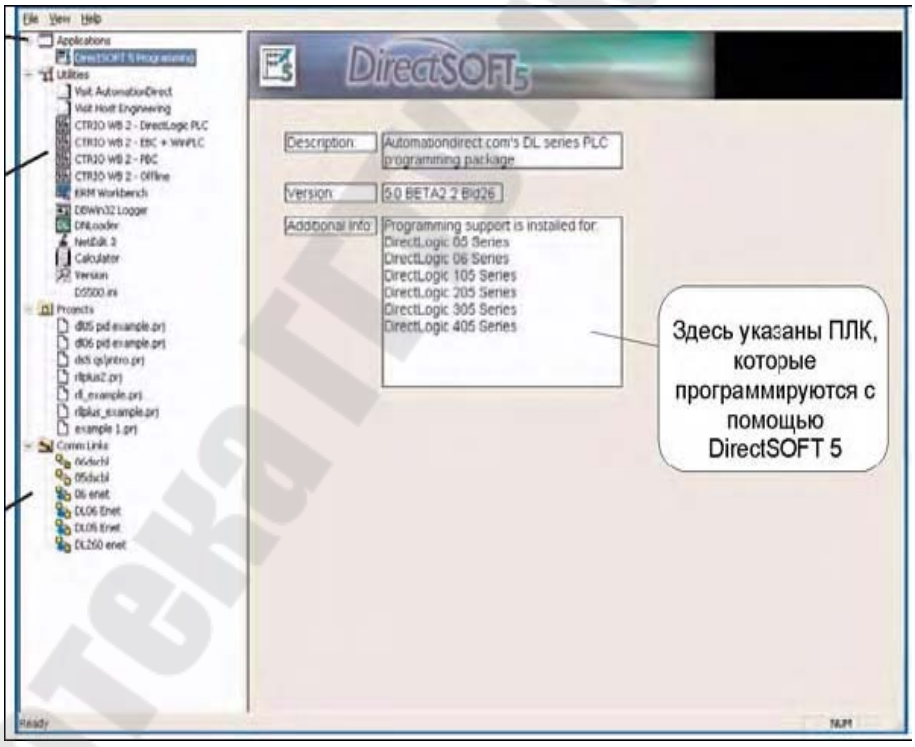


Рисунок 6.6 – Начальная страница Direct Soft

Из этого окна, как из единого центра, могут быть запущены дополнительные подпрограммы, такие как NetEdit, CTRIO WB и другие. Это окно может быть также использовано для создания программ ПЛК и управления этими программами, а также для управления коммуникационными каналами между компьютером и

Библиотека ГГТУ им. П.О.Сухого

ПЛК. Обратите внимание на различные области, которые указаны слева от окна запуска.

• Applications (Приложения) – Это приложения, к настоящему времени установленные в DirectSOFT 5. Ярлычки этих приложений можно увидеть в древовидном меню под папкой/иконкой Applications, эти ярлычки связаны с приложениями, которые разработаны для запуска из DirectSOFT 5. Например, для создания новой программы дважды щелкните по имени DirectSOFT 5 Programming.

• Utilities (Подпрограммы) – Под этой папкой/иконкой имеется несколько подпрограмм. Если вы уже установили такие подпрограммы, как NetEdit, CTRIO Workbench, то они будут показаны здесь. Обратите внимание, что теперь имеются три новые подпрограммы: ярлычок к калькулятору Microsoft®, который (калькулятор) поставляется с Microsoft® Windows, прямое соединение с сайтом AutomationDirect и прямое соединение с сайтом Host Engineering. Ярлычки к наиболее часто используемым подпрограммам можно добавить в раздел Utilitiesфайла DS500.ini.

• Projects (Проекты) – Это программы, которые создаются с помощью DirectSOFT 5. Проект – это общее название для создаваемой программы и сопутствующей ей документации. Когда вы создаете новый проект или работаете над существующим проектом, то имя этого проекта будет в древовидном меню в списке проектов под папкой/иконкой Projects. Для того чтобы открыть существующий проект, дважды щелкните по имени проекта. Для того чтобы открыть проект, который не находится в списке проектов, щелкните правой клавишей мыши по Projects, затем выберите Browse (Обзор) для поиска проекта, потом выберите его.

• Comm Links (Коммуникационные каналы) - Под этой папкой/иконкой указываются коммуникационные каналы, используемые для связи компьютера с одним или несколькими ПЛК. Если в предыдущей версии DirectSOFT уже были созданы коммуникационные каналы, то они появятся здесь. Новые коммуникационные каналы также появится в этой папке после того, как они будут созданы.

Для того чтобы начать проект дважды щелкните по DirectSOFT 5 Programming под папкой Applications в древовидном меню. Появится окно, приведенное ниже. Это окно New Project (Новый проект) предназначено для ввода основной информации о новом проекте. Дайте имя новому проекту, затем передвиньте курсор в область

Библиотека ГГТУ им. П.О.Сухого

Family (Семейство) и выберите семейство, к которому принадлежит программируемый вами ПЛК. Потом выберите тип процессорного модуля. После ввода всей этой информации щелкните по ОК. Имейте в виду, что имеющиеся в DirectSOFT мнемосхемы, правила обработки и инструментальные панели будут согласованы с вашим выбором семейства и типа процессорного модуля.

