Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Комсомольский-на-Амуре государственный технический университет»

В. И. Суздорф

ПРОЕКТИРОВАНИЕ СИСТЕМ АВТОМАТИЗАЦИИ И УПРАВЛЕНИЯ

Рекомендовано федеральным государственным бюджетным
образовательным учреждением высшего профессионального образования
«Московский государственный технологический университет
“СТАНКИН”» в качестве учебного пособия для студентов высших
учебных заведений, обучающихся по направлению подготовки
«Автоматизация технологических процессов и производств»

Комсомольск-на-Амуре
2014

УДК 681.51.001.2-52(07)
ББК 32.965-05я7
С893

Рецензенты:

Кафедра «Автоматика и системотехника» ФГБОУ ВПО «Тихоокеанский государственный университет», заведующий кафедрой доктор технических наук, профессор Чье Ен Ун;

С. В. Власьевский, доктор технических наук, профессор кафедры «Электротехника, электроника и электромеханика» ФГБОУ ВПО «Дальневосточный государственный университет путей сообщения»

Суздорф, В. И.

С893 Проектирование систем автоматизации и управления : учеб. пособие / В. И. Суздорф. – Комсомольск-на-Амуре : ФГБОУ ВПО «КнАГТУ», 2014. – 201 с.

ISBN 978-5-7765-1024-3

В учебном пособии изложены основные сведения по проектированию различных программных и аппаратных средств промышленной автоматизации технических систем, необходимые для изучения курса. Приведены базовые понятия проектирования систем управления на основе типовых программно-технических комплексов, технических средств и систем автоматизации, управления, контроля, автоматизированных технологий производства, средств и систем автоматизации и управления процессами. Особенное внимание уделено проблемам реального времени автоматизированных систем управления технологическими процессами и электроприводами.

Предназначено для студентов направления 220700 – Автоматизация технологических процессов и производств.

УДК 681.51.001.2-52(07)
ББК 32.965-05я7

ISBN 978-5-7765-1024-3

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящее пособие посвящено проектированию различных программных и аппаратных средств промышленной автоматизации технических систем. Сбор информации, ее обработка, передача при возможном участии человека в информационном процессе выполняются в реальном времени и при одновременном участии различных ресурсов управления. Совокупность процессов управления объединяется ресурсами, которые условно можно разделить на технические средства, математическое обеспечение и информационные технологии.

Технические системы как объекты организации управления имеют свои специфические особенности, которые могут определяться иерархической структурой технических систем как уровней производства. На современной стадии развития производства принято выделять ***административный*** (стратегическое планирование), ***диспетчерский*** (оперативное управление, человеко-машинный интерфейс) и ***технологический*** (оборудование, контроль технологических процессов, управление) уровни. Соответственно, и организация управления, и потоки данных, и информационные технологии, обеспечивающие сбор информации, ее обработку, передачу, хранение и другие функции, существенно отличаются на каждом уровне. Особо следует выделить проблему реального времени, характерную для двух нижних уровней иерархии технических систем.

Все перечисленные уровни представления производства существенно отличаются в своей интерпретации. Проектирование подобных систем является сегодня сложной, взаимосвязанной, но и очень востребованной современным производством задачей.

1. ЗАДАЧИ АДМИНИСТРАТИВНОГО УРОВНЯ

Обработка данных, описание информационных процессов, решение задач средствами информационных технологий являются областями исследования компьютерной лингвистики. Объекты, рассматриваемые этой дисциплиной, представлены на рис. 1.1.

Атомарной лингвистической единицей является входной символ информационной системы – терминал или литерал, именующий неделимый элемент данных. К символам относятся: буквы, цифры, знаки, управляющие символы [1].

Упорядоченные множества символов объединяются в алфавиты. Конечные упорядоченные последовательности символов, принадлежащих одному алфавиту, образуют внутренние лингвистические ***конструкции*** – слова, называемые также не терминами, или лексемами.

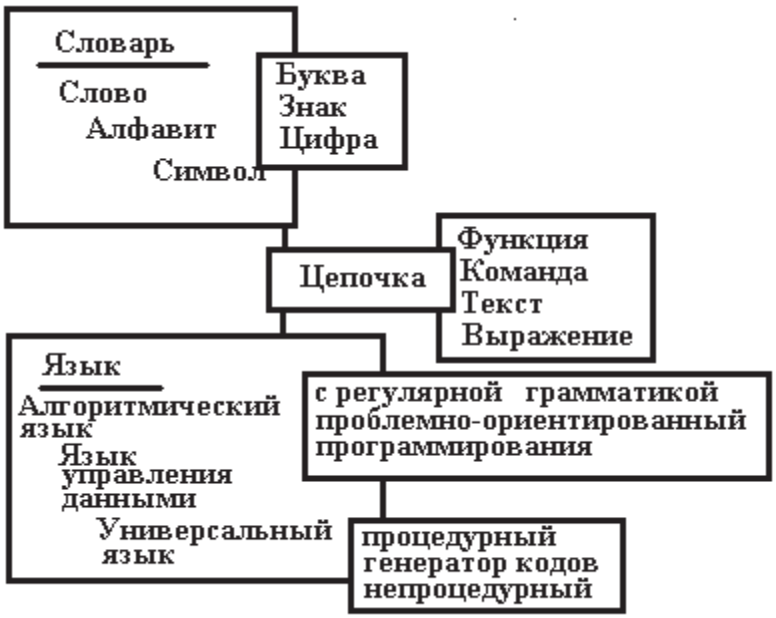


Рис. 1.1. Объекты компьютерной лингвистики

Из упорядоченного множества символов и слов строится ***словарь*** – структура данных, обеспечивающая доступ к информации по имени. В свою очередь, элементы словаря сливаются в символьные последовательности, или цепочки. Так образуются выражения (т.е. группы данных, вырабатывающие новые значения), команды (т.е. выражения, являющиеся процессом), функции, тексты (т.е. описания процессов последовательностями выражений).

Словарь вместе со всеми доступными цепочками образует ***язык*** – знаковую систему для обмена информацией. Искусственный язык, пригодный для записи алгоритмов, использования в компьютерах, публикациях, проведения формального анализа данных, называется ***алгоритмическим языком***.

Алгоритмические языки являются предметом изучения компьютерной лингвистики. К основным разделам этой дисциплины относятся:

- семиотика, исследующая свойства знаков и знаковых систем;
- лексика, описывающая словарный состав языка;
- синтаксис, изучающий способы соединения слов в текстах, типы и значения словосочетаний и выражений;
- грамматика – раздел, рассматривающий строй языка, т.е. систему языковых форм и правил словообразования на базе синтаксических конструкций;
- семантика, раскрывающая связь знака и значения, т.е. конструкций, имеющих смысл с точки зрения конкретной предметной области;

- прагматика, обрабатывающая сообщения, образованные системами предложений и текстов.

Компьютерная лингвистика оценивает языки рядом качественных показателей:

- универсальностью, т.е. единством применения в различных прикладных областях;
- сложностью синтаксических форм: стилем языковых конструкций, расшифровываемостью, однородностью, диалоговыми возможностями;
- семантической силой, выражаемой объемом данных, необходимым для описания объектов;
- логичностью, мнемоничностью, минимальностью средств, полнотой, определяющей возможность всестороннего описания, иерархичностью, т.е. способностью к последовательному описанию процессов, алгоритмичностью;
- перспективностью применения и развития: распространенностью, расширяемостью, т.е. возможностью распространения на новые языковые конструкции, реализуемостью языка.

В соответствии с этими свойствами различают языки с регулярной грамматикой, проблемно-ориентированные языки и языки программирования.

Среди языков с регулярной грамматикой выделяется группа автоматных языков: языков релейных и функциональных схем, логических формул, символических кодов. Первые основаны на соответствии языковых конструкций обозначениям схем цифровой автоматики и релейно-контакторной аппаратуры. Для работы с такими языками используются специальные пульты, с которых оператор последовательно заносит схему в память ЭВМ. Языки наглядны и удобны в коммутации и настройке промышленного оборудования и бытовой микропроцессорной техники.

Более универсальны вследствие своей абстрагируемости от конкретной практической реализации языки функциональных схем. Компактнее выглядят системы команд на языках логических формул, но далеко не каждый объект может быть запрограммирован на базе одной только логики, поэтому чаще встречаются языки символических кодов, в основе которых лежат мнемонические (пиктографические) обозначения операций и заданий.

Описание задач на языках с регулярной грамматикой имеет ряд недостатков: громоздкость раскрытия сложных алгоритмов, длительность программирования и полная зависимость от особенностей объекта.

Специфика объекта хорошо учитывается проблемно-ориентированными языками. В них меньше аппаратурных атрибутов, и они проще в усвоении. Их словарь бывает богатым, цепочки насыщенными и удобными для развития. К проблемно-ориентированным относятся языки графов и

языки схем алгоритмов, используемые в документации на лингвистические объекты, а также входные языки интерфейсов пользователя.

Языки программирования делятся на процедурные, непроцедурные и генераторы кодов. Программа на процедурном языке представляется в виде последовательности шагов (действий), описываемых элементарными операциями и группами операций, организованными в функции. В процессе реализации изменяются переменные, а функции возвращают определенные результаты.

Первым шагом на пути решения любой из процедурных задач и основным условием такого решения является алгоритмизация. ***Алгоритмом*** называется система правил, предписывающая последовательность действий, этапов решения задачи. Алгоритм характеризуется рядом свойств:

- понятность применительно к конкретному исполнителю – вычислительной системе, человеку – означает доступность для него средств реализации алгоритма;
- массовость проявляется в независимости алгоритма от входных данных, позволяющей использовать один и тот же алгоритм для решения различных однотипных задач.

Например, выражения «выйти из дома, сесть в трамвай, доехать до работы, выйти из трамвая» представляют собой алгоритм, реализуемый тысячами людей на всем земном шаре. В то же время для каждого алгоритма определено конечное множество объектов, которые допустимы в качестве входных.

1.1. Структура информационного обеспечения управленческих систем

Типами организационных информационных систем (ИС) организации могут быть следующие [2]:

- системы обработки транзакций (СОТ);
- системы получения отчетов (СПО);
- системы выработки решений (СВР);
- исполнительные информационные системы (ИИС);
- офисные информационные системы (ОИС).

Информационная система большой организации обычно включает по несколько систем каждого типа, обслуживающих, например, различные подразделения внутри предприятия. Типы систем отличаются по объектам, с которыми они работают, и по своим задачам. Это отличие показано в табл. 1.1.

Таблица 1.1

Типы информационных систем

Тип ИС	Время появления	Основные цели
Системы обработки транзакций (СОТ)	Середина 50-х гг.	Корпоративная информация. Обработка рутинных транзакций и поддержка баз данных (БД)
Системы получения отчетов (СПО)	Начало 60-х гг.	Генерация рутинных отчетов по БД
Системы выработки решений (СВР)	Начало 70-х гг.	Поддержка на основе аналитических методов и БД циклов принятия решений
Исполняющие информационные системы (ИИС)	Середина 80-х гг.	Непосредственный доступ к внутренней и внешней информации с использованием графики
Офисные информационные системы (ОИС)	Конец 70-х гг.	Обработка документов и сообщений (голос, данные, текст, видео)

1.1.1. Системы обработки транзакций

СОТ создают обширные базы данных (БД) корпоративной информации. В данном контексте под транзакцией понимают весь процесс от предложения продукции компании (предприятия) до полной оплаты всех услуг по поставке как свершившегося факта. Примеры подобных систем можно видеть в фирмах, специализирующихся на продаже компьютерной и офисной техники, запасных частей автомобилей и других, где клиент может непосредственно участвовать в выполнении транзакции.

1.1.2. Системы получения отчетов

СПО являются наиболее сложными из управленческих компонентов ИС. Основной задачей СПО является предоставление нижнему и среднему управленческому составу возможности получения печатных отчетов и осуществления запросов для облегчения управления предприятием.

СПО имеет следующие характеристики:

1) СПО обычно создаются профессиональными разработчиками ИС, а не конечными пользователями, в течение продолжительного периода времени с помощью методологий, рассчитанных на весь жизненный цикл

системы (в отличие от построения вначале простого прототипа системы и последующего его улучшения с ростом опыта пользователя). Разработка подобных систем ведется с большой осторожностью из-за большого количества системных интерфейсов с различными пользователями и базами данных.

2) СПО создаются для ситуаций, в которых информационные потребности достаточно хорошо определены, и ожидается, что они будут достаточно стабильными. Модификация таких систем, как и их создание, является достаточно сложным делом. Это ограничивает информационную гибкость СПО, но обеспечивает стабильную информационную среду.

3) СПО не поддерживают напрямую процесс выработки решений для поиска альтернативных решений проблем. Обычно информация, полученная от СПО, используется менеджером в процессе выработки решений. В СПО встроены только хорошо структурированные правила получения решений, как, например, расчет количества товаров для более экономичной их закупки.

4) СПО обычно ориентированы на получение отчетов о будущем и настоящем, а не планирование будущего.

5) СПО обычно имеют ограниченные аналитические возможности – они не используют сложные модели, а полагаются на обобщение и извлечение данных по некоторому критерию. Отчеты, полученные в результате простых обобщений и выборок данных, выводятся на бумагу (или, если объем позволяет, на экран) в предопределенной форме.

6) СПО обычно выдает отчеты по внутренней деятельности компании, а не по информации, полученной из внешних источников.

СПО может генерировать отчеты либо по БД, созданной СОТ, либо по БД, специально созданной для этой цели. Специальные БД могут создаваться по разным причинам, например, таким как избежание конфликтов и задержек при обработке транзакции, обеспечение безопасности центральных БД или экономия затрат на телекоммуникации для работы с центральной БД путем создания локальных БД.

СПО предоставляет следующие формы отчетов:

- ***Плановые*** (***периодические***) ***отчеты.*** Эти отчеты генерируются каждый день, каждую неделю, каждые две недели или с какой-либо другой периодичностью в зависимости от необходимости принятия решений. Еженедельный аналитический отчет о продажах может использоваться менеджером по продажам для оценки эффективности продаж в различных районах.

Форма и информационное содержание плановых отчетов определяются заранее. Однако очень важно определить необходимую для различных менеджеров информацию, чтобы помочь им в выработке решений и предотвратить информационную перегрузку. Обычно для этого использу-

ется концепция специализированных отчетов: менеджеры получают отчеты по той области деятельности, в которой они специализируются.

- ***Внеплановые отчеты.*** Еще одним средством избежания информационной перегрузки является использование внеплановых (исключительных) отчетов, которые генерируются только тогда, когда случается какая-либо исключительная (в соответствии с заданными критериями) ситуация, и которые содержат только информацию, касающуюся этой ситуации. Например, менеджеры, отвечающие за поставки, могут получать внеплановые отчеты в случае, если поставщики задерживают доставку грузов больше чем на неделю. Такой отчет может генерироваться вне плана при обнаружении задержки грузов от одного поставщика или периодически, если есть несколько поставщиков, задерживающих доставку грузов. Отчет может содержать: список поставщиков, задерживающих поставку, задержку и товары, заказанные у каждого из них. Внеплановые отчеты помогают избежать регулярного получения ненужной информации.
- ***Специализированные отчеты***. Предоставление менеджеру возможности делать запросы и получать соответствующие отчеты делает использование СПО более гибким и даёт конечному пользователю (каждому отдельному менеджеру) возможность запрашивать ту информацию, которая наилучшим образом удовлетворяет его информационные потребности. Осуществление подобных запросов реализуется с помощью языка запросов, предоставляемого системой управления базами данных (СУБД).

1.1.3. Системы выработки решений

Все ИС косвенно поддерживают выработку решений. СВР являются типом ИС, специально разработанным для поддержки процесса выработки решений. СВР обеспечивает диалог между пользователем, рассматривающим альтернативные решения проблемы, и системой, имеющей встроенные модели и доступ к БД. База данных СВР обычно является выборкой из главной БД предприятия. Эти системы являются интерактивными, и обычно предлагается на выбор несколько сценариев. Например, менеджер, пытающийся установить цену для нового товара, может использовать маркетинговую СВР, в которую может быть встроена электронная таблица. СВР содержит модель, учитывающую различные факторы – цену товара, себестоимость, расходы на продвижение товара на рынок – для определения планируемых прибылей (или потерь) от продаж этого товара в течение первых пяти лет. Меняя цену товара в этой модели, менеджер может сравнивать прогнозируемые результаты, а затем выбрать цену.

СВР имеет следующие характеристики:

1) СВР обычно разрабатывается с помощью менеджеров для облегчения выработки решений, которые им обычно приходится принимать. Неко-

торые из этих систем достаточно просты, и их можно создать с помощью электронных таблиц (например, Excel или Lotus 1-2-3), возможно, с использованием поставляемых с ними шаблонов, которые позволяют настроить электронную таблицу для конкретного приложения. Если для создания СПО обычно привлекаются профессиональные разработчики ИС, то созданием и модификацией СВР обычно занимаются конечные пользователи.

2) СВР создаются с учетом будущих модификаций. Сам процесс разработки и условия использования СВР влекут за собой постоянную адаптацию этих систем к изменяющимся требованиям пользователей. СВР являются очень гибкими и адаптируемыми инструментами принятия решений.

3) СВР напрямую поддерживают процесс выработки решений. Более того, класс этих систем поддерживает групповую выработку решений. В отличие от СПО, СВР способны поддерживать неструктурированные или не полностью структурированные решения, в которых некоторые зависимости между факторами и вывод заключений содержатся в модели, а некоторые предоставляются менеджером, работающим с системой. Таким образом, СВР предлагает модели для структурированных (программируемых) частей проблемы и позволяет менеджеру использовать личное суждение при принятии окончательного решения. Возможность прямого общения с системой позволяет пользователю мгновенно получать количественный анализ. Кроме того, в процессе общения с системой они также учатся принимать наилучшие решения.

4) Мощным свойством СВР является прогнозирование возможного будущего во время процесса планирования. Можно использовать два принципиальных режима анализа. В режиме «что-если» пользователь рассматривает альтернативные сценарии и их результаты. Например: «Что будет, если мы увеличим затраты на рекламу на 5 %?». В режиме поиска цели пользователь спрашивает: «Какие потребуются затраты, чтобы достигнуть определенной производительности?»

5) Главной причиной использования СВР являются аналитические возможности их моделей. Некоторые достаточно сложные системы имеют даже возможности управления моделями, которые позволяют системе самой выбирать модель, наиболее подходящую для данной проблемы, и, следовательно, освобождают пользователя от необходимости глубокого понимания сути моделей.

6) Во многих СВР большое значение придается комбинации внутренней и внешней информации.

7) Графические возможности СВР позволяют представлять данные в более сжатой форме, чем таблицы.

1.1.4. Исполняющие информационные системы

ИИС предоставляют прямую информационную поддержку для равных менеджеров. Обычно главные менеджеры располагают большим количеством неформальных источников информации, так что компьютерные ИС могут предоставить только ограниченную помощь. Однако исполнительный директор, главный и исполнительный вице-президенты и совет директоров должны иметь возможность следить за деятельностью своей компании и различных ее подразделений, оценивать деловую среду и вырабатывать стратегические направления развития компании в будущем. В частности, им требуется большое количество разнообразной внешней информации для сравнения деятельности их компании с конкурентами и исследования основных направлений в экономике государств, в которых компания ведет свою деятельность.

ИИС имеют следующие характеристики:

1) ИИС предоставляет простой и быстрый доступ к информации, отражающей ключевые факторы успеха компании и ее подразделений.

2) Привлекательные и удобные интерфейсы, такие как цветная графика и видео, позволяющие пользователю с первого взгляда определить имеющиеся тенденции. Здесь главной ценностью считается время пользователя.

3) ИИС предоставляют доступ к множеству БД (как внутренних, так и внешних) через единый интерфейс. Таким образом от пользователя скрывается тот факт, что система работает с различными БД.

4) ИИС предоставляют информацию как о существующем положении дел, так и прогнозы на будущее. При этом должна быть возможность сравнения различных перспектив.

5) ИИС должна легко адаптироваться к нуждам различных пользователей или групп пользователей (например, кабинет генерального директора или совет корпорации).

6) ИИС должна предоставлять возможность «углубляться» в данные, т.е. возможность просматривать информацию с различной степенью подробности.

ИИС являются мощным средством, помогающим осуществлять контролирующие функции менеджмента. Благодаря этим системам многие исполнительные менеджеры получили возможность расширить сферу своего контроля – другими словами, увеличить количество людей, непосредственно им подчиненных.

1.1.5. Офисные информационные системы

ОИС поддерживают обмен информацией в пределах офиса. Разработка ОИС началась в конце 1970-х гг., когда появление персонального компьютера произвело революцию в этой области. С тех пор слияние офисной технологии с компьютерной и коммуникационной технологиями привело к тому, что ОИС быстро становятся «офисом будущего». Целью ОИС является поддержка мультимедиа-связей внутри фирмы и создание шлюзов во внешний мир. Примерами поддерживаемых ОИС видов деятельности являются: обработка документов, персональное планирование времени, проведение компьютерных конференций и обмен сообщениями в различных форматах. ОИС позволяет создавать, хранить и передавать сообщения в самых разных форматах: документы, бинарные данные, голосовые сообщения, голограммы, графика и анимация; кроме того, они поддерживают проведение конференций с самыми различными уровнями присутствия участников – от обычных систем обмена сообщениями типа USENET до телеконференций.

Роль экспертных систем. Экспертные системы – это передовая технология, появившаяся в результате исследований в области искусственного интеллекта и получившая свое применение в ИС в середине 1980-х гг. Экспертные системы предлагают решение, основанное на компьютеризованном процессе, напоминающем логическое мышление. Этот процесс опирается на базу знаний об узкой области его применения, конкретные факты, по которым надо принять решение, и встроенный механизм обоснования. Экспертные системы могут включаться во все типы организационных ИС или использоваться как самостоятельные системы. В частности, они все больше интегрируются с обычными технологиями программирования в СОТ и СВР. Экспертные системы используются для выбора самых дешевых способов отправки почты, диагностики неисправностей оборудования, планирования инвестиций или компоновки сложных заказов на оборудование. Обычно эти системы не заменяют специалиста, а помогают человеку принимать решения.

Основными компонентами баз знаний являются эвристико-неформальные, субъективные элементы знаний в области применения экспертной системы. Опора на базу знаний является основной отличительной чертой этих систем. База знаний основывается и пополняется во время тестирования системы на тестовых примерах, а затем в процессе использования системы.

Таким образом, экспертные системы – это основанные на знаниях программы, которые имитируют процесс рассуждения, чтобы получить решение проблемы в своей области применения.

Простейшие системы обычно реализуются как оболочки экспертных систем – системы, основанные на знаниях, с пустыми базами знаний. Всё, что нужно сделать разработчику (инженеру знаний или конечному пользователю), это наполнить базу знаний информацией, специфичной для данной предметной области.

1.2. Базы данных

В процессе развития вычислительной техники сформировались два основных направления:

1) выполнение численных расчетов;

2) использование вычислительных средств в автоматизированных информационных системах.

Традиционная ИС – это целый пакет программ, который предназначен для надежного хранения информации в постоянной памяти, манипуляции с данными и вычислений, обеспечения удобного и «интуитивно-понятного» интерфейса. Зачастую массивы данных, обрабатываемых ИС, могут иметь большой объем, а модель данных имеет достаточно сложную, не всегда формализуемую структуру.

Второе направление связано с тем, что первые компьютеры обладали ограниченными возможностями по объему хранимой информации. Хранение информации в статических и динамических запоминающих устройствах изначально создавалось для возможного отключения электрического питания (обычно ориентировались на 2 выходных дня, т.е. 48 ч). Оперативная память сохраняет информацию только при работающем устройстве. Накопители данных создавались на основе магнитных карт, лент, магнитных барабанов. В целом первые хранители данных обладали малым быстродействием, малой емкостью (от сотен килобайт до единиц мегабайт) и обладали значительными габаритами.

Развитие систем управления данными на внешних носителях началось с создания магнитных дисков. Появилась возможность формировать архивы именованных данных и структурировать их во внешней памяти.

Файловые системы. Именованная часть внешней памяти, где можно размещать и откуда можно считывать данные или прикладную программу, может рассматриваться как атомарная форма данных – файл. От конкретной системы управления файлами зависит распределение внешней памяти, отображение имен файлов и обеспечение доступа к данным.

Структуры файлов. Современные ЭВМ оперируют с внешними устройствами памяти, которыми являются магнитные диски с подвижными головками для хранения файлов. Магнитные диски [9] представляют собой пакеты магнитных пластин, между которыми на одном рычаге двигается пакет магнитных головок. Шаг движения пакета головок является дис-

кретным, и каждому положению пакета головок логически соответствует цилиндр магнитного диска. На каждой поверхности цилиндр размечает дорожку так, что каждая поверхность содержит число дорожек, равное числу цилиндров. При разметке магнитного диска каждая дорожка размечается на одно и то же количество блоков таким образом, что в каждый блок можно записать по максимуму одно и то же число байтов. Таким образом, для произведения обмена с магнитным диском на уровне аппаратуры нужно указать номер цилиндра, номер поверхности, номер блока на соответствующей дорожке и число байтов, которое нужно записать или прочитать от начала этого блока.

Именование файлов. Каталоги – это специальная структура для организации хранения данных во внешней памяти при их многоуровневом именовании. Каталоги содержат имена файлов. В свою очередь, имя файла состоит из списка имен каталогов, а также имени файла в каталоге.

Защита файлов [9]. Системы управления файлами должны обеспечивать авторизацию доступа к файлам. В общем виде подход состоит в том, что по отношению к каждому зарегистрированному пользователю данной вычислительной системы для каждого существующего файла указываются действия, которые разрешены или запрещены данному пользователю, поэтому в большинстве современных систем управления файлами применяется подход к защите файлов (ОС UNIX). В этой системе каждому зарегистрированному пользователю соответствует пара целочисленных идентификаторов: идентификатор группы, к которой относится этот пользователь, и его собственный идентификатор в группе. Соответственно, при каждом файле хранится полный идентификатор пользователя, который создал этот файл, и отмечается, какие действия с файлом может производить он сам, какие действия с файлом доступны для других пользователей той же группы и что могут делать с файлом пользователи других групп. Эта информация очень компактна, при проверке требуется небольшое количество действий, и этот способ контроля доступа удовлетворителен в большинстве случаев.

Режим многопользовательского доступа. При одновременной попытке доступа к данным разными пользователями необходима синхронизация их работы.

Области применения файлов. Прежде всего, это хранение текстовых данных. Эти файлы образуются с помощью различных текстовых редакторов. Структура текстовых файлов представляет собой последовательность записей, содержащих строки текста, либо последовательность байтов.

Обычно файловые системы [9] обеспечивают хранение слабо структурированной информации, оставляя дальнейшую структуризацию прикладным программам. В перечисленных выше случаях использования

файлов это даже хорошо, потому что при разработке любой новой прикладной системы, опираясь на простые, стандартные и сравнительно дешевые средства файловой системы, пользователь может реализовать те структуры хранения, которые наиболее естественно соответствуют специфике данной прикладной области.

Потребности информационных систем [9]. Многие ИС главным образом ориентированы на хранение, выбор и модификацию постоянно существующей информации. Структура информации зачастую очень сложна, и хотя структуры данных различны в разных ИС, между ними часто бывает много общего. Но поскольку ИС требуют сложных структур данных, эти дополнительные индивидуальные средства управления данными являлись существенной частью информационных систем и практически повторялись от одной системы к другой. Стремление выделить и обобщить общую часть ИС, ответственную за управление сложно структурированными данными, явилось побудительной причиной создания систем управления базами данных (СУБД). Невозможно обойтись общей библиотекой программ, реализующей над стандартной базовой файловой системой более сложные методы хранения данных.

Понятие ***согласованности данных*** является ключевым понятием баз данных. Фактически, если информационная система поддерживает согласованное хранение информации в нескольких файлах, то можно говорить о том, что она поддерживает базу данных.

Традиционных возможностей файловых систем оказывается недостаточно для построения даже простых ИС. Если прикладная ИС опирается на некоторую систему управления данными, обладающую определенными свойствами, то эта система управления данными является *системой управления базами данных* (*СУБД*).

1.2.1. Системы управления базами данных

В порядке перечисления:

1) ***Управление данными во внешней памяти.***

2) ***Управление буферами оперативной памяти.***

3) ***Управление транзакциями.*** Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется и СУБД фиксирует изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД.

С управлением транзакциями в многопользовательской СУБД связаны важные понятия [9] ***сериализации транзакций*** и ***сериального плана выполнения смеси транзакций***. Под сериализацией параллельно выполняющихся транзакций понимается такой порядок планирования их работы,

при котором суммарный эффект смеси транзакций эквивалентен эффекту их некоторого последовательного выполнения. Сериальный план выполнения смеси транзакций – это такой план, который приводит к сериализации транзакций.

Существует несколько базовых алгоритмов сериализации транзакций. В централизованных СУБД наиболее распространены алгоритмы, основанные на синхронизационных захватах объектов БД. При использовании любого алгоритма сериализации возможны ситуации конфликтов между двумя или более транзакциями по доступу к объектам БД. В этом случае для поддержания сериализации необходимо выполнить откат (ликвидировать все изменения, произведенные в БД) одной или более транзакций.

4) ***Журнализация***. Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя.

Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД.

5) ***Поддержка языков БД.*** Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language), а также QBE (Query by Example).

1.2.2. Иерархические и сетевые системы управления базами данных

Иерархические структуры данных. Иерархическая БД состоит из упорядоченного набора деревьев, напоминающего (для тех, кто еще помнит) структуру хранения файлов в MS DOS.

Тип дерева состоит из одного «корневого» типа записи и упорядоченного набора из нуля или более типов поддеревьев (каждое из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор типов записи. Пример типа дерева изображен на рис. 1.2.

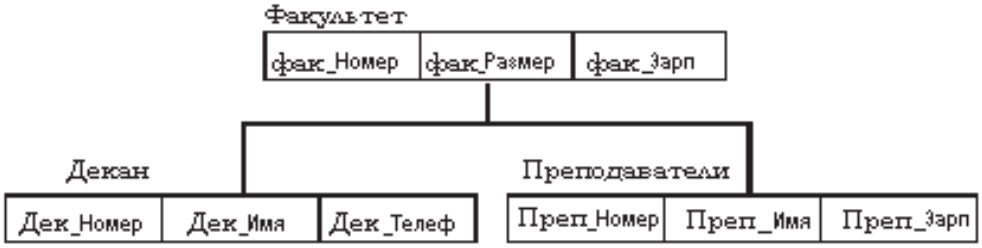


Рис. 1.2. Схема иерархической БД

Здесь Факультет является предком для Декан и Преподаватели, а Декан и Преподаватели – потомки Факультет. Между типами записи поддерживаются связи.

База данных с такой схемой могла бы выглядеть так, как на рис. 1.3.

Все экземпляры данного типа потомка с общим экземпляром типа предка называются ***близнецами***.

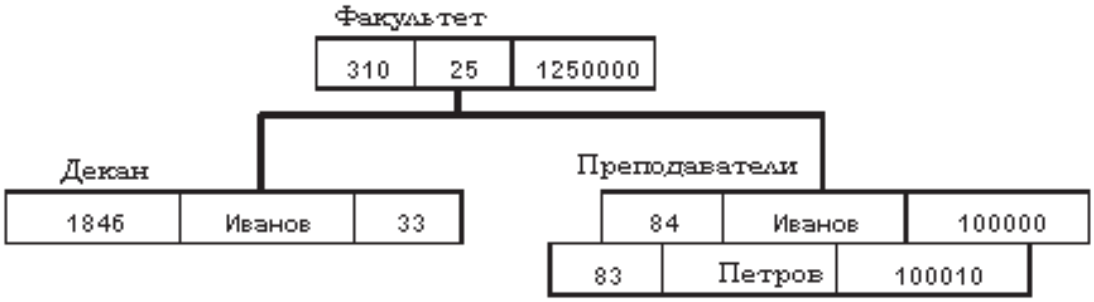


Рис. 1.3. Модифицированная система иерархической БД

Манипулирование данными. Примерами типичных операторов манипулирования иерархически организованными данными могут быть следующие [9]:

- найти указанное дерево БД;
- перейти от одного дерева к другому;
- перейти от одной записи к другой внутри дерева;
- перейти от одной записи к другой в порядке обхода иерархии;
- вставить новую запись в указанную позицию;
- удалить текущую запись.

Ограничения целостности. Автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

В иерархических системах поддерживается форма представлений БД на основе ограничения иерархии. Примером представления приведенной выше БД может быть иерархия на рис. 1.4.

Рис. 1.4. БД на основе ограничения иерархии

Сетевые структуры данных. Сетевой подход к организации данных является расширением иерархического [9]. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков.

Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно, из набора экземпляров каждого типа из заданного в схеме БД набора типов записи и набора экземпляров каждого типа из заданного набора типов связи.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора экземпляров типа записи потомка.

Простой пример сетевой схемы БД изображен на рис. 1.5.

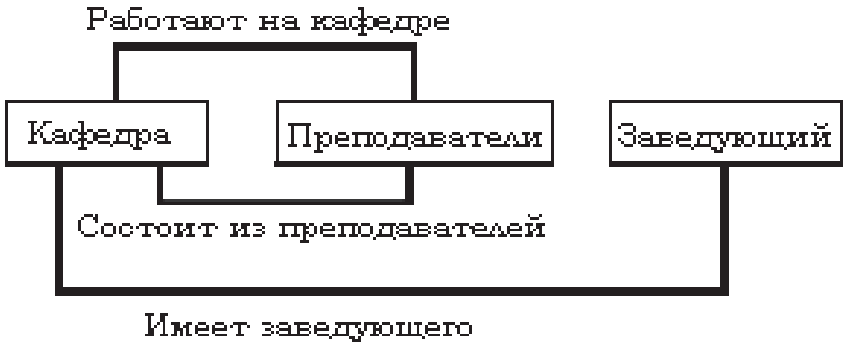


Рис. 1.5. Простая сетевая схема БД

Достоинства СУБД:

- развитые средства управления данными во внешней памяти на низком уровне;
- возможность построения вручную эффективных прикладных систем;
- возможность экономии памяти за счет разделения подобъектов (в сетевых системах).

Недостатки СУБД:

- слишком сложно пользоваться;
- фактически необходимы знания о физической организации;
- прикладные системы зависят от этой организации;
- их логика перегружена деталями организации доступа к БД.

1.2.3. Реляционная модель организации данных

Базовыми понятиями этой модели организации данных являются: тип данных, домен, атрибут, кортеж, первичный ключ и отношение.

Необходимо показать смысл этих понятий, что можно сделать на примере отношения ПРЕПОДАВАТЕЛИ, где размещена информация о преподавателях некоторого вуза (рис. 1.6).

Понятие ***тип данных*** в реляционной модели данных соответствует понятию типа данных в языках программирования. В реляционных БД [9] допускается хранение символьных, числовых данных, битовых строк, спе-

циализированных числовых данных (таких как «деньги»), а также специальных «темпоральных» данных (дата, время, временной интервал).

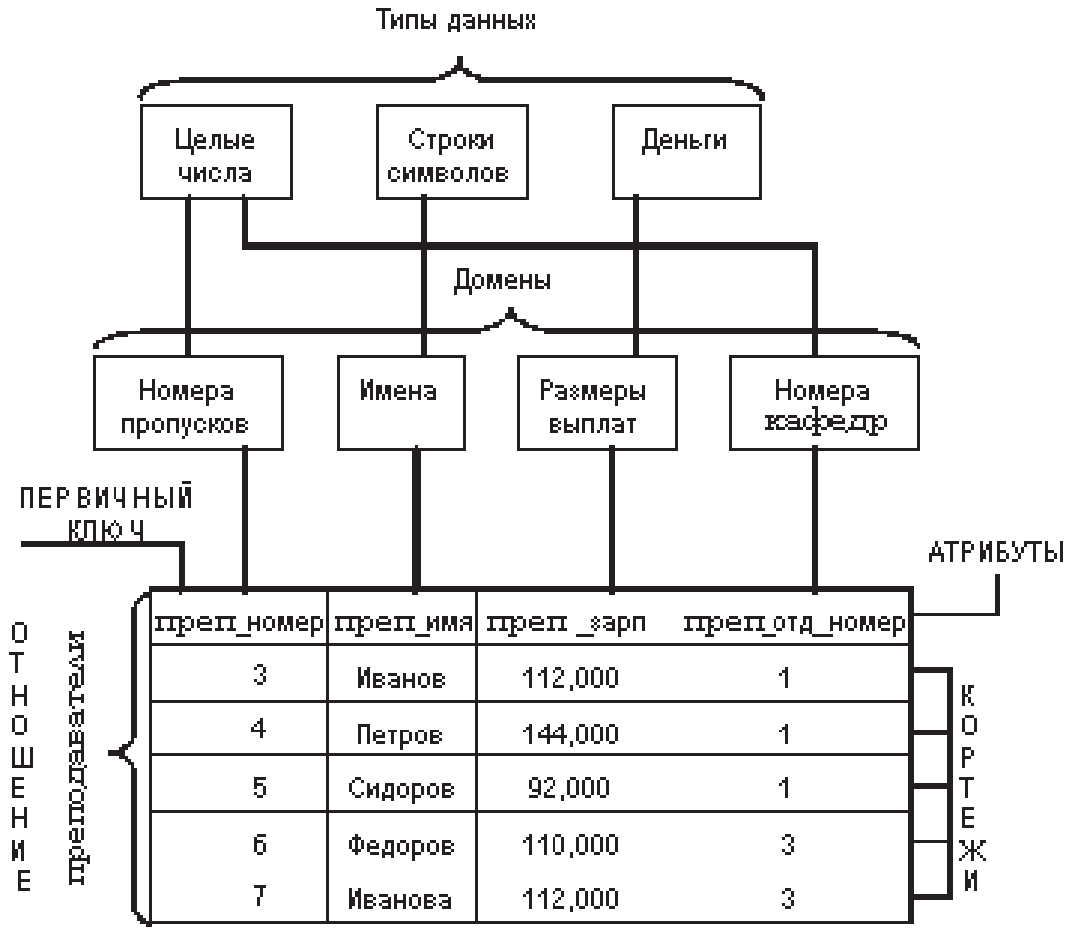


Рис. 1.6. Базовые понятия реляционных БД

Домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат «истина», то элемент данных является элементом домена.

Понятие домена трактуется как допустимое множество значений данного типа.

Схема отношения – это именованное множество пар {имя атрибута, имя домена (или типа, если понятие домена не поддерживается)}. Степень или «арность» схемы отношения – мощность этого множества. Степень отношения ПРЕПОДАВАТЕЛИ равна четырем, т.е. оно является 4-арным.

Схема БД (в структурном смысле) – это набор именованных схем отношений.

Кортеж, соответствующий данной схеме отношения, – это множество пар {имя атрибута, значение}, которое содержит одно вхождение

каждого имени атрибута, принадлежащего схеме отношения. «Значение» является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или «арность» кортежа, т.е. число элементов в нем, совпадает с «арностью» соответствующей схемы отношения. Попросту говоря, кортеж – это набор именованных значений заданного типа.

Отношение – это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят «отношение-схема» и «отношение-экземпляр», иногда схему отношения называют *заголовком* отношения, а отношение как набор кортежей – *телом* отношения.

Обычным житейским представлением отношения является таблица, заголовком которой является схема отношения, а строками – кортежи отношения-экземпляра; в этом случае имена атрибутов именуют столбцы этой таблицы, поэтому иногда говорят «столбец таблицы», имея в виду «атрибут отношения».

Реляционная БД – это набор отношений, имена которых совпадают с именами схем отношений в схеме БД.

Свойства отношений [9]:

1) ***Отсутствие кортежей-дубликатов***. То, что отношения не содержат кортежей-дубликатов, следует из определения отношения как множества кортежей. В классической теории множеств по определению каждое множество состоит из различных элементов.

Из этого свойства вытекает наличие у каждого отношения так называемого первичного ключа – набора атрибутов, значения которых однозначно определяют кортеж отношения. Для каждого отношения по крайней мере полный набор его атрибутов обладает этим свойством. Понятие *первичного ключа* является исключительно важным в связи с понятием целостности баз данных.

2) ***Отсутствие упорядоченности кортежей***. Свойство отсутствия упорядоченности кортежей отношения также является следствием определения отношения-экземпляра как множества кортежей. Отсутствие требования к поддержанию порядка на множестве кортежей отношения дает дополнительную гибкость СУБД при хранении баз данных во внешней памяти и при выполнении запросов к базе данных.

3) ***Отсутствие упорядоченности атрибутов***. Атрибуты отношений не упорядочены, поскольку по определению схема отношения есть множество пар {имя атрибута, имя домена}. Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута. Это свойство теоретически позволяет, например, модифицировать схемы существующих отношений не только путем добавления новых атрибутов, но и путем удаления существующих атрибутов.