Окно для программирования немного отличается от такого же окна в предыдущих версиях пакета программирования DirectSOFT. Этот вид окна появляется по умолчанию всегда, когда открывается новый проект. Панели инструментов “Online” (Программирование ПЛК в реальном времени) и “Offline” (Программирование, когда ПЛК отключен) расположены так же, как и в окнах программирования предыдущей версии DirectSOFT, только кнопки на них имеют другие пиктограммы. Обратите внимание на то, что некоторые пиктограммы на кнопках панели инструментов бледно-серого цвета – эти кнопки недоступны, когда как другие кнопки с цветными пиктограммами активны. Во время редактирования программы некоторые из бледно-серых кнопок становятся доступными (цветными). Панель инструментов Online – серая и остается в таком виде до тех пор, пока компьютер не связан с ПЛК.

По умолчанию, когда открывается новый проект, окно программирования по является в виде двух окон. Одно окно Cross Reference View (Окно перекрестных ссылок) размещено слева и другое Ladder View (Окно программы) - справа. Окно перекрестных ссылок – это одно из новых перемещаемых («плавающих») окон в DirectSOFT 5. Оно включает в себя также окна Data Views (Окно данных) и Output (Окно вывода). Эти окна могут быть закреплены («пришвартованы») к любому краю окна программирования или они могут быть «отшвартованы» и перемещены в любую часть экрана или даже на другой дисплей, если у вас к компьютеру подключено несколько мониторов. Если окно пришвартовано, то его можно спрятать, щелкнув по кнопке свертывания в правом верхнем углу этого окна. Окно при этом свернется слева от окна программы в закладку, на которой будет написано имя окна. Чтобы снова развернуть окно, наведите курсор мыши на имя окна в закладке. Если окно совсем не нужно, то закройте его, щелкнув по кнопке X, расположенной справа от кнопки свертывания окна. Обратите внимание, что панель инструментов программирования расположена справа от окна программы. Если не активирован режим

Библиотека ГГТУ им. П.О.Сухого

редактирования, то кнопки элементов программы на панели инструментов программирования будут серыми. Для активации панели инструментов программирования щелкните по любой кнопке EDIT Mode (Режим редактирования) (См. Рис. 6.7); одна кнопка расположена на панели инструментов Offline и другая – на панели инструментов редактирования сверху. Эту панель инструментов программирования можно расположить в любом месте экрана, захватив курсором мыши полоску перетаскивания этой панели и передвигая ее в новое место.

С окном программы можно работать в двух режимах: в режиме просмотра - Display Mode и в режиме редактирования – Edit Mode. Когда открывается новая или существующая программа, окно программы находится в режиме просмотра. В этом режиме программу нельзя редактировать. Для редактирования программы необходимо перейти в режим редактирования. Для включения режима редактирования или щелкните по кнопке режима редактирования на панели инструментов Offline, или щелкните по кнопке режима редактирования на панели инструментов программирования. Когда режим редактирования включен, курсор в виде прямоугольника заливается сплошным однородным цветом, вокруг кнопок включения режима редактирования появляется окантовка в виде прямоугольника и элементы на панели инструментов программирования подсвечиваются.

Панель инструментов программирования (См. Рис. 6.8), показанная ниже, может выглядеть не совсем так, как вы ее увидите на экране компьютера. Набор инструментов на панели зависит от типа процессорного модуля в ПЛК. В примере на рисунке показаны только инструменты, общие для всех процессорных модулей.

Для ввода первой команды в программе воспользуйтесь панелью инструментов для программирования. Сначала щелкните по одной из кнопок Edit Mode (Режим редактирования). После щелчка прямоугольный курсор редактирования будет залит однородным цветом (См. Рис. 6.9). Курсор редактирования следует поместить слева в начале ступеньки 1. Первая команда программы – обычно релейный контакт или элемент - будет помещена в месте расположения курсора. Щелкните по символу Normally Open Contact (Нормально разомкнутый контакт) на панели инструментов программирования. Курсор превратится в окошко ввода элемента, на котором будет нарисован разомкнутый релейный контакт, а также

Библиотека ГГТУ им. П.О.Сухого

помещены текстовое поле с мерцающим курсором после подсвеченного адреса C0 и зеленые индикаторы подтверждения правильности ввода.

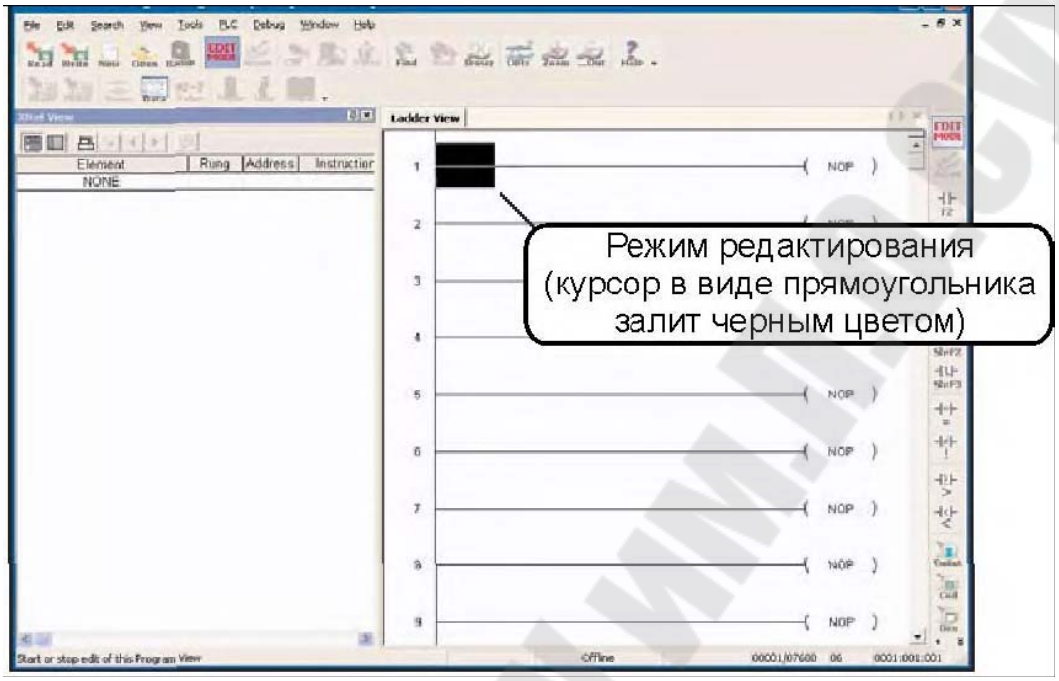
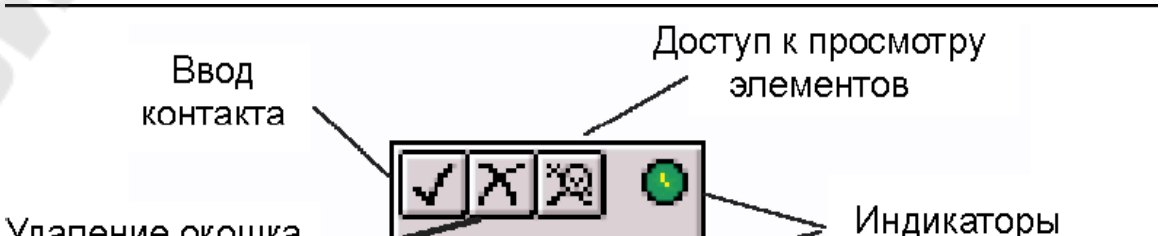