4) ***Атомарность значений атрибутов***. Значения всех атрибутов являются атомарными. В реляционных БД допускаются только нормализованные отношения или отношения, представленные в первой нормальной форме. Пример ненормализованного отношения представлен на рис. 1.7.

Рис. 1.7. Пример ненормализованного отношения

Можно сказать, что здесь бинарное отношение, значениями атрибута КАФЕДРА которого являются отношения. Исходное отношение ПРЕПОДАВАТЕЛИ является нормализованным вариантом отношения КАФЕДРА:

ПРЕП_НОМЕР	ПРЕП_ИМЯ	ПРЕП_ЗАРП	ПРЕП_КАФ_НОМЕР
11	Иванов	12,000	3
11	Петров	11,500	4
11	Сидоров	12,000	5
12	Федоров	10,000	6
13	Иванова	12,000	2

Нормализованные отношения составляют основу классического реляционного подхода к организации баз данных. Они обладают некоторыми ограничениями (не любую информацию удобно представлять в виде плоских таблиц), но существенно упрощают манипулирование данными. Возможны два идентичных оператора занесения кортежа:

1) зачислить преподавателя Сидорова (пропуск номер 3000, зарплата 12,000) на кафедру 11;

2) зачислить преподавателя Сидорова (пропуск номер 3000, зарплата 12,000) на кафедру 12.

Если информация о преподавателях представлена в виде отношения ПРЕПОДАВАТЕЛИ, то оба оператора будут выполняться одинаково (вставить кортеж в отношение ПРЕПОДАВАТЕЛИ). Если же работать с ненормализованным отношением КАФЕДРА, то первый оператор выразится в занесение кортежа, а второй – в добавление информации о Сидорове в множественное значение атрибута КАФЕДРА кортежа с первичным ключом 11.

Реляционная модель данных [9].

Модель данных описывает некоторый набор понятий и признаков, которыми должны обладать все СУБД и управляемые ими БД, если они основываются на этой модели. Наличие модели данных позволяет сравнивать конкретные реализации, используя один общий язык.

Согласно К. Дейту, реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода: *структурной*, *манипуляционной* и *целостной*.

В структурной части модели фиксируется, что единственной структурой данных, используемой в реляционных БД, является нормализованное *n*-арное отношение.

Целостность сущности и ссылок.

В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД. Первое требование называется *требованием целостности сущностей*. Объекту или сущности реального мира в реляционных БД соответствуют кортежи отношений. Конкретно требование состоит в том, чтобы любой кортеж любого отношения был отличим от любого другого кортежа этого отношения, т.е. другими словами, любое отношение должно обладать первичным ключом.

Второе требование называется *требованием целостности по ссылкам* и является несколько более сложным. При соблюдении нормализованности отношений сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений.

Требование целостности по ссылкам, или требование внешнего ключа, состоит в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать).

Ограничения целостности сущности и по ссылкам должны поддерживаться СУБД. Для соблюдения целостности сущности достаточно гарантировать отсутствие в любом отношении кортежей с одним и тем же значением первичного ключа. С целостностью по ссылкам дела обстоят несколько более сложно.

1.2.4. Проектирование реляционных БД

Проектирование реляционных БД с использованием нормализации [9]. Процесс проектирования представляет собой процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает свойствами лучшими, чем предыдущая.

Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений. Примером набора ограничений является ограничение ***первой нормальной формы*** – значения всех атрибутов отношения атомарны. Поскольку требование первой нормальной формы является базовым требованием классической реляционной модели данных, то считается, что исходный набор отношений уже соответствует этому требованию.

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1***NF***);
- вторая нормальная форма (2***NF***);
- третья нормальная форма (3***NF***);
- нормальная форма Бойса-Кодда (***BCNF***);
- четвертая нормальная форма (4***NF***);
- пятая нормальная форма, или нормальная форма проекции-соединения (5***NF***, или ***PJ***/***NF***).

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле лучше предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных свойств сохраняются.

В основе процесса проектирования лежит метод нормализации, декомпозиция отношения, находящегося в предыдущей нормальной форме, в два или более отношения, удовлетворяющих требованиям следующей нормальной формы.

ER-диаграммы, семантическое моделирование данных.

Широкое распространение реляционных СУБД и их использование в самых разнообразных приложениях показывает, что реляционная модель данных достаточна для моделирования предметных областей. Однако проектирование реляционной БД в терминах отношений на основе кратко рассмотренного нами механизма нормализации часто представляет собой очень сложный и неудобный для проектировщика процесс.

При этом проявляется ограниченность реляционной модели данных в следующих аспектах:

- Модель не предоставляет достаточных средств для представления смысла данных. Семантика реальной предметной области должна независи-

мым от модели способом представляться в голове проектировщика. В частности, это относится к проблеме представления ограничений целостности.

- Для многих приложений трудно моделировать предметную область на основе плоских таблиц. В ряде случаев на самой начальной стадии проектирования проектировщику приходится производить насилие над собой, чтобы описать предметную область в виде одной (возможно, даже ненормализованной) таблицы.
- Хотя весь процесс проектирования происходит на основе учета зависимостей, реляционная модель не предоставляет каких-либо средств для представления этих зависимостей.
- Несмотря на то что процесс проектирования начинается с выделения некоторых существенных для приложения объектов предметной области («сущностей») и выявления связей между этими сущностями, реляционная модель данных не предлагает какого-либо аппарата для разделения сущностей и связей.

1.2.5. Семантические модели данных

На практике семантическое моделирование используется на первой стадии проектирования БД. При этом в терминах семантической модели производится концептуальная схема БД, которая затем вручную преобразуется к реляционной (или какой-либо другой) схеме. Этот процесс выполняется под управлением методик, в которых достаточно четко оговорены все этапы такого преобразования.

Реже реализуется автоматизированная компиляция концептуальной схемы в реляционную. При этом известны два подхода:

1) на основе явного представления концептуальной схемы как исходной информации для компилятора;

2) построения интегрированных систем проектирования с автоматизированным созданием концептуальной схемы на основе интервью с экспертами предметной области.

Наиболее близко ко второму подходу находятся современные объектно-ориентированные СУБД, модели данных которых по многим параметрам близки к семантическим моделям.

Основные понятия модели «Сущность-Связи».

Модель «Сущность-Связи» (Entity-Relationship) часто называют кратко ER-моделью. На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных). Модель была предложена П. Ченом (Chen) в 1976 г. Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концепту-

альных схем баз данных ER-модели получили широкое распространение в системах CASE, поддерживающих автоматизированное проектирование реляционных БД. Среди множества разновидностей ER-моделей одна из наиболее развитых применяется в системе CASE фирмы ORACLE.

Основными понятиями ER-модели являются сущность, связь и атрибут.

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. В диаграммах ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа. Для большей выразительности и лучшего понимания имя сущности может сопровождаться примерами конкретных объектов этого типа.

На рис. 1.8 изображена сущность АЭРОПОРТ с примерными объектами Домодедово и Хитроу.

Рис. 1.8. Сущность АЭРОПОРТ

Каждый экземпляр сущности должен быть отличим от любого другого экземпляра той же сущности (это требование в некотором роде аналогично требованию отсутствия кортежей-дубликатов в реляционных таблицах).

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указываются имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи).

Связь представляется в виде линии, связывающей две сущности или ведущей от сущности к ней же самой. При этом в месте «стыковки» связи с сущностью используются трехточечный вход в прямоугольник сущности, если для этой сущности в связи могут использоваться много (many) экземпляров сущности, и одноточечный вход, если в связи может участвовать только один экземпляр сущности. Обязательный конец связи изображается сплошной линией, а необязательный – прерывистой.

Как и сущность, связь – это типовое понятие, все экземпляры обеих пар связываемых сущностей подчиняются правилам связывания.

Рис. 1.9. Связь

В изображенном на рис. 19. примере связь между сущностями БИЛЕТ и ПАССАЖИР связывает би-

леты и пассажиров. Конец сущности с именем «для» позволяет связывать с одним пассажиром более одного билета, причем каждый билет должен быть связан с каким-либо пассажиром. Конец сущности с именем «имеет» означает, что каждый билет может принадлежать только одному пассажиру, причем пассажир не обязан иметь хотя бы один билет.

Лаконичной устной трактовкой изображенной диаграммы является следующая:

- каждый БИЛЕТ предназначен для одного и только одного ПАССАЖИРА;
- каждый ПАССАЖИР может иметь один или более БИЛЕТОВ.

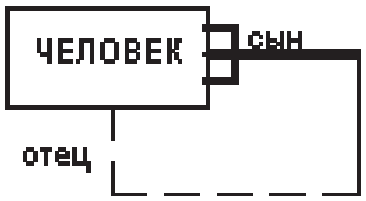


Рис. 1.10.
Рекурсивная связь

На рис. 1.10 изображена рекурсивная связь, связывающая сущность ЧЕЛОВЕК с ней же самой. Конец связи с именем «сын» определяет тот факт, что у одного отца может быть более чем один сын. Конец связи с именем «отец» означает, что не у каждого человека могут быть сыновья.

Лаконичной устной трактовкой изображенной диаграммы является следующая:

- каждый ЧЕЛОВЕК является сыном одного и только одного ЧЕЛОВЕКА;
- каждый ЧЕЛОВЕК может являться отцом для одного или более ЛЮДЕЙ («ЧЕЛОВЕК»).

Атрибутом сущности является любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Имена атрибутов заносятся в прямоугольник, изображающий сущность, под именем сущности и изображаются малыми буквами, возможно, с примерами.

Пример:

Уникальным идентификатором сущности является атрибут, комбинация атрибутов, комбинация связей или комбинация связей и атрибутов, уникально отличающая любой экземпляр сущности от других экземпляров сущности того же типа.

Нормальные формы ER-схем.

Как и в реляционных схемах баз данных, в ER-схемах вводится понятие нормальных форм, причем их смысл очень близко соответствует смыслу реляционных нормальных форм. Формулировки нормальных форм ER-схем делают более понятным смысл нормализации реляционных схем. Ниже приведены только очень краткие и неформальные определения трех первых нормальных форм.

В *первой нормальной форме* ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов, т.е. производится выявление неявных сущностей, «замаскированных» под атрибуты.

Во *второй нормальной форме* устраняются атрибуты, зависящие только от части уникального идентификатора. Эта часть уникального идентификатора определяет отдельную сущность.

В *третьей нормальной форме* устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Эти атрибуты являются основой отдельной сущности.

Получение реляционной схемы из ER-схемы.

Шаг 1. Каждая простая сущность превращается в таблицу. Простая сущность – это сущность, не являющаяся подтипом и не имеющая подтипов. Имя сущности становится именем таблицы.

Шаг 2. Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения, а столбцы, соответствующие обязательным атрибутам, – не могут.

Шаг 3. Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы. Если имеется несколько возможных уникальных идентификаторов, то выбирается наиболее используемый. Если в состав уникального идентификатора входят связи, то к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно). Для именования этих столбцов используются имена концов связей и/или имена сущностей.

Шаг 4. Связи «многие-к-одному» и «один-к-одному» становятся внешними ключами, т.е. делается копия уникального идентификатора с конца связи «один» и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения, а обязательные связи – столбцам, не допускающим неопределенные значения.

Шаг 5. Индексы создаются для первичного ключа (уникальный индекс), внешних ключей и тех атрибутов, на которых предполагается в основном базировать запросы.

Шаг 6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа:

1) все подтипы в одной таблице;

2) для каждого подтипа – отдельная таблица.

При применении первого способа таблица создается для наиболее внешнего супертипа, а для подтипов могут создаваться представления. В таблицу добавляется по крайней мере один столбец, содержащий код ТИПА; он становится частью первичного ключа.

При использовании второго способа для каждого подтипа первого уровня (для более нижних – представления) супертип воссоздается с

помощью представления UNION (из всех таблиц подтипов выбираются общие столбцы – столбцы супертипа).

Шаг 7. Имеются два способа работы при наличии исключающих связей:

1) общий домен;

2) явные внешние ключи.

Если остающиеся внешние ключи все в одном домене, т.е. имеют общий формат (первый способ), то создаются два столбца: идентификатор связи и идентификатор сущности. Столбец идентификатора связи используется для различения связей, покрываемых дугой исключения. Столбец идентификатора сущности используется для хранения значений уникального идентификатора сущности на дальнем конце соответствующей связи.

Если результирующие внешние ключи не относятся к одному домену, то для каждой связи, покрываемой дугой исключения, создаются явные столбцы внешних ключей; все эти столбцы могут содержать неопределенные значения.

1.2.6. Архитектура «клиент-сервер»

Распространение архитектуры «клиент-сервер» стало возможным благодаря развитию и широкому внедрению в практику концепции ***открытых систем***.

Технологии и стандарты открытых систем обеспечивают реальную и проверенную практикой возможность производства системных и прикладных программных средств со свойствами мобильности (portability) и интероперабельности (interoperability). Свойство мобильности означает сравнительную простоту переноса программной системы в широком спектре аппаратно-программных средств, соответствующих стандартам. Интероперабельность означает упрощение комплексирования новых программных систем на основе использования готовых компонентов со стандартными интерфейсами.

Использование подхода открытых систем выгодно и производителям, и пользователям. Прежде всего, открытые системы обеспечивают естественное решение проблемы поколений аппаратных и программных средств. Производители таких средств не вынуждены решать все проблемы заново; они могут, по крайней мере, временно продолжать комплексировать системы, используя существующие компоненты.

Преимуществом для пользователей является то, что они могут постепенно заменять компоненты системы на более совершенные, не утрачивая работоспособности системы. В частности, в этом кроется решение проблемы постепенного наращивания вычислительных, информационных и других мощностей компьютерной системы.

Клиенты и серверы локальных сетей.

В основе широкого распространения локальных сетей компьютеров лежит известная идея разделения ресурсов. Высокая пропускная способность локальных сетей обеспечивает эффективный доступ из одного узла локальной сети к ресурсам, находящимся в других узлах.

Развитие этой идеи приводит к функциональному выделению компонентов сети: разумно иметь не только доступ к ресурсам удаленного компьютера, но также получать от этого компьютера некоторый сервис, который специфичен для ресурсов данного рода и программные средства для обеспечения которого нецелесообразно дублировать в нескольких узлах. Так, различают *рабочие станции* и *серверы локальной сети*.

Рабочая станция предназначена для непосредственной работы пользователя или категории пользователей и обладает ресурсами, соответствующими локальным потребностям данного пользователя. Специфическими особенностями рабочей станции могут быть: объем оперативной памяти (далеко не все категории пользователей нуждаются в наличии большой оперативной памяти), наличие и объем дисковой памяти (достаточно популярны бездисковые рабочие станции, использующие внешнюю память дискового сервера), характеристики процессора и монитора (некоторым пользователям нужен мощный процессор, других в большей степени интересует разрешающая способность монитора, для третьих обязательно требуются средства убыстрения графики и т.д.). При необходимости можно использовать ресурсы и/или услуги, предоставляемые сервером.

Сервер локальной сети должен обладать ресурсами, соответствующими его функциональному назначению и потребностям сети. В связи с ориентацией на подход открытых систем правильнее говорить о логических серверах (имея в виду набор ресурсов и программных средств, обеспечивающих услуги над этими ресурсами), которые располагаются не обязательно на разных компьютерах. Особенностью логического сервера в открытой системе является то, что если по соображениям эффективности сервер целесообразно переместить на отдельный компьютер, то это можно сделать без потребности в какой-либо переделке как его самого, так и использующих его прикладных программ.

Примерами серверов могут служить:

- сервер телекоммуникаций, обеспечивающий услуги по связи данной локальной сети с внешним миром;
- вычислительный сервер, дающий возможность производить вычисления, которые невозможно выполнить на рабочих станциях;
- дисковый сервер, обладающий расширенными ресурсами внешней памяти и предоставляющий их в использование рабочим станциями и, возможно, другим серверам;

- файловый сервер, поддерживающий общее хранилище файлов для всех рабочих станций;
- сервер баз данных – фактически обычная СУБД, принимающая запросы по локальной сети и возвращающая результаты.

Сервер локальной сети предоставляет ресурсы (услуги) рабочим станциям и/или другим серверам.

Принято называть клиентом локальной сети, запрашивающим услуги у некоторого сервера и сервером, – компонент локальной сети, оказывающий услуги некоторым клиентам.

Системная архитектура «клиент-сервер».

В общем случае, чтобы прикладная программа, выполняющаяся на рабочей станции, могла запросить услугу у некоторого сервера, как минимум требуется некоторый интерфейс, поддерживающий такого рода взаимодействие.

Система разбивается на две части – *клиентскую* и *серверную*, которые могут выполняться в разных узлах сети. Прикладная программа или конечный пользователь взаимодействуют с клиентской частью системы, которая в простейшем случае обеспечивает просто надсетевой интерфейс. Клиентская часть системы при потребности обращается по сети к серверной части. В развитых системах сетевое обращение к серверной части может и не понадобиться, если система может предугадывать потребности пользователя и в клиентской части содержатся данные, способные удовлетворить его следующий запрос.

Интерфейс серверной части определен и фиксирован, поэтому возможно создание новых клиентских частей существующей системы (пример интероперабельности на системном уровне).

Основной проблемой систем, основанных на архитектуре «клиент-сервер», является то, что в соответствии с концепцией открытых систем от них требуется мобильность в как можно более широком классе аппаратно-программных решений открытых систем. Даже если ограничиться UNIX-ориентированными локальными сетями, в разных сетях применяется разная аппаратура и протоколы связи. Попытки создания систем, поддерживающих все возможные протоколы, приводит к их перегрузке сетевыми деталями в ущерб функциональности.

Серверы баз данных.

Термин «сервер баз данных» обычно используют для обозначения всей СУБД, основанной на архитектуре «клиент-сервер», включая и серверную, и клиентскую части. Такие системы предназначены для хранения и обеспечения доступа к базам данных.

Хотя обычно одна БД целиком хранится в одном узле сети и поддерживается одним сервером, серверы баз данных представляют собой простое

и дешевое приближение к распределенным базам данных, поскольку общая БД доступна для всех пользователей локальной сети.

Распределенные БД.

Основная задача систем управления распределенными БД состоит в обеспечении средства интеграции локальных БД, располагающихся в некоторых узлах вычислительной сети с тем, чтобы пользователь, работающий в любом узле сети, имел доступ ко всем этим базам данных как к единой БД.

При этом должны обеспечиваться:

- простота использования системы;
- возможности автономного функционирования при нарушениях связности сети или при административных потребностях;
- высокая степень эффективности.

Информационные хранилища.

Информационные хранилища (DW) выполнены в архитектуре «клиент-сервер» и радикально упрощают поиск данных в гигантских БД. На их основе создаются программы принятия решений.

DW – интегрированная, предметно-ориентированная и не разрушаемая БД, которая выполняет функции информационных советников в системах принятия решений.

DW реализовали две функции: быстроту доступа к данным и устранение их несогласованности. Содержание DW доступно только для чтения, поэтому оно вторично, а первичные данные накапливаются и структурируются в обычных СУБД.

Разделение данных по целевому признаку выдвигает особые требования к их администрированию.

Данные в предметно-ориентированных БД хранятся в виде логических групп, чтобы упростить пользователю восприятие информации. Как следствие появляется избыточная информация. Добавление информации в DW осуществляется через первичные БД, в результате чего появилась возможность анализировать не только текущие показатели, но и их историю.

Проектирование DW ведется по принципу от сервера к ПК. Основное внимание уделяется источникам информации, а не самим данным.

Этапы проектирования DW:

1) Анализ существующих в организациях БД. Нужно уловить смысл данных, определить их физические характеристики. Нужно помнить, что изменение структуры DW не должно влиять на БД нижнего уровня; для большей скорости поиска и обработки данных нужно использовать избыточность.

2) Анализ пожеланий пользователя к интерфейсу системы. Нужно интерпретировать пользовательские интерфейсы нижнего уровня и предвидеть пожелания пользователей в будущем.

3) Квантование времени. Отсюда необходимо различать первичные и производные данные.

4) Соглашения по именованию данных конфликтов.

Администрирование DW.

Простейшее: экспорт из СУБД нижнего уровня в DW. Для надежности администрирование должно осуществляться автоматически, но это усложнит систему и, как правило, снизит ее надежность, поэтому в DW введены метаданные (данные о данных), которые определяют источник, приемник и алгоритм трансформации данных при их переносе из источника в приемник.

Извлечение данных.

Наполнение и обслуживание DW состоит из трех этапов: *экстракция*, *трансформация* и *загрузка данных*.

Экстракция – идентификация базовой СУБД, где содержатся первичные данные. Она осуществляется программами разработчика, которые посредством метаданных извлекают необходимую информацию.

После загрузки первоначальной информации в DW процедуры экстракции должны считывать только новую или обновленную информацию, что позволяет существенно снизить требования к вычислительным ресурсам системы.

Данные в DW – точная копия информации в исходной СУБД. Следовательно, они в DW импортируются напрямую. Если обнаружены отличия или оказалось, что данные в DW есть проекция нескольких таблиц в СУБД, то необходима ***трансформация данных***. Операция трансформации данных содержит три алгоритма:

1) суммирование – агрегирование данных в виде дневных, месячных, годовых итогов;

2) консолидацию – объединение нескольких таблиц из разных источников, но имеющих одинаковые ключи и атрибуты;

3) коррекцию – разрешение конфликтов.

Загрузка данных идет после процедур считывания и трансформации первичных данных в нужные форматы. Этот процесс синхронизирован временными интерфейсами или внешними событиями. Периодически необходимо производить архивацию и чистку.

Требования к DW:

1) При создании DW организация данных должна отражать потребность пользователя.

2) Для извлечения, трансформации и загрузки данных в DW необходимо использовать утилиты конвертирования, поставляемые с СУБД.

3) Нужны информационные каталоги для того, чтобы администраторы и пользователи могли отделить источники данных, методы доступа, дать последние модификации.

4) При работе с особенно запутанной информацией, для доступа к которой необходима организация сложных запросов, можно делать вспомогательные хранилища и использовать средства тиражирования данных.

5) Конечный расчет: доступ пользователей осуществляется через интеллектуальные бизнес-приложения.

1.3. Пример разработки АРМ расчета нормативных потерь тепловой энергии сетевой компании в среде системы управления базами данных MS ACCESS

Расчетной задачей является определение нормативных потерь тепловой энергии при ее транспорте по трубопроводам сетевой компании к потребителям. Методика расчета опубликована в Инструкции[1] Минэнерго РФ и определена Приказом № 325. В примере в качестве теплоносителя принята вода.

К нормируемым технологическим затратам теплоносителя относятся:

- затраты теплоносителя на заполнение трубопроводов тепловых сетей перед пуском после плановых ремонтов и при подключении новых участков тепловых сетей;
- технологические сливы теплоносителя средствами автоматического регулирования теплового и гидравлического режимов, а также режима защиты оборудования;
- технически обоснованные затраты теплоносителя на плановые эксплуатационные испытания тепловых сетей и другие регламентные работы.

К нормируемым технологическим потерям теплоносителя относятся технически неизбежные в процессе передачи и распределения тепловой энергии потери теплоносителя с его утечкой через неплотности в арматуре и трубопроводах тепловых сетей в пределах, установленных правилами технической эксплуатации электрических станций и сетей, а также правилами технической эксплуатации тепловых энергоустановок.

Нормативные значения потерь теплоносителя за год с его нормируемой утечкой, м^3, определяются по формуле

$$G_{\text{ут.н}} = aV_{\text{год}}\, n_{\text{год}} \cdot 10^{-2} = m_{\text{ут.год.н}}\, n_{\text{год}},$$

где a – норма среднегодовой утечки теплоносителя, установленная правилами технической эксплуатации электрических станций и сетей, а также правилами технической эксплуатации тепловых энергоустановок в пределах 0,25 % среднегодовой емкости трубопроводов тепловых сетей в час, м^3/(ч·м^3); $V_{\text{год}}$ – среднегодовая емкость трубопроводов тепловых сетей,

[1] http://www.norm-load.ru/SNiP/Data1/58/58168/index.htm

эксплуатируемых теплосетевой организацией, м^3; $n_{год}$ – продолжительность функционирования тепловых сетей в году, ч; $m_{ут.год.н}$ – среднегодовая норма потерь теплоносителя, обусловленных утечкой, м^3/ч.

Значение среднегодовой емкости трубопроводов тепловых сетей, м^3, определяется из выражения

$$V_{год} = (V_{от}n_{от} + V_{л}n_{л}) / (n_{от} + n_{л}) = (V_{от}n_{от} + V_{л}n_{л}) / n_{год},$$

где $V_{от}$ и $V_{л}$ – соответственно емкость трубопроводов тепловых сетей в отопительном и неотопительном периодах, м^3; $n_{от}$ и $n_{л}$ – соответственно продолжительность функционирования тепловых сетей в отопительном и неотопительном периодах, ч.

При расчете значения среднегодовой емкости необходимо учесть:

- емкость трубопроводов, вновь вводимых в эксплуатацию, и продолжительность использования данных трубопроводов в течение календарного года;
- емкость трубопроводов, образуемую в результате реконструкции тепловой сети (изменения диаметров труб на участках, длины трубопроводов, конфигурации трассы тепловой сети), и период времени, в течение которого введенные в эксплуатацию участки реконструированных трубопроводов задействованы в календарном году;
- емкость трубопроводов, временно выводимых из использования для ремонта, и продолжительность ремонтных работ.

Прогнозируемая продолжительность отопительного периода принимается как средняя из соответствующих фактических значений за последние 5 лет или в соответствии со строительными нормами и правилами по строительной климатологии.

Затраты теплоносителя, обусловленные вводом в эксплуатацию трубопроводов тепловых сетей как новых, так и после плановых ремонтов или реконструкции, принимаются в размере 1,5-кратной емкости соответствующих трубопроводов тепловых сетей.

Затраты теплоносителя, обусловленные его сливом средствами автоматического регулирования и защиты, предусматривающими такой слив, определяются конструкцией указанных приборов и технологией обеспечения нормального функционирования тепловых сетей и оборудования.

Значения годовых потерь теплоносителя в результате слива, м^3, определяются из формулы

$$G_{а.н} = \sum_{1}^{k} mNn_{год.авт},$$

где m – технически обоснованный расход теплоносителя, сливаемого каждым из действующих приборов автоматики или защиты одного типа,

$м^3$/ч; N – количество действующих приборов автоматики или защиты одного типа, шт.; $n_{год.авт}$ – продолжительность функционирования однотипных приборов в течение года, ч; k – количество групп однотипных действующих приборов автоматики и защиты.

Нормативные технологические потери и затраты тепловой энергии при ее передаче включают:

- потери и затраты тепловой энергии, обусловленные потерями и затратами теплоносителя;
- потери тепловой энергии теплопередачей через изоляционные конструкции теплопроводов и оборудование тепловых сетей.

Определение нормативных технологических затрат и потерь тепловой энергии, обусловленных потерями и затратами теплоносителя (воды).

Определение нормативных технологических потерь тепловой энергии, Гкал, обусловленных потерями теплоносителя, производится по формуле

$$Q_{у.н} = m_{у.год.н}\, \rho_{год}\, c\, [b\tau_{1год} + (1 - b)\tau_{2год} - \tau_{х.год}]\, n_{год}\, 10^{-6},$$

где $\rho_{год}$ – среднегодовая плотность теплоносителя при средней (с учетом b) температуре теплоносителя в подающем и обратном трубопроводах тепловой сети, кг/$м^3$; b – доля массового расхода теплоносителя, теряемого подающим трубопроводом тепловой сети (при отсутствии данных можно принимать 0,5…0,75); $\tau_{1год}$ и $\tau_{2год}$ – среднегодовые значения температуры теплоносителя в подающем и обратном трубопроводах тепловой сети по температурному графику регулирования тепловой нагрузки, °С; $\tau_{х.год}$ – среднегодовое значение температуры исходной воды, подаваемой на источник теплоснабжения и используемой для подпитки тепловой сети, °С; c – удельная теплоемкость теплоносителя, ккал/кг·°С.

Среднегодовые значения температуры теплоносителя в подающем и обратном трубопроводах рассчитываются как средневзвешенные по среднемесячным значениям температуры теплоносителя в соответствующем трубопроводе с учетом числа часов работы в каждом месяце. Среднемесячные значения температуры теплоносителя в подающем и обратном трубопроводах определяются по эксплуатационному температурному графику отпуска тепловой энергии в соответствии с ожидаемыми среднемесячными значениями температуры наружного воздуха.

Ожидаемые среднемесячные значения температуры наружного воздуха определяются как средние из соответствующих статистических значений по информации метеорологической станции за последние 5 лет, или в соответствии со строительными нормами и правилами по строительной климатологии и климатологическим справочником.

Средневзвешенные значения температуры теплоносителя в подающих $\tau_{1год}$ и обратных $\tau_{2год}$ трубопроводах тепловой сети, °С, можно определить по формулам

$$\tau_{1год} = \Sigma(\tau_{1i}\, n_i) / (n_{от} + n_{л}) = \Sigma(\tau_{1i}\, n_i) / n_{год};$$

$$\tau_{2год} = \Sigma(\tau_{2i}\, n_i) / (n_{от} + n_{л}) = \Sigma(\tau_{2i}\, n_i) / n_{год},$$

где τ_{1i} и τ_{2i} – соответственно значения температуры теплоносителя в подающем и обратном трубопроводах тепловой сети по эксплуатационному температурному графику отпуска тепловой энергии при средней температуре наружного воздуха соответствующего месяца, °С.

При отсутствии достоверной информации по температурам исходной воды допустимо принимать $\tau_{х.от} = 5$ °С, $\tau_{х.л} = 15$ °С.

Нормативные технологические затраты тепловой энергии на заполнение новых участков трубопроводов и после плановых ремонтов, Гкал, определяются

$$Q_{зап} = 1{,}5 V_{тр.з}\, \rho_{зап}\, c\, (\tau_{зап} - \tau_{х}) 10^{-6},$$

где $V_{тр.з}$ – емкость заполняемых трубопроводов тепловых сетей, эксплуатируемых теплосетевой организацией, м^3; $\rho_{зап}$ – плотность воды, используемой для заполнения, кг/м^3; $\tau_{зап}$ – температура воды, используемой для заполнения, °С; $\tau_{х}$ – температура исходной воды, подаваемой на источник тепловой энергии в период заполнения, °С.

Нормативные технологические потери тепловой энергии со сливами из приборов автоматического регулирования и защиты, Гкал, определяются по формуле

$$Q_{а.н} = G_{а.н}\, \rho_{сл}\, c\, (\tau_{сл} - \tau_{х}) 10^{-6},$$

где $G_{а.н}$ – годовые потери теплоносителя в результате слива, м^3; $\rho_{сл}$ – среднегодовая плотность теплоносителя в зависимости от места установки автоматических приборов, кг/м^3; $\tau_{сл}$ и $\tau_{х}$ – соответственно температура сливаемого теплоносителя и исходной воды, подаваемой на источник теплоснабжения в период слива, °С.

Определение нормативных значений часовых потерь тепловой энергии производится в следующем порядке:

1) Для всех участков тепловых сетей на основе сведений о конструктивных особенностях теплопроводов (тип прокладки, год проектирования, наружный диаметр трубопроводов, длина участка) и норм тепловых потерь (теплового потока), указанных в таблицах приложений 1, 2, 3 и 4 к Инструкции Минэнерго РФ, пересчета табличных значений удельных норм на среднегодовые (среднесезонные) условия эксплуатации определяются значения часовых тепловых потерь теплопередачи через

теплоизоляционные конструкции трубопроводов, эксплуатируемых теплосетевой организацией.

2) Для участков тепловой сети, характерных для нее по типам прокладки и видам изоляционной конструкции и подвергавшихся испытаниям на тепловые потери, в качестве нормативных принимаются полученные при испытаниях значения фактических часовых тепловых потерь, пересчитанные на среднегодовые условия эксплуатации тепловой сети.

3) Для участков тепловой сети, аналогичных подвергавшимся тепловым испытаниям по типам прокладки, видам теплоизоляционных конструкций и условиям эксплуатации, в качестве нормативных принимаются значения часовых тепловых потерь, определенные по соответствующим нормам тепловых потерь (теплового потока) с введением поправочных коэффициентов, определенных по результатам испытаний.

Значения нормативных часовых тепловых потерь в тепловой сети в целом при среднегодовых (среднесезонных) условиях эксплуатации определяются суммированием значений часовых тепловых потерь на отдельных участках.

Определение нормативных значений часовых тепловых потерь для среднегодовых (среднесезонных) условий эксплуатации трубопроводов тепловых сетей производится согласно значениям норм тепловых потерь (теплового потока) в соответствии с годом проектирования конкретных участков тепловых сетей.

Значения нормативных удельных часовых тепловых потерь при среднегодовых (среднесезонных) условиях эксплуатации, отличающихся от значений, приведенных в соответствующих таблицах, ккал/ч·м, определяются линейной интерполяцией или экстраполяцией.

Определение нормативных значений часовых тепловых потерь для среднегодовых (среднесезонных) условий эксплуатации трубопроводов тепловых сетей производится в зависимости от года проектирования теплопроводов:

- спроектированных с 1959 по 1989 г. включительно;
- спроектированных с 1990 по 1997 г. включительно;
- спроектированных с 1998 по 2003 г. включительно;
- спроектированных с 2004 г.

Определение нормативных значений часовых тепловых потерь, Гкал/ч, для среднегодовых (среднесезонных) условий эксплуатации трубопроводов тепловых сетей производится по формуле

$$Q_{\text{из.н.год}} = \Sigma(q_{\text{из.н}} L \beta) 10^{-6},$$

где $q_{\text{из.н}}$ – удельные часовые тепловые потери трубопроводами каждого диаметра, определенные пересчетом табличных значений норм удельных

часовых тепловых потерь на среднегодовые (среднесезонные) условия эксплуатации, ккал/ч·м; L – длина участка трубопроводов тепловой сети, м; β – коэффициент местных тепловых потерь, учитывающий тепловые потери запорной и другой арматурой, компенсаторами и опорами (принимается 1,2 при диаметре трубопроводов до 150 мм и 1,15 при диаметре 150 мм и более, а также при всех диаметрах трубопроводов бесканальной прокладки независимо от года проектирования).

Значения нормативных часовых тепловых потерь, Гкал/ч, участков трубопроводов тепловых сетей, аналогичных участкам трубопроводов, подвергавшихся испытаниям на тепловые потери, по типу прокладки, виду изоляционных конструкций и условиям эксплуатации, определяются для трубопроводов подземной и надземной прокладки отдельно по формуле

$$Q_{\text{из.н.год}} = \Sigma(k_{\text{и}} q_{\text{из.н}} L \beta) 10^{-6},$$

где $k_{\text{и}}$ – поправочный коэффициент для определения нормативных часовых тепловых потерь, полученный по результатам испытаний на тепловые потери.

Значения поправочного коэффициента $k_{\text{и}}$ определяются по формуле

$$k_{\text{и}} = Q_{\text{из.год.и}} / Q_{\text{из.год.н}},$$

где $Q_{\text{из.год.и}}$ – тепловые потери, определенные в результате испытаний на тепловые потери, пересчитанные на среднегодовые условия эксплуатации каждого испытанного участка трубопроводов тепловой сети, Гкал/ч; $Q_{\text{из.год.н}}$ – потери, определенные по нормам для тех же участков, Гкал/ч.

Примеры создания отношений, их связей, формирования отчета по вычисляемым данным (вторичным), а также программы на встроенном языке программирования VBA приведены на рис. 1.11-1.14.

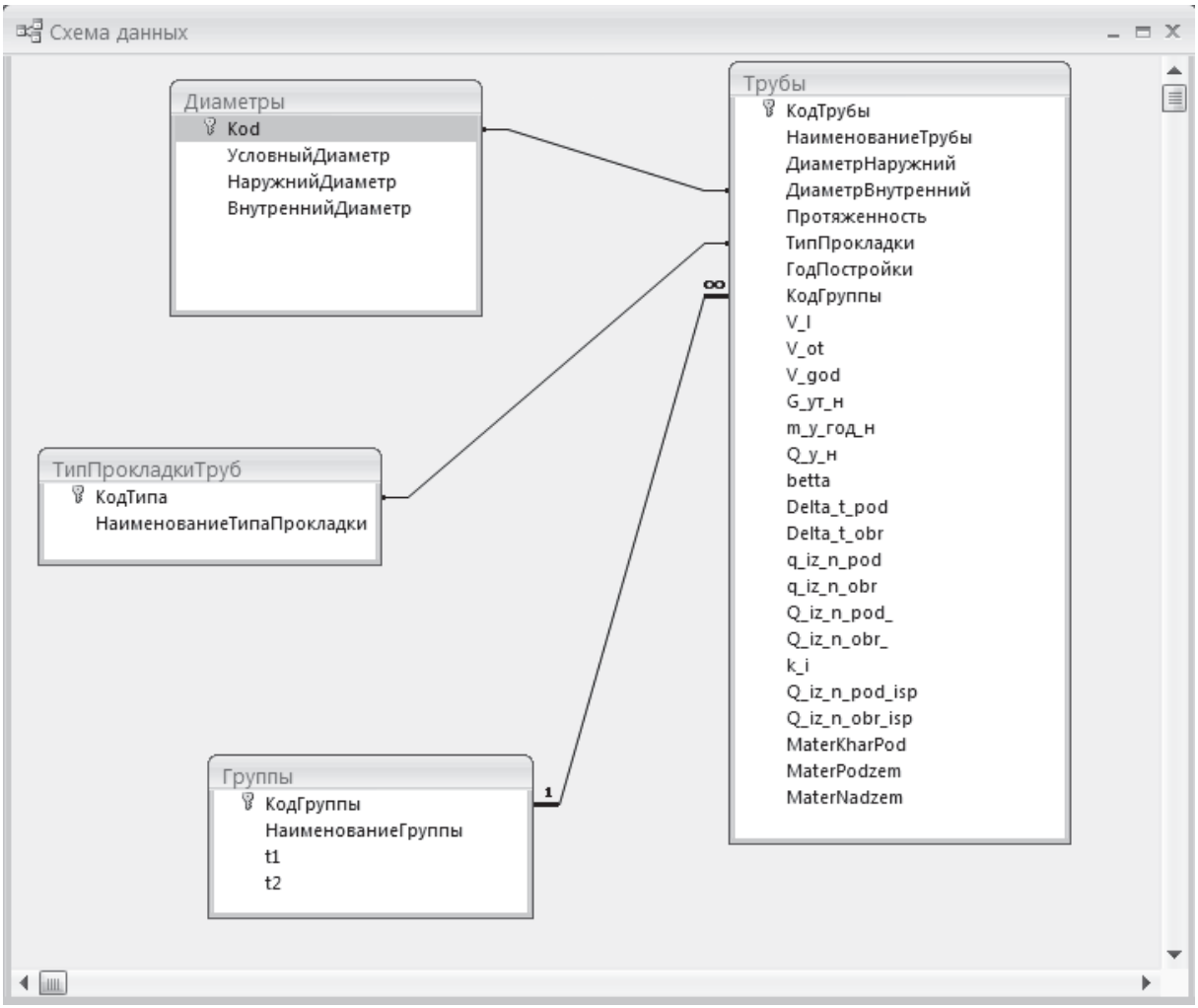


Рис. 1.11 Схема данных (обозначения отношений приведены в тексте программы)

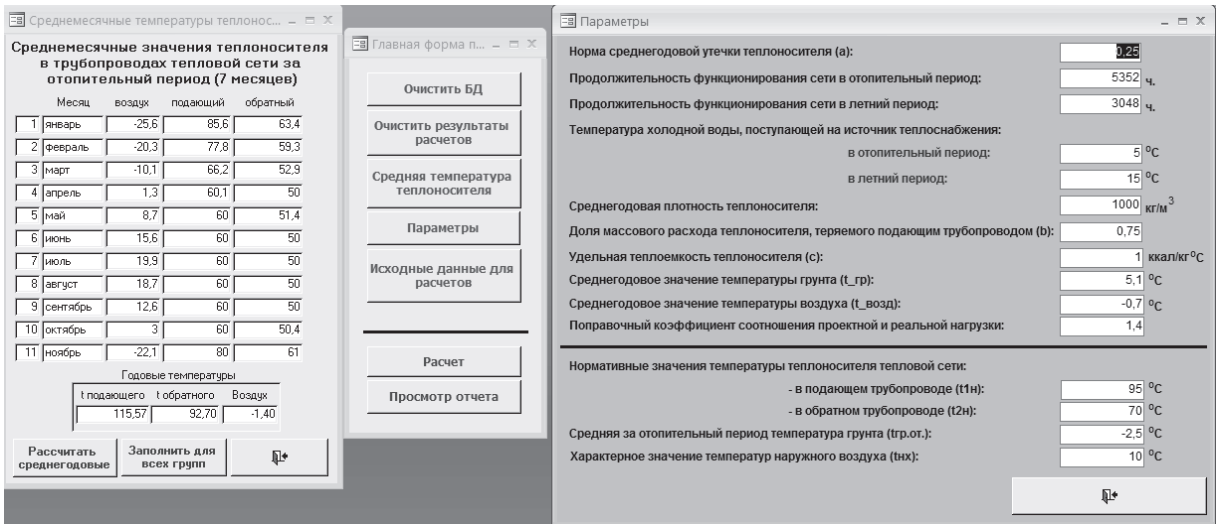


Рис. 1.12 Форма для ввода исходных данных о теплоносителе

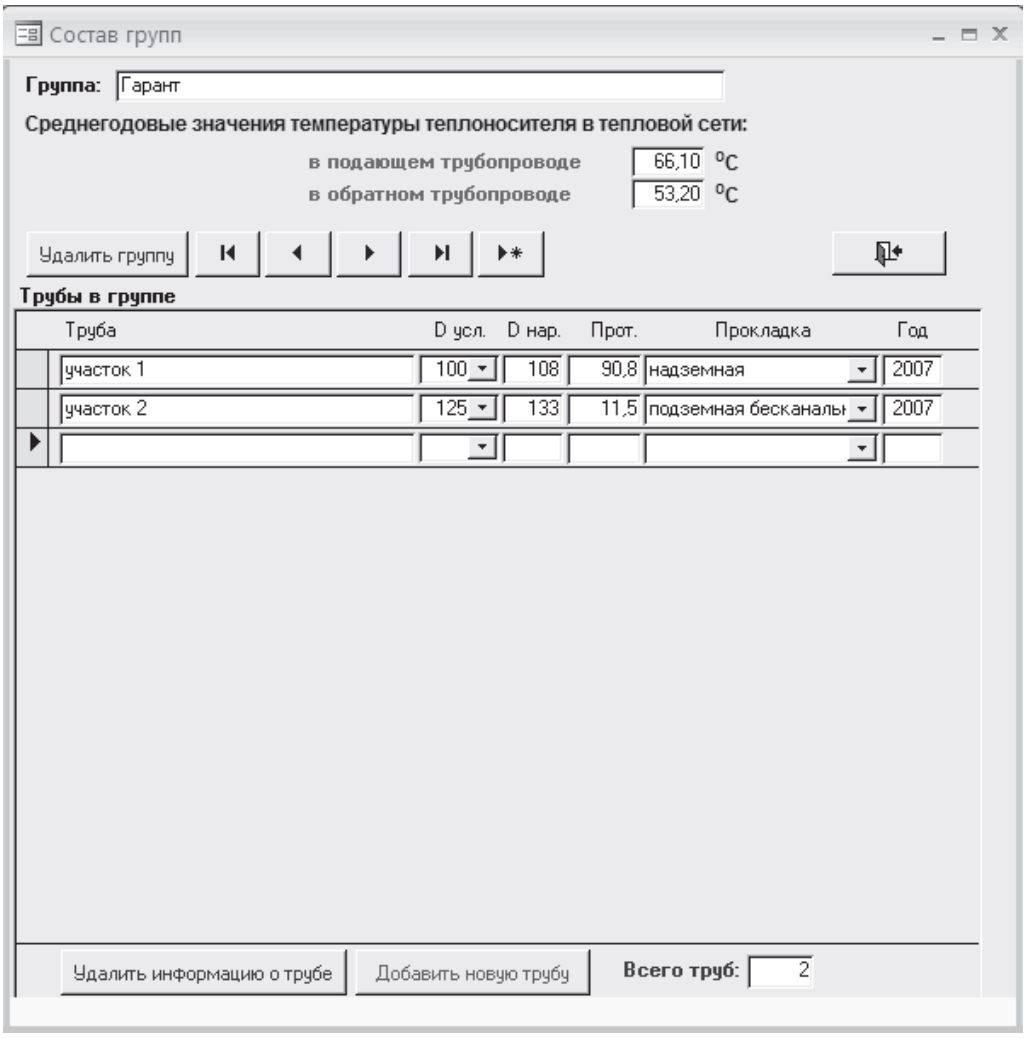


Рис. 1.13. Состав групп

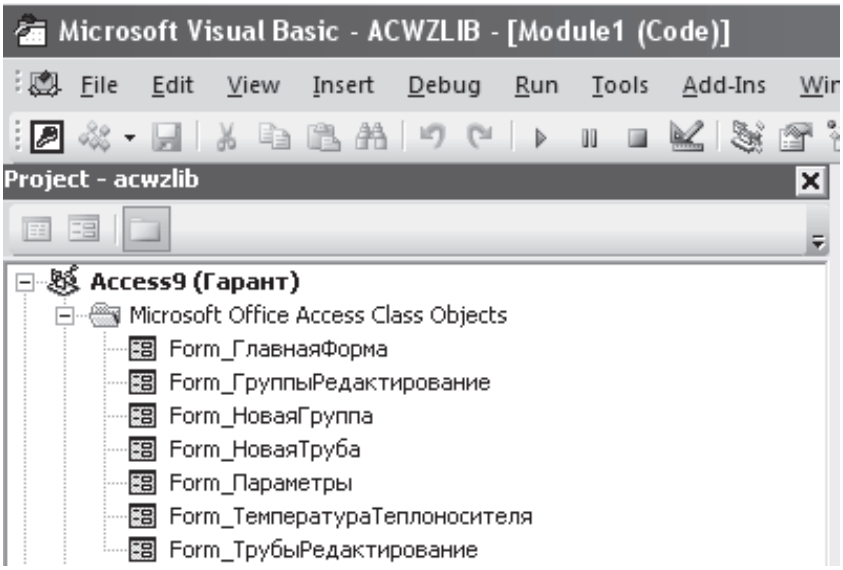


Рис. 1.14. Вид программы на встроенном языке программирования VBA

Главная форма:

```
Option Compare Database
' общие параметры расчета
Dim a As Double      ' норма среднегодовой утечки теплоносителя
Dim n_ot As Integer  ' число часов работы сети в отопительный период
Dim n_l As Integer   ' число часов работы сети в летний период
Dim t_h_ot As Double ' температура х.в., подаваемой в отопительный период
Dim t_h_l As Double  ' температура х.в., подаваемой в летний период
Dim Ro As Double     ' средняя плотность теплоносителя
Dim b As Double      ' доля массового расхода теплоносителя
Dim c As Double      ' удельная теплоемкость теплоносителя
Dim t1 As Double     ' среднегодовая температура в подающем трубопроводе
Dim t2 As Double     ' среднегодовая температура в обратном трубопроводе
Dim t_h_god As Double ' среднегодовое значение температуры холодной воды,
подаваемой на источник теплоснабжения для подпитки тепловой сети
Dim t_gr As Double   ' среднегодовое значение температуры грунта
Dim t_voz As Double  ' среднегодовое значение температуры воздуха
Dim k_i As Double    ' Поправочный коэффициент соотношения проектной и реальной
нагрузки

Private Sub Кнопка1_Click()
On Error GoTo Err_Кнопка1_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = ChrW(1055) & ChrW(1072) & ChrW(1088) & ChrW(1072) & ChrW(1084)
& ChrW(1077) & ChrW(1090) & ChrW(1088) & ChrW(1099)
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Кнопка1_Click:
    Exit Sub

Err_Кнопка1_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка1_Click

End Sub
Private Sub Кнопка2_Click()
On Error GoTo Err_Кнопка2_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = ChrW(1058) & ChrW(1077) & ChrW(1084) & ChrW(1087) & ChrW(1077)
& ChrW(1088) & ChrW(1072) & ChrW(1090) & ChrW(1091) & ChrW(1088) &
ChrW(1072) & ChrW(1058) & ChrW(1077) & ChrW(1087) & ChrW(1083) & ChrW(1086)
```

```
& ChrW(1085) & ChrW(1086) & ChrW(1089) & ChrW(1080) & ChrW(1090) &
ChrW(1077) & ChrW(1083) & ChrW(1103)
   DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Кнопка2_Click:
   Exit Sub

Err_Кнопка2_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка2_Click

End Sub
Private Sub Кнопка3_Click()
On Error GoTo Err_Кнопка3_Click

   Dim stDocName As String
   Dim stLinkCriteria As String

   stDocName = ChrW(1043) & ChrW(1088) & ChrW(1091) & ChrW(1087) & ChrW(1087)
& ChrW(1099) & ChrW(1056) & ChrW(1077) & ChrW(1076) & ChrW(1072) &
ChrW(1082) & ChrW(1090) & ChrW(1080) & ChrW(1088) & ChrW(1086) & ChrW(1074)
& ChrW(1072) & ChrW(1085) & ChrW(1080) & ChrW(1077)
   DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Кнопка3_Click:
   Exit Sub

Err_Кнопка3_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка3_Click

End Sub

' функция расчета всех параметров
Private Sub Кнопка4_Click()
   ' считываем параметры
   If (SolutionParameters = 0) Then
      ' рассчитываем для труб все остальные цифры
      Raschet
   Else
      MsgBox "Возникли ошибки чтения параметров для расчета!"
   End If
End Sub

' функция расчета тепловых потерь
Private Function Raschet()
On Error GoTo Err_Raschet
   Dim CurDB As Database
   Dim rst As DAO.Recordset
```

```
Dim rstGroup As DAO.Recordset
Dim j As Integer
Dim i As Integer
Dim materKharaktPodz As Double
Dim NadzemTr As Double
Dim PodzemTr As Double
Dim tt1 As Double
Dim tt2 As Double

' открываем текущую БД и таблицу труб
Set CurDB = CurrentDb
Set rst = CurDB.OpenRecordset("ТрубыЗапрос")
Set rstGroup = CurDB.OpenRecordset("MaterKharakter")
rst.MoveLast
rst.MoveFirst
rstGroup.MoveLast

' перебираем все строки таблицы труб
For i = 1 To rst.RecordCount
   ' считываем температуры из таблицы групп
   rstGroup.MoveFirst
   t1 = 0
   t2 = 0
   materKharaktPodz = 0
   ' перебираем все группы, пока не найдем
   For j = 1 To rstGroup.RecordCount
      If (rstGroup.Fields(0).Value = rst.Fields(7).Value) Then
         t1 = rstGroup.Fields(2).Value
         t2 = rstGroup.Fields(3).Value
         materKharaktPodz = rstGroup.Fields(1).Value
         NadzemTr = rstGroup.Fields(5).Value
         PodzemTr = rstGroup.Fields(6).Value
         Exit For
      End If
      rstGroup.MoveNext
   Next j

   ' начинаем редактирование записи
   rst.Edit

   ' объем сети в отопительный период
   rst.Fields(9).Value = 2# * rst.Fields(3).Value * rst.Fields(3).Value * _
      rst.Fields(4).Value * 3.141592654 / (4# * 1000000)
   ' объем сети в летний период
   rst.Fields(8).Value = 0.8 * rst.Fields(9).Value
   ' Среднегодовая емкость тепловой сети
   rst.Fields(10).Value = (rst.Fields(9).Value * n_ot + n_l * rst.Fields(8).Value) / _
      (n_ot + n_l)
   ' Нормативные значения годовых потерь теплоносителя с его утечкой
```

```
rst.Fields(11).Value = rst.Fields(10).Value * a * (n_ot + n_l) / 100#
' Среднечасовая годовая норма потерь теплоносителя, обусловленная его утечкой
rst.Fields(12).Value = rst.Fields(10).Value * a / 100#
' Нормативные значения годовых технологических потерь с утечкой
' теплоносителя из трубопроводов
rst.Fields(13).Value = rst.Fields(12).Value * Ro * c * (b * t1 + _
   (1 - b) * t2 - t_h_god) * (n_l + n_ot) / 1000000
' Коэффициент местных тепловых потерь betta
Select Case rst.Fields(5).Value
' подземная
Case 1
   rst.Fields(14).Value = 1.15
' надземная
Case 2
   rst.Fields(14).Value = 1.25
' канальная
Case 3
   If (rst.Fields(3).Value <= 150) Then
      rst.Fields(14).Value = 1.2
   Else
      rst.Fields(14).Value = 1.15
   End If
' если тип неизвестен
Case Else
   rst.Fields(14).Value = 0
End Select
' среднегодовые разности значений температур теплоносителя и грунта (или
воздуха)
If (rst.Fields(5).Value = 2) Then
   rst.Fields(15).Value = t1 - t_voz
   rst.Fields(16).Value = t2 - t_voz
Else
   rst.Fields(15).Value = (t1 + t2) / 2# - t_gr
   'rst.Fields(16).Value = (t1 + t2) / 2# - t_gr
End If
' удельные часовые тепловые потери трубопроводов
If (rst.Fields(5).Value = 2) Then
   ' выбираем температуру по году
   If (rst.Fields(6).Value > 1990) Then
      tt1 = t1
      tt2 = t2
   Else
      tt1 = rst.Fields(15).Value
      tt2 = rst.Fields(16).Value
   End If
   ' q_iz_n_pod
   rst.Fields(17).Value = OprTable(rst.Fields(6).Value, 0, rst.Fields(5).Value _
   , rst.Fields(3).Value, tt1)
   ' q_iz_n_obr
```

```
      rst.Fields(18).Value = OprTable(rst.Fields(6).Value, 1, rst.Fields(5).Value, _
         rst.Fields(3).Value, tt2)
   Else
      ' выбираем температуру по году
      If (rst.Fields(6).Value > 1990) Then
         tt1 = t1
      Else
         tt1 = rst.Fields(15).Value
      End If
      ' q_iz_n
      rst.Fields(17).Value = OprTable(rst.Fields(6).Value, 0, rst.Fields(5).Value, _
         rst.Fields(3).Value, tt1)
      ' q_iz_n_obr
      'rst.Fields(18).Value = OprTable(rst.Fields(6).Value, 1, 1, rst.Fields(3).Value, _
      '    rst.Fields(16).Value)
   End If
   ' часовые тепловые потери трубопроводов
   rst.Fields(19).Value = rst.Fields(17).Value * rst.Fields(14).Value * _
      rst.Fields(4).Value / 1000000#
   If (rst.Fields(5).Value = 2) Then
      rst.Fields(20).Value = rst.Fields(18).Value * rst.Fields(14).Value * _
         rst.Fields(4).Value / 1000000#
   End If
   ' поправочный коэффициент
   rst.Fields(21).Value = Koeff(materKharaktPodz, rst.Fields(5).Value)
   ' часовые тепловые потери трубопроводов на испытания
   rst.Fields(22).Value = rst.Fields(19).Value * rst.Fields(21).Value
   If (rst.Fields(5).Value = 2) Then
      rst.Fields(23).Value = rst.Fields(20).Value * rst.Fields(21).Value
   End If
   ' сохраняем значение материальной характеристики
   ' подземной части труб
   rst.Fields(24).Value = materKharaktPodz
   ' числитель для подземной части труб
   rst.Fields(25).Value = PodzemTr
   ' числитель для надземной части труб
   rst.Fields(26).Value = NadzemTr

   ' сохраняем изменения
   rst.Update
   ' переходим к следующей строке
   rst.MoveNext
Next i

' закрываем таблицы
rstGroup.Close
Set rstGroup = Nothing
rst.Close
Set rst = Nothing
```

```
    Set CurDB = Nothing

    Raschet = 0
Exit_Raschet:
    Exit Function

Err_Raschet:
    MsgBox Err.Description
    Raschet = -1
    Resume Exit_Raschet
End Function

' функция определения поправочного коэффициента
' MaterKhar - значение материальной характеристики для подземной части
' Uroven - уровень нахождения трубы, 2 - надземная, 1 - подземная бесканальная
'          3 - подземная канальная
Private Function Koeff(MaterKhar As Double, Uroven As Integer)
On Error GoTo Err_Koeff
    Dim CurDB As Database
    Dim rst As DAO.Recordset
    Dim i As Integer

    Koeff = 0

    ' открываем текущую БД и таблицу труб
    Set CurDB = CurrentDb
    Set rst = CurDB.OpenRecordset("Поправки")
    rst.MoveFirst

    ' перебираем строки
    For i = 1 To rst.RecordCount - 1
        If (rst.Fields(1).Value <= MaterKhar) Then
            Exit For
        End If
        rst.MoveNext
    Next i

    ' перебираем столбцы
    ' интервал от 0.6 до 0.8
    If (k_i <= 0.8) Then
        i = 2
    ' интервал от 0.8 до 0.9
    ElseIf (k_i <= 0.9) Then
        i = 4
    ' интервал от 0.9 до 1.0
    ElseIf (k_i <= 1#) Then
        i = 6
    ' интервал от 1.0 до 1.1
    ElseIf (k_i <= 1.1) Then
```

```
        i = 8
    ' интервал от 1.1 до 1.2
    ElseIf (k_i <= 1.2) Then
        i = 10
    ' интервал от 1.2 до 1.3
    ElseIf (k_i <= 1.3) Then
        i = 12
    ' интервал от 1.3 до 1.4
    Else
        i = 14
    End If

    ' определяем коэффициент
    ' надземная прокладка
    If (Uroven = 2) Then
        Koeff = rst.Fields(i + 1).Value
    ' подземная прокладка
    Else
        Koeff = rst.Fields(i).Value
    End If

    ' закрываем таблицы
    rst.Close
    Set rst = Nothing
    Set CurDB = Nothing

Exit_Koeff:
    Exit Function

Err_Koeff:
    MsgBox Err.Description
    Resume Exit_Koeff

End Function

' функция определения таблицы и расчет удельных часовых тепловых потерь
' God - год создания трубопровода
' TypeTrubi - тип трубопровода, 0 - подающая, 1 - обратная
' Uroven - уровень нахождения трубы, 2 - надземная, 1 - подземная бесканальная
'          3 - подземная канальная
' D -  диаметр трубы
' temperatura - среднегодовая разность температур
Private Function OprTable(God As Integer, TypeTrubi As Integer, _
    Uroven As Integer, D As Double, temperatura As Double)
On Error GoTo Err_OprTable
    Dim sss As String

    ' выбираем имя таблицы
    If (God >= 1959) And (God <= 1990) Then
```

```
    ' подающая труба
    If (TypeTrubi = 0) Then
       ' надземный трубопровод
       If (Uroven = 2) Then
          sss = "q_nadzem_1959_1990"
       ' подземный трубопровод
       Else
          sss = "q_podzem_1959_1990"
       End If
    ' обратная труба
    Else
       ' надземный трубопровод
       If (Uroven = 2) Then
          sss = "q_nadzem_1959_1990"
       ' подземный трубопровод
       Else
          sss = "q_podzem_1959_1990"
       End If
    End If
 ElseIf (God <= 1998) Then
    ' подающая труба
    If (TypeTrubi = 0) Then
       ' надземный трубопровод
       If (Uroven = 2) Then
          sss = "q_nadzem_1990_1998_pod"
       ' подземный бесканальный трубопровод
       ElseIf (Uroven = 1) Then
          sss = "q_podzem_1990_1998_pod_no_kanal"
       ' подземный канальный трубопровод
       Else
          sss = "q_podzem_1990_1998_pod_kanal"
       End If
    ' обратная труба
    Else
       ' надземный трубопровод
       If (Uroven = 2) Then
          sss = "q_nadzem_1990_1998_obr"
       ' подземный бесканальный трубопровод
       ElseIf (Uroven = 1) Then
          sss = "q_podzem_1990_1998_obr_no_kanal"
       ' подземный канальный трубопровод
       Else
          sss = "q_podzem_1990_1998_obr_kanal"
       End If
    End If
 ElseIf (God <= 2003) Then
    ' подающая труба
    If (TypeTrubi = 0) Then
       ' надземный трубопровод
```

```
        If (Uroven = 2) Then
            sss = "q_nadzem_1998_2003_pod"
        ' подземный трубопровод
        Else
            sss = "q_podzem_1998_2003_pod"
        End If
    ' обратная труба
    Else
        ' надземный трубопровод
        If (Uroven = 2) Then
            sss = "q_nadzem_1998_2003_obr"
        ' подземный трубопровод
        Else
            sss = "q_podzem_1998_2003_obr"
        End If
    End If
Else   ' 2004 и менее
    ' подающая труба
    If (TypeTrubi = 0) Then
        ' надземный трубопровод
        If (Uroven = 2) Then
            sss = "q_nadzem_2004_pod"
        ' подземный бесканальный трубопровод
        ElseIf (Uroven = 1) Then
            sss = "q_podzem_2004_pod_no_kanal"
        ' подземный канальный трубопровод
        Else
            sss = "q_podzem_2004_pod_kanal"
        End If
    ' обратная труба
    Else
        ' надземный трубопровод
        If (Uroven = 2) Then
            sss = "q_nadzem_2004_obr"
        ' подземный бесканальный трубопровод
        ElseIf (Uroven = 1) Then
            sss = "q_podzem_2004_obr_no_kanal"
        ' подземный канальный трубопровод
        Else
            sss = "q_podzem_2004_obr_kanal"
        End If
    End If
End If

OprTable = Raschet_q(sss, D, temperatura)

' если попадаем в промежуток от 1991 до 2004 года и подземная труба, то
' умножить полученное значение на 2
'If (God > 1990) And (God <= 2003) And (Uroven <> 2) Then
```

```
    '    OprTable = OprTable * 2#
    'End If

Exit_OprTable:
    Exit Function

Err_OprTable:
    MsgBox Err.Description
    Resume Exit_OprTable
End Function

' функция расчета удельных часовых тепловых потерь
Private Function Raschet_q(NameTable As String, D As Double, _
    temperatura As Double)
On Error GoTo Err_Raschet_q
    Dim CurDB As Database
    Dim rst As DAO.Recordset
    Dim temp(10) As Double
    Dim i As Integer
    Dim temp_i1, temp_i2 As Integer     ' номера столбцов с меньшей и большей
температурой
    Dim kk As Double         ' коэффицент смещения параметров
    Dim diametr As Double    ' больший диаметр
    Dim q1, q2 As Double     ' значения удельных часовых тепловых потерь
    Dim t1, t2 As Double     ' вилка температур в таблице
    Dim nTemp As Integer     ' количество столбцов температур
    Raschet_q = 0

    ' открываем текущую БД и таблицу параметров
    Set CurDB = CurrentDb
    Set rst = CurDB.OpenRecordset(NameTable)
    ' переходим к первой строке и считываем температуры
    rst.MoveFirst
    nTemp = rst.Fields.Count   ' считываем количество столбцов в таблице
    For i = 2 To nTemp - 1
        temp(i - 2) = rst.Fields(i).Value
    Next i
    rst.MoveNext

    For i = 1 To (rst.RecordCount - 1)
        ' выполняем проверку на достижение требуемого диаметра
        If (rst.Fields(1).Value >= D) Then Exit For

        ' переходим к следующей строке
        rst.MoveNext
    Next i

    diametr = rst.Fields(1).Value     ' считываем значение большего диаметра
```

```
   ' выполняем проверку на выход температур из диапазона таблицы
   ' выход справа
   If (temperatura >= temp(nTemp - 3)) Then
      t1 = temp(nTemp - 3)
      t2 = temp(nTemp - 3)
      q1 = rst.Fields(nTemp - 1).Value
      q2 = rst.Fields(nTemp - 1).Value
      temp_i1 = nTemp - 1
      temp_i2 = nTemp - 1
      GoTo qqq
   End If

   ' выход слева
   If (temperatura <= temp(0)) Then
      t1 = temp(0)
      t2 = temp(0)
      q1 = rst.Fields(2).Value
      q2 = rst.Fields(2).Value
      temp_i1 = 2
      temp_i2 = 2
      GoTo qqq
   End If

   ' ищем температуры в таблице
   For i = 2 To nTemp - 1
      If (temperatura <= temp(i - 2)) Then
         t2 = temp(i - 2)           ' большая температура
         q2 = rst.Fields(i).Value    ' часовые потери, соответствующие большей
температуре
         temp_i2 = i                ' номер столбца с большей температурой

         ' проверяем, не находимся ли мы в начале таблицы
         If (i <= 2) Then
            ' дошли
            t1 = temp(i - 2)         ' присваиваем меньшую температуру
            q1 = rst.Fields(i).Value  ' часовые потери, соответствующие меньшей
                                 ' температуре
            temp_i1 = i              ' номер столбца с большей температурой
         Else
            ' не дошли
            t1 = temp(i - 3)           ' присваиваем меньшую температуру
            q1 = rst.Fields(i - 1).Value ' часовые потери, соответствующие меньшей
                                   ' температуре
            temp_i1 = i - 1            ' номер столбца с меньшей температурой
         End If

         ' после расчета выход из цикла
         Exit For
      End If
```

```
    Next i

qqq:
    ' для расчета теперь берем строку, соответствующую большему диаметру
    GoTo wwww

    ' выполняем проверку на точное равенство диаметров труб
    ' если диаметры не совпадают, то интерполируем значения
    If (rst.Fields(1) <> D) Then
        ' переходим к предыдущей записи
        rst.MovePrevious
        ' если запись была первая, то выходим
        If Not rst.BOF Then
            ' проводим интерполяцию
            ' считаем коэффицент
            kk = (diametr - D) / (diametr - rst.Fields(1).Value)
            q1 = q1 - kk * (q1 - rst.Fields(temp_i1).Value)
            q2 = q2 - kk * (q2 - rst.Fields(temp_i2).Value)
        End If
    End If

wwww:
    ' рассчитываем удельные часовые потери по трубе
    If (t2 - t1) = 0 Then
        Raschet_q = q1
    Else
        Raschet_q = q1 + (q2 - q1) * (temperatura - t1) / (t2 - t1)
    End If

    ' закрываем таблицу
    rst.Close
    Set rst = Nothing
    Set CurDB = Nothing

Exit_Raschet_q:
    Exit Function

Err_Raschet_q:
    MsgBox Err.Description
    Resume Exit_Raschet_q
End Function
' процедура считывания данных из таблицы параметров в переменные
Private Function SolutionParameters()
On Error GoTo Err_SolutionParameters
    Dim CurDB As Database
    Dim rst As DAO.Recordset
    'Dim ss As Integer

    ' открываем текущую БД и таблицу параметров
```

```
    Set CurDB = CurrentDb
    Set rst = CurDB.OpenRecordset("Параметры")
    rst.MoveFirst
    ' считываем параметры
    a = rst.Fields(0).Value
    n_ot = rst.Fields(1).Value
    n_l = rst.Fields(2).Value
    t_h_ot = rst.Fields(3).Value
    t_h_l = rst.Fields(4).Value
    Ro = rst.Fields(5).Value
    b = rst.Fields(6).Value
    c = rst.Fields(7).Value
    t_gr = rst.Fields(8).Value
    t_voz = rst.Fields(9).Value
    k_i = rst.Fields(10).Value

    ' рассчитываем параметры
    t_h_god = (t_h_ot * n_ot + t_h_l * n_l) / (n_ot + n_l)

    ' закрываем таблицу
    rst.Close
    Set rst = Nothing
    Set CurDB = Nothing

    SolutionParameters = 0
Exit_SolutionParameters:
    Exit Function

Err_SolutionParameters:
    MsgBox Err.Description
    SolutionParameters = -1
    Resume Exit_SolutionParameters
End Function

' очистка базы данных
Private Sub Кнопка5_Click()
On Error GoTo Err_Кнопка5_Click
    Dim CurDB As Database
    Dim rst As DAO.Recordset

    ' запрос у пользователя на очистку
    If (MsgBox("Удалить все исходные данные (кроме параметров) из БД?", _
    vbOKCancel, "Очистка БД") = vbCancel) Then
        Exit Sub
    End If

    ' открываем текущую БД и таблицу труб

    Set CurDB = CurrentDb
```

```
    Set rst = CurDB.OpenRecordset("Трубы")
    While (rst.RecordCount > 0)
       rst.MoveFirst
       rst.Delete
    Wend
    ' закрываем таблицу труб
    rst.Close
    Set rst = Nothing

    ' открываем таблицу групп
    Set rst = CurDB.OpenRecordset("Группы")
    While (rst.RecordCount > 0)
       rst.MoveFirst
       rst.Delete
    Wend
    ' закрываем таблицу групп
    rst.Close
    Set rst = Nothing

    Set CurDB = Nothing

Exit_Кнопка5_Click:
    Exit Sub

Err_Кнопка5_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка5_Click
End Sub

Private Sub Кнопка6_Click()
On Error GoTo Err_Кнопка6_Click
    Dim CurDB As Database
    Dim rst As DAO.Recordset
    Dim i, j As Integer

    ' открываем текущую БД и таблицу труб
    Set CurDB = CurrentDb
    Set rst = CurDB.OpenRecordset("Трубы")
    rst.MoveFirst

    ' удаляем все рассчитанные значения
    For i = 1 To rst.RecordCount
       ' начинаем удаление значения
       rst.Edit
       For j = 8 To rst.Fields.Count - 1
          rst.Fields(j).Value = Empty
       Next j
       ' сохраняем выполненные изменения
       rst.Update
```

```
        rst.MoveNext
    Next i

    ' закрываем таблицу труб
    rst.Close
    Set rst = Nothing
    Set CurDB = Nothing
Exit_Кнопка6_Click:
    Exit Sub

Err_Кнопка6_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка6_Click
End Sub
Private Sub Кнопка7_Click()
On Error GoTo Err_Кнопка7_Click

    Dim stDocName As String

    stDocName = "GeneralResult"
    DoCmd.OpenReport stDocName, acPreview

Exit_Кнопка7_Click:
    Exit Sub

Err_Кнопка7_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка7_Click

End Sub
```

Группы, редактирование:

```
Private Sub Кнопка6_Click()
On Error GoTo Err_Кнопка6_Click


    DoCmd.GoToRecord , , acNext

Exit_Кнопка6_Click:
    Exit Sub

Err_Кнопка6_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка6_Click

End Sub
Private Sub Кнопка7_Click()
On Error GoTo Err_Кнопка7_Click
```

```
    DoCmd.GoToRecord , , acPrevious

Exit_Кнопка7_Click:
    Exit Sub

Err_Кнопка7_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка7_Click

End Sub
Private Sub Кнопка8_Click()
On Error GoTo Err_Кнопка8_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_Кнопка8_Click:
    Exit Sub

Err_Кнопка8_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка8_Click

End Sub
Private Sub Кнопка9_Click()
On Error GoTo Err_Кнопка9_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = ChrW(1043) & ChrW(1088) & ChrW(1091) & ChrW(1087) & ChrW(1087)
& ChrW(1099)
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Кнопка9_Click:
    Exit Sub

Err_Кнопка9_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка9_Click

End Sub
' добавление новой группы труб
Private Sub Кнопка10_Click()
On Error GoTo Err_Кнопка10_Click

    Dim stDocName As String
    Dim stLinkCriteria As String
```

```
    stDocName = ChrW(1053) & ChrW(1086) & ChrW(1074) & ChrW(1072) & ChrW(1103)
& ChrW(1043) & ChrW(1088) & ChrW(1091) & ChrW(1087) & ChrW(1087) &
ChrW(1072)
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_Кнопка10_Click:
    Exit Sub

Err_Кнопка10_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка10_Click

End Sub
Private Sub Кнопка12_Click()
On Error GoTo Err_Кнопка12_Click


    DoCmd.GoToRecord , , acFirst

Exit_Кнопка12_Click:
    Exit Sub

Err_Кнопка12_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка12_Click

End Sub
Private Sub Кнопка13_Click()
On Error GoTo Err_Кнопка13_Click


    DoCmd.GoToRecord , , acNext

Exit_Кнопка13_Click:
    Exit Sub

Err_Кнопка13_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка13_Click

End Sub
Private Sub Кнопка14_Click()
On Error GoTo Err_Кнопка14_Click


    DoCmd.GoToRecord , , acPrevious

Exit_Кнопка14_Click:
    Exit Sub
```

```
Err_Кнопка14_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка14_Click

End Sub
Private Sub Кнопка15_Click()
On Error GoTo Err_Кнопка15_Click

    DoCmd.GoToRecord , , acLast

Exit_Кнопка15_Click:
    Exit Sub

Err_Кнопка15_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка15_Click

End Sub
Private Sub Кнопка16_Click()
On Error GoTo Err_Кнопка16_Click


    DoCmd.GoToRecord , , acNewRec

Exit_Кнопка16_Click:
    Exit Sub

Err_Кнопка16_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка16_Click

End Sub
Private Sub Кнопка17_Click()
On Error GoTo Err_Кнопка17_Click


    DoCmd.Close

Exit_Кнопка17_Click:
    Exit Sub

Err_Кнопка17_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка17_Click

End Sub
```

Новая группа:

```
' вставка новой группы труб
Private Sub Кнопка4_Click()
On Error GoTo Err_Кнопка4_Click


    DoCmd.GoToRecord , , acNewRec
    DoCmd.Close
    Forms![ГруппыРедактирование].Refresh
    Form.Refresh

Exit_Кнопка4_Click:
    Exit Sub

Err_Кнопка4_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка4_Click

End Sub
Private Sub Кнопка6_Click()
On Error GoTo Err_Кнопка6_Click


    DoCmd.Close

Exit_Кнопка6_Click:
    Exit Sub

Err_Кнопка6_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка6_Click

End Sub
```

Новая труба:

```
Private Sub Form_Open(Cancel As Integer)
    Dim sss As String
    'Form![Поле30].SetFocus
    'sss = Form![Поле30]
End Sub

Private Sub ДиаметрВнутренний_Change()
On Error GoTo Err_ДиаметрВнутренний_Change
    Me.ДиаметрНаружний = (Me.ДиаметрВнутренний.Column(2))
Exit_ДиаметрВнутренний_Change:
    Exit Sub
```

```
Err_ДиаметрВнутренний_Change:
   'MsgBox Err.Description
   Resume Exit_ДиаметрВнутренний_Change
End Sub

Private Sub Кнопка27_Click()
On Error GoTo Err_Кнопка27_Click

   Dim sss As Integer
   'sss = Form![КодГруппы]
   DoCmd.GoToRecord , , acNewRec
   Forms![ГруппыРедактирование].Refresh
   'Form![КодГруппы] = sss
   DoCmd.Close

Exit_Кнопка27_Click:
   Exit Sub

Err_Кнопка27_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка27_Click

End Sub

Private Sub Кнопка28_Click()
On Error GoTo Err_Кнопка28_Click


   DoCmd.Close

Exit_Кнопка28_Click:
   Exit Sub

Err_Кнопка28_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка28_Click

End Sub
```

Параметры:

```
Private Sub k_i_AfterUpdate()
   ' предупредить пользователя о выходе за диапазон
   If (Me.k_i.Value < 0.6) Or (Me.k_i.Value > 1.4) Then
      MsgBox "Значение коэффициента должно лежать в пределах от 0,6 до 1,4!", vbIn-
formation
   End If
End Sub
```

```
Private Sub Кнопка15_Click()
On Error GoTo Err_Кнопка15_Click
    DoCmd.Close

Exit_Кнопка15_Click:
    Exit Sub

Err_Кнопка15_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка15_Click

End Sub
```

Температура теплоносителя:

```
' заполнение среднегодовых температур теплоносителя для всех групп
Private Sub Кнопка13_Click()
On Error GoTo Err_Кнопка13_Click
    Dim CurDB As DAO.Database
    Dim rst As DAO.Recordset
    Dim i As Integer
    Dim t1, t2 As Double

    Set CurDB = CurrentDb
    Set rst = CurDB.OpenRecordset("ТемператураТеплоносителяЗапрос")
    rst.MoveFirst

    t1 = rst.Fields(0).Value
    t2 = rst.Fields(1).Value

    ' закрытие наборов
    rst.Close
    Set rst = Nothing

    Set rst = CurDB.OpenRecordset("Группы")
    ' изменяем температуры для групп
    For i = 1 To rst.RecordCount
        rst.Edit
        rst.Fields(2).Value = t1
        rst.Fields(3).Value = t2
        rst.Update
    Next i

    ' закрытие наборов
    rst.Close
    Set rst = Nothing
    Set CurDB = Nothing

Exit_Кнопка13_Click:
    Exit Sub
```

```
Err_Кнопка13_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка13_Click
End Sub

Private Sub Кнопка6_Click()
On Error GoTo Err_Кнопка6_Click


   DoCmd.Close

Exit_Кнопка6_Click:
   Exit Sub

Err_Кнопка6_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка6_Click

End Sub
```

Трубы, редактирование:

```
' расчет среднегодовых значений температуры теплоносителя
' в подающем и обратном трубопроводах сети
Private Sub Кнопка8_Click()
On Error GoTo Err_Кнопка8_Click

   Dim rst As DAO.Recordset
   Dim i As Integer
   Dim t1 As Double
   Dim t2 As Double

   ' обновляем данные запроса
   Me.Refresh
   ' выходим из процедуры
   Exit Sub

      ' устанавливаем начальные значения
   Set rst = Me.Recordset
   t1 = 0
   t2 = 0
   ' переходим к первой строке
   rst.MoveFirst
   ' перебираем все месяцы
   For i = 1 To rst.RecordCount
      t1 = t1 + rst.Fields(1).Value
      t2 = t2 + rst.Fields(2).Value
      rst.MoveNext
   Next i
```

```
    ' вычисляем средние значения
    t1 = t1 / 12#
    t2 = t2 / 12#
    ' запись данных в таблицу параметров
    DoCmd.RunSQL ("UPDATE Параметры SET Параметры.t1 = " + _
        Str(t1) + ", Параметры.t2 = " + Str(t2) + ";")

    Exit_Кнопка8_Click:
    Exit Sub

Err_Кнопка8_Click:
    MsgBox Err.Description
    Resume Exit_Кнопка8_Click
End Sub
Option Compare Database

Private Sub Form_AfterUpdate()
    Me.Refresh
End Sub

Private Sub Кнопка24_Click()
On Error GoTo Err_Кнопка24_Click


    DoCmd.DoMenuItem acFormBar, acEditMenu, 8, , acMenuVer70
    DoCmd.DoMenuItem acFormBar, acEditMenu, 6, , acMenuVer70

Exit_Кнопка24_Click:
    Exit Sub

Err_Кнопка24_Click:
    'MsgBox Err.Description
    Resume Exit_Кнопка24_Click

End Sub
Private Sub Кнопка26_Click()
On Error GoTo Err_Кнопка26_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = ChrW(1053) & ChrW(1086) & ChrW(1074) & ChrW(1072) & ChrW(1103)
& ChrW(1058) & ChrW(1088) & ChrW(1091) & ChrW(1073) & ChrW(1072)
    DoCmd.OpenForm stDocName, , , stLinkCriteria
    KodGruppi = 1

Exit_Кнопка26_Click:
    Exit Sub
```

```
Err_Кнопка26_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка26_Click

End Sub
Private Sub Кнопка27_Click()
On Error GoTo Err_Кнопка27_Click

   Dim stDocName As String
   Dim st As Integer

   ' если не задана группа, то ничего не делать
   If (IsNull(Forms![ГруппыРедактирование]![КодГруппы])) Then
      MsgBox ("Название группы не заполнено!")
      Exit Sub
   End If

   stDocName = ChrW(1052) & ChrW(1072) & ChrW(1082) & ChrW(1088) & ChrW(1086)
& ChrW(1089) & ChrW(49)
   DoCmd.RunMacro stDocName

Exit_Кнопка27_Click:
   Exit Sub

Err_Кнопка27_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка27_Click

End Sub

Private Sub ПолеСоСписком30_AfterUpdate()
On Error GoTo Err_ПолеСоСписком30_AfterUpdate

   Me.ДиаметрНаружний.Value = (Me.ПолеСоСписком30.Column(2))

Exit_ПолеСоСписком30_AfterUpdate:
   Exit Sub

Err_ПолеСоСписком30_AfterUpdate:
   'MsgBox Err.Description
   Resume Exit_ПолеСоСписком30_AfterUpdate
End Sub
```

Результатом расчета данных является документ «Сводные результаты расчета» (рис. 1.15). Окно редактирования отчета в режиме конструктора приведено на рис. 1.16.