Рисунок 6.7 – Режим редактирования

Рисунок 6.8 – Панель инструментов



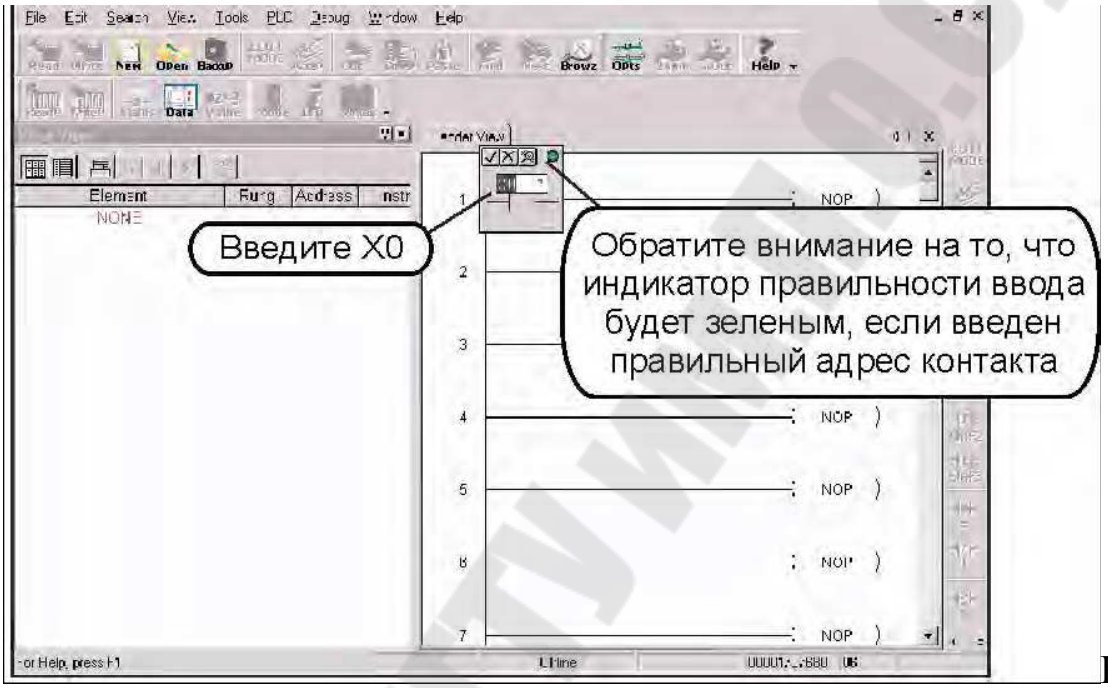


Рисунок 6.9 – Панель редактирования команды

Если зеленый цвет индикатора правильности ввода меняется на красный, то это значит, что введен неверный адрес, недействительный или неправильный символ. Например, если вы ввели букву «О» вместо цифры «0», то индикатор станет красным и будет красным до тех пор, пока не будет исправлена ошибка. В нашем примере введите X0 вместо C0. Индикатор правильности ввода должен остаться зеленым, означая, что адрес введен правильно. Теперь или щелкните по «галочке» (□□), или нажмите клавишу Enter (Ввод).

Элемент будет введен, и курсор передвинется в следующую позицию ввода. Слева от ступеньки 1 появится желтая вертикальная полоса (См. Рис. 6.10). Эта желтая полоса указывает на то, что вы уже ввели команду или команды, но программа еще не принята (не скомпилирована).

Библиотека ГГТУ им. П.О.Сухого

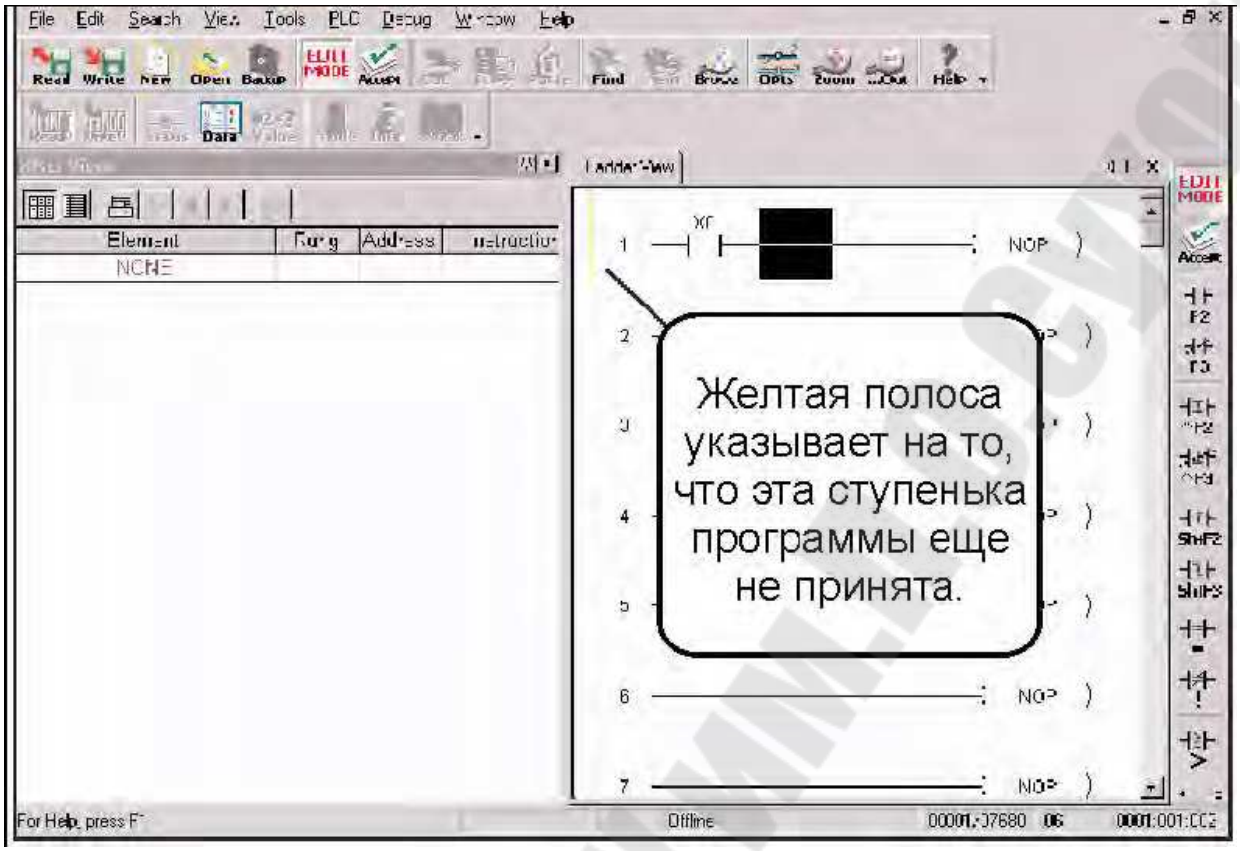


Рисунок 6.10 – Промежуточный вид окна редактирования

Теперь передвиньте курсор в конец ступеньки, поместив его на NOP (No Operation – Нет команды). Щелкните по кнопке Browse Coils (Просмотр обмоток) на панели инструментов программирования. Появится окно Instruction Browser (Просмотр команд), в котором по умолчанию выбрана Standard Coil (Стандартная обмотка). Щелкните по ОК для ввода стандартной обмотки. Имейте в виду, что вы можете выбрать какую-либо другую обмотку.