Сводные результаты расчета

Параметры расчета:

норма среднегодовой утечки теплоносителя 0,25
среднегодовая плотность теплоносителя 1000 **кг/м³**
удельная теплоемкость теплоносителя 1 **ккал/кг °С**
среднегодовое значение температуры грунта 5,1 ***°С***
среднегодовое значение температуры воздуха -0,7 ***°С***
Среднегодовые значения температуры теплоносителя в тепловой
в подающем трубопроводе: ***66,1 °С*** ***в обратном трубопроводе:*** ***53,2 °С***

температура холодной воды, поступающей на источник теплоснабжения в отопительный 5***°С***
температура холодной воды, поступающей на источник теплоснабжения в летний 15***°С***
продолжительность функционирования сети в отопительный период 5352 **ч**
продолжительность функционирования сети в летний период 3048 **ч**
доля массового расхода теплоносителя, теряемого подающим трубопроводом 0,75
Значение материальной характеристики подземной **0,20** **3,99** ***м²***
Значение материальной характеристики надземной **0,80** **16,42** ***м²***

Тип прокладки надземная **Протяженность: 76,00** ***м***

Наименование	**D нар. (мм)**	**D ус. (мм)**	**L (м.п.)**	**Год**	**V л. (м³)**	**V от. (м³)**	**V г. (м³)**	**G ут.н.**	**m**	**Q у.н.**	**betta**	**q из.п.**	**q из.об.**	**Q под.**	**Q обр.**
Участок 1	108	100	76	2007	0,96	1,19	1,11	23,25	0,003	1,26	1,25	20,186	16,832	0,0019	0,0016

Часовые тепловые потери по проектным нормам: Q из.под.: 0,0019 **Гкал/ч, Q** 0,0016 **Гкал/ч;**

Тип прокладки подземная канальная **Протяженность: 15,00** ***м***

Наименование	**D нар. (мм)**	**D ус. (мм)**	**L (м.п.)**	**Год**	**V л. (м³)**	**V от. (м³)**	**V г. (м³)**	**G ут.н.**	**m**	**Q у.н.**	**betta**	**q из.п.**	**q из.об.**	**Q**
Участок 2	133	125	15	2007	0,29	0,37	0,34	7,17	0,001	0,39	1,2	34,264		0,0006

Часовые тепловые потери по проектным нормам: Q из.под.: 0,0006 **Гкал/ч, Q** **Гкал/ч; суммарные потери: Q из.под.:** 0,0009 **Гкал/ч,**

Значения материальной характеристики трубопроводов тепловой сети:
- подземная прокладка: **3,99** ***м²***
- надземная прокладка: **16,42** ***м²***

среднегодовое значение температуры грунта (t_гр): 5,1 ***°С***
среднегодовое значение температуры воздуха (tвозд): -0,7 ***°С***
Среднегодовые значения температуры теплоносителя в тепловой сети:

Нормативные значения часовых тепловых потерь через изоляционные конструкции:
- подземная прокладка: **0,000617 Гкал/ч**
- надземная прокладка (подающие): **0,001918 Гкал/ч**
- надземная прокладка (обратные): **0,001599 Гкал/ч**

в подающем трубопроводе: 66,1 °С
в обратном трубопроводе: 53,2 °С

Протяженность (общая): 91,00 ***м***
Протяженность (надземная часть): 76,00 ***м***
Протяженность (подземная бесканальная часть): 0,00 ***м***
Протяженность (подземная канальная часть): 15,00 ***м***

Среднегодовая емкость тепловой сети (V год.): 1,45 ***м³***
Среднегодовая емкость тепловой сети в отопительный период (V от.): 1,56 ***м³***
Среднегодовая емкость тепловой сети в неотопительный период (V л.): 1,25 ***м³***
Нормативные значения годовых потерь теплоносителя с его утечкой (G) 30,42 ***м³***

ИТОГО: **Суммарные часовые тепловые потери (изоляция):**
Q из.: 0,004133 **Гкал/ч**

Суммарные годовые тепловые потери по проектным нормам:
Q из.: 34,721080 **Гкал**
Утечка Q у.н.: 1,650217 **Гкал**

Рис. 1.15. Сводные результаты расчета

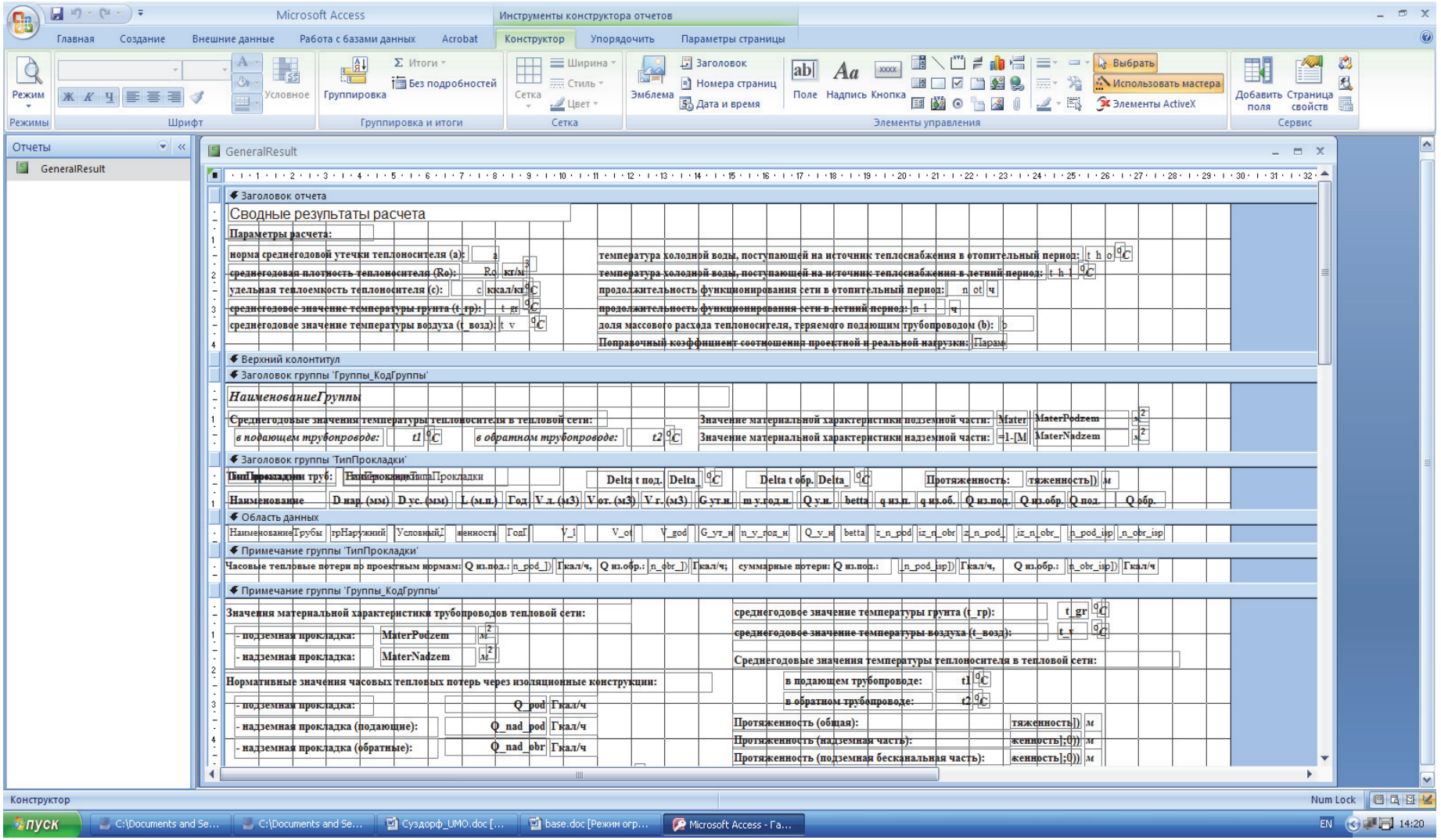


Рис. 1.16. Режим конструктора отчетов

Контрольные вопросы:

1) Назовите уровни обобщения данных для принятия решений.
2) Раскройте смысл выражения «семантика данных».
3) Дайте определение терминам «база данных» и «банк данных».
4) Какова роль информационных хранилищ?
5) Что такое модель данных?
6) Повторите «для себя» определения: схема отношений, атрибут, кортеж, ключ.
7) В чем смысл нормализации отношений?
8) Чем отличаются языки QBE и SQL?
9) Что такое транзакция?
10) Охарактеризуйте идеологию создания распределенных БД.
11) Опишите сервер БД в сетевой архитектуре предприятия.

2. СРЕДСТВА ПРОЕКТИРОВАНИЯ ДИСПЕТЧЕРСКОГО УРОВНЯ УПРАВЛЕНИЯ ТЕХНИЧЕСКИМИ СИСТЕМАМИ

2.1. Системы сбора, хранения и визуализации данных

В автоматизированных системах управления технологическими процессами (АСУТП), часто называемых системами промышленной автоматизации, можно выделить свои иерархические уровни.

На *верхнем* (диспетчерском) *уровне* АСУТП осуществляются сбор и обработка данных о состоянии оборудования и протекании производственных процессов для принятия решений по загрузке станков и выполнению технологических маршрутов. Эти функции возложены на систему диспетчерского управления и сбора данных, называемую **SCADA** (Supervisory Control and Data Acquisition). Кроме диспетчерских функций система SCADA выполняет роль инструментальной системы разработки программного обеспечения (ПО) для промышленных систем компьютерной автоматизации.

На *уровне управления технологическим оборудованием* (на уровне контроллеров) в АСУТП выполняются: запуск, тестирование, выключение станков, сигнализация о неисправностях, выработка управляющих воздействий для рабочих органов программно-управляемого оборудования. Для этого в составе технологического оборудования используются системы управления на базе программируемых контроллеров – компьютеров, встроенных в технологическое оборудование, поэтому системы промышленной автоматизации часто называют ***встроенными системами*** (Embedded Computing Systems).

Техническое обеспечение АСУТП представлено персональными ЭВМ и микрокомпьютерами, распределенными по контролируемым

участкам производства и связанными друг с другом с помощью промышленных шин. Программное обеспечение АСУТП представлено операционными системами, программами SCADA, драйверами и прикладными программами контроллеров.

Функции систем SCADA:

- сбор первичной информации от датчиков;
- хранение, обработка и визуализация данных;
- управление и регистрация аварийных сигналов;
- связь с корпоративной информационной сетью;
- автоматизированная разработка прикладного ПО.

SCADA-системы состоят из терминальных компонентов, диспетчерских пунктов и каналов связи. Различаются SCADA-системы типами поддерживаемых контроллеров и способами связи с ними, операционной средой, типами алармов, числом трендов (тенденций в состоянии контролируемого процесса) и способом их вывода, особенностями человеко-машинного интерфейса (HMI) и др.

Связь с контроллерами и приложениями в SCADA-системах обычно осуществляется посредством технологий DDE, OLE, OPC или ODBC. В качестве каналов связи используют последовательные промышленные шины Profibus, CANbus, Foundation Fieldbus и др.

При выходе значений контролируемых параметров или скоростей их изменения за границы допустимых диапазонов фиксируются **алармы**.

Число одновременно выводимых трендов может быть различным, их визуализация возможна в реальном времени или с предварительной буферизацией. Предусматриваются возможности интерактивной работы операторов.

Разработка программ для программируемых контроллеров выполняется на языках C/C++, VBA или оригинальных языках, разработанных для конкретных систем. Программирование обычно выполняют не профессиональные программисты, а заводские технологи, поэтому желательно, чтобы языки программирования были достаточно простыми, построенными на визуальных изображениях ситуаций. Во многих системах дополнительно используются различные схемные языки. Ряд языков стандартизован и представлен в международном стандарте IEC 1131-3.

Одной из широко известных SCADA-систем является система **Citech** австралийской компании Ci Technology, работающая в среде Windows. Это масштабируемая система «клиент-сервер» со встроенным резервированием для повышения надежности. Состоит из пяти подсистем: ввода-вывода, визуализации, оповещения (алармов), трендов, отчетов. Подсистемы могут быть распределены по разным узлам сети. Используется оригинальный язык программирования – **Cicode**.

SCADA-система **Trace Mode** для крупных АСУТП в различных отраслях промышленности и в городских службах создана компанией AdAstra. Система состоит из инструментальной части и исполнительных модулей. Предусмотрены управление технологическими процессами, разработка автоматизированного рабочего места (АРМ) руководителей цехов и участков, диспетчеров и операторов. Возможно использование операционных систем QNX, OS9, Windows.

Другой пример популярной SCADA-системы – **BridgeVIEW** (другое название – Lab VIEW SCADA) компании National Instruments. Ядро системы управляет базой данных, взаимодействует с серверами устройств, реагирует на алармы. Подсистема HMI предназначена для интерфейса с пользователями и для исполнения задаваемых ими программ. При настройке системы на конкретное приложение пользователь конфигурирует входные и выходные каналы, указывая для них такие величины, как частота опроса, диапазоны значений сигнала и другие и создает программу работы приложения. Программирование ведется на графическом языке блок-диаграмм.

С развитием сетевой инфраструктуры появляется возможность более тесной интеграции автоматизированных систем управления предприятием (АСУП) и АСУТП, ранее развивавшихся автономно. И спользование информации непосредственно от технологических процессов позволяет более рационально планировать производство и управлять предприятием. Интеграция выражается в использовании на этих уровнях общих программных средств, баз данных, связей с Internet на основе развития PC-совместимых контроллеров и сетей Industrial Ethernet и т.п.

К оперативной системе (ОС) реального времени предъявляется ряд специфических требований, основными из них являются требования высокой скорости реакции на запросы внешних устройств, устойчивости системы (т.е. способности работы без зависаний) и экономного использования имеющихся в наличии системных ресурсов.

В SCADA-системах в основном применяют операционные системы UNIX или Windows NT.

Операционные системы Windows NT и Windows 2000 оказывается возможным использовать в системах реального времени, дополнив их, например, средой RTX компании VenturCom. Развитый программный интерфейс RTX API, основанный на Win32 AIM, обеспечивает создание драйверов и приложений реального времени. Кроме того, Microsoft разработала специальную версию Windows NT для встроенных приложений Windows NT Embedded.

Перспективной считается ОС LynxOS – многозадачная, многопользовательская, UNIX совместимая система. Есть средства кросс-разработки программ. Сетевые средства предусмотрены для TCP/IP, ATM, FR, ISDN и др.

Авторы одной из концепций построения АСУТП рекомендуют ОС OS-9, QNX или расширения Windows NT для реального времени в случае CompactPCI и ОС QNX или VxWorks в случае использования аппаратуры на базе VMEbus.

Другая популярная ОС для встраиваемых приложений OS-9 относится к многозадачным, многопользовательским системам реального времени. В системе поддерживаются коммуникационные протоколы X.25, FR, ATM, ISDN, SS7 и др. Для разработки приложений в OS-9 имеется интегрированная кросс-среда Hawk, она включает: редактор, браузер исходных кодов, отладчики, компиляторы C/C++.

ОС QNX канадской фирмы QSSL – открытая, модульная и легко модифицируемая, она функционирует в «защищенном режиме», поддерживает шины ISA, PCI, CompactPCI, PC/104, VME, STD32 и др.

ОС реального времени VxWorks выполняет функции планирования и управления задачами. Может функционировать в мультипроцессорных системах как с общей памятью, так и в слабосвязанных с использованием распределенных очередей сообщений. Система VxWorks поддерживает все сетевые средства, обычные для UNIX, а также OPC-интерфейсы (OLE for Process Control). Она вместе с инструментальной системой Tornado является кросс-системой для разработки прикладного ПО.

Для разработки ПО реального времени используют пакеты типа Component Integrator. К числу известных комплексов Component Integrator относятся: FIX, Factory Suite 2000, ISaGRAF и др. Назначение прикладного ПО – анализ производства, воздействие на него в реальном времени.

Комплекс Factory Suite 2000 компании WonderWare, используемый при проектировании систем промышленной автоматизации от АСУТП до АСУП, включает в себя следующие подсистемы:

- InTouch 7.0 – SCADA-система – для создания распределенных приложений, визуализации процессов управления;
- InControl – для управления контроллерами;
- InTrack – для управления производством (контроль материально-технических запасов, незавершенного производства, загрузки оборудования). В частности, подсистема InTrack интегрирована в известную систему планирования ресурсов предприятия iBaan;
- InBatch – для управления процессами непрерывного производства;
- Industrial SQL Server – для хранения статистики, истории производственного процесса, данных о ситуациях;
- Scout – удаленный доступ к данным о технологическом процессе.

Одной из развитых инструментальных сред разработки приложений реального времени является система **Tornado**, разработанная для мультизадачной ОС VxWorks фирмой Wind River. Разработка приложений ведется на инструментальном компьютере, которым могут быть ПЭВМ или

рабочие станции Sun, HP, IBM, DEC. В базовую конфигурацию Tornado входят: компиляторы C/C++, отладчики, симулятор целевой машины, командный интерпретатор, браузер объектов целевой системы, средства управления проектом и др. Для разработки ПО для встраиваемых сигнальных процессоров Tornado применяют вместе со специальной ОС WISP. Инструментальная среда Tornado Prototyper и симулятор ОС VxWorks, работающий под Windows, могут быть получены бесплатно по Internet, что позволяет провести предварительную разработку прикладной программы, а уже затем закупать полную версию кросс-системы.

Инструментальную среду ISaGRAF используют для разработки прикладного ПО программируемых контроллером PLC. Среда реализует методологию граф-схем Flowchart и пять языков программирования по стандарту МЭК61131-3 (IEC 1131-3). Это графические языки функциональных схем SFC, блоковых диаграмм FBD, диаграмм релейной логики LD и текстовые языки – паскалеподобный ST и низкоуровневый язык инструкций IL.

2.2. Программное обеспечение автоматизации проектирования АСУ и АСУТП

GENESIS-32 является набором 32-разрядных приложений для Windows 95, Windows 98 и Windows NT, построенных в соответствии со спецификацией OPC, который предназначен для создания программного обеспечения сбора данных и оперативного диспетчерского управления верхнего уровня систем промышленной автоматизации. В состав GENESIS-32 также входит среда редактирования сценарных процедур Advanced VBA Scripting, обеспечивающая возможность разработки части ПО средствами Microsoft Visual Basic for Applications 5.0 (Visual Basic для приложений), входящего в популярный пакет MS Office 97. Все программные компоненты реализованы на базе многопоточной модели и поддерживают технологию ActiveX.

OLE™ for Process Contro (OPC™) (механизм связывания и внедрения объектов для сбора данных и управления в системах промышленной автоматизации) является наиболее общим способом организации взаимодействия между различными источниками и приемниками данных, такими как устройства, базы данных и системы визуализации информации о контролируемом объекте автоматизации. OPC обеспечивает интерфейс между приложениями-клиентами и серверами путем реализации стандартного механизма связи между источниками данных (серверами) и получателями данных (клиентами). Иными словами, OPC является аналогом технологии Plug-n-Play для программного обеспечения, выполняющего функции человеко-машинного интерфейса в сфере промышленной автоматизации.

В традиционной идеологии «клиент-сервер» различные серверы или устройства имеют различные интерфейсы или драйверы для каждого приложения-клиента. Поскольку аппаратные средства разных производителей имеют различные и притом фиксированные протоколы обмена, архитектура приложений-клиентов также является уникальной в каждом конкретном случае. Это приводит к увеличению времени разработки и стоимости АСУТП, а любое изменение, внесенное производителем в устройство или протокол обмена, требует внесения изменений в функционирующую систему. Архитектура «клиент-сервер», основанная на технологии OPC, позволяет решить данную проблему.

В данном случае устройство каждого производителя имеет единственный стандартный драйвер, совместимый с OPC (OPC-сервер). Приложения, соответствующие спецификации, выработанной для клиента OPC (OPC-клиенты), могут при этом обмениваться данными с устройствами любого производителя при наличии OPC-совместимых драйверов для указанных устройств. Уже в настоящее время производителям аппаратных средств предлагается свыше 2000 серверов OPC.

В отличие от многих известных SCADA-систем, имеющих либо собственный формат драйверов аппаратуры, либо встроенную поддержку аппаратуры ограниченного числа производителей, GENESIS-32 представляет наиболее универсальный способ взаимодействия с аппаратными средствами любого производителя. Для фирм, занимающихся самостоятельным производством устройств сбора данных и управления, GENESIS-32 предлагает комплект разработчика OPC ToolWorX, который позволяет в кратчайшие сроки создавать серверы OPC для собственных технических средств. При этом разработанный OPC-сервер будет совместим с любыми приложениями-клиентами, поддерживающими спецификацию OPC 1.1 и выше.

Приложения, соответствующие спецификации клиента OPC, исполняющиеся на рабочих станциях, которые объединены в локальную вычислительную сеть, могут иметь доступ к каналам ввода/вывода аппаратуры, обслуживаемой серверами OPC, которые исполняются на любых узлах сети. Одно из уникальных качеств, присущих данной технологии, состоит в том, что клиенты OPC имеют возможность получения данных от удаленных серверов OPC через глобальную сеть Интернет. Разработчики систем промышленной автоматизации по достоинству оценили указанную функциональную возможность. Теперь не придется выезжать к заказчикам, расположенным за тысячи километров, для контроля состояния технических средств системы и модификации реализованных системных функций. Все эти операции могут быть выполнены с помощью браузера Интернет и GENESIS-32.

GENESIS-32 включает в себя следующие приложения, являющиеся клиентами OPC:

- GraphWorX 32;
- TrendWorX 32;
- AlarmWorX 32.

GENESIS-32 также содержит среду разработки сценарных процедур VBA Scripting, которая является сервером OPC.

Кроме того, в состав пакета входят сервер системного администрирования Security Config и сервер фоновой архивации данных Persistent Trending.

2.2.1. GraphWorX32

GraphWorX32 является инструментальным средством, предназначенным для визуализации контролируемых технологических параметров и оперативного диспетчерского управления на верхнем уровне АСУТП, которое полностью соответствует требованиям к клиенту OPC и поддерживает технологии ActiveX и OLE.

Основные характеристики GraphWorX32:

- многопоточное 32-разрядное приложение;
- возможность обмена данными с любыми серверами OPC;
- мощные инструменты для создания экранных форм и динамических элементов отображения;
- возможность встраивания элементов управления ActiveX и объектов OLE;
- встроенная среда редактирования сценарных процедур Microsoft Visual Basic for Applications 5.0;
- динамизация элементов отображения с временем обновления графической информации 50 мс;
- средства разработки шаблонов экранных форм, содержащих наиболее часто используемые слои графических объектов;
- возможность встраивания в HTML-страницы и серверы OLE (MS Word, MS Excel, MS Access и др.);
- возможность просмотра браузерами Интернет, такими как MS Internet Explorer;
- обширная библиотека элементов отображения, ориентированных на построение мнемосхем промышленных объектов;
- возможность встраивания графиков TrendWorX32 и журналов событий и тревог AlarmWorX32;
- средства импорта графических метафайлов (WMF) и растровых изображений (BMP).

Набор средств разработки графических форм отображения.

GraphWorX32 имеет в своем составе полный набор средств рисования и анимации, объединенных в объектно-ориентированную среду разработки технологической графики.

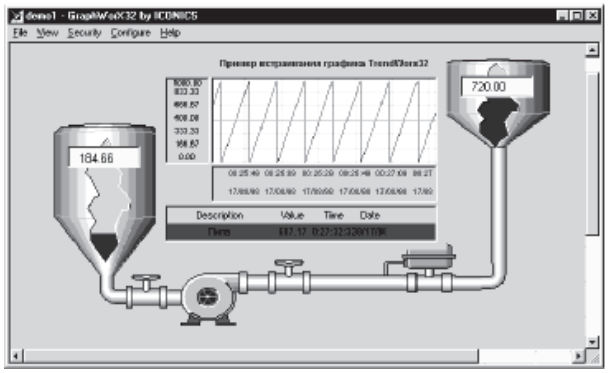


Рис. 2.1. Экранная форма

Проект состоит из совокупности экранных форм, каждая из которых хранится в отдельном файле в каталоге проекта. Разработка экранных форм выполняется с помощью инструментов рисования, встраивания графических изображений из метафайлов и файлов растровых изображений, а также путем использования обширной библиотеки встроенных стандартных символов. Каждый элемент отображения может быть связан с переменной процесса и отображать ее значение или состояние (рис. 2.1). Переключение между окнами во время исполнения проекта с загрузкой соответствующего файла экранной формы может выполняться путем нажатия командных кнопок при возникновении различных событий в контролируемой прикладной области либо по команде из сценарной процедуры.

Средства анимации.

Экранные формы, создаваемые с помощью GraphWorx32, могут быть насыщены графическими объектами и элементами отображения, форма и положение которых зависят от значений и состояний контролируемых параметров в прикладной области, а также от действий персонала, эксплуатирующего систему. Скорость обновления анимации составляет 50 мс, что стало возможным благодаря применению самых передовых технологий в области объектно-ориентированной графики и 32-разрядного многопоточного программирования. Любому элементу отображения и графическому объекту может быть поставлен в соответствие один или несколько способов динамизации (изменение размера по вертикальной и горизонтальной осям, изменение цвета, перемещение, вращение и др.) в зависимости от связанного с ним параметра. Средства установления связи между каналами ввода-вывода устройств и другими переменными проекта и элементами отображения позволяют задавать закон предварительной обработки параметра, представляемый в виде комбинации арифметических, логических, бинарных и функциональных преобразований, а также условных операций.

Библиотека встроенных символов технологической и деловой графики.

GraphWorx32 содержит библиотеку символов технологической и деловой графики, которая позволяет значительно улучшить внешний вид экранных форм и создавать интуитивно понятные мнемосхемы автоматизируемых технологических процессов в кратчайшие сроки. Одним из наиболее привлекательных качеств библиотеки символов является возможность ее расширения и дополнения пользователем путем создания собственных символов.

2.2.2. TrendWorX32

TrendWorX32 является многооконным приложением, которое предназначено для выполнения следующих ***функций***:

- представление контролируемых параметров в виде графиков (трендов) различных типов в реальном масштабе времени;
- архивирование значений контролируемых параметров;
- вычисление статистических характеристик выборок значений контролируемых параметров;
- извлечение значений контролируемых параметров из архивов и представление в виде графиков различных типов;
- вывод графиков на печатающее устройство.

TrendWorX32 является клиентом OPC, поддерживающим технологии ActiveX и OLE, и может использоваться как совместно с другими компонентами GENESIS-32, так и с приложениями других производителей.

TrendWorX32 является многооконным контейнером ActiveX, который может запускаться как автономно, так и одновременно с другими компонентами GENESIS-32. Каждое окно TrendWorX32 содержит элементы управления TWXView32 ActiveX, с помощью которых выполняется графическое представление неограниченного количества контролируемых параметров и внутренних переменных проекта. Поскольку TrendWorX32 является клиентом OPC, то имеется возможность построения графиков значений параметров, сбор которых ведется на узлах глобальной сети Интернет и локальной вычислительной сети предприятия.

TrendWorX32 поддерживаются следующие виды трендов:

- зависимость параметра от времени;
- логарифмическая зависимость параметра от времени;
- гистограмма параметра;
- зависимость параметра от времени с использованием единиц времени в качестве вертикальной оси;
- зависимость одного параметра от другого.

Конфигурирование трендов во время исполнения.

Двойной щелчок левой клавишей мыши в окне TrendWorX32 во время исполнения приводит к появлению инструментальной панели Trend Viewer, которая позволяет выполнить конфигурирование трендов, добавить и удалить отображаемые параметры, изменить диапазоны вдоль осей, вывести статистику отображаемых параметров на текущем интервале (математическое ожидание, минимальное и максимальное значения).

TrendWorX32 позволяет во время исполнения «заморозить» картинку и выполнить детализацию требуемого фрагмента графика, что обеспечивает возможность оперативного анализа характера изменения контролируемых параметров.

Архивирование значений контролируемых параметров.

Архивирование контролируемых параметров выполняется при помощи сервера фоновой архивации Persistent Trending, который является клиентом OPC и сервером OLE Automation. Данная утилита предназначена для сбора информации о контролируемых параметрах технологического процесса и ее сохранения в специальных буферах памяти, а также в определенных пользователем файлах. Приложения-клиенты могут запрашивать данные из буферов сервера фоновой архивации посредством механизма OLE Automation, что существенно повышает эффективность их работы. Элемент управления TWXView32 ActiveX, с помощью которого отображаются графики контролируемых параметров, устанавливает связь с сервером фоновой архивации для инициализации графика каждого параметра.

Таким образом, сервер фоновой архивации обеспечивает выполнение следующих ***функций***:

- буферизация данных, собираемых в процессе работы системы от серверов OPC;
- передача буферизованных данных клиентам OLE Automation;
- регистрация данных в файлах архивов, настраиваемых пользователем.

В окне сервера Persistent Trending создаются группы параметров, для которых в процессе исполнения будут сформированы отдельные многоуровневые архивы. Для каждой группы параметров могут быть заданы различные алгоритмы записи информации в архивы. Для групп контролируемых параметров поддерживаются следующие ***методы*** архивирования:

- сохранение в одном файле всех значений параметров для выбранной группы через определенный интервал времени после запуска проекта на исполнение;
- сохранение значений параметров для выбранной группы по команде из сценарной процедуры;
- сохранение значений параметров для выбранной группы с открытием нового файла архива для каждого интервала архивации.

Для обеспечения возможности последующего использования информация в архивах представлена в символьном формате.

Вывод графиков на печатающее устройство.

TrendWorX32 позволяет выводить на печатающее устройство графики контролируемых параметров в том виде, в котором выполняется их отображение на экране монитора. При этом имеется возможность предварительного просмотра листов, которые будут выведены на печать, и выбора требуемого.

2.2.3. AlarmWorX32

AlarmWorX32 является мультимедийным приложением, которое предназначено для выполнения следующих ***функций***:

- голосовое оповещение персонала об обнаруженных аварийных ситуациях;
- рассылка электронных извещений об аварийных событиях посредством пейджинговой связи и электронной почты;
- оповещение персонала путем автоматического дозвона по коммутируемым каналам связи с передачей сообщений об аварийных событиях и приемом подтверждений восприятия от ответственных лиц;
- персональное планирование оповещения для задействования только дежурного персонала;
- анализ аварийных событий и действий ответственного персонала;
- объединение всех аварийных событий и подтверждений восприятия системных сообщений ответственным персоналом в сводки аварийных событий;
- отображение вспомогательной информации для аварийных событий, позволяющей локализовывать и устранять причины аварии;
- связь с аппаратными средствами системы через интерфейсы OPC;
- связь с другими приложениями посредством технологии ODBC.

AlarmWorX32 является клиентом OPC, поддерживающим технологии ActiveX и OLE, и может использоваться как совместно с другими компонентами GENESIS-32, так и с приложениями других производителей.

Настройка параметров аварийных событий.

Для каждого технологического параметра, контролируемого системой, могут быть заданы условия, наступление которых воспринимается системой как аварийная ситуация. Для аналоговых сигналов могут быть определены верхние и нижние допустимые и предельные значения, зона нечувствительности, а также количество попыток оповещения, при которых в систему не вводится подтверждение от дежурного персонала. Каждому событию при этом ставится в соответствие текстовая строка, которая будет отображаться в журнале событий, а также тревожный звуковой сиг-

нал и звуковой файл, в котором содержится речевое сообщение об аварийной ситуации. Кроме того, для каждого аварийного события может быть задана краткая инструкция, которая будет доведена до дежурного персонала при возникновении аварии.

Технологические параметры, проверяемые на выполнение условий аварийных ситуаций, объединяются в группы, каждой из которых может назначаться свой приоритет, а также устанавливаться список лиц, ответственных за принятие мер по устранению причины и ликвидации последствий аварии.

Персональное планирование оповещения.

AlarmWorX32 позволяет составить персональный план оповещения для всех лиц, ответственных за принятие мер по устранению аварии.

Голосовое оповещение персонала об аварийных ситуациях.

Для каждого аварийного события может быть создан и впоследствии воспроизведен звуковой файл, содержащий речевое сообщение об аварии. Создание и воспроизведение звуковых файлов выполняются средствами ОС с помощью звуковой карты Sound-Blaster.

Отображение информации об аварийных и других событиях.

AlarmWorX32 отображает информацию об аварийных и других событиях, связанных с системой, в окнах журнала событий и архива событий. Имеется возможность просмотра сводок аварийных событий и действий персонала как в текущий момент времени, так и за прошедшее время (рис. 2.2).

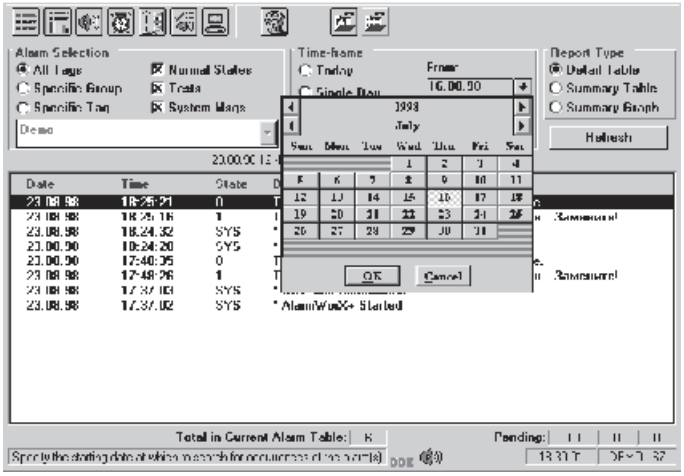


Рис. 2.2. Сводка аварийных событий за истекший период

Системное администрирование и управление правами доступа.

Пакет GENESIS-32 имеет встроенные средства системного администрирования и управления правами доступа к информации, связанной с системой. Все лица, имеющие отношение к автоматизированной системе

управления технологическим процессом, с помощью сервера системного администрирования Security Config вносятся в список допускаемых к системной информации.

Наивысшими правами доступа обладает ***системный администратор***. Остальной персонал, допущенный к системным функциям и системной информации, получает ограниченные права, настройка которых выполняется в диалоговых панелях сервера системного администрирования. Для каждого ответственного лица или группы может быть создана собственная учетная запись сервера системного администрирования, на которую распространяются права доступа к определенным технологическим параметрам, файлам, узлам сети и системным операциям, таким как нажатие кнопок, запуск и останов системы, переключение окон и т.д.

Доступ ответственных лиц к системным функциям, доступ персонала к технологическим параметрам.

Для учетных записей сервера системного администрирования могут быть созданы еженедельные и посменные расписания доступа к системной информации и системным функциям, а также разработана политика, в соответствии с которой определяются период действия присвоенного учетной записи пароля, минимальная длина пароля, период времени, по истечении которого возобновляется действие ранее использованного пароля. Весьма важной функцией, определяемой при настройке политики администрирования учетной записи, является блокировка всех действий пользователя после ряда ошибочных или злонамеренных действий.

Среда редактирования сценарных процедур VBA Scripting.

Visual Basic для приложений (VBA) является мощной средой программирования, которая фактически приобрела статус промышленного стандарта. Microsoft VBA обеспечивает наиболее доступный и быстрый способ разработки и модернизации приложений Microsoft Windows. GENESIS-32 поставляется со встроенной средой разработки и исполнения сценарных процедур Microsoft Visual Basic for Applications 5.0, которая позволяет адаптировать пакет GENESIS-32 в соответствии со специфическими требованиями к создаваемой системе промышленной автоматизации. VBA позволяет:

- разрабатывать, отлаживать и запускать на исполнение сценарные процедуры и макрокоманды;
- создавать процедуры обработки событий, связанных с программными компонентами GENESIS-32;
- модифицировать встроенные объекты программных компонентов GENESIS-32;
- устанавливать связь и обмениваться данными между объектами ActiveX и встроенными объектами программных компонентов GENESIS-32;

- обмениваться данными с другими приложениями Windows и драйверами аппаратуры.

Поддержка аппаратуры. Серверы OPC.

В настоящее время производителями контроллеров и других аппаратных средств разработано свыше 1000 серверов OPC, что стало возможным благодаря наличию комплекта разработчика OPC ToolWorX. Данный комплект содержит пример проекта сервера OPC для промышленной шины Modbus, а также простое клиентское приложение, позволяющее проверить работоспособность сервера. OPC ToolWorX нашел признание у производителей аппаратуры за счет того, что с его помощью можно создавать серверы OPC, совместимые с любыми приложениями-клиентами, соответствующими спецификации OPC 1.1 и выше.

Назначение основных серверов OPC:

- GEN-OPC – для организации взаимодействия между программными компонентами GENESIS-32 и драйверами аппаратуры GENESIS 3.xx;
- GEN DDE – для обмена данными между программными компонентами GENESIS-32 и серверами динамического обмена данными (DDE), что позволяет устанавливать связь между GENESIS-32 и любыми 32- и 16-разрядными приложениями Windows, поддерживающими механизм DDE, а также с аппаратными средствами, для которых поставляются серверы DDE.

2.2.4. SCADA-система Trace Mode

Программа Trace Mode фирмы AdAstra Research Group к настоящему времени обладает мощной графикой и автоматизирует полный цикл разработки АСУ, начиная с настройки на устройства телемеханики и кончая генерацией отчетов. Все этапы разработки АСУ осуществляются в графических редакторах без использования языков программирования. Для отладки АСУ не требуется покидать систему или подключать контроллеры – Trace Mode может эмулировать работу создаваемой АСУ.

Разработка системы автоматизации так называемого «верхнего уровня» (рабочее место оператора) в системе Trace Mode осуществляется в три этапа. Сначала разрабатывается математическая основа АСУ. В объектном «редакторе базы каналов» Trace Mode следует «набросать» общую структуру системы автоматизации технологического объекта, а именно перечислить количество цехов, установок в них, указать число измеряемых и контролируемых параметров. На основании этих данных в Trace Mode создается граф объектов, определяющий информационную структуру создаваемой АСУ. Обычно начальные сведения, необходимые для создания графа, уже содержатся в проекте, поэтому разработка информационной структуры не вызывает сложностей.

Каждый объект Trace Mode включает в себя «каналы», связанные с устройствами ввода-вывода информации с технологического объекта.

Канал – это базовое понятие всей системы. Данные с внешних устройств записываются в каналы. Данные из каналов посылаются на внешние устройства и выводятся в различных формах на экран монитора. В каналы оператор заносит управляющие данные. Значения из каналов записываются в архивы, оперативные отчеты и все генерируемые документы. Все преобразования данных происходят в каналах. Меняя значения на системных каналах, можно управлять выводимой на экран информацией, звуковыми эффектами, архивами и другим, т.е. всей системой.

Совокупность всех каналов – ***база каналов*** – составляет математическую основу системы управления. Конфигурирование базы каналов осуществляется в редакторе базы каналов.

Настройка каналов для приема и обработки данных осуществляется в редакторе базы каналов (РБК).

Каналом в Trace Mode называется информационная структура, включающая в себя совокупность переменных, методов формирования и преобразования значений этих переменных, а также ряд констант. Каждому каналу приписаны три основных и несколько вспомогательных переменных. В АСУТП, часто называемых системами промышленной автоматизации, можно выделить свои иерархические уровни.

Атрибуты могут иметь начальные значения, задаваемые в РБК, и могут изменяться в реальном времени.

В системе Trace Mode значения каналов подразделяются на следующие три типа:

1) *аппаратные* (_A_),
2) *реальные* (_R_),
3) *пользовательские* (_U_).

Значения канала также являются его атрибутами.

Аппаратное и реальное значения одного канала связаны между собой процедурой трансляции. Пользовательское значение канала может формироваться либо процедурой преобразования по значениям атрибута каналов, либо копированием его реального значения. Копирование реального значения в пользовательское осуществляется при отключенной процедуре преобразования. Пользовательское значение может быть скопировано в реальное значение того же канала или в аппаратное значение другого канала.

Таким образом, можно выделить пять основных составляющих канала: три значения (аппаратное, реальное и пользовательское) и две процедуры (трансляция и преобразование). Процедура трансляции связывает между собой аппаратное и реальное значения одного канала. Процедура пре-

образования формирует пользовательское значение канала по значениям атрибутов одного или нескольких каналов.

Каналы в Trace Mode могут иметь разные назначения: от приема и обработки данных с внешних устройств до коррекции системных параметров, таких как время цикла системы, период архивирования и др. Назначение канала определяется его классом, типом, подтипом и дополнениями к подтипу. Класс, тип, подтип и дополнения к подтипу канала задаются в РБК.

Все каналы Trace Mode делятся на два класса: внешние и внутренние.

Внешние каналы предназначены для связи с аппаратурой (контроллеры, УСО (устройство связи с оборудованием) и др.), обмен данными с которой не поддерживается внутри системы и требует использования специальной программы – *драйвера*. Связь внешних каналов с драйвером всегда осуществляется через аппаратные значения.

Внутренние каналы используются для получения данных и управления устройствами, поддержка которых реализована внутри системы. Кроме того, внутренние каналы предназначены для управления задачами сбора, обработки и архивирования данных, вывода данных на экран и другими системными задачами. Все операции по внутренним каналам производятся внутри системы без использования внешних программ.