Окно Instruction Browser (Просмотр команд) сменится окошком ввода элемента. Адрес по умолчанию, C0, будет подсвечен. Введите Y0 и проверьте, что индикатор правильности ввода остался зеленым, означая, что адрес введен правильно. Теперь или щелкните по «галочке» (□□), или нажмите клавишу Enter (Ввод) для ввода элемента.

Ступенька 1 запрограммирована (См. Рис. 6.11). Для того чтобы эту ступеньку программы можно было загрузить в ПЛК, необходимо добавить еще одну недостающую ступеньку с обмоткой END Coil (Конечная обмотка), которой должны заканчиваться все программы.

Библиотека ГГТУ им. П.О. Сухого

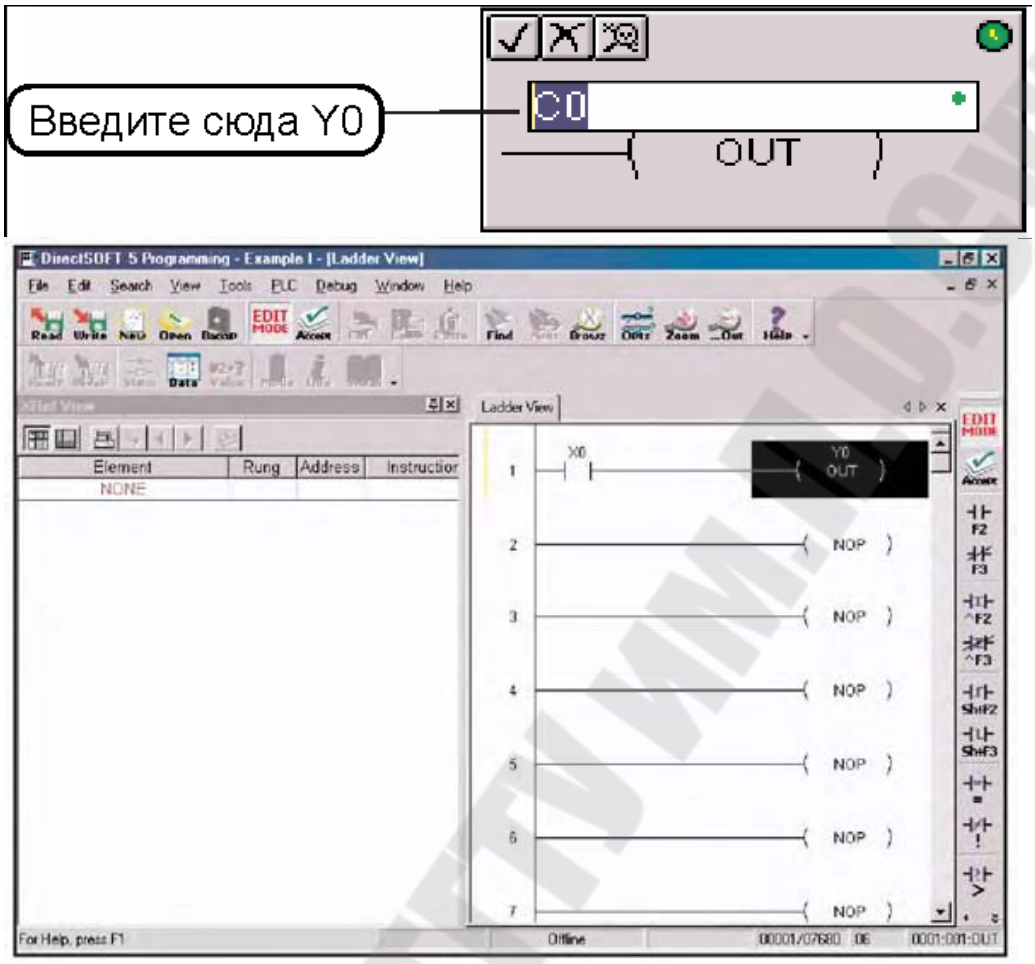


Рисунок 6.11 – Окончательный вид запрограммированного действия

Для программирования этой ступеньки поместите курсор на NOP (No Operation – Нет команды) в конце ступеньки 2 и щелкните по кнопке Browse Coils (Просмотр обмоток) на панели инструментов программирования. Появится окно Instruction Browser (Просмотр команд). На этот раз выберите позицию Program Control (Управление программой) в наборе команд Coil Class (Группа обмоток). Затем выберите позицию END (Конец) в наборе команд Coils (Обмотки). Щелкните по ОК, затем Enter.

Для того чтобы загрузить программу в ПЛК, ее сначала нужно принять. Как видно на рисунке 6.12, есть две кнопки Accept (Принять). Щелкните по любой из этих кнопок для того, чтобы скомпилировать программу. Если ступеньки программы принимаются компилятором без ошибок, то желтая полоса изменится на зеленую,

кнопки Accept станут бледно-серыми и в окне перекрестных ссылок (Cross Reference – XRef) появятся два элемента, которые и были запрограммированы.

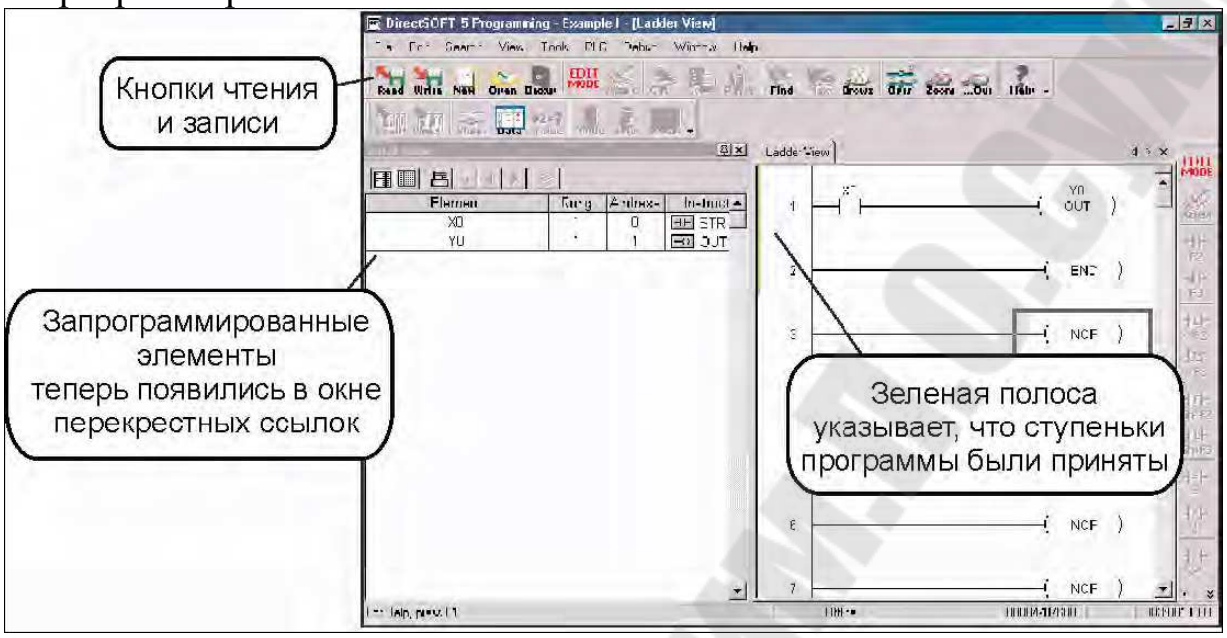


Рисунок 6.12 – Окно записи/чтения готового проекта

Обратите внимание, что две кнопки Read (Прочитать) и Write (Записать), распо-ложенные слева на панели инструментов Offline, теперь включены и не бледно-серые. Программу теперь можно сохранить на диск компьютера. Для записи программы на диск щелкните по кнопке Write (Записать). Совсем не обязательно сохранять программу на диске, если вы хотите загрузить ее в ПЛК, однако, со-хранять программу после ее редактирования – это хорошая привычка. Иногда может быть сделана ошибка в программе и нужно вернуть программу в состояние, в котором она была до ошибки. Если ошибка была сделана, и вы хотите восстановить программу, щелкните по кнопке Read (Прочитать). На экране появится ранее сохраненная версия вашей программы.

Если программа сохраняется нажатием кнопки Write (только на диск), то сохраняется только сама лестничная программа. Если же вы отредактировали программу и включили в нее всю сопутствующую документацию, то необходимо сохранить все, что вы сделали. Это осуществляется выбором в главном меню File > Save Project > to disk. Можно также щелкнуть по Backup (Резервное копирование), чтобы выполнить то же самое действие, при этом будет сохранена резервная копия файла.

Для того чтобы загрузить программу в ПЛК, необходимо создать коммуникационный канал (канал связи) (См. Рис. 6.13).

Соедините кабелем программирования последовательный порт компьютера с последовательным портом ПЛК. Включите ПЛК и убедитесь, что переключатель контроллера RUN/TERM/STOP находится в положении TERM. Щелкните по PLC (ПЛК) в главном меню, затем в выпадающем меню выберите Connect (Соединить), появится диалоговое окно Select Link (Выберите тип канала связи). Поскольку пока еще нет ни одного канала для выбора, щелкните по Add (Добавить). Появится диалоговое окно помощника создания канала связи, в котором пере-числены коммуникационные порты. Выберите порт, который вы будете исполь-зовать (обычно COM1), и щелкните по Next (Далее).