В зависимости от направления движения информации (от аппаратного значения через процедуру трансляции к реальному или наоборот) каналы подразделяются на входные (тип INPUT) и выходные (тип OUTPUT). Для внешних каналов это имеет чисто физический смысл: информация либо принимается от драйвера, связанного с внешним устройством, либо передается ему. Аппаратные значения внутренних каналов типа INPUT формируются системой и определяются подтипом каналов. Это может быть текущее время, код клавиши, нажатой на клавиатуре, состояние системы, данные из сети или от плат ввода измерительных сигналов, поддерживаемых Trace Mode и т.п.

Внутренние каналы типа OUTPUT позволяют управлять системными задачами, передавать данные по сети и внешним устройствам, поддержка которых реализована в Trace Mode. В зависимости от подтипа каналов типа OUTPUT аппаратные значения, сформированные на этих каналах, могут определять выводимый на экран графический фрагмент, проигрываемый звуковой файл, управлять архивами, режимом выхода из системы и т.д.

В системе Trace Mode предусмотрены два типа выходных каналов:

1) A_OUTPUT;
2) M_OUTPUT.

Тип выходного канала определяет, каким образом на нем формируется реальное значение.

У канала типа A_OUTPUT реальное значение формируется процедурой *автоматическое управление* по одному из законов автоматического управления (П, ПИ, ПИД, позиционный и т.п.).

Автоматическое управление – это еще одна из основных процедур канала. Она формирует по определенным законам реальные значения канала типа A_OUTPUT из значений атрибутов других каналов.

Реальное значение канала типа M_OUTPUT может формироваться несколькими способами:

- вручную оператором;
- копированием пользовательского значения;
- по заранее сформированной таблице;
- по сети;
- интерпретатором формул;
- другим каналом.

Кроме описанных выше в Trace Mode предусмотрен еще один тип каналов – GENERATE. Каналы этого типа являются генераторами: с их помощью можно смоделировать физический процесс во время отладки системы. Кроме того, данный тип каналов используется для создания анимационных эффектов.

Аппаратные значения каналов типа GENERATE генерируются системой по математическим законам (синус, косинус, пила, импульс и прочее, всего 16). В остальном каналы этого типа работают так же, как каналы типа INPUT: реальные значения формируются из аппаратных процедурой трансляции, затем пользовательские из реальных процедурой преобразования.

Все внутренние каналы Trace Mode имеют подтипы.

Для каналов типа INPUT подтип определяет характеристику получаемой ими информации (подтип СОРУ FROM – значение у другого канала, СТАТУС – состояние системы, ПЕРИОД – временные настройки системы и прочее, всего 16 подтипов).

Подтип внутренних каналов A_OUTPUT и M_OUTPUT определяет, чем управляет конкретный канал (подтип ЗВУК – номер проигрываемого звукового файла, GOTO – выводимый на экран графический фрагмент, ПЕРИОД – временные настройки системы и прочее, всего 16 подтипов).

Для каналов типа GENERATE подтип определяет математический код, в соответствии с которым генерируются аппаратные значения каналов.

Большинство подтипов каналов имеют дополнения к подтипу. С помощью этих дополнений настраиваются или уточняются выполняемые данным каналом функции. Так, например, для подтипа канала FROM NET с помощью трех дополнений задается сетевой адрес и тип источника данных, а для подтипа ИЗМЕНИТЬ – имя канала и изменяемый атрибут канала.

Значения канала могут быть представлены в одном из следующих форматов:

- число с плавающей точкой одинарной точности – предназначен для хранения и обработки аналоговых переменных;
- 24-битовое целое число – предназначен для работы с дискретными переменными и для осуществления логических операций.

Применительно к каналам, получающим данные от аппаратуры ввода/вывода (тип INPUT), трансляция служит для первичной математической обработки поступающих данных (исправление систематических ошибок датчиков и масштабирование, фильтрация данных, коррекции температуры холодных спаев или расчет значений по аппроксимирующему полиному для термопар и т.д.). В этом случае процедура трансляции осуществляет преобразование полученных аппаратных данных в реальные физические значения.

Если канал является выходным (тип M_OUTPUT или A_OUTPUT) и передает данные на аппаратуру ввода/вывода, тогда процедура трансляции реализует обратную задачу – преобразование реальных физических значений к виду, воспринимаемому внешними устройствами. Для внутренних каналов процедура трансляции не имеет столь явно выраженной функции, как для внешних каналов – она просто дает возможность ввести дополнительное преобразование данных при реализации алгоритмов управления или вычислений.

Процедура преобразования предназначена для вычисления пользовательского значения текущего канала по математическим законам из реальных значений указанных каналов. Законы, по которым осуществляются эти вычисления, выбираются из предлагаемого списка в редакторе базы каналов.

Если встроенные математические функции не позволяют осуществить необходимое преобразование, то можно использовать интерпретатор ТехноБЕЙСИК. Для этого необходимо в файл с именем, совпадающим с именем базы каналов, и с расширением **frm** записать формулы, реализующие требуемые вычисления. После этого процедурой трансляции или преобразования следует обращаться к строке данного файла, в которой записано требуемое выражение или группа выражений.

Кроме того, с помощью процедуры преобразования можно обратиться к одной из трех математических моделей. Математическая модель разрабатывается на языке **Си** и оформляется как резидентная программа. При этом надо согласовать структуру обмена данными этой программы с Trace Mode. Независимо от того, используются ли в процедуре преобразования встроенные математические функции, интерпретатор ТехноБЕЙСИК или математическая модель, осуществляется формирование пользовательского значения канала.

С помощью процедуры автоматического управления формируются реальные значения каналов типа A_OUTPUT. Формирование реальных значений с помощью этой процедуры осуществляется путем преобразования значений на других каналах по одному из 20 законов автоматического управления. Для этой процедуры могут использоваться значения любых атрибутов других каналов (аппаратное, реальное и пользовательское значения, аварийные границы, среднее, сумма, счетчик, тенденция, достоверность, интервал и др.).

Настройка канала осуществляется в редакторе базы каналов. Здесь устанавливаются параметры, определяющие режимы его работы, а также источники или приемники данных.

Ввод параметров канала осуществляется с помощью мыши нажатием левой или правой кнопки в соответствующих областях экрана редактора базы каналов. Значения некоторых параметров задаются вводом с клавиатуры после нажатия одной из кнопок мыши в областях, определенных для ввода значений этих параметров.

Для каждого канала независимо от его класса задаются следующие параметры:

- имя канала;
- класс канала;
- тип канала;
- период работы канала;
- начальное состояние канала;
- начальное значение канала;
- вид представления данных канала;
- апертура/ограничение скорости изменения;
- аварийные границы канала.

Кроме того, для каждого канала можно установить ряд флагов, определяющих режимы пересчета и архивирования данных:

- флаг АРХИВ;
- флаг ОТЧЕТ ТРЕВОГ;
- флаг РЕГИСТРАТОР СОБЫТИЙ;
- флаг СРЕДНЕЕ;
- флаг ИЗМЕНЕНИЕ/ПЕРЕХОД ЧЕРЕЗ ГРАНИЦУ;
- флаг СБРОСА/АНАЛИЗА ДОСТОВЕРНОСТИ;
- флаг ОТРАБОТАТЬ/ИГНОРИРОВАТЬ;
- флаг АВТОПОСЫЛКИ;
- флаг ВКЛЮЧЕНИЕ ПРЕОБРАЗОВАНИЯ.

Если канал специфицирован как внешний, то необходимо будет задать его удаленный адрес. Если же канал внутренний, то необходимо определить его подтип и дополнения к подтипу.

Кроме перечисленных выше, каждый канал обладает еще рядом атрибутов. Эти атрибуты вычисляются для него системой или формируются по заданным алгоритмам. К таким атрибутам относятся следующие:

- среднее значение канала;
- счетчик;
- сумма;
- интервал;
- тенденция;
- достоверность.

Каждый канал имеет два параметра, которые идентифицируют его в базе каналов – это номер и имя. По ним осуществляются ссылки на него в базе каналов, привязка к его значениям форм отображения на экране монитора, данных в архивах и в генерируемых формах и отчетах.

Оба эти параметра формируются в редакторе базы каналов и являются неизменными, тогда как большинство остальных характеристик канала (кроме класса, типа, подтипа и удаленного адреса) могут быть изменены в процессе работы системы в реальном времени.

Номер присваивается каналу автоматически по порядку в базе каналов. При удалении или добавлении каналов в базу номера всех существующих каналов автоматически пересчитываются.

Имя канала задается пользователем, исходя из условий кодировки параметров, принятых в проекте (например, LRC-R1 – уровень в реакторе R1). Для задания имени канала можно использовать любые сочетания букв и цифр (в том числе и пробелы). Существуют два ограничения: имя не может включать более 10 символов и не должно начинаться с цифры или символов «-», «@», «#».

Каждый канал Trace Mode может находиться в одном из двух состояний: включен и выключен. Если канал включен (ON), то значения на нем пересчитываются с частотой, определяемой периодом канала. В случае когда канал находится в выключенном состоянии (OFF), значения на нем не пересчитываются и остаются неизменными.

Состоянием канала можно управлять в реальном времени. Для управления состоянием канала можно использовать те же методы, которые используются для управления периодом канала. Разница заключается в том, что в качестве управляемого атрибута надо указывать состояние канала _C_.

Возможность в реальном времени включать и выключать каналы удобна для реализации таких операций, как, например, квитирование аварийного звукового сигнала или отключение опроса датчиков вспомогательного оборудования, используемого только в моменты пуска или останова технологического процесса. Ниже подробно приведен пример квитирования аварийного звукового сигнала.

В систему Trace Mode встроена поддержка последовательных портов RS232/485 с протоколами основных контроллеров, используемых в настоящее время в России. Среди них популярны MODBUS (AEG Modicon, PEP Modular Computers, Autolog и др.), OMRON SYSMAC, Optomux, используемые в небольших контроллерах ADAM 4000, а также основные АЦП для MicroPC фирмы Octagon Systems. Кроме того, Trace Mode обеспечивает встроенную поддержку распространенных российских контроллеров, таких как ЛОМИКОНТ, Ш-711 и др. Причем одновременно можно пользоваться несколькими встроенными протоколами, что позволяет задействовать разнотипные контроллеры в пределах одной системы. Кроме этого, любые устройства могут быть подключены через внешний драйвер программы, формат которого открыт и документирован.

После завершения организации ввода-вывода данных приступают к заданию их математической основы дискретного и аналогового управления. Для математических расчетов и анализа есть мощный интерпретатор пользовательских формул, способный решать не только алгебраические, но и сложные логические задачи. Кроме того, в системе имеется возможность подключения математических моделей на языке **Си**. Набор математических функций Trace Mode достаточен для разработки управляющих систем любой сложности.

Когда математическая основа будущей системы сформирована, переходят к созданию графического интерфейса АСУ. Для этих целей в Trace Mode предусмотрены два мощных объектных редактора с векторной масштабируемой графикой. В одном из них создается статичный рисунок мнемосхем объекта, а в другом производится его динамизация путем наложения форм динамического отображения, увязанных с каналами ввода-вывода. Эти редакторы называются «редактор рисунка» и «редактор форм отображения».

Особенностью редактора рисунка является наличие виртуального графического поля размером в 100 экранов монитора. На этом пространстве при помощи редактора рисунка изображается объект технологического мониторинга. Большой размер графического поля связан с особенностью организации представления информации в реальном времени: рисунки в Trace Mode могут быть масштабированы произвольным образом. В любой момент времени оператор АСУ имеет возможность вывести на экран изображение как всего технологического объекта, так и любой его части, причем в произвольном масштабе по каждой из координатных осей.

Графические средства редактора рисунка включают набор из 128 пользовательских примитивов и 36 элементов рисования, часть из которых основана на кривых Безье. Кроме того, на векторные рисунки могут быть наложены небольшие растровые изображения в форматах CUT (Media Cybernetics drHALO) и BMP. Надписи на экране осуществляются

четырьмя векторными масштабируемыми шрифтами пятнадцати градаций жирности.

В редакторе форм отображения, как и в редакторе базы каналов, любая часть рисунка может быть выделена в виде объекта и, в свою очередь, привязана к объекту базы каналов ввода-вывода Trace Mode. Подобная организация проектирования позволяет начиная с некоторого этапа создавать прикладные системы автоматизации путем тиражирования и манипулирования объектами Trace Mode, что благоприятно сказывается на производительности труда разработчика.

Trace Mode имеет 216 форм представления данных в реальном времени, включая трехмерные графики. Так, АСУ, созданная при помощи Trace Mode, предоставляет оператору экранные «кнопки», «ползунки» и прочее для управления оборудованием. Кроме того, в системе предусмотрена возможность создания мультипликации, управляемой значениями на каналах, причем мультипликационные серии могут передвигаться по рабочему полю. Мультипликация в программе позволяет отображать такие традиционно сложные для систем SCADA объекты, как конвейеры и технологические линии, а также отслеживать движение объектов в компьютерных охранных системах. Trace Mode в рабочем режиме ведет несколько типов отчетов, допускающих графическую визуализацию. Это значит, что пользователь АСУ может просматривать архив мониторинга как фильм в режиме playback, причем с заданной скоростью. Данное качество является весьма ценным при разборе аварийных ситуаций.

Для информации о критических превышениях измеряемых параметров предусмотрен специальный отчет событий.

После завершения разработки графического интерфейса пользователя, отладки и небольшой заключительной настройки система автоматизации готова к запуску. Trace Mode имеет run-time-системы для Windows, которые совместимы по формату файлов. Это дает разработчику дополнительную свободу: одни и те же проекты могут быть запущены под управлением любой из этих операционных систем.

Логика развития компьютерных АСУТП предполагает объединение отдельных рабочих станций оператора в единую информационную систему цеха, а потом и предприятия. Сетевая идеология Trace Mode наследует принципы, заложенные в ее предыдущие версии.

Сетевые комплексы Trace Mode имеют иерархическую организацию. В них выделяются два уровня: ***оперативный*** – для диспетчера и ***административный*** – для технических руководителей. ПО рабочих мест оперативного уровня предназначено для сбора, визуализации данных, управления, ведения архивов и автоматического составления отчетов. На административном уровне системы введена дополнительная возможность – графический Playback архива. Термин «playback» заимствован из видеотехни-

ки и означает «воспроизведение записи». Руководитель не обязан постоянно наблюдать за процессом на экране, но он должен быть постоянно в курсе всех происходящих событий. Именно поэтому на своем компьютере под управлением Монитора архива Trace Mode он сможет при необходимости переместиться в любую временную точку и просмотреть (playback) ход технологического процесса с удобной для него скоростью.

В новой версии Trace Mode сетевые функции существенно расширены. Сетевые станции способны вести обмен данными 4096 каналов ввода-вывода с частотой до 0,055 с. Сетевой обмен между компьютерами оперативного уровня ведется в режиме «точка-точка». При этом пользователю доступны такие способы сетевого взаимодействия, как «запрос-ответ» и «автопосылка». Автопосылка данных может быть как индивидуальной, так и групповой, т.е. один компьютер способен рассылать данные группе сетевых станций. Подобная организация обмена позволяет инженерам в реальном времени осуществлять дистанционную коррекцию параметров управления технологическим процессом, включать и выключать устройства, менять установки, динамически загружать и редактировать рецепты.

МикроМРВ, специализированная run-time-система Trace Mode, предназначен для работы на получивших в последнее время широкое распространение IBM-совместимых промышленных контроллерах, на которых средствами Trace Mode возможно программировать задачи нижнего уровня АСУ. При этом как операторские станции верхнего, так и слепые узлы нижнего уровней системы создаются при помощи одного объектного редактора базы каналов Trace Mode. С помощью Trace Mode и МикроМРВ можно создавать распределенные сетевые конфигурации, включающие до 200 контроллеров и ПК на базе локальных сетей или RS232/485. Математические возможности МикроМРВ Trace Mode позволяют решать задачи автоматического и ручного управления, а время реакции системы (от 120 мс на простом контуре регулирования, а в версиях 5.x и 6.x – значительно меньшее) является приемлемым для большинства промышленных задач.

На ПК устанавливается операторская станция, работающая под управлением монитора реального времени Trace Mode. С операторской станции ведется загрузка управляющей программы в контроллер и принимаются сигналы о состоянии электрооборудования станка (в том числе и сигналы о сбоях).

На нижнем уровне АСУ должен работать PC-контроллер под управлением МикроМРВ. Он должен осуществлять следующие операции: прием управляющей программы, расчет интерполяции, выдачу управляющих воздействий на электроприводы главного движения и подач, управление элекроавтоматикой станка, опрос состояния электрооборудования станка.

Для разработки интерфейса графической системы необходимо сначала в редакторе рисунка Trace Mode нарисовать статический рисунок. Затем создать базу каналов для обмена данными с контроллерами. В нашем случае контроллер работает под управлением МикроМРВ и связан с компьютером, например, с помощью сети M-link. Для обмена данными по этому протоколу необходимо использовать каналы с подтипом MPB. В дополнениях подтипа необходимо указать сетевой номер контроллера, номер и атрибут канала. Затем в редакторе представления данных необходимо создать динамические формы, которые необходимо связать с каналами, которые принимают данные из контроллера или посылают их.

Для посылки значений в канал можно использовать формы управления «кнопка» и «область». При работе в реальном времени, когда оператор щелкает мышкой в области этих форм, в указанный атрибут связанного с этой формой канала посылается определенное число.

Контрольные вопросы:

1) Назовите основные функции SCADA.
2) Что такое канал SCADA?
3) Как формируется локальная сеть удаленного контроля?
4) Перечислите классы каналов.
5) Охарактеризуйте смысл OPC.
6) Проанализируйте проблему реального времени диспетчерских систем.
7) Что называется тегами, трендами, алармами?
8) Как происходит интеграция УЧПУ с информационными системами?
9) О каких стандартах и протоколах обмена данными Вы узнали?

3. ПРОЕКТИРОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ И КОМПЛЕКСАМИ

Информационная поддержка этапа производства продукции осуществляется АСУП и АСУТП. К АСУП относятся: системы планирования и управления предприятием ERP (Enterprise Resource Planning), планирования производства и требований к материалам MRP-2 (Manufacturing Requirement Planning), производственная исполнительная система MES (Manufacturing Execution Systems), а также SCM (Supply Chain Management) и система управления взаимоотношениями с заказчиками CRM (Customer Requirement Management).

Функционально задачи управления взаимосвязаны, как показано на рис. 3.1.

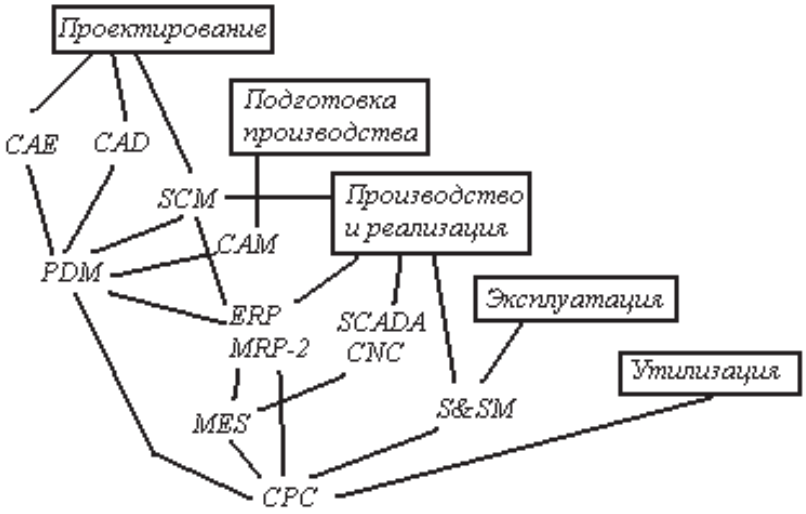


Рис. 3.1. Этапы жизненного цикла промышленных изделий
и используемые автоматизированные системы

Принято выделять системы функционального, конструкторского и технологического проектирования. Первые из них называют системами расчетов и инженерного анализа или системами CAE (Computer Aided Engineering). Системы конструкторского проектирования называют системами CAD (Computer Aided Design). Проектирование технологических процессов составляет часть технологической подготовки производства, выполняется в системах CAM (Computer Aided Manufacturing). Функции координации работы систем CAE/CAD/CAM, управления проектными данными и проектирования возложены на систему управления проектными данными PDM (Product Data Management).

Уже на стадии проектирования требуются услуги системы управления цепочками поставок (SCM), иногда называемой системой Component Supplier Management (CSM). На этапе производства эта система управляет поставками необходимых материалов и комплектующих.

Системы CNC (Числовое программное управление) обеспечивают непосредственное управление технологическим оборудованием через соответствующие устройства сопряжения (УСО) и реализуют обмен данными и их обработку в реальном масштабе времени.

Из рис. 3.1 следует, что именно CNC являются основой построения производственных автоматизированных систем.

3.1. Интеграция устройства числового программного управления с информационными системами

Интеграция устройства числового программного управления (УЧПУ) с информационными системами, а также разработка ПО адаптации, его объединение с программным обеспечением УЧПУ приводят к созданию интеллектуальных систем ЧПУ.

Для выполнения какой-либо работы УЧПУ должно получить соответствующие указания о том, какое задание из предусмотренных при проектировании УЧПУ должно быть исполнено, а для выполнения некоторых заданий еще и детальное указание задач, которые при этом должны быть исполнены. Эти указания УЧПУ может получить или от оператора, или по каналам связи с верхним уровнем управления МПС и периферийными устройствами УЧПУ. Для связи УЧПУ с оператором предусмотрены аппаратно-программные средства, обеспечивающие выбор задания и определение задач задания, а также контроль за исполнением задания и задач. К таким средствам относятся: клавишный пульт оператора или пульт обучения, дисплей УЧПУ, сигнальная индикация, фотосчитывающее устройство, кассеты памяти, диски и другие устройства, размещенные на панели (пульте) оператора. Общение оператора и УЧПУ происходит на нескольких языках, среди которых можно выделить: ***язык заданий***, ***язык задач***, ***язык дисплея и язык индикации*** (рис. 3.2).

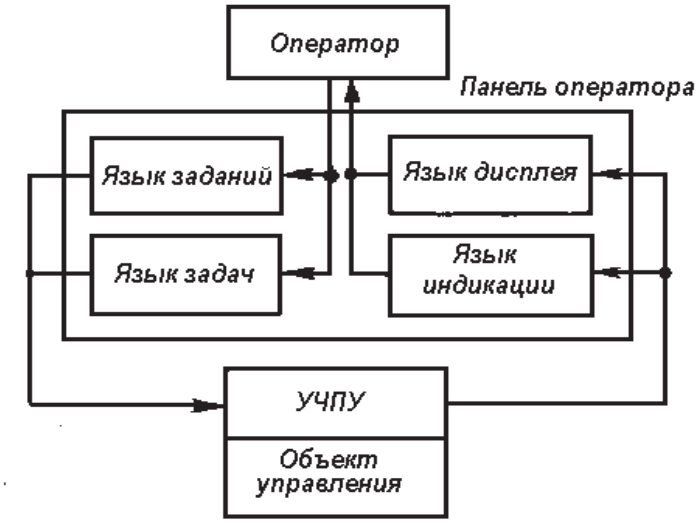


Рис. 3.2. Связь оператора с УЧПУ

Язык заданий реализуется клавишными средствами панели оператора, при помощи которых оператор определяет задание (режим работы) УЧПУ, выдает директиву УЧПУ для его настройки на выполнение определенного вида работ. Язык заданий достаточно многообразен, т.к. отражает

не только специфику УЧПУ, но и специфику объекта управления. Язык заданий УЧПУ нашел свое отражение в выделении функциональных полей (зон) панели оператора, мнемонических обозначениях функциональных клавиш полей и правилах пользования ими для выполнения заданий. Некоторые мнемонические обозначения функциональных клавиш определены в ГОСТ 24505-80.

Язык задач предназначен для описания задач, выполняемых при исполнении заданий. Описание задач, выполняемых при управлении объектом, задается с помощью управляющей программы (УП), которая составляется на том или ином языке пользователя. Так, например, для составления технологических УП УЧПУ станками используется международный код ИСО, нашедший отражение в ГОСТ 20999-83, определяющем содержание УП. В режимах подготовки к управлению объектом оператор может обращаться с УЧПУ на языке, определенном правилами тестирования, клавишами панели оператора и дисплея, языке подготовки УП в режиме «Меню» с использованием графических представлений и средств или без их использования.

Языки дисплея и индикации определяются правилами эксплуатации УЧПУ и используемых языков заданий и задач, а также мнемоникой функциональных клавиш дисплея и индикации и правилами пользования ими. С помощью языка дисплея и индикации задаются объем и содержание визуально контролируемой информации, ведется диалог, выполняется плоская и объемная графика и т.д. Индикация может сопровождаться подсветкой или звуком при неправильном вводе информации, обнаружении ошибок при контроле.

Расширители стандартных функций микропроцессорных УЧПУ необходимы для повышения производительности МПС (микропроцессорные системы) при выполнении операций, входящих в базовый набор арифметических функций. Наиболее часто выполняемая в микропроцессорных УЧПУ арифметическая функция (умножение слов длиной в 16 и 32 разряда), как правило, требует такого расширителя. Микропроцессор, память, расширители стандартных функций, объединенные системной магистралью, составляют вычислительное ядро, или вычислитель микропроцессорной системы (рис. 3.3), с которым сопрягаются внешние устройства, необходимые для управления объектом и связи с оператором. При использовании однопроцессорного вычислителя возможны две основные структуры. В структуре с общей магистралью (см. рис. 3.3, *а*) все контроллеры внешних устройств сопрягаются с вычислителем (ВЧС) через эту магистраль. В этой структуре нагрузочная способность системной магистрали (СМГ) ВЧС ограничивает возможности расширения системы из-за ухудшения помехоустойчивости. Для улучшения этих характеристик нередко используется вариант однопроцессорной структуры с двумя маги-

стралями (см. рис. 3.3, *б*), сопряжение между которыми осуществляется через специальное устройство согласования магистралей – адаптер.

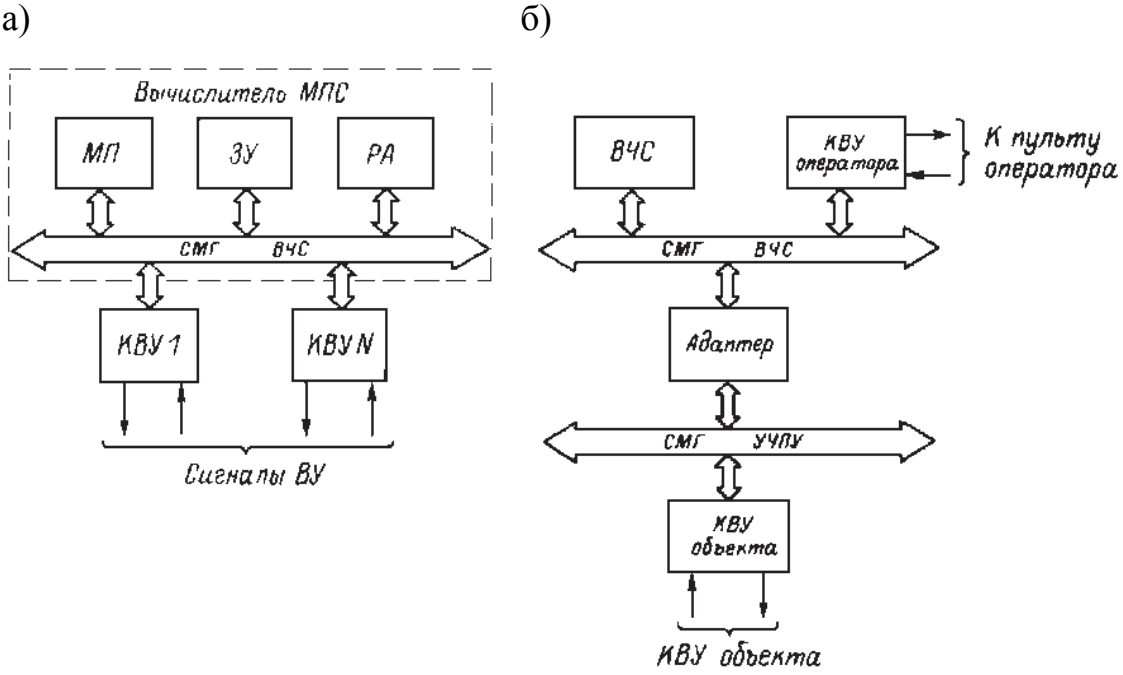


Рис. 3.3. Структуры однопроцессорных УЧПУ

В этой структуре контроллер внешних устройств (КВУ) оператора (дисплея и клавиатуры) сопрягаются обычно с магистралью ВЧС, а КВУ объекта (электроприводов, датчиков состояния объекта, устройств управления электроавтоматикой) – с магистралью УЧПУ (СМГ УЧПУ).

Использование однопроцессорной структуры с двумя магистралями оказывается эффективным, т.к. КВУ объекта для связи с несколькими однотипными внешними устройствами обычно выполняется на одной плате и является по отношению к магистрали одной единицей нагрузки.

Уже в однопроцессорных УЧПУ в полной мере определились основные принципы организации МПС УЧПУ, обеспечивающие возможность расширения системы при сохранении функциональной гибкости и обеспечении надежности функционирования при малом времени восстановления в случае отказа. К их числу относятся:

- ***принцип программно-аппаратной реализации функций управления***, в соответствии с которым основными функциями аппаратных средств КВУ являются функции преобразования информации вычислителя в сигналы управления объектом и обратное преобразование сигналов состояния объекта в машинную форму, а обработка информации производится программными средствами вычислителя;
- ***магистрально-модульный принцип построения аппаратных средств***, в соответствии с которым аппаратные средства, входящие в

состав базовой конфигурации микропроцессорного УЧПУ, выполняются в виде унифицированных по конструкции модулей и имеют унифицированные средства сопряжения с системными магистралями;

- ***принцип «встроенной» диагностики аппарата УЧПУ***, в соответствии с которым каждый блок, входящий в ВЧС или КВУ, имеет свои аппаратные диагностические средства, что позволяет быстро локализовать неисправности и снизить время восстановления устройства в случае отказа.

Однопроцессорные УЧПУ накладывают ограничения на количественное и качественное расширения функциональных возможностей и надежность управления локальными устройствами ЧПУ, обусловленные следующими причинами:

1) Ограничение вычислительной мощности процессора. Даже чисто количественное увеличение выполняемых МПС одинаковых функций, например числа управляемых осей исполнительных устройств объекта, имеющее место в тяжелых и уникальных станках с ЧПУ, приводит к недопустимому увеличению загрузки процессора.

2) Ограничение возможностей привязки микропроцессорного УЧПУ к объекту. Эти ограничения связаны с проблемой управления устройствами электроавтоматики. Если схема ЭЛА (электроавтоматика) является сравнительно простой, то обработка дискретных входных сигналов и управление дискретными выходами ЭЛА возлагаются на МПС устройства ЧПУ. Однако для объектов со сложной схемой ЭЛА такое решение обычно неэффективно. В этом случае функции привязки осуществляются специальным командоаппаратом. Если такой командоаппарат выполняется на базе МПС, то он называется *программируемым логическим контроллером* (ПЛК). В настоящее время ПЛК является наиболее универсальным средством автоматизации устройств автоматики во многих отраслях промышленности. В области систем управления ГПС (гибкая производственная система) на основе этих контроллеров строится управление транспортно-накопительными системами.

3) Ограничение, связанное с тем обстоятельством, что локальный объект автоматизации в ГПС по природе своей является пространственно-распределенным объектом. В большей или меньшей степени это справедливо для станков, роботов и ГПС. Естественным является стремление выполнять отдельные компоненты УЧПУ в виде функционально и конструктивно завершенных модулей, имеющих минимальное число линий связи с другими модулями и максимально приближенных к управляемому объекту. Но это выполнимо только в том случае, если каждый из модулей будет иметь свою локальную МПС (ЛМПС), обладающую возможностью обмена с другими ЛМПС, входящими в общую мультимикропроцессорную структуру.

4) Ограничение, связанное с конечной нагрузочной способностью системной магистрали. Увеличение вычислительной мощности процессора

целесообразно, если можно расширять число модулей, подключаемых к СМГ без снижения надежности обмена данными. На практике это условие оказывается значительно более жестким, нежели ограничение вычислительной мощности процессора. Из сказанного следует, что переход к мультимикропроцессорным структурам УЧПУ является объективной тенденцией развития этого класса устройств. В настоящее время распространены мультимикропроцессорные (ММПС) УЧПУ, в которых увеличение функциональных возможностей достигается в результате применения в качестве ЛМПС однородных по элементной базе функционально законченных микропроцессорных модулей, каждый из которых решает собственную вычислительную задачу. Различие функций, выполняемых модулями, определяется различием ПО, занесенного в ЗУ ВЧС, входящих в состав этих модулей. Такой подход позволяет создавать модульные микропроцессорные УЧПУ, расположенные в одном блоке или же распределенные по конструктивным блокам системы управления. При расположении микропроцессорных модулей ММПС в одном блоке агрегатирование отдельных вычислителей осуществляется с использованием двухпортовых блоков общей памяти (БОП) в соответствии с рис. 3.4.

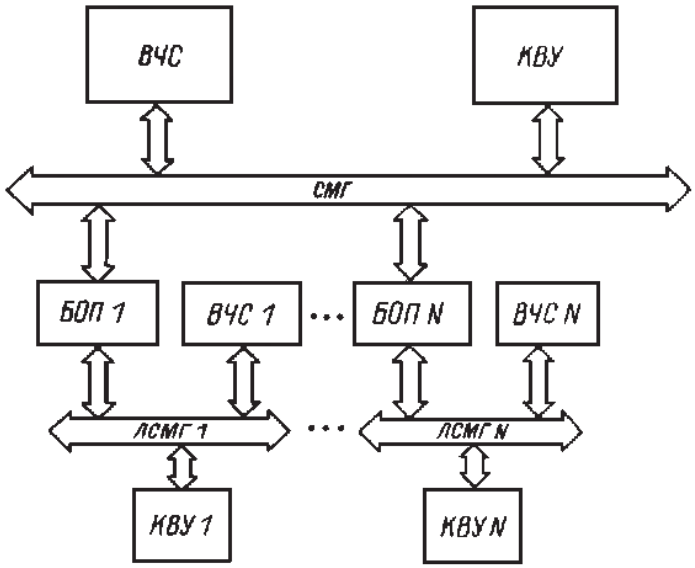


Рис. 3.4. Структура многопроцессорного УЧПУ
с обменом через БОП

Центральный ВЧС УЧПУ связан через системную магистраль СМГ с КВУ (ими могут быть контроллер дисплея и клавиатуры пульта) и локальными вычислителями (ВЧС1-В4CN), имеющими собственные системные магистрали ЛСМГ и решающими конкретные функциональные задачи, например управления приводами и ЭЛА. Связь через двухпортовые БОП обеспечивает наименьшие потери производительности ММПС, обусловленные конфликтами при одновременном обращении вычислителей к

БОП. Так как интенсивность обмена данными между локальными ВЧС и центральным ВЧС невелика, то БОП может быть совмещен с оперативным запоминающим устройством локального вычислителя. При использовании параллельного обмена в рассмотренной структуре потери производительности, обусловленные временем обмена, становятся пренебрежимо малыми.

Большая пропускная способность требуется от канала обмена данными при использовании структуры ММПС, способной обеспечить функциональные возможности, аналогичные тем, которые обеспечивает структура на рис. 3.4, но в распределенном по конструктивным блокам варианте. Вариант структуры «распределенного» УЧПУ приведен на рис. 3.5.

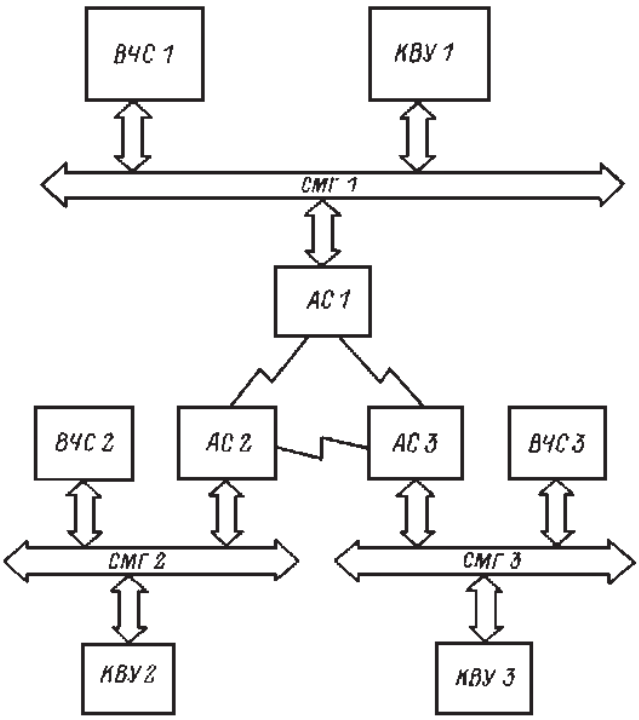


Рис. 3.5. Структура многопроцессорного УЧПУ
с обменом через адаптеры связи

В отличие от предыдущей, в этой структуре микропроцессорные вычислители имеют равные возможности обмена данными. Обмен осуществляется через адаптеры связи АС1…АСЗ, подключенные к системным магистралям соответствующих вычислителей и имеющие выходы на два последовательных канала обмена. Несомненными *достоинствами* рассмотренной структуры являются конструктивная автономность локальных ВЧС, возможность их встраивания в аппаратуру, расположенную в различных местах общей системы управления при минимальном числе линий обмена, и хорошая, а в перспективе при использовании оптических каналов обмена – идеальная помехозащищенность. *Недостаток* структуры – ограниченная пропускная способность последовательного канала обмена.

Комбинируя принципы построения мультимикропроцессорных структур, можно получить оптимальную по выбранному критерию структуру, удовлетворяющую требованиям использования в конкретном УЧПУ.

3.2. Информационный обмен между ЭВМ и системой программного управления

Одновременная работа в системе управления ГПС различных УЧПУ станками, роботами и СПУ транспортно-складскими системами, которые обмениваются с ЭВМ большими массивами информации, требует унификации аппаратных средств и стандартизации ПО, связанного с обменом информации между ЭВМ и системой программного управления (СПУ).

В настоящее время при обмене информацией на расстоянии до 500 м, т.е. в пределах участка, линии, цеха, используются аппаратные средства стандартных каналов последовательного обмена, рассмотренные выше. Наиболее перспективным решением в области стандартизации обмена информацией является разработка протокола связи в автоматизированном производстве (MAP), базирующегося на эталонной модели соединения открытых систем. Эта модель в самом общем случае включает в себя семь уровней связи:

1) ***физический***, на котором определены физические среды передачи данных: коаксиальный кабель, витая пара проводов, оптоволоконная линия;

2) ***канальный***, на котором определены топологии сетей (шина и кольцо) и способы обращения к каналам передачи с использованием маркерного метода, определены в общей форме услуги, обеспечиваемые концептуальным интерфейсом, между канальным и физическим уровнями, а также специфицированы временные и размерные параметры канального уровня;

3) ***сетевой***, на котором разрабатываются стандарты и межсетевые протоколы для связи различных сетей ГПС;

4) ***транспортный***, на котором определяется стандарт, обеспечивающий единый протокол транспортирования данных для соединения нескольких сетей, и описывается межсетевой пакет, обеспечивающий транспортирование данных между несколькими сетями. Этот пакет (датаграмма) передается между сетями с помощью буферных коммутирующих узлов (межсетевых шлюзов);

5) ***сеансовый***, на котором определяются следующие функции: управление началом и концом сеанса связи, управление взаимодействием пользователей в ходе сеанса, контроль ошибок и их корректировка в ходе сеанса;

6) ***представительский***, который включает в себя: синтаксические функции и форматные преобразования текстовых и управляющих символов, типов данных, строк и страниц; структурные преобразования синтак-

сического и семантического типа, необходимые для доступа к файлам и массивам данных; структурные преобразования синтаксического и семантического типа, необходимые для получения отображения данных на терминалах;

7) ***прикладной***, который охватывает функции: подключения (отключения) пользователя от системы, проверки права доступа к ресурсам системы и справочнику сетевых услуг и т.д.; передачи файлов, ввода заданий, формирования и адресования пакетов, текстовой обработки и т.д.; проблемно-ориентированные функции, такие как информационный поиск, генерация ответов, управление базами данных.

Создание средств связи производственных систем с протоколами по типу МАР является перспективой развития системы управления ГПС.

В настоящее время для системы управления (СУ) производственными участками используются более простые системы связи, обеспечивающие радиальную топологию со стандартизованным физическим и канальным уровнем. Общими характеристиками канала являются: соединение СПУ и ЭВМ типа «точка-точка», последовательная асинхронная передача информации, разделение цепей приема и передачи, полудуплексный тип передачи, отсутствие коммутации канала.