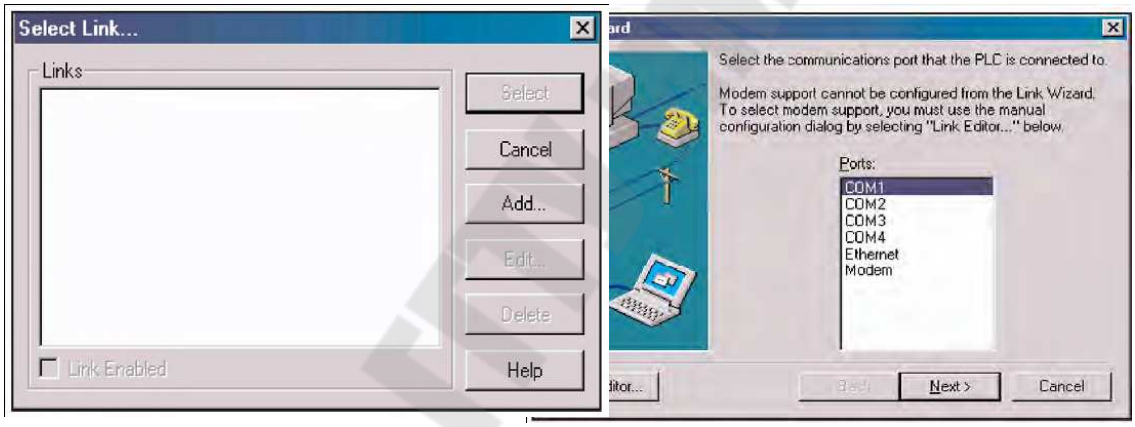


Рисунок 6.13 – Последовательность действий по созданию канала связи

В следующем окне представлен список семейств ПЛК (PLC Families). Выберите семейство ПЛК, щелкнув по нужной строчке списка. Если вы не уверены в том, какое семейство выбрать, но знаете, какой коммуникационный протокол будет использовать, выберите «Not Sure» (Не уверен). Если используется ПЛК, совместимый с контроллерами DirectLOGIC, то помощник создания канала связи попытается определить тип ПЛК автоматически. Когда закончите выбор, щелкните по Next (Далее).

На этом шаге нужно выбрать или DirectNET, или K-Sequence. Предположим, что вы выбрали семейство контроллеров DirectLOGIC (только не DL305), то тогда по умолчанию будет подсвечиваться K-

Библиотека ГГТУ им. П.О. Сухого

Sequence. Протокол обмена данными KSequence позволяет производить операции записи в индивидуальные дискретные каналы ввода/вывода и в управляющие реле. Протокол DirectNET эти операции выполнить не может. (См. Приложение А, в котором приведен перечень протоколов обмена данными для контроллеров DirectLOGIC и совместимых с ними ПЛК. Если вашему ПЛК задан узловой адрес, отличный от 1, то введите этот адрес сейчас. Щелкните по Next (Далее). Если помощник в создании коммуникационного канала установил связь с ПЛК, то в следующем окне вам будет предложено ввести уникальное имя канала связи и, если нужно, описание этого канала. Описание канала связи может содержать до 32 символов(См. Рис. 6.14). Введите имя канала связи и его описание, затем щелкните по Finish (Готово), появится диалоговое окно Select Link (Выберите канал связи), в списке коммуникационных каналов которого будет содержаться и имя созданного канала связи.

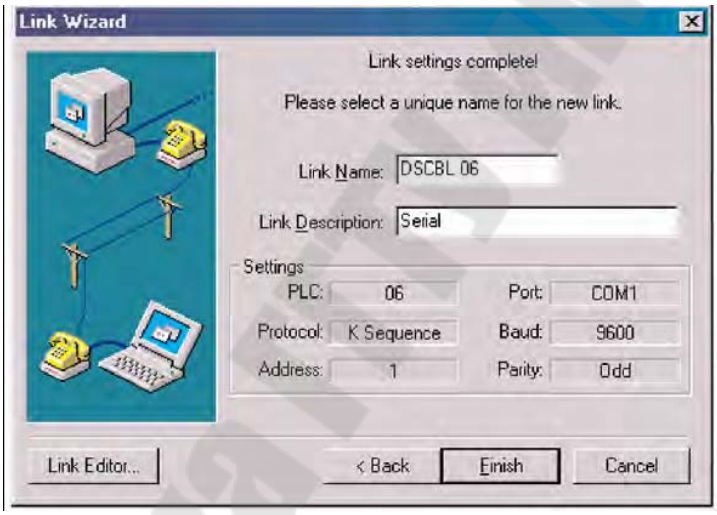


Рисунок 6.14 – Окно идентификации канала связи

Поскольку только что созданный канал связи единственный именованный канал в списке окна Select Link (Выберите канал связи), щелкните по Select (Выбрать) для запуска процесса соединения с ПЛК.

DirectSOFT 5 автоматически сравнит открытую в настоящий момент программу с программой, которая сохранена в ПЛК. Появится диалоговое окно Online/Offline Differences (Отличия программ в компьютере и в ПЛК), в котором спрашивается, какую копию лестничной программы следует вывести на экран компьютера: копию из ПЛК или копию из компьютера.Поскольку в настоящий момент мы

Библиотека ГГТУ им. П.О.Сухого

имеем дело с новой программой, выберите кнопку Use Disk (Воспользоваться программой на диске компьютера). Кнопку Use Disk следует использовать всякий раз, когда на компьютере, который соединен с ПЛК каналом связи, вы сделали изменения в программе и собираетесь загрузить ее в ПЛК. Если нажать кнопку Details (Подробности), то бок о бок друг с другом появятся отличающиеся ступеньки программ в ПЛК и в компьютере, например, такие, как показанные ниже в окне Compare Programs (Сравнение программ). Выбор нужной программы можно также сделать в этом диалоговом окне.

После нажатия на кнопку Use Disk (Воспользоваться программой на диске ком-пьютера) окно программирования будет выглядеть несколько иначе. Обратите внимание на то, что рисунки (иконки) на кнопках панели инструментов Online уже не бледно-серые. Индикаторы внизу окна программирования подсказывают вам, что ПЛК исправен, компьютер связан с ПЛК каналом передачи данных и ПЛК на-ходится в режиме программирования (Program Mode). На данном этапе про-грамма ещё не записана в ПЛК. Также обратите внимание, что две самые левые кнопки на панели инструментов Online (ReadP (Прочитать программу) и WriteP (Записать программу) подсвечены. Для записи программы в ПЛК выберите кнопку WriteP. Появится всплывающая индикаторная панель, которая позволяет видеть, что программа записывается в ПЛК.

После того как программа будет записана в ПЛК, всё что останется сделать это перевести ПЛК в режим выполнения программы (RUN Mode). Щелкните по кнопке Mode (Режим) на панели инструментов Online. Появится диалоговое окно PLC Modes (Режимы ПЛК). Щелкните по RUN, затем ОК и ПЛК перейдет в режим вы-полнения программы (RUN Mode). Обратите внимание на зелёный индикатор под окном программы. Он показывает, что ПЛК сейчас находится в режиме выполнения программы.

Библиотека ГГТУ им. П.О. Сухого

Список литературы

1. Николайчук О. И. Н28 Системы малой автоматизации / О. И. Николайчук — М.: СОЛОН-Пресс, 2003. 256 с.
2. Основы микропроцессорной техники. Курс лекций. Учебное пособие / Издание второе, исправленное / Новиков Ю.В., Скоробогатов П.К. / М.: ИНТУИТ.РУ «Интернет-университет Информационных Технологий», 2004. — 440 с.
3. Густав Олссон, Джангуидо Пиани. Цифровые системы автоматизации и управления. — СПб.: Невский Диалект, 2001.-557 с.: ил
4. Грушвицкий Р.И., Мурсаев А.Х. Угрюмов Е.П. Проектирование систем на микросхемах с программируемой структурой. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006. - 736с.
5. CoDesSys интегрированный комплекс МЭК 61131-3 программирования. Описание программного комплекса. Смоленск. – «Пролог», 2014- 34с.
6. Бергер Г. Автоматизация посредством STEP 7 с использованием STL и SCL и программируемых контроллеров SIMATIC S7-300/400. Siemens AG, Нюрнберг, 2001.
7. Программирование с помощью STEP 7 V5.3. Руководство 6ES7810-4CA07-8BW1. Siemens AG, Нюрнберг, 2004.
8. Романов В.П. Основы языка программирования STEP 7 и базового программного обеспечения промышленных контроллеров Siemens. Учебно-методическое пособие. Новокузнецк.-Изд. ФГУО СПО «КИТ». - 2009г. – 50с.
9. Блудов А.В. Единый программный проект CX-One./ Автоматизация в промышленности / М. – Изд. Дом «ИнфоАвтомтизация», №8, 2012г.
10. Бибило П. Н. Основы языка VHDL. Изд. 3-е, доп.- М.: Издательство ЛКИ, 2007. -328с.
11. Поляков А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М.: СОЛОН-Пресс, 2003. -320с.
12. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2004. – 256 с.: ил.

Библиотека ГГТУ им. П.О.Сухого

13. Анашкин А.С., Кадыров Э.Д., Харазов В.Г. Техническое и программное обеспечение распределенных систем управления. – С. Петербург: «П-2», 2004. – 368 с., ил.

14. Руководство пользователя контроллера DL06, номер руководства D0-06USER-M-RUS часть 1. перевод ООО «ПЛКСистемы».

15. Программное обеспечение для программирования PC-DSOFT5-M часть 1. Перевод ООО «ПЛКСистемы».

16. Системы автоматизированного управления электроприводами: Учеб. пособие / Г.И. Гульков, Ю.Н. Петренко, Е.П. Раткевич О Л. Симоненкова; Под общ. ред. Ю.Н. Петренко. — Мн.: Новое знание, 2004. — 384 с.: ил.

Библиотека ГГТУ им.П.О.Сухого

СОДЕРЖАНИЕ

Библиотека ГГТУ им. П.О. Сухого

Ковалев Алексей Викторович
Литвинов Дмитрий Александрович

ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ КОНТРОЛЛЕРЫ

Практикум
по одноименной дисциплине для студентов специальности 1-53 80 01 «Автоматизация и управление технологическими процессами и производствами» дневной и заочной форм обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 31.03.20.

Рег. № 43Е.
http://www.gstu.by

Библиотека ГГТУ им.П.О.Сухого