Физический уровень в таких системах связи обеспечивается витой парой проводов или коаксиальным кабелем, канальный уровень – программно-аппаратными средствами в соответствии с описанными в ГОСТ 23675-79 стандартами на стык С2-ИС и интерфейс ИРПС. Аппаратные средства поддерживают старт-стопную передачу и прием восьмибитовых знаков и символов. Программная поддержка передачи и приема осуществляется специальными программами (драйверами) канала связи, обеспечивающими обмен данными в соответствии с этими стандартами. При соединении типа «точка-точка» сетевой и транспортный уровни отсутствуют, сеансовый уровень поддерживается драйвером канала связи, фактически объединяясь с канальным уровнем.

Представительский уровень реализуется с использованием специального языка обмена между ЭВМ и СПУ. Язык предусматривает две модификации: первая ориентирована на связь с УЧПУ станка или робота, а вторая – на связь с СПУ транспортно-складских систем. При связи с УЧПУ от ЭВМ передаются: управляющие программы, модули ПО, программы диагностики, данные о коррекциях, параметрах станка и инструменте. От УЧПУ передаются: отредактированные УП и модули ПО, запросы на передачу УП, данные о состоянии технологического оборудования и УЧПУ, подтверждение выполнения команд.

Передача данных по инициативе приемника называется запланированной, а по инициативе передатчика – незапланированной. Для проведе-

ния незапланированной передачи необходимо подтверждение готовности приема.

УЧПУ является проблемно-ориентированной ЭВМ, предназначенной для управления в реальном времени приводами и цикловыми механизмами. Влияние на УЧПУ объекта управления проявляется чрезвычайно сильно в том смысле, что поставленные объектом задачи управления полностью определяют облик системы управления.

Процесс управления можно расчленить на **фазы** информационных преобразований и вычислений.

Первая фаза информационного преобразования в процессе ЧПУ состоит во вводе управляющей программы СПУ, а также выполнении подготовительных расчетов в машинном масштабе времени. Традиционной формой управляющей программы являлась перфолента с текстом в коде ISO-7 bit, однако сегодня применяют другие носители (блоки полупроводниковой памяти, гибкие и твердые диски и др.). В современных УЧПУ обязательно предусмотрены возможность ручного ввода и редактирования управляющих программ с панели оператора и хранение архива управляющих программ в памяти. Существует тенденция использования в рамках первой фазы процесса ЧПУ элементов автоматизированной разработки управляющих программ с привлечением диалога и техники меню.

Вторую фазу в процессе ЧПУ составляют вычисления, выполняемые в реальном времени. Сюда относятся: интерполяция (сложность которой обусловлена формой интерполируемой траектории и размерностью интерполируемого пространства); расчеты разгонов и торможений; стыковка кадров управляющей программы; анализ рабочего пространства; коррекция подачи; анализ и логическая обработка информации о состоянии цикловых механизмов и др. Вычисления сопровождаются подготовкой данных для визуализации всей оперативной информации. Вычисления второй фазы, как правило, требуют сложных и точных расчетов, повторяющихся с высокой частотой.

Третью фазу в процессе ЧПУ составляет управление приводами, осуществляемое в реальном времени. Наибольшую долю занимают здесь расчеты, связанные с замыканием контуров следящих приводов подачи и коррекцией систематических погрешностей (например, винтовых пар станка). Трудоемкость расчетов прямо пропорциональна числу управляемых координат, общее число которых достигается иногда 10…12. Другая часть работы, выполняемой в пределах третьей фазы, относится к управлению многочисленными приводами цикловых механизмов и приему сигналов, определяющих состояния (положения) указанных приводов.

Различное наполнение представленных трех фаз конкретными задачами ЧПУ и создает предпосылки для тех или иных повторений конкрет-

ной системы управления. Окончательный вариант УЧПУ определяется выбором вычислительной архитектуры и электронной элементной базы.

Ядром любого архитектурного решения является универсальный или специально разработанный вычислитель, располагающий традиционным набором компонентов (микропроцессор, шина ЭВМ, память, стандартные интерфейсы, стандартная периферия ЭВМ) и дополненный специальным набором компонентов, определяемым спецификой ЧПУ (шина ЧПУ, дополнительная память, специальные интерфейсы, специальная периферия ЧПУ). По мере увеличения сложности поставленных перед системой управления задач исходную архитектуру развивают: путем наращивания мощности вычислителя; путем «интеллектуализации» интерфейсов; путем умножения числа вычислительных ядер.

Отличительная особенность современного УЧПУ как проблемно-ориентированной управляющей вычислительной машины состоит в совмещении разнородных функций управления и разнородных режимов переработки информации. В самом деле, УЧПУ одновременно осуществляет функции программного управления системами автоматического регулирования (каковыми являются следящие приводы подачи станка, приводы роботов) и функции логического управления конечными автоматами (к числу которых относятся устройства станочной автоматики). Наряду с этим на процессы переработки информации в режиме реального времени, связанные с формированием управляющих команд, наложены множественные процессы, выполняемые в машинном масштабе времени: автоматизированное программирование, подготовка данных и др. Таким образом, можно сказать, что УЧПУ совмещает функции программного регулятора, ПЛК и активного терминала.

Программирование контроллеров и интеллектуальных датчиков производится различными программными средствами или специальными программаторами, поставляемыми с оборудованием.

Исторически первым и до сих пор достаточно популярным языком программирования является язык релейно-контактных символов (РКС). Язык основан на применявшейся методике создания схем электроавтоматики с жесткими структурами (реле, контакторами, бесконтактными логическими элементами). В качестве исходного документа для программирования принимают принципиальные релейные электрические семы (как документы, привычные для электриков-конструкторов и электриков-ремонтников). Другими видами исходных документов могут быть: словесное описание, тактограммы, циклограммы и т.д.

Помимо языка РКС получили признание следующие языки:

- языки ассемблерного типа;
- графические языки логических схем;

- языки мнемонического символьного кодирования в виде набора строк-уравнений сложных булевых структур, сохраняющие методику проектирования программ на языке РКС;
- языки мнемонического задания многоситуационных процессов управления с естественным логическим описанием управления (ЕСЛИ … ТО … ИНАЧЕ …), не имеющие РКС-эквивалентов;
- оригинальные проблемно-ориентированные мнемонические языки высокого уровня с элементами организации работы в реальном времени;
- процедурные языки высокого уровня общего назначения (Бейсик, Паскаль), адаптированные для программирования электроавтоматики;
- языки графического отображения текущего состояния и взаимодействия процессов программ управления (например, GRAFCET).

Такое обилие языков не особенно удобно, однако в условиях большого рыночного разнообразия процессоров и шин, неустоявшихся стандартов использование специальных программаторов казалось единственным выходом. Но ситуация изменилась: с момента массового распространения IBM PC совместимых контроллеров (PC-контроллеров) появилась возможность унифицировать ПО для операторских персональных компьютеров и ПЛК.

3.3. Обработка дискретных сигналов на примере электроавтоматики станка и роботехнических систем

Работа электрооборудования определяется принципиальной схемой электрической.

Станок может работать в трех режимах: «Программа», «Ручное управление» и «Установка в ноль».

Работа в режиме «***Программа***» заключается в отработке управляющей программы. Управляющая программа содержит всю геометрическую и технологическую информацию, необходимую для обработки детали. Пуск и остановка программы осуществляеются с пульта ЧПУ. Временное прерывание программы с остановом подачи или подачи и шпинделя может быть осуществлено соответствующим переключателем.

В режиме «***Ручное управление***» может быть осуществлено перемещение рабочих органов стола по координатам X и Z на быстром ходу, переключение механических диапазонов с включением шпинделя в толчковом режиме, зажим-разжим патрона, подвод-отвод пиноли.

В режиме «***Установка в ноль***» (кнопка на пульте станка) происходит автоматический выход рабочих органов станка в некоторую точку, фиксированную конечными выключателями.

Электрооборудование обеспечивает работу станка во всех режимах, которые заложены в системе ЧПУ.

Часть электрооборудования работает независимо от выбранного режима работы и обеспечивает выполнение следующих функций:

- ограничение перемещения каретки и суппорта;
- аварийный останов.

В случае аварийного останова для перемещения каретки или суппорта в противоположную сторону необходимо перейти в ручной режим, нажать на кнопку, шунтирующую аварийные конечные выключатели, и с помощью переключателя выбрать направление перемещения каретки или суппорта, нажав кнопку «Быстрый ход», осуществить съезд с конечных выключателей.

Выключение смазки шпиндельной бабки и направляющих осуществляется автоматически системой ЧПУ.

Для принудительного включения смазки имеется кнопка «Толчок смазки».

Включение шпинделя.

Алгоритм включения шпинделя представлен на рис. 3.6.

Скорость шпинделя задается технологической программой или преднабором с пульта ЧПУ. Скорость шпинделя определяется сигналом «Задание», поступающим от ЧПУ на вход преобразователя главного привода.

При останове шпинделя команды на вращение сохраняются в памяти ЧПУ, и при возвращении переключателя в первоначальное положение вращение восстанавливается.

При получении сигнала о включении шпинделя проверяется состояние патрона и пиноли. Если патрон разжат или пиноль отведена, то выдается соответствующий сигнал о сбое, в противном случае выдаются сигналы: преобразователь главного привода, деблокировка главного привода и включение смазки шпинделя. Если в течение 3 с сигналы, подтверждающие, что смазка включена, питание главного привода включено и привод деблокирован, не появились на выходе станка, то в этом случае выдаются сообщения о соответствующих сбоях в системе электрооборудования. В противном случае выдается управляющее воздействие на электропривод главного движения.

Второй алгоритм – управление выключением шпинделя. Для выключения шпинделя снимаются сигналы – преобразователь главного привода и деблокировка главного привода.

С учетом распайки линий связи на входные и выходные разъемы УЧПУ алгоритмы включения шпинделя и его смазки будут выглядеть в соответствии с разрядами представления информации о состоянии оборудования, как показано в табл. 3.1 (учитывая, что все входные сигналы объединены на разъеме А1, а выходные – на разъеме А2).

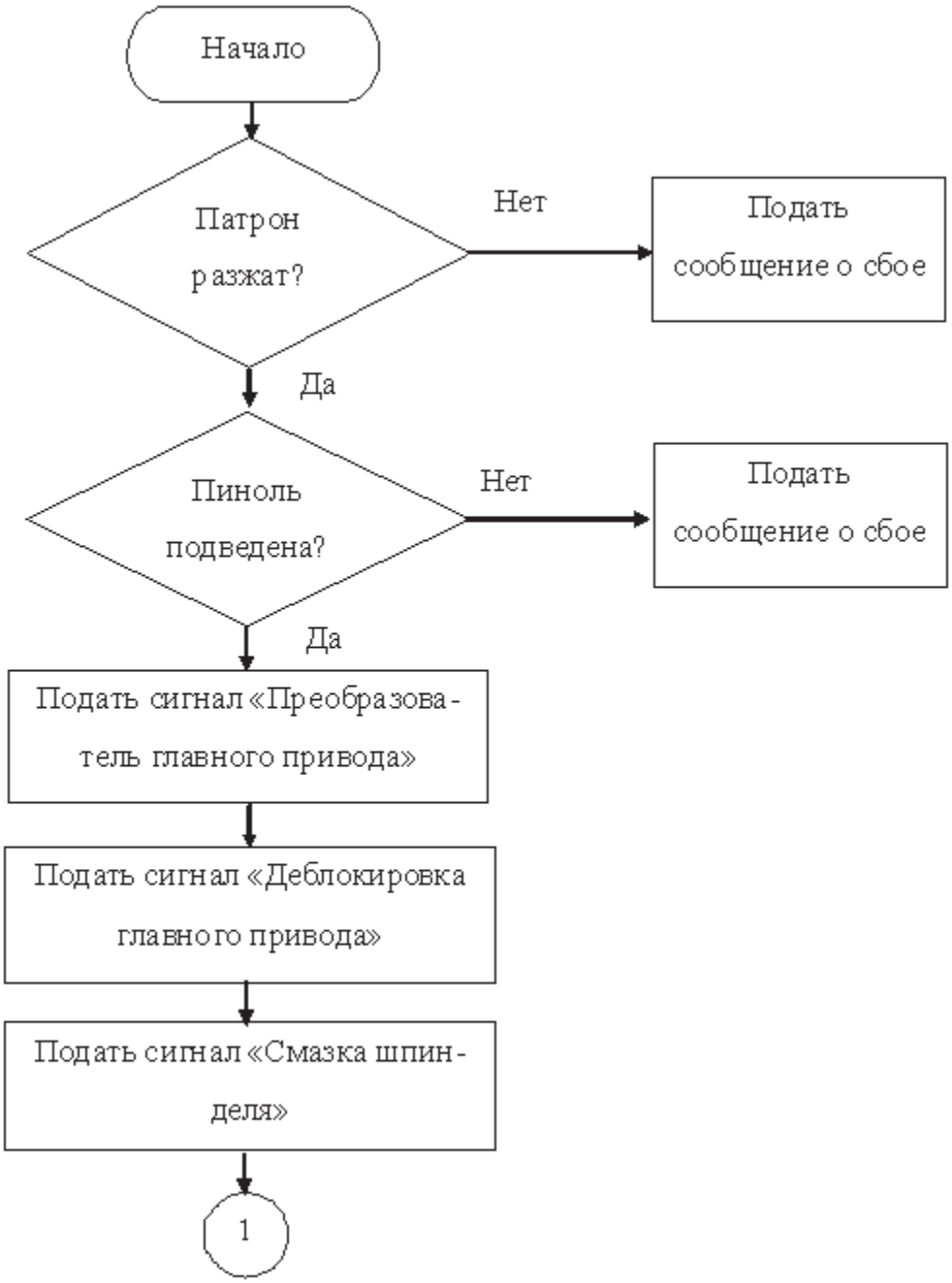


Рис. 3.6. Алгоритм включения шпинделя

С учетом табл. 3.1 алгоритм первой части работы программы приобретет вид, показанный на рис. 3.7 (здесь А2/8 обозначает: выходной разъем А2, контролируемый бит информации – 8-й разряд: 1 – «главный привод деблокирован», 0 – нет; т.е. если 00000001000, то «главный привод деблокирован», а если 000000000000, то нет).

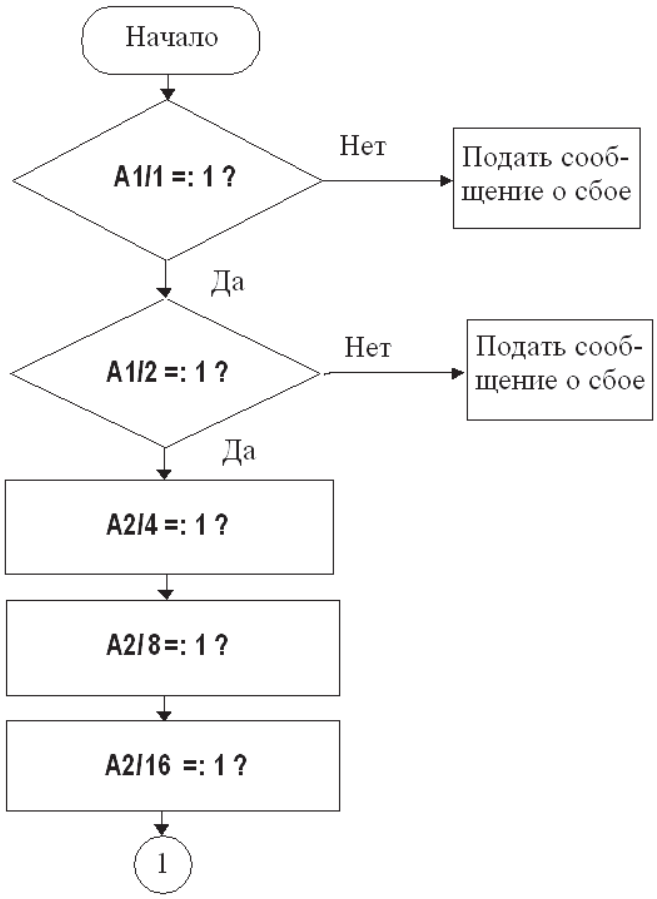


Рис. 3.7. Программа включения шпинделя

Таблица 3.1

Входные и выходные сигналы станка

A1			A2		
Контакт	Разряд	Сигнал	Контакт	Разряд	Сигнал
1	1	«Патрон зажат»	1	1	Подать сигнал «патрон зажать»
2	2	«Пиноль подведена»	2	2	Подать сигнал «пиноль подвести»
3	4	«Питание главного привода включено»	3	4	«Питание главного привода включить»
4	8	«Главный привод деблокирован»	4	8	Подать сигнал «главный привод деблокировать»
5	16	«Смазка включена»	5	16	Подать сигнал «смазка включена»
6	32	«Напряжение задания подано»	6	32	Подать сигнал «напряжение задания подать»
7	64		7	64	
8	128		8	128	
…	…		…	…	

3.4. Программируемые контроллеры

Программируемые контроллеры имеют жесткое разбиение памяти для размещения различных типов данных. Это необходимо учитывать при программировании и записи адресов переменных.

Например, в контроллерах фирмы Modicon адреса данных распределяются следующим образом:

0XXXX – дискретные выходы и промежуточные переменные (биты);

1XXXX – дискретные входы (биты);

3XXXX – регистры входов (16-разрядные двоичные слова);

4XXXX – регистры выходов и промежуточных значений, в том числе таймеров и счетчиков (16-разрядные двоичные слова);

5XXXX – регистры слов двойной длины (32-разрядные двоичные слова). Диапазон чисел XXXX определяется конкретным типом, модификацией и настройкой контроллера.

В контроллерах фирмы Omron, в частности в контроллере C200H, иное распределение памяти. Прежде всего, следует сказать, что здесь адрес бита задается адресом слова и номером бита в слове (две цифры в диапазоне 00…15). Например: 127 – адрес слова, тогда 12 700 – адрес самого младшего, а 12 715 – адрес самого старшего бита в этом слове. Адрес слова зависит от области памяти, в которой оно находится.

Для контроллеров Omron C200H выделяются девять областей памяти данных и память для управляющей программы. Их назначение заключается в следующем:

1) Область данных ввода/вывода IR используется для хранения данных модулей ввода/вывода и для хранения промежуточных переменных, если слова не используются модулями ввода/вывода (рабочее слово). Всего в этой области 236 слов, адреса в диапазоне 000…235 (адреса бит соответственно 00000…23 515). Адреса слов, занимаемых модулями, зависят от типа модуля.

2) Область флагов управления и состояния контроллера SR содержит 20 слов с адресами в диапазоне 236…255 (адреса бит 236 000… 25 515 соответственно). Следует отметить, что все биты импульсных синхронизирующих сигналов имеют скважность, равную 2, т.е. половина периода состояний бита – единичное (ON), половина – нулевое (OFF). При программировании полезны флаги констант 25 313 (ON) и 25 314 (OFF), флаг первого сканирования программы 25 315 для начальной установки и арифметические флаги 25 504…25 507, устанавливаемые сразу после выполнения арифметически инструкций программы.

3) Область флагов работы с сетями SYSMAC Link и SYSMAC-NET, календаря и контроля времени сканирования AR содержит 28 слов с адресами в диапазоне AR00…AR27 (адреса бит AR0000…AR2715). Информа-

ция календаря и часов представлена в двоично-десятичном коде и расположена в словах AR18…AR21 согласно табл. 3.2.

Таблица 3.2

Информация календаря и часов

Адреса бит	Параметр	Диапазон значений
AR 1800…AR 1807	Секунды	00…59
AR 1808…AR 1815	Минуты	00…59
AR 1900…AR 1907	Часы	00…24
AR 1908…AR 1915	День	01…31
AR 2000…AR2007	Месяц	01…12 (январь-декабрь)
AR 2008…AR 2015	Год	00…99
AR 2100…AR 2107	Дни недели	00…06 (воскресенье-суббота)
AR 2108…AR 2115	Управление календарем и часами	

Максимальное AR26 и текущее AR27 времена сканирования тоже представлены в двоично-десятичном коде и могут иметь значения 000,0… 999,9 мс.

4) Область шестнадцатиразрядных двоичных данных DM включает 2000 слов. Особенностью этой области памяти является то, что запись и чтение выполняются словами и обращение к отдельным битам невозможно. При исчезновении питания информация в области DM сохраняется. Только в этой области допустима косвенная адресация к ячейкам памяти. Например, в ячейке DM500 находится число 375 и запись «DM500» означает, что используются данные из ячейки с адресом, указанным в DM500, т.е. из ячейки DM375.

В области DM выделяются:

- 968 слов (DM0…DM968) общего назначения для чтения и записи;
- 30 слов (DM969…DM999) для размещения архива ошибок в формате по три слова: первое слово – код ошибки; второе слово – секунды и минуты; третье слово – часы и день месяца;
- 1000 слов для использования специальными модулями (возможна запись с консоли программирования), причем каждый модуль использует слово с адресом DM1N00…DM1N99, где N – номер специального модуля.

5) Область данных постоянного хранения HR содержит 100 слов в диапазоне HR00…HR99 (адреса бит HR0000…HR9915). Значения данных сохраняются в области при прерывании питания, изменении режима работы и остановке программируемого контроллера.

6) Область данных таймеров и счетчиков TC включает 512 ячеек с адресами в диапазоне 000…511. Каждая ячейка этой области памяти хранит три типа данных:

- значение предварительной установки (SV) в двоично-десятичном коде (значения сохраняются при выключении питания);
- текущее значение (PV) в двоично-десятичном коде (при выключении питания и программном сбросе у таймеров текущее значение сбрасывается, у счетчиков – сохраняется);
- битовая переменная (флаг завершения), которая устанавливается при равенстве значений предварительной установки и текущего времени (перед адресом флага завершения для таймеров добавляются символы TIM, для счетчиков – CNT).

7) Область общих данных связи LR содержит 64 слова в диапазоне LR00…LR63 (адреса бит LR0000…LR6315) и используется при работе контроллеров в сетях связи PC-Link, SYSMAC-Link или SYSMAC-NET. Если сетей нет, то слова и биты этой области можно использовать как рабочие. При перерыве питания информация не сохраняется.

8) Область TR содержит 8 бит (TR0…TR7), которые используются в качестве промежуточных переменных для программирования сложных ветвящихся логических условий.

9) Область памяти для управляющей программы объемом 4 или 8 Кслов выполнена на энергонезависимом ОЗУ или перепрограммируемом ПЗУ, расположенном в модуле памяти.

Модули ввода/вывода и распределение области памяти IR.

К модулям ввода/вывода контроллера Omron C200H относятся: стандартные, модули «группы 2» и специальные. Стандартные модули предназначены для ввода или вывода 5, 8, 12 или 16 (в зависимости от модификации) дискретных битовых сигналов и используют не более одного слова области IR. Адреса входов и выходов этих модулей зависят от места размещения модуля на базовой панели и от типа блока. Адрес состоит из 5 цифр и определяется согласно рис. 3.8.

Все входы и выходы дискретных сигналов имеют гальваническую развязку и объединены в группы по 8. В зависимости от модификации модуля входы позволяют вводить сигналы напряжением 220 или 24 В. Выходы представляют собой замыкающие релейные контакты или транзисторы (коммутируемое напряжение 5…24 В постоянного тока), или тиристоры (для коммутации 220 В переменного тока).

Модули «группы 2» предназначены для ввода или вывода 32 или 64 дискретных сигналов и занимают соответственно 2 или 4 слова в области IR с адресами в диапазоне 030…049. Конкретные адреса используемых модулем слов определяются его номером, который устанавливается переключателем (табл. 3.3).

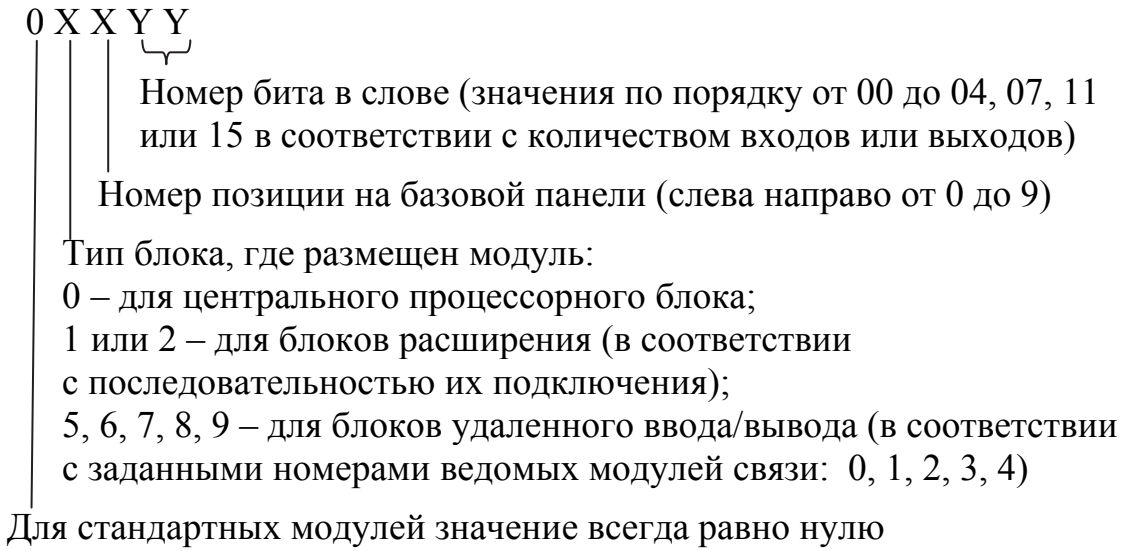


Рис. 3.8. Формат адресов блоков ввода/вывода

Таблица 3.3

Модули «группы 2»

32-точечные модули		64-точечные модули	
Номер	Слова	Номер	Слова
0	030…031	0	030…033
1	032…033	1	032…035
2	034…035	2	034…037
3	036…037	3	036…039
4	038…039	4	038…041
5	040…041	5	040…043
6	042…043	6	042…045
7	044…045	7	044…047
8	046…047	8	046…049
9	048…049	9	Не используется

При установке номеров 64-точечных модулей необходимо следить, чтобы адреса используемых слов не перекрывались с адресами других модулей. Конструктивно входы/выходы модулей «группы 2» также имеют гальваническую развязку, напряжения входных сигналов 5…24 В, выходы транзисторные. Сигналы входа и выхода заводятся в модуль через разъем: ряд A разъема CN1 занимает первое слово, ряд B CN1 – второе слово, ряд A CN2 – третье слово и ряд B CN2 – четвертое слово. Контроль состояния входов и выходов, так же как и у стандартных модулей, может осуществляться по светодиодам на лицевой панели блока, но у 64-точечных блоков всего 32 светодиода и имеется переключатель для контроля первого и второго слова или третьего и четвертого.

Специальные модули также занимают более одного слова в области IR (максимально до 10), и адреса используемых слов задаются номером модуля, который устанавливается на лицевой панели модуля с помощью переключателя (рис. 3.9).

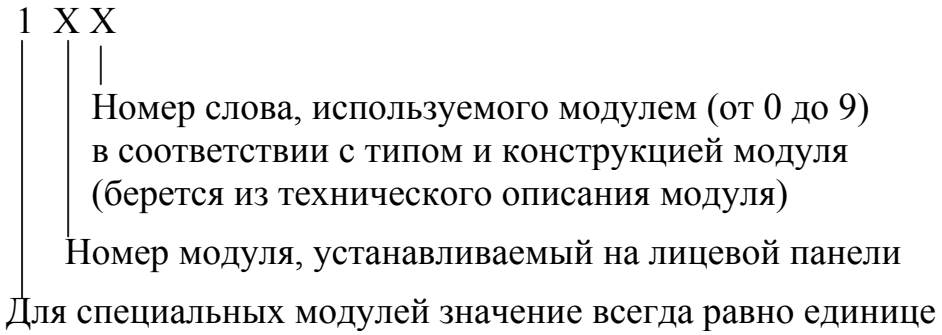


Рис. 3.9. Формат модуля переключателя

Специальные модули – наиболее разнообразная группа модулей. Среди них выделяют:

1) Модули многоточечного ввода/вывода дискретных сигналов, которые позволяют вводить или выводить 32 дискретных сигнала (параметры сигналов аналогичны модулям «группы 2»). Существуют специальные модули, позволяющие выводить и вводить 16 дискретных сигналов одновременно. Эти модули могут подключаться в динамическом режиме, что позволяет организовать последовательный опрос 128 кнопок или переключателей.

2) Модули аналогового ввода имеют один аналого-цифровой преобразователь (АЦП) и коммутаторы аналоговых сигналов на 4 или 8 входов, а модули аналогового вывода имеют один цифро-аналоговый преобразователь (ЦАП) и коммутатор на 2 или 4 выхода. Переключателями устанавливается вид преобразуемых аналоговых сигналов: 4…20 мА, 1…5 В или 0…10 В, которые в цифровой форме представлены 12-разрядными двоичными числами. Каждый канал в области IR занимает одно слово.

3) Модули температурных датчиков имеют 4 входных канала для подключения термопар или термосопротивленпй, позволяют измерять температуру в диапазоне 0…1000 °С (термопары) и -50…400 °С (термосопротивления) с точностью ±1 %, время преобразования 1,2 с. Модули требуют 10 слов области IR.

4) Модули регулирования температуры имеют 2 контура регулирования. В зависимости от типа подключаемого датчика они имеют несколько диапазонов измерения температуры, которые в целом перекрывают диапазон 200…1800 °С. Для управления используется токовый выход, выход напряжения или транзисторный выход, возможна реализация про-

порционально-интегро-дифференцирующего (ПИД) или релейного закона управления.

5) Модули ПИД-регулирования также имеют 2 контура с АЦП для токовых сигналов и сигналов напряжения. Период квантования составляет 100 мс, установка постоянных времени в диапазоне 0…100 с.

6) Модули высокоскоростного счетчика предназначены для реверсивного счета импульсов с импульсных датчиков скорости или положения с частотой до 75 кГц. Модуль имеет несколько режимов, позволяющих линейно считать импульсы, сравнивать с заданным значением, вычитать из него, управлять счетом от внешнего сигнала.

7) Модули систем позиционного управления предназначены для управления шаговым или следящим электроприводом и выдают импульсы унитарного кода с частотой 1…250 кГц (это соответствует скорости 150 м/мин при перемещении 10 мкм на импульс) в диапазоне +8 388 606…-388 607 импульсов. В модуле реализовано управление по одной оси (модуль запоминает 20 позиций) или по двум осям (53 позиции).

8) ASCII-модуль контроллера Omron предназначен для связи с периферийным устройством (принтером, модемом, дисплеем, клавиатурой), обработки данных и представления информации в требуемом виде. Программируется модуль на языке Бейсик, имеет 24 Кбайт электрически перепрограммируемого ППЗУ и 24 Кбайт ОЗУ, два порта RS-232C: один для программирования от ПЭВМ, другой для подключения периферийного устройства.

Фирмы постоянно работают над совершенствованием и разработкой новых модулей, поэтому, естественно, представленный обзор не охватывает всех выпускаемых и готовящихся к выпуску модулей.

Основные этапы подготовки и составления программ:

1) Определите, что должна делать управляемая система (объект управления); цели управления и другие дополнительные функции, выполняемые системой управления; последовательность выполняемых операций исполнительных механизмов с привязкой ко времени и к состоянию датчиков и устройств управления.

2) Определите, какие модули требуются для подключения датчиков, устройств управления, исполнительных механизмов, средств сигнализации и другой аппаратуры, необходимой в управлении объектом. При выборе модулей проведите анализ параметров сигналов входа/выхода (уровни напряжений и тока) и функциональных возможностей модулей. Обратите внимание на возможность применения интеллектуальных модулей, которые могут выполнять предварительную обработку входных сигналов и некоторые функции локального управления, что позволит значительно уменьшить сложность программирования.

3) Составьте схему или таблицу подключения к модулям контроллера всех входных и выходных устройств объекта управления и аппаратуры, используемой в процессе управления.

4) Составьте перечень или таблицу с указанием адресов слов или бит областей памяти, которые заняты выбранными модулями, где укажите распределение бит и слов по конкретным устройствам и выполняемым функциям в управлении объектом и работой интеллектуальных модулей ввода/вывода и модулей связи.

5) Определите рабочие биты и слова (промежуточные данные) областей памяти контроллера и запишите в таблицу с указанием, как вы их используете в программе. Аналогичные таблицы составьте для таймеров, счетчиков и управляющих данных (номера подпрограмм, шагов, переходов и т.д.).

6) Составьте релейно-контактную схему программы (схему GRAFCET при использовании последовательно-функционального программирования), которая реализует заданную последовательность операций, их взаимосвязь и отрабатывает возможные аварийные ситуации.

7) Введите программу и все требуемые рабочие параметры в контроллер. При вводе программы с консоли необходимо предварительно преобразовать ее в мнемокод.

8) Проверьте программу на синтаксические ошибки и откорректируйте их, а затем на ошибки выполнения и тоже внесите соответствующие исправления.

9) После того как система управления будет полностью собрана, проверьте программу в реальном процессе управления и, если необходимо, проведите более точную настройку.

10) Запишите две копии окончательной программы и храните их в разных местах.

3.4.1. Принципы релейно-контактного программирования

Система команд (инструкций) контроллера составляет 12 базовых и 160 специальных инструкций. При программировании контроллеров Omron используются три варианта представления программы: в виде мнемокода, в виде релейно-контактной схемы и в виде функциональной схемы (схемы на логических элементах). Представление программы в виде релейно-контактной схемы считается наиболее удобным при реализации логики управления, широко распространено и часто встречается также в контроллерах других фирм. Представление программы в виде мнемокода используется при программировании и отладке с переносной портативной консоли, что удобно в производственных условиях, поэтому целесообразно остановиться на этих двух формах представления программы.

Базовые инструкции позволяют программировать логику управления в виде последовательной логической зависимости (цепочки) некоторых событий (условий), представляемых операндами-битами. Базовые инструкции LOAD и LOAD NOT задают первое условие в новой цепочке. После задания это условие считается текущим и последовательно изменяется базовыми инструкциями AND, AND NOT, OR, OR NOT. Каждая из этих инструкций вводит новый операнд-бит и выполняет соответствующую логическую операцию между ним и текущим условием. Результат этой операции становится новым текущим условием.

Цепочки базовых инструкций могут образовывать сложные условия, но всегда заканчиваются одним или несколькими базовыми инструкциями вывода OUT или OUT NOT, которые присваивают текущее значение или его инверсию операнду-биту управляемого события или подаются на управляющие входы базовых инструкций таймера (TIM), счетчика (CNT) или специальных инструкций. Некоторые специальные функции управления программой, выполняющие роль меток, не имеют условий выполнения.

Релейно-контактная схема состоит из одной вертикальной линии, от которой вправо отходят последовательности (цепочки) базовых инструкций, определяющие логическую функцию управления.

Базовые инструкции LOAD, AND и OR изображаются замыкающей контактной группой (две вертикальные короткие черты) с указанным рядом операндом-битом, который определяет условие в цепочке инструкции: активное (ON) или неактивное (OFF). Базовые инструкции LOAD NOT, AND NOT и OR NOT изображаются размыкающей контактной группой (две вертикальные контактные черты с косой линией между ними) и определяют условие, противоположное значению указанного рядом операнда-бита. Аналогично базовая инструкция вывода OUT, изображаемая двумя круглыми скобками, присваивает указанному рядом операнду-биту полученное в цепочке впереди стоящих инструкций условие выполнения, а инструкция вывода OUT NOT присваивает инверсные значения этого условия.

На рис. 3.10 дан пример цепочки управления состоянием бита, представленный в мнемокоде (рис. 3.10, *а*) и в виде релейно-контактной схемы (рис. 3.10, *б*).

В некоторых случаях программируемое условие может представлять собой две или более последовательно или параллельно соединенные сложные цепочки. В этом случае используется понятие ***логического блока***, когда каждая такая цепочка программируется отдельно, начиная с инструкции LOAD или LOAD NOT, а затем выполняется действие над условиями блоков с помощью инструкции AND LOAD или OR LOAD. Пример с последовательными логическими блоками представлен на рис. 3.11.

а) б)

LD	0005
AND NOT	0006
OR	0007
AND	0008
OR NOT	0009
AND	0010
OUT	0201

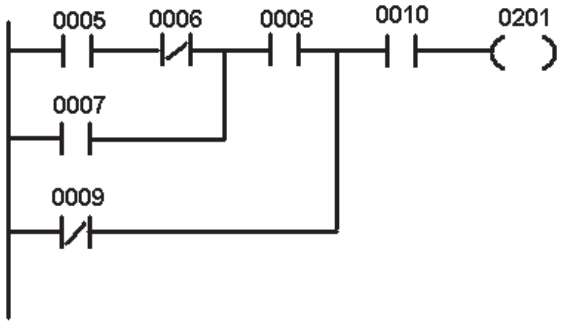


Рис. 3.10. Цепочки управления состоянием бита

Действия над логическими блоками можно пояснить следующей схемой: когда появляется команда LD, вводящая первое условие нового блока, результат предыдущего логического блока сохраняется в стеке.

а)

LD	0001	Логические блоки
OR NOT	0002	
LD	0003	
OR	0004	
AND LD		
TIM	1# 50	

б)

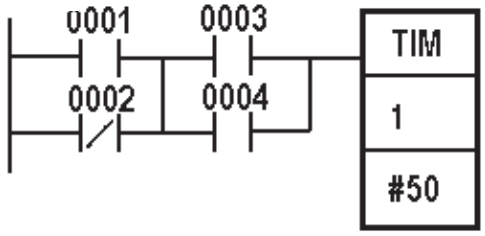


Рис. 3.11. Цепочка с последовательными логическими блоками

После выполнения второго блока инструкции AND LD или OR LD выполняют действие между условием текущего блока и вызванным из стека условием предыдущего блока. При программировании логических блоков возможны два подхода:

1) сначала определить условия всех блоков, а затем выполнить действие над ними (рис. 3.12, *б*);

2) последовательно определить условия блоков и сразу выполнить действия между условием блока и текущим значением условия основной цепи (рис. 3.12, *в*).

В данном примере две записи программы в виде мнемокода соответствуют одной и той же релейной схеме (рис. 3.12, *а*). Первый подход не позволяет описать более 8 блоков, тогда как при втором подходе таких ограничений нет.

При программировании ветвящихся цепочек используют переменные области TR, в которых инструкциями OUT TR фиксируются условия в точ-

ках ветвления. Например, релейная схема, представленная на рис. 3.13, *а*, в мнемокоде может быть записана в виде, представленном на рис. 3.13, *б*.

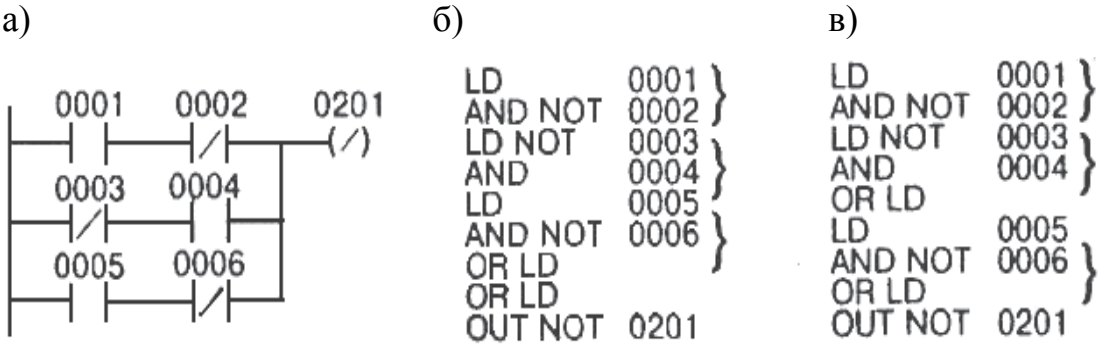


Рис. 3.12. Две записи программы на LD

Так как в области TR всего 8 бит, то цепочка может иметь максимально 8 точек ветвления.

Если программируется специальная инструкция, имеющая несколько входов, то для каждого входа формируется отдельное условие. На рис. 3.14 представлен пример программирования специальной инструкции сдвига, которая имеет 3 входа.

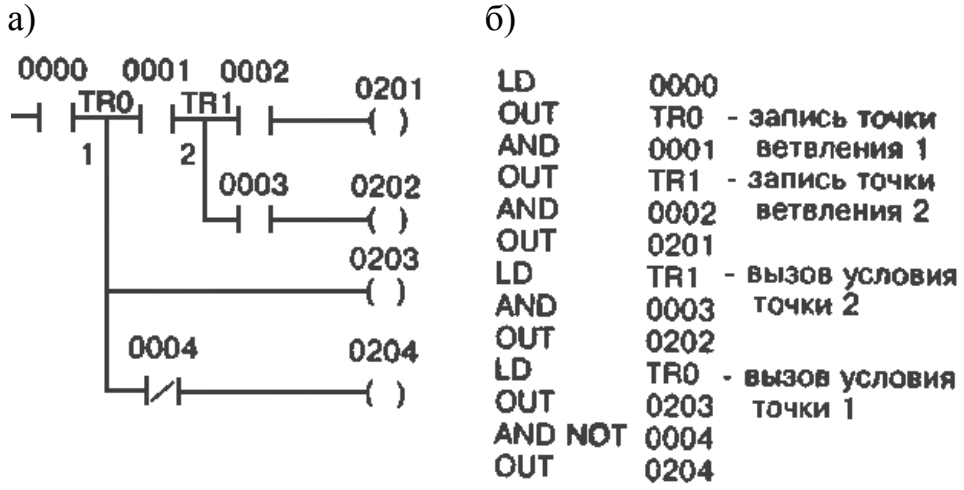


Рис. 3.13. Пример с ветвящимися цепочками

Последняя цепочка программы должна иметь одну инструкцию END(01), которая не имеет условий. Если этой инструкции нет, то программа не будет выполняться. Если END(01) поставить в середине программы, то будет выполняться часть программы от начала до END.

Если какая-либо инструкция должна выполняться постоянно, то в качестве условия для нее можно использовать флаг константы «ON» из области SR (бит 25313). Если вход инструкции всегда должен быть неак-

тивным (например, нет сброса на входе *R* в функции сдвига SFT(10)), то можно использовать флаг константы «OFF» (бит 25314).

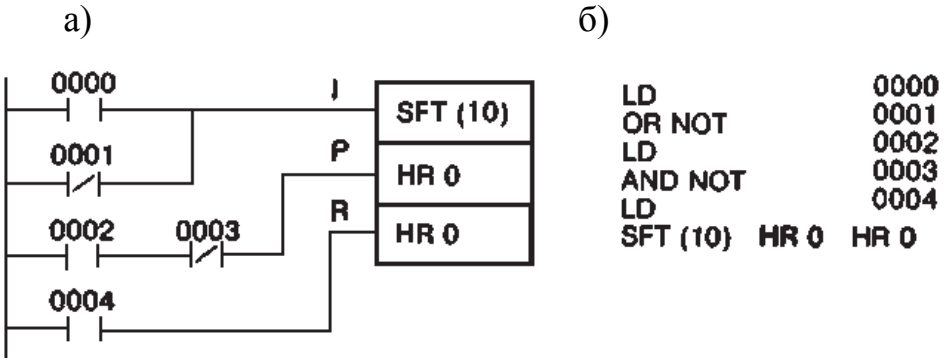


Рис. 3.14. Пример с функцией сдвига

Для слов и бит из областей IR и SR указываются только их адреса, для слов и бит из других областей памяти обязательно указание букв областей: HR, AR, LR, DM, TR. Указание бит завершения области TC осуществляется буквами: TIM – для таймеров, CNT – для счетчиков. Косвенная адресация в области DM обозначается звездочкой перед DM, например: *DM0001. Числовые константы впереди имеют символ #.

Большинство инструкций имеет дифференциальную форму, которая обозначается символом @ перед их мнемоникой. Инструкции недифференциальной формы выполняются при каждом сканировании, пока условие находится в ON. Инструкции дифференциальной формы выполняются только один раз, когда условие их выполнения переходит из OFF в ON. Для этих же целей можно использовать функции дифференцирования DIFU(13) и DIFD(14).

Некоторые инструкции при их выполнении воздействуют на арифметические флаги области SR (например: инструкции сравнения, арифметических операций, пересылок). Если состояние флагов используется в алгоритме управления, то его необходимо сразу запомнить в промежуточных рабочих битах до исполнения другой инструкции, влияющей на эти флаги.

3.4.2. Последовательно-функциональное программирование

Последовательно-функциональное программирование заключается в разбиении процесса управления на отдельные этапы (шаги), которые начинают выполняться (становятся активными) после наступления определенного события. Такой подход удобен для управления процессами, включающими ряд последовательных операций. Графическое представление и ввод программ с использованием принципов последовательно-функционального программирования получили название GRAFCET.

Версия GRAFCET включает постоянно выполняемую часть программы Permanent и совокупность этапов (шагов) с условиями их запуска и направлением переходов (аналогично блок-схеме алгоритма). Схема GRAFCET может включать следующие элементы:

- ***Шаг*** – часть программы, имеющая единое функциональное значение в управлении. Шаг связывается с какой-то битовой переменной и становится активным, если значение переменной ON, или неактивным, если значение этой переменной OFF. Шаг изображается в виде прямоугольника с указанием адреса битовой переменной (рис. 3.15, *а*).

Каждый шаг GRAFCET и постоянная часть программы Permanent программируются в режиме релейно-контактного программирования или в режиме мнемокода. В общем случае он может содержать много цепочек. Шаги программы не должны содержать цепочки с функцией END(01), но в конце постоянной части такая цепочка необходима.

Точка входа в GRAFCET указывается начальным шагом (двойные линии вверху и внизу прямоугольника, рис. 3.15, *б*). Активизируется начальный шаг программы при запуске контроллера. Шаг может также представлять подсхему GRAFCET. В этом случае он изображается с двойными линиями по боковым сторонам (рис. 3.15, *в*).

- ***Переход*** – барьер, разделяющий шаги и определяющий условия запуска следующего шага программы. Переход также связывается с битовой переменной, и если условие выполняется, то эта переменная принимает значение ON, а если не выполняется – значение OFF. Графическое изображение перехода имеет вид линии (рис. 3.15, *г*) с пересечением, и рядом записывается адрес переменной, связанной с этим переходом. Переход программируется только одной цепочкой релейно-контактной схемы, которая задает условие для переменной (выхода), связанной с данным переходом.

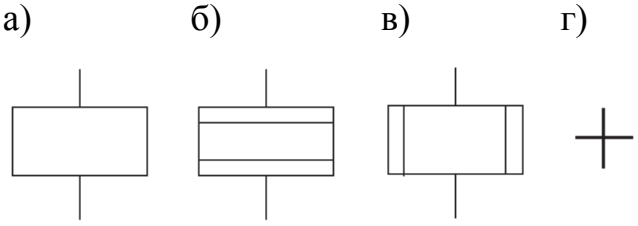


Рис. 3.15. Элементы структуры

- ***Связь*** – указание последовательности взаимодействия шагов и переходов GRAFCET. Используется четыре типа связи:

1) *Вертикальная* – для указания последовательности «шаг-переход» или «переход-шаг» (изображается вертикальной прямой линией).

2) *Селективная* – для указания нескольких альтернативных переходов (изображается горизонтальной одинарной линией), например:

шаг 2 активизируется, если активен шаг 1 и выполняется условие перехода ***A*** или/и шаг 3 активизируется, если активен шаг 1 и выполняется условие перехода ***B*** («расходимость по схеме ИЛИ», рис. 3.16, *а*);

шаг 3 активизируется, если активен шаг 1 и выполняется условие перехода ***A*** или активен шаг 2 и выполняется условие перехода ***B*** («сходимость по схеме ИЛИ», рис. 3.16, *б*).

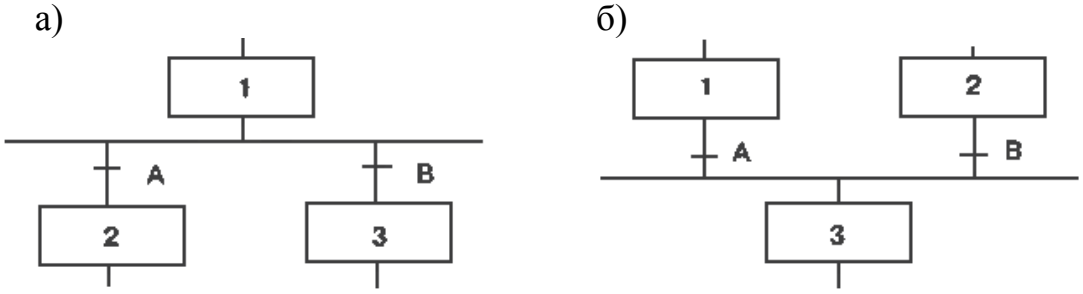


Рис. 3.16. Типы связей шагов программы

3) *Одновременная* – для активизации одновременно нескольких шагов программы (отображается горизонтальной двойной линией), например:

если активен шаг 1 и выполняется условие перехода ***A***, то одновременно начинают выполняться и шаг 2, и шаг 3 («расходимость по схеме И», рис. 3.17, *а*);

шаг 3 становится активным, если одновременно активны и шаг 1, и шаг 2, и выполняется условие перехода ***A*** («сходимость по схеме И», рис. 3.17, *б*).

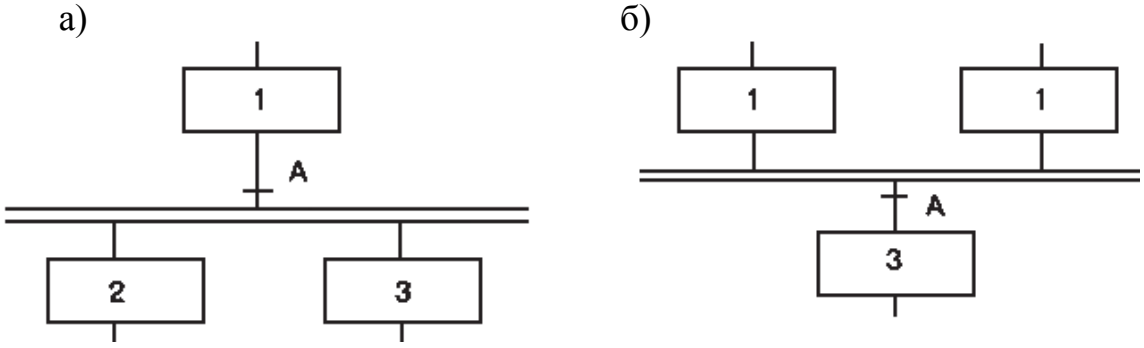


Рис. 3.17. Типы переходов программы

4) *Ссылка* – связь, указываемая по номеру (адресу переменной), с выхода перехода на шаг (отображается стрелкой, направленной вниз, с указанием номера шага, куда выполняется переход, при этом в изображении шага тоже появляется знак стрелки).

Основные правила GRAFCET:

- шаги всегда должны быть разделены переходами, между переходами всегда должны быть шаги;
- шаг становится активным, если был активным предыдущий шаг и выполнено условие перехода, разделяющего эти шаги;
- шаг становится неактивным, если стал активным следующий шаг после выполнения условия, запрограммированного в разделяющем их переходе.

На рис. 3.18, *а* представлен пример схемы GRAFCET управления дорожным светофором.

На рис. 3.18, *б* представлена постоянная часть программы, а на рис. 3.19 представлены программы шагов и переходов GRAFCET.

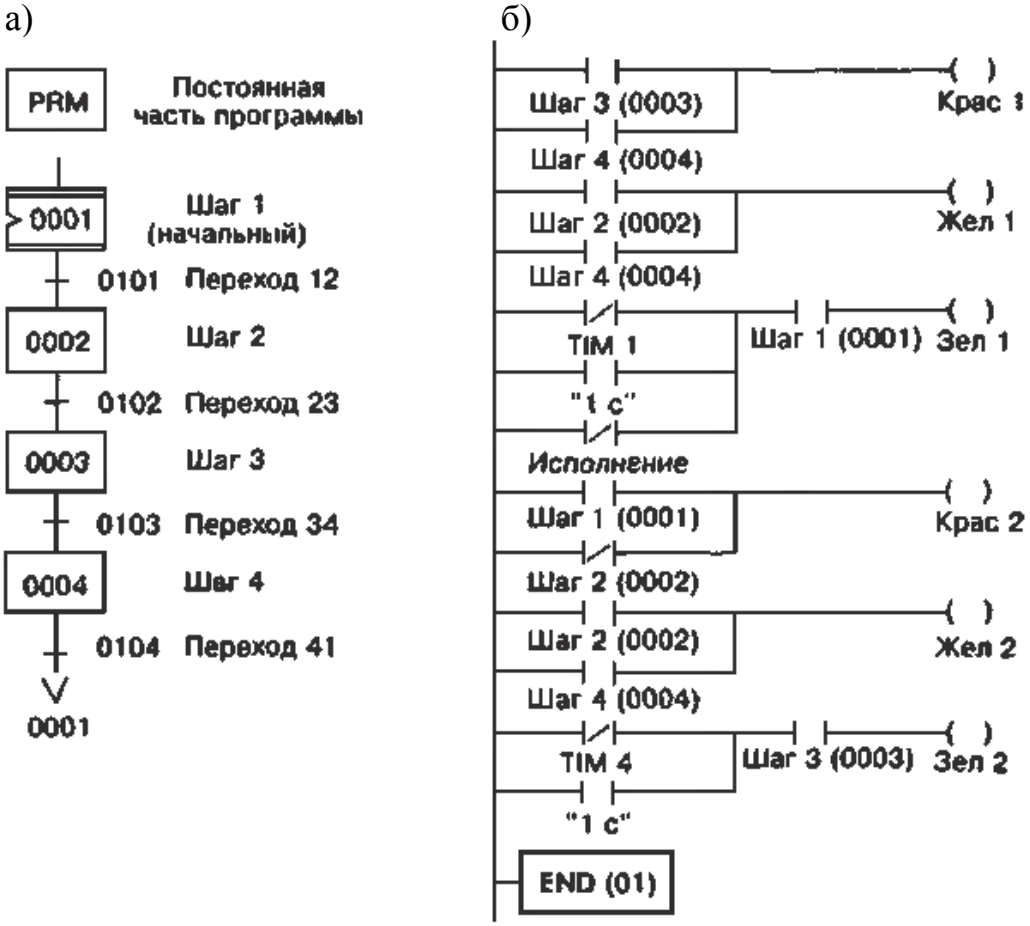


Рис. 3.18. Пример схемы GRAFCET

Исходное состояние соответствует разрешенному движению по главной дороге (огни группы 1) и запрещенному движению по второстепенной дороге (огни группы 2). В схеме GRAFCET в этом состоянии активен начальный шаг 1, и таймером TIM1 отсчитывается 15 с, в течение которых это состояние сохраняется и горит зеленый свет первой группы (Зел1).

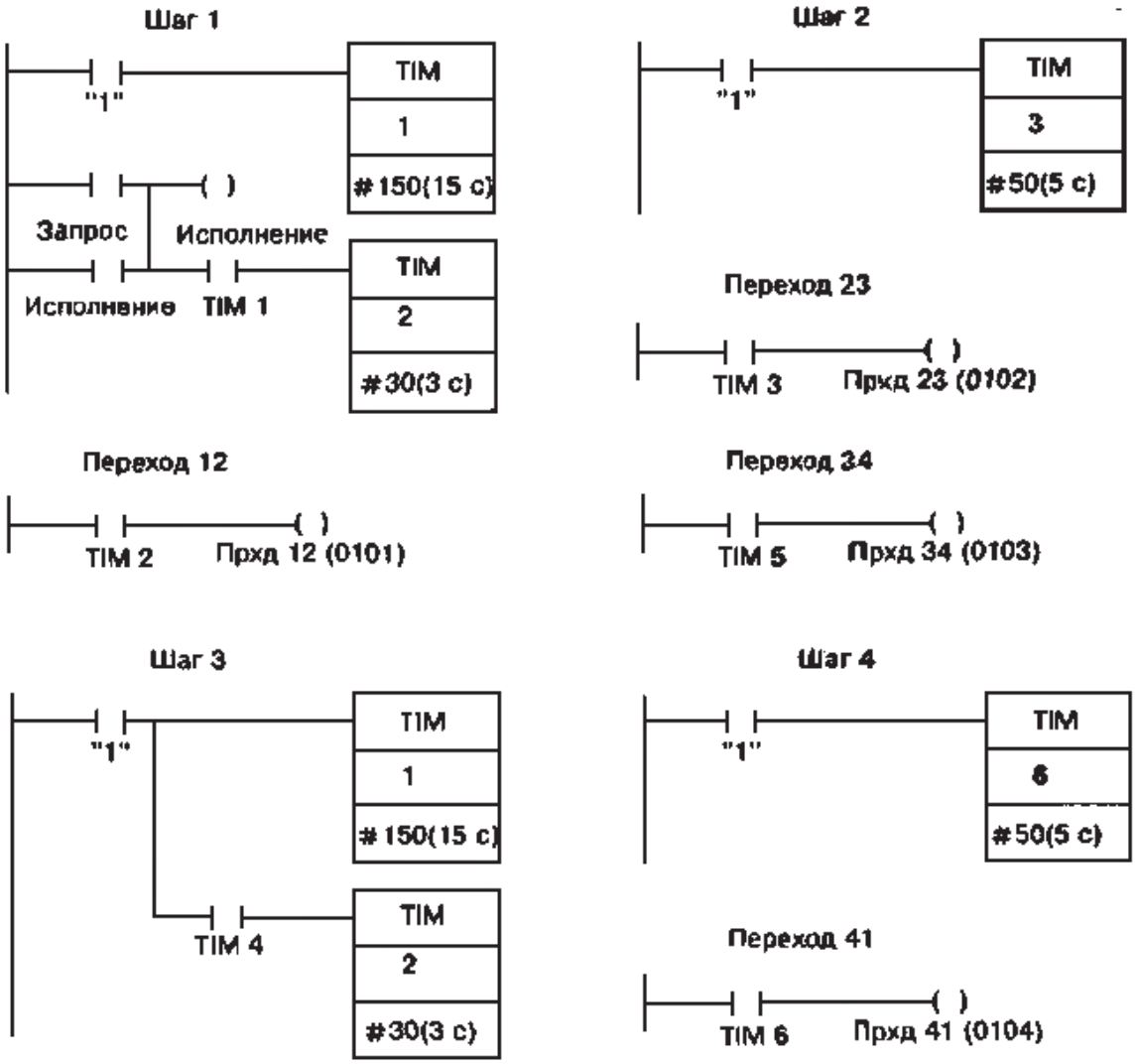


Рис. 3.19. Программы шагов и переходов GRAFCET

Рабочий цикл начинается с поступления запроса на движение по второстепенной улице. Бит «Запрос» устанавливается в ON, при этом бит «Исполнение» также переходит в ON и ставится на блокировку. В цепи выхода Зел1 размыкаются контакты «Исполнение», и через пульсирующий бит «1 с» начнет мигать Зел1 (предупреждение, что будет переключение светофора). Через 3 с таймер TIM2 установит в положение ON свой бит TIM2 в цепи перехода 12. Переменная, связанная с этим переходом, установится в ON, что сделает шаг 2 активным, а шаг 1 – пассивным. Перейдя в пассивное состояние, шаг 1 выключит зеленый свет, а активизация шага 2 через бит, связанный с этим шагом, включит желтый свет Жел1 и одновременно через постоянно установленный в ON бит «1» запустит таймер TIM3. После отсчета таймером TIM3 выдержки времени 5 с бит TIM3 в цепи перехода 23 установится в ON и установит в ON переменную, связанную с этим переходом, вследствие чего шаг 3 станет активным, а шаг 2 –

пассивным. Изменение активности шагов выключит желтый свет Жел1 и включит красный Крас1. Кроме того, шаг 3 через постоянно установленный в ON бит «1» запустит таймер TIM4, который обеспечит постоянное горение в течение 15 с зеленого света Зел2, разрешающего движение по второстепенной дороге. По окончании 15 с TIM4 запустит на счет TIM5 и снимет блокировку с пульсирующего бита «1 с», что обеспечит мигание в течение 3 с зеленого света Зел2 для предупреждения о предстоящем переключении. Переход активности к шагу 4, а затем к шагу 1 будет аналогичным.

3.4.3. Программирование контроллеров SIMATIC S7

Перед каждым проходом обработки управляющей программы ОС контроллера переписывает значения входов из входных модулей в памяти контроллера в область отображения входов. В процессе обработки программы значения входов считываются не из модулей ввода, а из этой области памяти. Таким образом исключается неоднозначность считывания значений входов, устраняется дребезг контактов, а также увеличивается скорость обработки информации, т.е. сокращается время обращения.

Выходные переменные в процессе обработки управляющей программы записывают в операционную память контроллера в область отображения выходов. По окончании обработки цикла программы ОС контроллера производит вывод области отображения выходов в модули вывода. Это производится для обеспечения однозначности состояния выходов в процессе обработки каждого цикла программ для исключения частых многократных переключений, а также для сокращения времени обращения к выходным переменным (табл. 3.4).

В контроллерах SIMATIC S7 возможно обращение к входам и выходам как через область отображения, так и минуя ее (для обработки аварийных ситуаций и для ввода/вывода аналоговых сигналов).

Таблица 3.4

Доступные области памяти, их описание и формат доступа

Область памяти	Функция	Возможности доступа
1	*2*	*3*
Входные и выходные переменные		
Область отображения входов	Обращение к входам через область отображения входов	E1.5 Бит EB2 Байт EW4 Слово ED4 Двойное слово

Продолжение табл. 3.4

1	*2*	*3*
Область отображения выходов	Вывод значений выходов в область отображения выходов	A8.5 Бит AB8 Байт AW8 Слово AD8 Двойное слово
Внутренние переменные контроллера		
Маркеры	Вспомогательные переменные программы для промежуточного хранения информации	M10.6 Бит MB4 Байт MW20 Слово MD36 Двойное слово
Периферийные входы Периферийные выходы	Непосредственный доступ к входам и выходам контроллера, минуя область отображения	PEB288 Байт PEW290 Слово PED292 Двойное слово PAB294 Байт PAW296 Слово PAD298 Двойное слово
Таймеры	Организация функций времени: формирование временных интервалов и временных задержек	T10 Номер таймера
Счётчики	Организация функции счёта	Z2 Номер счётчика
Блоки данных пользователя (data block)	Долговременное хранение данных. Блоки данных пользователя доступны из любого блока программы. Структуру таких блоков определяет разработчик программы	Для открытого блока данных: DBX2.0 Бит DBB10 Байт DBW16 Слово DBD4 Двойное слово
Блоки данных функционального блока (instance data block)	Структура блока данных «instance» повторяет структуру таблицы описания переменных связанного с ним функционального блока. В одно и то же время можно открыть два различных блока данных: один как блок данных пользователя (DB), другой как блок данных функционального блока (DI)	Для открытого блока: DIX2.0 Бит DIB10 Байт DIW16 Слово DID4 Двойное слово

Окончание табл. 3.4

1	*2*	*3*
Локальные данные	Содержат временные данные, которые могут использоваться только внутри программного блока (OB, FB или FC). Данные содержатся в так называемом L-стеке. После выполнения блока данные теряются и память освобождается	L2.0 Бит LB10 Байт LW6 Слово LD8 Двойное слово

Типы данных.

Контроллеры SIMATIC S7 могут работать со следующими типами данных:

- **Бит** (BOOL) – это единица, соответствующая одному двоичному разряду. Два возможных значения бита обозначаются «0» (FALSE) и «1» (TRUE).
- **Байт** (BYTE) состоит из 8 бит, которым соответствуют битовые адреса в диапазоне 0…7 (справа налево). Старшим является бит с большим адресом. Байт могут образовать только те биты, адрес младшего из которых кратен 8, например: 0, 8, 16 и т.д. В контроллерах SIMATIC S7 байт может интерпретироваться как просто байт (набор бит) или как ASCII-символ.
- **Слово** (WORD) – это следующая после байта по величине единица, ее длина 16 бит. Любые два соседних байта можно объединить в слово, старшим будет являться байт с меньшим адресом. Адрес слова – это адрес байта с меньшим адресом. В контроллерах SIMATIC S7 слово может интерпретироваться как просто слово (набор бит), целое число со знаком, дата, время и т.д.
- **Двойное слово** (DWORD) Любые два соседних слова можно объединить в двойное слово, его длина – 32 бита или 4 байта. Старшим словом (байтом) является слово (байт) с меньшим адресом. Адрес двойного слова – это адрес байта с меньшим адресом. В контроллерах SIMATIC S7 двойное слово можно интерпретировать как просто двойное слово, длинное целое число со знаком, вещественное число в формате IEEE и т.д.

Элементарные типы данных.

Каждый из приведённых ниже элементарных типов данных определяется форматом элемента данных:

BOOL

Тип данных	Длина, бит	Формат	Диапазон значений, пример записи
BOOL	1	Двоичный	TRUE, FALSE

Тип BOOL занимает в памяти контроллера 1 бит. Адрес бита состоит из байта, к которому он принадлежит, и битового адреса. Например, M3.0 указывает на 0-й бит 3-го байта меркерных данных. Контроллеры SIMATIC S7 имеют байтовую организацию памяти, поэтому обращение, например, M3.15 является недопустимым.

BYTE

Тип данных	Длина, бит	Формат	Диапазон значений, пример записи
BYTE	8	Шестнадцатеричный	B# 16#00…B# 16#FF

Тип BYTE занимает в памяти контроллера 1 байт. Старшим битом является имеющий больший битовый адрес:

Byte n

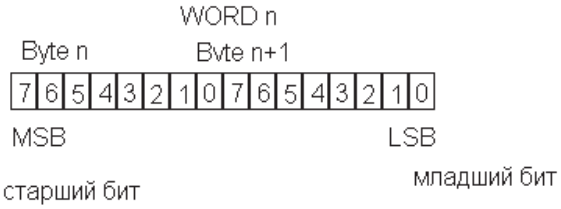
CHAR

Тип данных	Длина, бит	Формат	Диапазон значений, пример записи
CHAR	8	ASCII символ	‚A‚…

Тип CHAR занимает в памяти 1 байт и интерпретируется как ASCII-символ. Значением байта является ASCII-код символа.

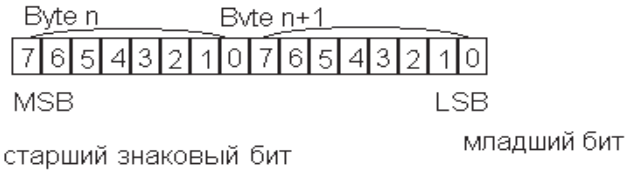
WORD

Тип данных	Длина, бит	Формат	Диапазон значений, пример записи
WORD	16	Двоичный Шестнадцатеричный Беззнаковый байтовый	2#0…2#1111 1111 1111 1111 W# 16#0000...W# 16#FFFF B#(0,,0)...B#(255,255)



INT

Тип данных	Длина, бит	Формат	Границы, пример записи
INT	16	Целый знаковый	-32768…32767



DATE

Тип данных	Длина, бит	Формат	Границы, пример записи
DATE	16	год-месяц-день	D#1990-01-01…D#2168-12-31

Дата определяется как беззнаковое число дней, прошедших с 1 января 1990 г.

Дата D#1998-01-30 будет представлена как 0000 1011 1000 0111.

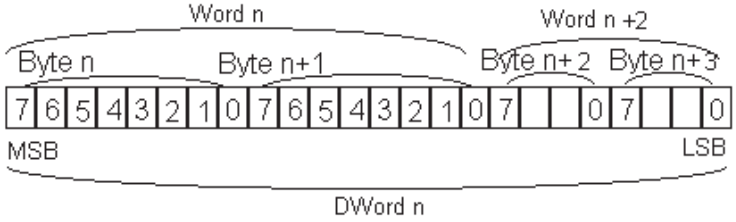
S5TIME

Тип данных	Длина, бит	Формат	Границы, пример записи
S5TIME	16	ч мин с	S5T#10ms…S5T#2h46m30s

DWORD

Тип данных	Длина, бит	Формат	Границы, пример записи
DWORD	32	Двоичный Шестнадцатеричный Беззнаковый байтовый	2#...2#11111111111111111111111111111111 DW#16#0…DW#16#FFFFFFFF B#(0,0,0,0)…B#(255,255,255,255)

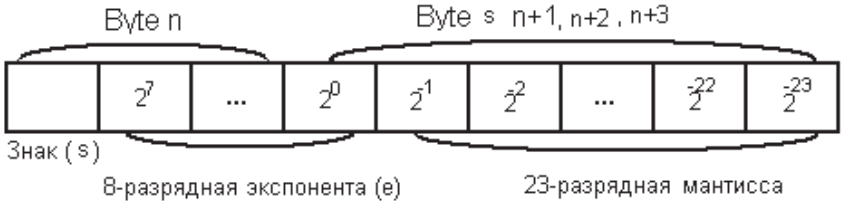
Тип DWORD соответствует двойному слову. Переменная типа DWORD состоит из двух слов или четырех байт, старшим из которых является тот, который имеет меньший адрес:



REAL

Тип данных	Длина, бит	Формат	Границы, пример записи
REAL	32	Вещественный с плавающей точкой	Ноль 0.0 Положит. +1.17e-38 Отрицат. -1.17e-38

Числа округляются до шестого знака после запятой.



Типы данных могут быть составными (пример в табл. 3.5).

Таблица 3.5

Составные типы данных DATE_AND_TIME или DT

Функция	Описание
1	*2*
Функции преобразования типов	
FC3 "D TOD DT"	Формирует из даты (в формате DATE) и времени (в формате TIME OF DAY) выходную величину в формате DT
FC6 "DT DATE"	«Вытаскивает» из входной величины в формате DT дату в формате DATE
FC7 "DT DAY"	«Вытаскивает» из входной величины в формате DT день недели в формате INT: 1 – воскресение, 2 – понедельник и т.д.
FC8 "DT TOD"	«Вытаскивает» из входной величины в формате DT время дня в формате TIME OF DAY
Функции сравнения	
FC9 "EQ_DT"	Сравнивает две входные величины (DT1 и DT2), заданные в формате DT. Если выполняется условие DT1 = DT2, то устанавливает в «1» выходную величину формата BOOL
FC12 "GE DT"	Сравнивает две входные величины (DT1 и DT2), заданные в формате DT. Если выполняется условие DT1 > DT2, то устанавливает в «1» выходную величину формата BOOL
FC14 "GT_DT"	Сравнивает две входные величины (DT1 и DT2), заданные в формате DT. Если выполняется условие DT1 > DT2, то устанавливает в «1» выходную величину формата BOOL
FC18 "LEJDT"	Сравнивает две входные величины (DT1 и DT2), заданные в формате DT. Если выполняется условие DT1 < DT2, то устанавливает в «1» выходную величину формата BOOL

Продолжение табл. 3.5

1	*2*
FC23 "LT_DT"	Сравнивает две входные величины (DT1 и DT2), заданные в формате DT. Если выполняется условие DTKDT2, то устанавливает в «1» выходную величину формата BOOL
FC28 "NE_DT"	Сравнивает две входные величины (DT1 и DT2), заданные в формате DT. Если выполняется условие DT1*DT2, то устанавливает в «1» выходную величину формата BOOL
Функции добавления/вычитания смещения	
FC1 "AD_DT_TM H"	Добавляет смещение, заданное входной величиной в формате TIME, к входной величине формата DT. Результат записывает в выходную переменную формата DT
FC34 "SB DT DT"	Находит разницу двух входных параметров, заданных в формате DT, *i*-й результат записывает в выходную переменную формата TIME
FC35 "SB_DT_TM"	Вычитает смещение, заданное входной величиной в формате TIME, из входной величины формата DT. Результат записывает в выходную переменную формата DT

STRING[n] или STRING

Тип данных	Длина	Формат	Границы, пример записи
STRING[n] STRING	N + 2	Строка ASCII символов изменяемой длины, *n* – длина строки, макс. 254 символа	STRING[2] – ‘АБ’ STRING[10] –‘АБ Г’
Примечание – Порядок байт строки STRING[4] со значением ‘АБ’ Максимальный байт 0 – Длина строки: 4 Байт 1 – реальная длина ‘АБ’= 2; Байт 2 – ASCII код ‘А’; Байт 3 – ASCII код ‘Б’			

3.5. Пример программирования системы весодозирования и подачи сыпучих материалов дуговой электросталеплавильной печи

3.5.1. Технологический процесс плавки в дуговой электросталеплавильной печи

Выплавка стали в примере осуществляется в двух дуговых электропечах с основной футеровкой, водоохлаждаемыми стенами и сводом, оснащенными газокислородными горелками, кислородными фурмами и системами фирм «FUCHS» и «STEIN», разливкой на МНЛЗ.

Дуговая печь состоит из рабочего пространства (собственно печи) с электродами и токоподводами и механизмов, обеспечивающих наклон печи, удержание и перемещение электродов и загрузку шихты.

Плавку стали ведут в рабочем пространстве, ограниченном сверху куполообразным сводом, снизу сферическим подом и с боков стенками.

Огнеупорная кладка пода и стен заключена в металлический кожух. Съемный свод набран из огнеупорных кирпичей, опирающихся на опорное кольцо. Через три симметрично расположенных в своде отверстия в рабочее пространство введены токопроводящие электроды, которые с помощью специальных механизмов могут перемещаться вверх и вниз. Печь питается трехфазным током. Шихтовые материалы загружают на под печи, после их расплавления в печи образуется слой металла и шлака. Плавление и нагрев осуществляются за счет тепла электрических дуг, возникающих между электродами и жидким металлом или металлической шихтой. Выпуск готовой стали и шлака осуществляется через сталевыпускное отверстие и желоб путем наклона рабочего пространства. Рабочее окно, закрываемое заслонкой, предназначено для контроля за ходом плавки, ремонта пода и загрузки материалов.

Основной составляющей шихты (75…100 %) электроплавки является стальной лом. Лом не должен включать цветных металлов и должен иметь минимальное количество никеля и меди; желательно, чтобы содержание фосфора в ломе не превышало 0,05 %. При более высоком содержании фосфора продолжительность плавки возрастает. Лом не должен быть сильно окисленным (ржавым) – с ржавчиной (гидратом окиси железа) вносится в металл много водорода. Лом должен быть тяжеловесным, чтобы обеспечивалась загрузка шихты в один прием (одной бадьей). При легковесном ломе после частичного расплавления первой порции шихты приходится вновь открывать печь и подсаживать шихту, что увеличивает продолжительность плавки.

В качестве шлакообразующих в основных печах применяют известь, известняк, плавиковый шпат, боксит, шамотный бой; в кислых печах – кварцевый песок, шамотный бой, известь. В качестве окислителей исполь-

зуют железную руду, прокатную окалину, агломерат, железные окатыши, газообразный кислород. К шлакообразующим и окислителям предъявляются те же требования, что и при других сталеплавильных процессах: известь не должна содержать более 90 % CaO, менее 2 % SiO_2, менее 0,1 % S и быть свежеобоженной, чтобы не вносить в металл водород. Железная руда должна содержать менее 8 % SiO_2, поскольку он понижает основность шлака, менее 0,05 % S и менее 0,2 % P; желательно применять руду с размером кусков 40…100 мм, поскольку такие куски легко проходят через слой шлака и непосредственно реагируют с металлом. В плавиковом шпате, применяемом для разжижения шлака, содержание CaF_2 должно превышать 85 %. После окончания завалки электроды опускают почти до касания с шихтой и включают ток. Под действием высокой температуры дуг шихта под электродами плавится, жидкий металл стекает вниз, накапливаясь в центральной части подины. Электроды постепенно опускаются, проплавляя в шихте «колодцы» и достигая крайнего нижнего положения. По мере увеличения количества жидкого металла электроды поднимаются. Это достигается при помощи автоматических регуляторов для поддержания определенной длины дуги. Плавление ведут при максимальной мощности печного трансформатора.

Во время плавления происходит окисление составляющих шихты, формируется шлак, происходит частичное удаление в шлак фосфора и серы.

Окисление примесей осуществляется за счет кислорода воздуха, окалины и ржавчины, внесенных металлической шихтой.

За время плавления полностью окисляется кремний, 40…60 % марганца, частично окисляется углерод и железо. В формировании шлака наряду с продуктами окисления (SiO_2, MnO, FeO) принимает участие и окись кальция, содержащаяся в извести. Шлак к концу периода плавления имеет примерно следующий состав: 35…40 % CaO; 15…25 % SiO_2; 8…15 % FeO; 5…10 % MnO; 3…7 % Al_2O_3; 0,5…1,2 % P_2O_5. Низкая температура и наличие основного железистого шлака благоприятствуют дефосфорации. В зоне электрических дуг за время плавления испаряется 2…5 % металла, преимущественно железа.

В процессе расплавления возможна присадка в печь извести, а также твердых окислителей: железной руды, агломерата, железорудных окатышей, окалины. Для ускорения процесса проплавления металлошихты после завалки и подвалки используются стеновые или дверные газокислородные горелки. Подача кислорода, вводимого через сводовую водоохлаждаемую фурму, начинается после проплавления колодцев и образования жидкой ванны (через 10…15 мин после включения печи). В конце расплавления производится обновление шлака. При этом количество и свойства шлака в печи должны обеспечивать работу с максимально возможным заглублением дуг в шлак, для чего в течение всего периода шлак поддерживается во

вспененном состоянии периодическими присадками дробленого кокса порциями до 50 кг через сводовое загрузочное устройство. Задача окислительного периода плавки состоит в следующем:

- уменьшить содержание в металле фосфора до 0,01…0,015 %;
- уменьшить содержание в металле водорода и азота;
- нагреть металл до температуры, близкой к температуре выпуска (на 120…130 °С выше температуры ликвидуса).

Кроме того, за время периода окисляют углерод до нижнего предела его содержания в выплавляемой стали. За счет кипения (выделения пузырьков CO при окислении углерода) происходит дегазация металла и его перемешивание, что ускоряет процессы дефосфорации и нагрева. Окисление углерода производится газообразным кислородом, вводимым через сводовую водоохлаждаемую фурму. Окислительный период начинается с того, что из печи сливают 65…75 % шлака, образовавшегося в период плавления. Шлак сливают, не выключая печь, наклонив её в сторону рабочего окна на 10…12°. Слив шлака производят для того, чтобы удалить из печи перешедший в шлак фосфор. Удалив шлак, в печь присаживают шлакообразующие: 1…1,5 % извести и при необходимости 0,15…0,25 % плавикового шпата, шамотного боя или боксита. В течение всего окислительного периода производится присадка шлакообразующих и твердых окислителей для поддержания количества и состава шлака в печи. При этом шлак должен быть пенистым, достаточно жидкоподвижным и самотеком сходить через порог рабочего окна. Для обеспечения работы печи с максимально возможным заглублением дуг в шлак производятся периодические присадки дробленого кокса порциями до 50 кг через сводовое загрузочное устройство или с использованием манипулятора фирмы FUCHS с расходом порошка кокса 15…65 кг/мин и газообразного кислорода до 3000 $м^3$/ч. Присадка руды вызывает интенсивное кипение ванны – окисляется углерод, реагируя с окислами железа руды с выделением большого количества пузырьков CO. Под воздействием газов шлак вспенивается, уровень его повышается и он стекает в шлаковую чашу через порог рабочего окна. Во время окислительного периода производится отбор проб металла для определения химического состава металла. Для успешного протекания этой реакции необходимы высокие основность шлака и концентрация окислов железа в нем, а также пониженная температура. Раскисление стали производят диффузионным способом после образования жидкоподвижного шлака. Вначале, в течение 15…20 мин, раскисление ведут смесью, состоящей из извести, плавикового шпата и кокса в соотношении 8:2:1, иногда присаживают один кокс. Далее начинают раскисление молотым 45 или 75%-м ферросилицием, который вводят в состав раскислительной смеси, содержащей известь, плавиковый шпат, кокс и ферросилиций в соотношении 4:1:1:1. На некоторых марках стали в конце восста-

новительного периода в состав раскислительной смеси вводят более сильные раскислители – молотый силикокальций и порошкообразный алюминий, а при выплавке ряда низкоуглеродистых сталей диффузионное раскисление ведут без введения кокса в состав раскислительных смесей. Преимущество диффузионного раскисления заключается в том, что поскольку реакции раскисления идут в шлаке, выплавляемая сталь не загрязняется продуктами раскисления – образующимися окислами. Это способствует получению стали с пониженным содержанием неметаллических включений. По мере диффузионного раскисления постепенно уменьшается содержание FeO в шлаке и пробы застывшего шлака светлеют, а затем становятся почти белыми. Для сталей со среднемарочным содержанием углерода до 0,25 % производится присадка 1 кг/т алюминия. В процессе рафинирования производится раскисление шлака в печи порошком кокса, порошком или крупкой ферросилиция, порошком или дробью алюминия в количестве по 100 кг каждого. Температура металла в печи перед выпуском должна быть в пределах, указанных для данной марки стали. В начале выпуска в ковш производится присадка шлаковой смеси и алюминия. Для улучшения десульфурации в состав шлаковой смеси возможно введение до 400 кг глинозема или глиноземсодержащих шлаков. Присадка шлаковой смеси и ферросплавов заканчивается до наполнения трети высоты ковша.

3.5.2. Система весодозирования и подачи сыпучих материалов дуговой электросталеплавильной печи

Подавляющая часть материалов подается в электросталеплавильный цех (ЭСПЦ) конвейерным транспортом. Система весодозирования и подачи сыпучих материалов дуговой электросталеплавильной печи предназначена для транспортирования, распределения и равномерной подачи потребителям различных сыпучих материалов в процессе выплавки стали и дальнейшей обработки ее расплава.

Система подачи и дозирования сыпучих материалов и ферросплавов включает систему подачи и дозирования из приемных бункеров приемного устройства в расходные бункера и систему дозирования и подачи сыпучих материалов и ферросплавов из расходных бункеров в ковши ДЭСП-1, ДЭСП-2, АКП-1 и АКП-2.

В состав системы входит отделение шихты и ферросплавов, которое осуществляет прием, хранение, подготовку материала и его последующую подачу потребителям в ЭСПЦ. Подавляющая часть материалов подается в ЭСПЦ конвейерным транспортом. Отделение соединено с цехом конвейерной галереей общей протяженностью около 500 м. В нем имеется 12 исходных бункеров, подающих материал на горизонтальный конвейер. Материал, двигаясь по этому конвейеру, попадает на наклонный конвейер, на котором расположены еще 11 бункеров. Далее материал попадает на весо-

вой конвейер, который осуществляет учет количества материала. Контроль за общим количеством того или иного материала, прошедшего через весовой дозатор, необходим для создания рецептов смесей и общего контроля расхода сыпучих материалов и ферросплавов. После взвешивания материал попадает на горизонтальный конвейер с телегой, распределяющей материал по тридцати восьми приемным бункерам. Из этих бункеров материал распределяется по потребителям и формируются различные смеси. Бункеры расположены в следующем порядке: 7 бункеров для АКП-2, 12 бункеров для ДЭСП-2, 11 бункеров для ДЭСП-1 и 8 бункеров для АКП-1.

Автоматизация объекта управления (весового дозатора) заключается в обеспечении его работы с помощью устройства управления. Приведенная в примере АСУ функционирует с учетом параметров состояния объекта управления. Оператор осуществляет лишь общий контроль за ходом процесса и при необходимости берет на себя управление работой всей системы или ее части.

Измерение параметров процесса производится с применением датчиков, которые собирают информацию и придают ей нужную форму, зачастую преобразуя физическую природу измеряемых величин. Управляющие команды передаются к дозатору органами воздействия. Передача сопровождается изменением физической природы информации и усилением мощности управляющих команд. На базе введенной программы и результатов изменений устройство управления (программируемый контроллер S7-313C) формирует сигналы воздействия в соответствии с алгоритмом управления объектом.

Система управления весовым дозатором конвейерного типа предназначена для выполнения следующих функций:

- автоматизированного управления в реальном времени технологическими механизмами и исполнительными устройствами, транспортирующими и взвешивающими сырье и готовую продукцию в соответствии с требованиями технологии;
- автоматизированного контроля технологических параметров с предупреждением и отключением оборудования в аварийных ситуациях.

Целью проектирования является замена и модернизация оборудования, повышение качества дозирования и расхода компонентов шихты.

Широкое применение ленточные конвейеры получили потому, что обладают рядом преимуществ перед другими видами подъемно-транспортного оборудования, а именно:

- высокой производительностью, обеспечиваемой большой скоростью движения ленты;
- малыми энергозатратами;
- несложной конструкцией и простотой в эксплуатации;
- высокой надежностью.

Принцип действия дозатора с весовым контролем основан на поддержании заданного значения производительности путём регулирования скорости перемещения ленты с материалом в зависимости от уровня материала на транспортёрной ленте и текущего веса материала.

Специфические условия работы ленточного конвейера на металлургическом предприятии ставят ряд дополнительных требований: надежная работа весов конвейерных при больших скоростях движения конвейерной ленты и неравномерной загрузке, взрывобезопасное и защищенное от влаги и пыли исполнение, малые габариты, возможность установки на передвигающихся конвейерах и др. Весы должны быть устойчивыми (возвращаться в первоначальное положение после малого числа колебаний), надежными в работе и иметь возможно малую погрешность (не более 1 %).

Конвейерные весы по принципу действия разделяют на весы периодического взвешивания (суммирующие) и весы непрерывного взвешивания (интегрирующие). Интегрирующие конвейерные весы (весы непрерывного действия) имеют датчики веса и скорости.

Дозируемый материал поступает через приёмную воронку на транспортёрную ленту. Производительность дозатора зависит от высоты слоя дозируемого материала на транспортёрной ленте, скорости движения ленты и насыпной плотности материала.

Высота слоя материала, выходящего из выпускного отверстия приемной воронки, регулируется шиберной заслонкой в диапазоне 0,1…0,4 м, скорость движения ленты – изменением при помощи частотного преобразователя числа оборотов двигателя ленточного транспортёра в зависимости от толщины слоя материала на ленте. Диапазон изменения скоростей транспортера 0,05…3 м/с.

Нагрузка на измерительное устройство от массы взвешиваемого дозируемого материала преобразуется тензорезисторными датчиками в сигнал постоянного тока. Измерительный лоток подвешивается к двум датчикам за раму, а третий монтируется под лотком.

3.5.3. Разработка функциональной схемы весового дозатора

Весовой конвейер предназначен для автоматического учета количества материала, транспортируемого конвейером, и регулирования скорости его подачи.

Дозатор – это система, в состав которой входят:

- ленточный транспортер;
- мотор-барабан;
- приемная воронка с шиберной заслонкой;
- шкаф управления, предназначенный для размещения блока обработки информации и панели управления;

- шкаф с частотным преобразователем;
- весоизмерительный лоток;
- контрольно-измерительные приборы (КИП).

Принцип действия дозатора основан на непрерывном взвешивании погонного веса груза в зависимости от скорости движения конвейерной ленты, перемножении показателей и интегрировании по времени.

Ленточный транспортер представляет собой конвейерную ленту и необходим для складирования и перемещения сыпучего материала под действием силы трения между лентой и приводным барабаном.

В качестве привода дозатора используется мотор-барабан. Мотор-барабан разработан специально как привод для ленточных конвейеров. Мотор, редуктор и подшипники встроены в корпус барабана и герметично уплотнены. Тем самым они надежно защищены от внешних воздействий, таких как вода, пыль, грязь, химикаты, жиры, масла и т.п. Так как привод и подшипники смонтированы внутри барабана, требуется меньше места, чем для традиционного привода, нет необходимости в дополнительных компонентах. Мотор-барабан является одним из самых безопасных приводов, т.к. мотор и редуктор полностью закрыты и наружные цапфы осей неподвижны. Вращается только корпус барабана. Усилия от двигателя через редуктор передаются непосредственно на обечайку барабана. Благодаря меньшим потерям на трение достигается КПД до 97 %.

Приемная воронка осуществляет накопление материала, подаваемого с наклонного конвейера, сглаживает удар, который происходит при падении материала на ленту весового дозатора. В приемной воронке монтируется дискретный датчик уровня заполнения течки, который срабатывает при достижении верхнего уровня воронки. Шиберная заслонка регулирует уровень материала, подаваемого на конвейер, чтобы предотвратить пересыпание материала через бортовые заслонки конвейера.

Под конвейером располагается очиститель ленты, который представляет собой резиновые и капроновые щетки, плотно прилегающие к поверхности ленты.

Шкаф управления монтируется в отапливаемом электропомещении на некотором удалении от объекта управления, дабы исключить влияние вредных факторов (шум, пылевое загрязнение, вибрация, перепады температур и влажности окружающей среды) на систему управления и на оператора. К шкафу управления подводятся питание 220 В переменного напряжения, кабели управления. Внутри располагаются два блока питания, которые осуществляют преобразование 220 В переменного тока в 24 В постоянного тока. Один из них необходим для питания панели управления, второй – для питания КИП. Также в шкафу монтируются клеммные коробки для подсоединения кабелей и защитные автоматы, которые размыкают электрическую цепь в случае превышения допустимого значения по току.

В нижней части шкафа управления монтируется преобразователь OLM (Optical Link Module) электрического сигнала, передаваемого от панели управления по полевому кабелю Profibus DP, в оптическую энергию, передаваемую по оптико-волоконному кабелю, для передачи данных на главный контроллер, расположенный на значительном удалении. В качестве среды передачи используются световоды со стеклянными или пластиковыми волокнами. В данной системе использованы стеклянные световоды, т.к. они обеспечивают хорошее качество передачи данных на большие расстояния. Световодная техника устойчива к электромагнитным помехам и устанавливает безопасную разность потенциалов между участниками. Благодаря простой технике подключения световодов, специальным пластиковым световодам эта техника пришла на полевой уровень. Модуль OLM имеет два функционально разделенных электрических канала и выходы для двух оптических каналов. Модули OLM соединяются с отдельными участниками или сегментами шины через интерфейс RS-485. Профиль DP протокола Profibus применяет физический (первый) уровень и уровень передачи данных (второй) модели ISO/OSI, а также пользовательский интерфейс. Уровни с третьего по седьмой не используются (рис. 3.20). Благодаря такой архитектуре достигается быстрая передача данных. Этот профиль протокола Profibus оптимизирован для быстрого обмена данными специально для коммуникаций между системами автоматизации и децентрализованной периферией на полевом уровне.

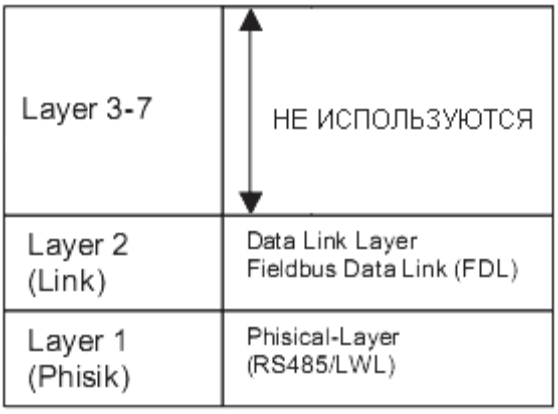


Рис. 3.20. Используемые уровни передачи данных в профиле DP интерфейса RS-485

На двери шкафа монтируется панель С7-613 со встроенным контроллером. В этом же электропомещении устанавливается шкаф с частотным преобразователем. К шкафу подводится питание 380 В переменного тока и через защитные автоматы подключается к преобразователю. Преобразова-

тель на выходе выдает сигнал пропорционально аналоговому сигналу, приходящему от управляющего контроллера.

На рис. 3.21 изображена общая схема подключения питания к частотному преобразователю и к приводу конвейера.

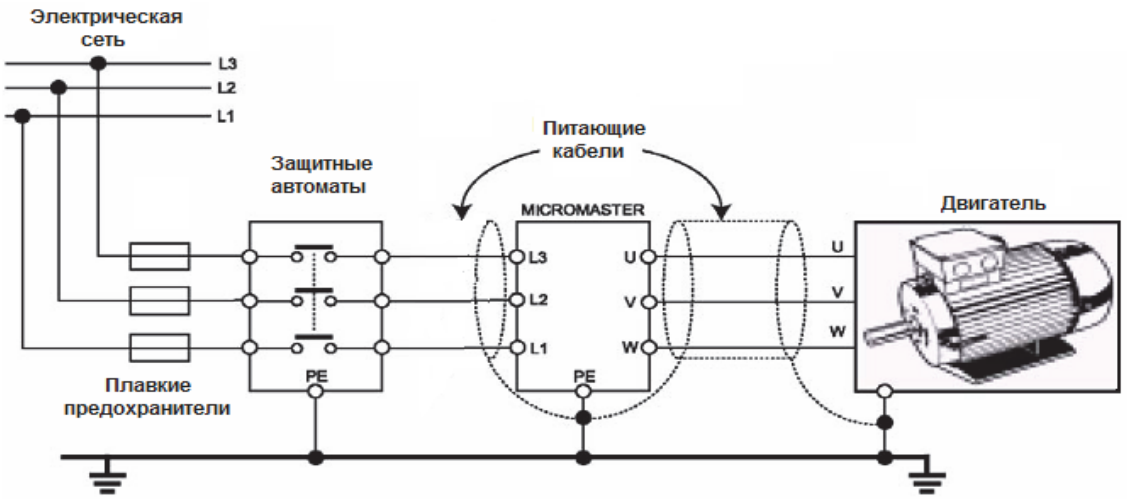


Рис. 3.21. Схема подключения питания к преобразователю Micromaster 440 и к электродвигателю

Весоизмерительный лоток представляет собой наклонную плоскость с системой силопередающих рычагов, установленную на неподвижном основании с помощью шарниров или призм. Сыпучий материал, двигаясь по поверхности лотка, воздействует на тензометрические датчики силы, входящие в состав силопередающих рычагов. Рабочая поверхность лотка покрыта сменной защитной футеровкой, представляющей собой стальной лист, прикрепленный к лотку болтами.

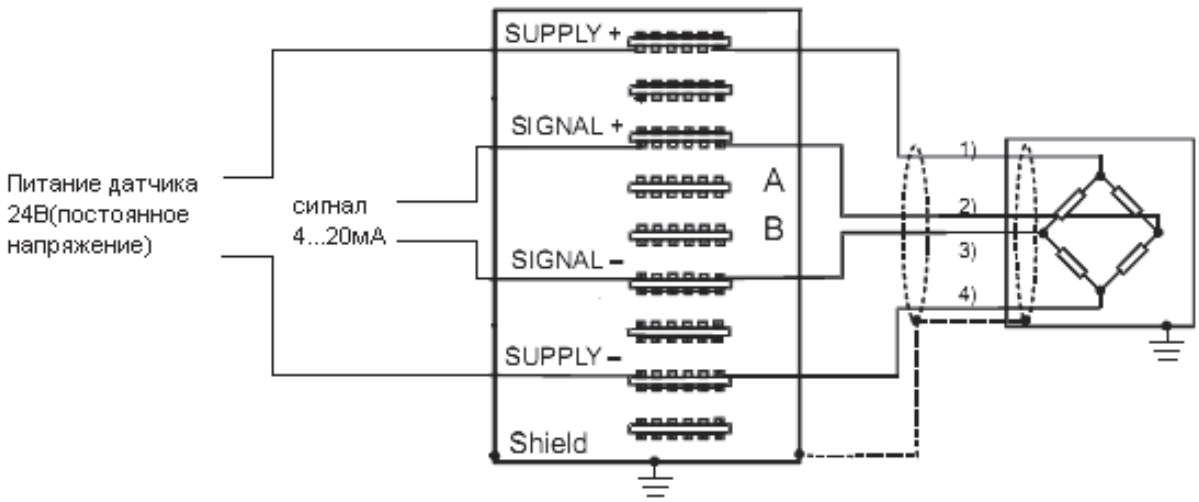


Рис. 3.22. Схема подключения датчика

К КИП относятся: дискретный датчик уровня забивки течки, датчик уровня материала на ленте, датчик схода ленты влево и датчик схода ленты вправо, три тензометрических датчика веса и датчик угла поворота.

В качестве весоизмерительных датчиков используются тензометрические датчики. Принцип работы тензометрических датчиков основан на способности специального резистора менять свое активное электрическое сопротивление в зависимости от деформации (растяжения или сжатия). Тензодатчик представляет собой упругий элемент, на поверхности которого наклонен проволочный преобразователь. В качестве преобразователя обычно применяют проволоку из константана диаметром 0,2…0,3 мм. В весоизмерительных устройствах используются тензодатчики, работающие на растяжение, сжатие и изгиб. Геометрические размеры, конструктивные и технологические особенности тензодатчиков определяются характером весовых нагрузок и задачами измерения. Конструкция тензодатчика должна надежно защищать проволочный элемент от механических повреждений и попадания влаги. Проволочные преобразователи включаются по схеме равноплечного электрического моста. При этом они наклеиваются таким образом, чтобы одна пара проволочных преобразователей испытала деформацию вдоль оси силоизмерительного упругого элемента. На рис. 3.22 изображена схема подключения датчика.

3.5.4. Разработка информационной структуры весового дозатора

Система КИП состоит из дискретных и аналоговых сигналов. Все сигналы КИП заводятся на контроллер без использования устройств распределенной периферии.

Контроллер данной локальной АСУТП выполняет следующие функции управления и контроля:

- управление скоростью вращения ведущего вала конвейера выдачей аналогового сигнала задания скорости;
- расчет и выдачу на верхний уровень АСУТП значения мгновенного и накопленного веса материала, прошедшего через данный конвейер;
- контроль и диагностику состояния системы (скорость перемещения и толщина слоя материала на ленте, контроль схода и обрыва ленты);
- управление режимами работы объекта;
- выдачу текущих, предупредительных и аварийных сообщений;
- самодиагностику;
- формирование аварийной и предупредительной сигнализации при выходе аналоговых сигналов за установленные пороги.

Контролируемый параметр в виде сигнала измерительной информации поступает на вход блока коммутации через модуль нормализации и усилитель, при этом сигнал приводится к стандартному (в случае необхо-

димости производится преобразование токового сигнала в сигнал напряжения или, наоборот, усиление уровня сигнала до требуемого уровня).

К контролируемым параметрам относятся:

- частота вращения барабана, которая для удобства преобразуется контроллером в линейную скорость перемещения ленты. Диапазон изменения скорости составляет 0,1…3 м/с;
- уровень материала на ленте конвейера – регулируется шиберной заслонкой. Также осуществляется ПИД-регулирование частоты вращения приводного барабана в зависимости от уровня материала на ленте. В результате регулирования поддерживается постоянная производительность, постоянный уровень материала на ленте, предотвращается засыпка конвейера. Уставка поддерживаемого уровня при регулировании зависит от насыпной плотности поступаемого материала;
- мгновенное значение веса сыпучего материала на весовом лотке, которое формируется исходя из показаний трех весоизмерительных датчиков. На рис. 3.23 приведен общий вид весоизмерительного лотка. В системе использовано три тензометрических датчика силы с оригинальной схемой крепления к весовому лотку [2]. Данная схема крепления позволяет наиболее точно измерить вес сыпучего материала, находящегося на лотке.

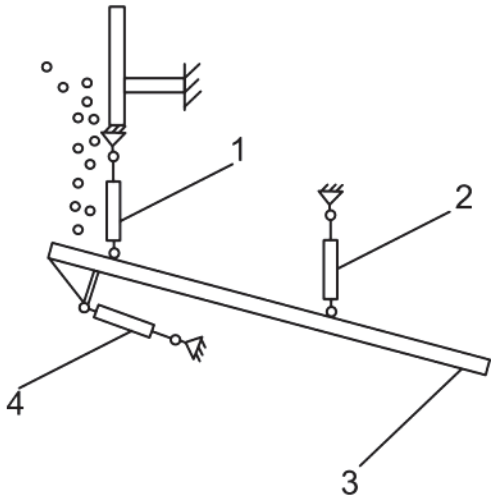


Рис. 3.23. Общий вид весоизмерительного лотка:
1, 2, 4 – тензометрические датчики силы; 3 – весоизмерительный лоток

Непосредственное управление дозатором производит программируемое устройство SIMATIC C7-613. Оно производит интерполяцию, выдачу управляющих воздействий на частотный преобразователь, собирает информацию с датчиков состояния электроавтоматики, управляет электроавтоматикой дозатора, а также выполняет функцию HMI (Human Machine Interface).

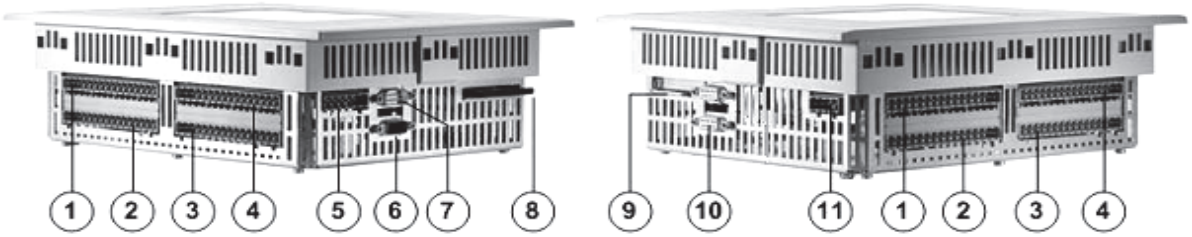
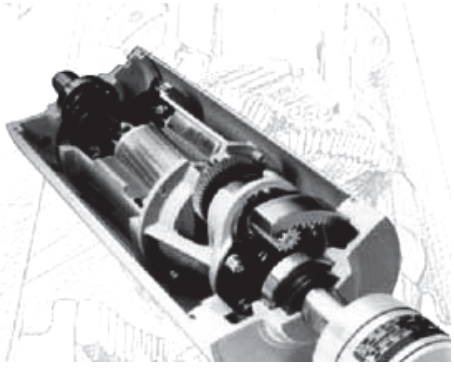


Рис. 3.24. Функциональная схема операторной панели С7-613:
1 –дискретные входы; 2 – дискретные выходы; 3 – аналоговые входы;
4 – питание дискретных входов-выходов и дискретные входы;
5 – аналоговые выходы; 6 – шина расширения системы локального ввода-вывода; 7 – интерфейс RS 222 (порт принтера); 8 – CF карта панели оператора; 9 – микрокарта памяти контроллера; 10 – интерфейс MPI;
11 – питание электроники контроллера и панели оператора

Устройства ввода информации служат для ввода диапазона измерения контролируемых параметров, величин аварийной и предупредительной сигнализации и т.д. На рис. 3.24 изображена функциональная схема панели С7-613.

3.5.5. Выбор датчиков и исполнительных устройств

Все средства измерений, участвующие в технологическим процессе и служащие для контроля качества готовой продукции, должны своевременно проверяться в метрологической службе завода (или в органах Государственной метрологической службы) с оформлением результатов поверки в паспорте или клеймом поверки.



В качестве привода конвейера используется мотор-барабан фирмы RULMECA. Он имеет бочкообразное исполнение. Привод встроен в свободное внутреннее пространство барабана и тем самым полностью закрыт. Мотор-барабан RULMECA часто оказывается экономичнее, чем традиционное решение привода, поскольку приходится монтировать меньше деталей. Мотор-барабан RULMECA практически не требует ухода, кроме смены масла через 100 000 машино-часов и замены уплотнений валов через 30 000 машино-часов. Иными словами, смена масла потребуется раз в пять лет при восьми рабочих часах в день и пяти рабочих днях в неделю.

Мотор-барабан RULMECA имеет значительно лучший КПД в отличие от традиционного привода. Встроенный редуктор действует непосред-

ственно на барабан. Благодаря этому достигается КПД до 97 % и обеспечивается абсолютная центровка ленты. Мотор-барабан RULMECA может эксплуатироваться с преобразователем частоты в диапазоне 15…65 Гц.

Мотор-барабан RULMECA является единым интегрированным приводным блоком. Мощность мотор-барабана составляет 5,5 кВт.

Управление приводом транспортера осуществляется с помощью преобразователя частоты MICROMASTER 440 производства фирмы Siemens, который обеспечивает высокую точность управления приводом с помощью встроенной микропроцессорной системы управления. MICROMASTER 440 был специально разработан для решения сложных функциональных задач с высокими требованиями к динамике. Система векторного управления обеспечивает высокое качество работы привода даже при резких изменениях нагрузки. С помощью быстрых входов и функции точного останова возможно точное позиционирование без использования энкодера. Благодаря интегрированному тормозному резистору привод работает с высокой точностью даже во время торможения и в режимах резкого замедления скорости. В системе реализована защита преобразователя и двигателя от перегрузок.

Для измерения веса используются активные тензометрические S-образные датчики сжатия-растяжения STCS 200 C3 фирмы Esit. Датчики такого типа применяются для измерения веса емкостей или иных конструкций, находящихся в подвешенном состоянии, а также для измерения усилий растяжения и сжатия. Используются в бункерных весах и дозаторах. Обладают высокой чувствительностью и компенсацией на широкий диапазон температур. В табл. 3.6 приведены технические характеристики этого датчика.

Таблица 3.6

Характеристики датчика C2H фирмы Тензо-М

Характеристика	Значение
Номинальная нагрузка, кг	200
Минимальная нагрузка, кг	0,02
Максимальная перегрузка, кг	300
Общая ошибка, %	< ±0,02
Коэффициент передачи, мВ/В	2 ± 0,010
Максимальное напряжение питания постоянного тока, В	15 В
Сопротивление входное, Ом	385 ± 20
Сопротивление выходное, Ом	350 ± 3
Сопротивление изоляции, Ом	≥ 500
Рабочий диапазон температур, °С	-40...+80
Материал датчика	Нержавеющая сталь

Датчик соответствует стандарту ГОСТ 30129-96 «Датчики весоизмерительные тензорезисторные. Общие технические требования».

В качестве функциональных модулей для измерения веса используются модули Siwarex U (рис. 3.25). В любой ситуации, когда необходимо точное и надежное измерение нагрузок и сил, модуль Siwarex U является наилучшим возможным решением. Модуль берет на себя все функции взвешивания в ходе технологического процесса. Siwarex U генерирует определенный сигнал, соответствующий измеренному весу, и сравнивает его с предельным значением (измеренное значение не должно превышать предельное). Модуль имеет высокую точность измерений (погрешность 0,05 %). Имеется два встроенных последовательных интерфейса: RS-232C (для подключения к компьютеру) и TTY (для подключения дистанционного дисплея).

Рис. 3.25. Модуль Siwarex U

Каждый модуль Siwarex U запитывается напряжением 5 В постоянного тока по задней шине и дополнительно 24 В к клеммам 1L+ и 2M, которое необходимо для питания активных датчиков.

Встроенный АЦП выдает результат измерения в виде 16-битного слова. Это соответствует разрешению в 65 535 долей. АЦП работает в однополярном режиме, однако при этом существует возможность отслеживания низких отрицательных значений напряжения. Предварительное значение измерения (значение преобразователя) определяется каждые 20 мс и представляет собой однополярный сигнал (т.е. без знака).

На рис. 3.26 изображен общий принцип корректировки весоизмерительных датчиков.

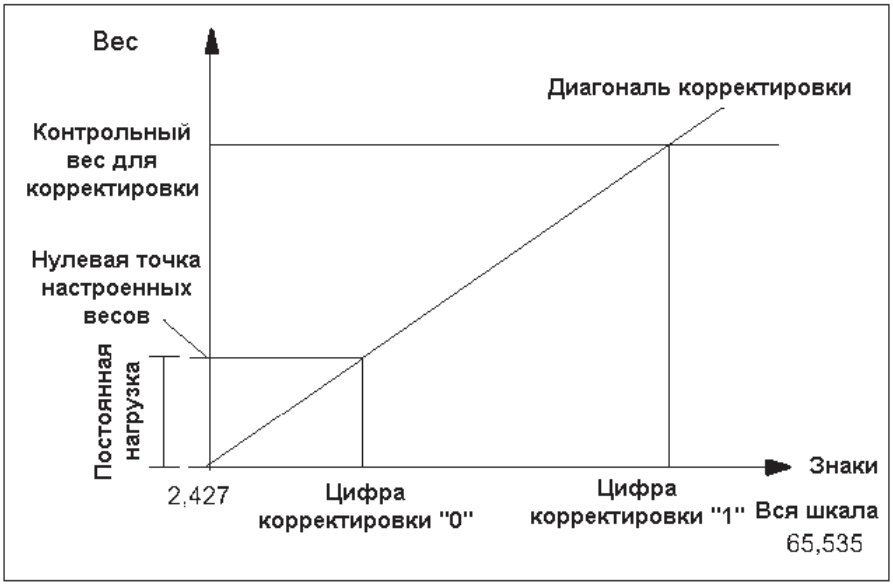


Рис. 3.26. Принцип корректировки датчика веса

Каждому значению веса соответствует код АЦП. Для настройки весов сначала задаем нулевую точку настроенных весов, которая соответствует постоянной нагрузке на датчик. В данной системе в качестве такой нагрузки выступает весовой лоток. Затем весы нагружают контрольным весом для корректировки веса. В данном случае это невозможно ввиду особенностей подключения датчиков, поэтому датчики корректируются отдельно в метрологической службе завода.

Для параметрирования модулей Siwarex U используется утилита SIWATOOL U, поставляемая с ними в комплекте.

В главном окне графического интерфейса (рис. 3.27) необходимо включить нужные каналы, а также задать верхние и нижние пределы измерения.

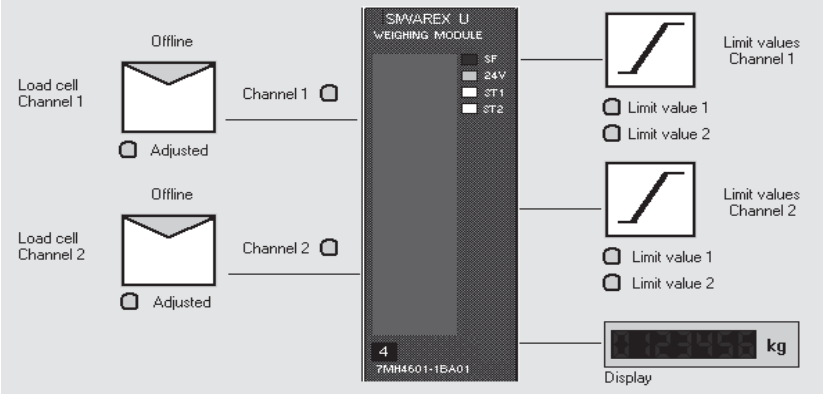


Рис. 3.27. Главное окно интерфейса утилиты SIWATOOL U

В настройках канала (рис. 3.28) включить фильтр, который нейтрализует помехи, вносимые постоянной вибрацией весовой площадки. Для этого нужно задать частоту вибрации в графе **Frequency Threshold**. Далее нужно указать чувствительность весоизмерительного датчика в графе **Characteristic value**. Рекомендуется коэффициент передачи 2 мВ/В, поэтому выбираем из ниспадающего меню это значение. В графе **Adjustment Weight** указывается средняя нагрузка. В строке **Zero setting digits** отображается постоянная нагрузка на датчик веса. В качестве постоянной нагрузки выступает металлический лоток. Его точный вес вычисляется автоматически нажатием кнопки **Set to Zero**.

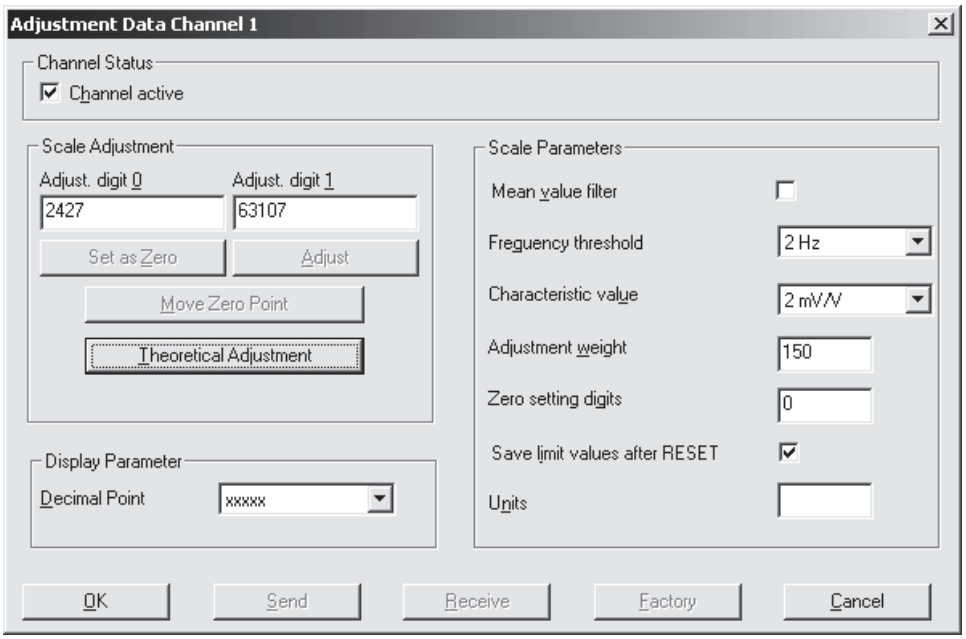


Рис. 3.28. Окно настроек канала

В качестве устройства управления дозатором используется программируемое устройство SIMATIC C7-613, которое включает в свой состав: центральный процессор CPU 313C; текстовую панель оператора с жидкокристаллическим дисплеем, позволяющим отображать текстовые строки по двадцать символов, и внутренней светодиодной подсветкой; интерфейс расширения системы локального ввода-вывода; интерфейс MPI; 24 дискретных входа; 16 дискретных выходов; 4 аналоговых входа; 2 аналоговых выхода.

Возможности C7-613:

- выбор режимов работы центрального процессора со встроенной клавиатуры (с защитой от несанкционированного доступа);
- необслуживаемое сохранение данных в микрокарте памяти при перебоях в питании контролера;

- подключение внешних цепей через соединители с контактами-защелками или с контактами под винт;
- возможность расширения системы локального ввода-вывода;
- встроенная панель оператора позволяет выполнять просмотр оперативных и сохраненных ранее сообщений, ввод-вывод значений параметров (на дисплее могут отображаться состояния флагов, входов и выходов, таймеров и счетчиков и модифицироваться с клавиатуры), определение назначения клавиш, устанавливать парольную защиту доступа к операциям мониторинга и оперативного управления, выбирать языки вывода системных, оперативных и аварийных сообщений, а также текстов помощи, выполнять регулировку контрастности изображения.

В табл. 3.7 представлены характеристики устройства С7-613.

Таблица 3.7

Характеристики программируемого устройства С7-613

Характеристика	Значение
Номинальное напряжение питания постоянного тока	24 В
Потребляемый ток при холостой работе	270 мА
Потребляемая мощность	11,3 Вт
Рабочий диапазон температур	0…50 °С
Относительная влажность окружающей среды во время работы	-5…95 %
Масса	0,915 кг

В табл. 3.8 представлены характеристики встроенного контроллера S7-313C.

Таблица 3.8

Характеристики контроллера S7-313C

Характеристика	Значение
Встроенная рабочая память для программ и данных (RAM)	80 КБайт
Объем загружаемой памяти (микрокарта памяти Flash-EEPROM)	До 8 МБайт
Минимальное время выполнения логических операций	0,1 мкс
Максимальное количество каналов ввода/вывода дискретных сигналов в системе локального ввода/вывода	168 шт.
Максимальное количество каналов ввода/вывода аналоговых сигналов в системе локального ввода/вывода	37 шт.
Максимальное количество функциональных модулей	4 шт.
Продолжительность хода встроенных системных часов при отключенном питании контроллера	6 недель

3.5.6. Разработка алгоритма управления весовым дозатором

Для работы с дозатором необходимо подать напряжение на все элементы системы управления. При этом должна гореть зеленая лампа на передней панели шкафа, что означает наличие напряжения на вводных клеммах. После запуска панели оператора, появится стартовый экран.

Весовой конвейер работает в четырех режимах: «Дистанционный автоматический регулируемый режим» («ДАВТРЕГ»), «Дистанционный ручной регулируемый режим» («ДРУЧРЕГ»), «Дистанционный ручной нерегулируемый режим» (ДРУЧНЕРЕГ) и «Местный» («МЕСТ»).

В режиме «ДАВТРЕГ» работа конвейера происходит с использованием всех параметров, алгоритмов измерений, регулирования, работы системы аварий и т.д., полученных с контроллера, который управляет работой всей системой весодозирования сыпучих материалов. Параметры вводятся в окне визуализации тракта подачи сыпучих материалов.

В режиме «ДРУЧРЕГ» работа весового конвейера происходит с использованием всех параметров, алгоритмов измерений, регулирования, работы системы аварий и т.д., предварительно введенных с панели оператора С7-613.

Режим «ДРУЧНЕРЕГ» по-другому можно назвать режимом ручного регулирования скорости дозатора. В этом режиме скорость движения конвейера задается вручную с панели оператора. Также возможно отключение системы аварийных сообщений.

Режим «МЕСТ» – это режим, предназначенный для наладки механической части дозатора. В этом режиме прокрутка идет на скорости, задаваемой с панели оператора. Система аварийных сообщений не работает. Запуск и останов дозатора происходит с пульта местного управления (ПМУ), с панели оператора управление заблокировано.

Режим работы определяется положением ключа «Авт»/«Мест» на местном пульте дозатора. Только в положении ключа «Авт» возможна работа в дистанционном автоматическом или ручном регулируемом режиме, а также дистанционном ручном нерегулируемом режиме.

В случай нажатия кнопки «Аварийный стоп» на ПМУ или на шкафу управления возле панели оператора происходит обесточивание привода барабана в обход контроллера и выдается сообщение на панели оператора об аварийном останове.

Для повторного запуска конвейера необходимо квитировать сообщение, т.е. подтвердить его, и лишь после этого запускать конвейер. Если отображается сообщение об аварии, которая не устранена, то конвейер не запустится до тех пор, пока не будет устранена авария.

Включение питания привода барабана происходит нажатием кнопки «ПУСК» на ПМУ в режиме управления «МЕСТ» либо нажатием кнопки К1 на панели оператора в остальных режимах управления.

Включение привода конвейера происходит при наличии готовности системы (отсутствие аварий и неквитированных сообщений на панели оператора). Подача и отключение питания на мотор-барабан происходит при поступлении сигнала от контроллера к частотному преобразователю.

Останов конвейера осуществляется нажатие кнопки «СТОП» на ПМУ в режиме управления «МЕСТ», нажатием кнопки К6 на панели оператора в остальных режимах управления либо сигналом от контроллера верхнего уровня, который управляет работой всего тракта подачи сыпучих материалов.

После запуска конвейера включается программный ПИД-регулятор, который поддерживает уровень, установленный для данного материала. Данные о материале приходят от контроллера управления всем трактом подачи сыпучих материалов. Параметры регулятора вводятся с панели оператора либо в окне визуализации общего тракта подачи. Минимальный уровень соответствует пустой ленте, максимальный уровень материала на ленте составляет 450 мм. Параметры ПИД-регулятора конвейера были рассчитаны и опробованы опытным путем. Коэффициент пропорциональности $K = 30$, время интегрирования $T_{\text{int}} = 2\,\text{с}$, время дифференцирования $T_{dif} = 9\,\text{с}$. Один оборот барабана соответствует сотне импульсов от датчика скорости.

3.5.7. Разработка программного обеспечения для управления весовым дозатором

В CPU всегда исполняются две программы:

- операционная система;
- программа пользователя.

Каждый CPU содержит ОС, которая организует все функции и последовательности в CPU, не связанные с конкретной задачей управления. Задачи ОС:

- обработка перезапуска;
- обновление таблицы образа процесса для входов и вывод таблицы образа процесса для выходов;
- вызов программы пользователя;
- обнаружение прерываний и вызов подпрограмм прерываний;
- обнаружение и обработка ошибок;
- управление областями памяти;
- обмен информацией с устройствами программирования и другими коммуникационными партнерами.

Программа пользователя содержит все функции, необходимые для обработки конкретной задачи автоматизации. Задачи программы пользователя:

- определение условий для перезапуска CPU (например, инициализация сигналов с определенным значением);
- обработка данных процесса (например, логическая комбинация двоичных сигналов, считывание и анализ аналоговых сигналов, задание двоичных сигналов для вывода, вывод аналоговых значений);
- определение реакции на прерывания;
- обработка нарушений в нормальном исполнении программы.

Программное обеспечение STEP 7 дает возможность структурировать пользовательскую программу, иными словами, разбивать программу на отдельные автономные программные секции. Это дает следующие преимущества:

- отдельные программные секции могут быть стандартизованы;
- упрощается организация программы;
- легче производить модификацию программы;
- отладка упрощается, т.к. можно тестировать отдельные секции.

Имеется несколько различных типов блоков, которые можно использовать внутри пользовательской программы STEP7 (табл. 3.9).

Таблица 3.9

Описание типов блоков программы в STEP7

Блок	Краткое описание функции
Организационные блоки (OB)	Определяют структуру программы пользователя
Системные функциональные блоки (SFB) и системные функции (SFC)	Встроены в CPU S7 и обеспечивают доступ ко всем важным системным функциям
Функциональные блоки (FB)	Это блоки с «памятью», которые можно программировать самостоятельно
Функции (FC)	Содержат программы для часто встречающихся функций
Экземплярные блоки данных (экземплярные DB)	Связываются с блоком, когда вызывается FB/SFB. Они создаются автоматически при компиляции
Блоки данных (DB)	Это области данных для хранения данных пользователя. Кроме данных, соответствующих функциональному блоку, могут быть определены также данные, совместно используемые любыми блоками

Программа пользователя может быть разделена на «подпрограммы» и распределена между различными организационными блоками. Если программа пользователя должна реагировать на важный сигнал, который появляется относительно редко (например, датчик граничного значения для измерения уровня в резервуаре сообщает, что достигнут максимальный уровень), то подпрограмма, которая должна обрабатываться, когда выдается этот сигнал, может быть помещена в ОВ, обработка которого управляется событиями (рис. 3.29)

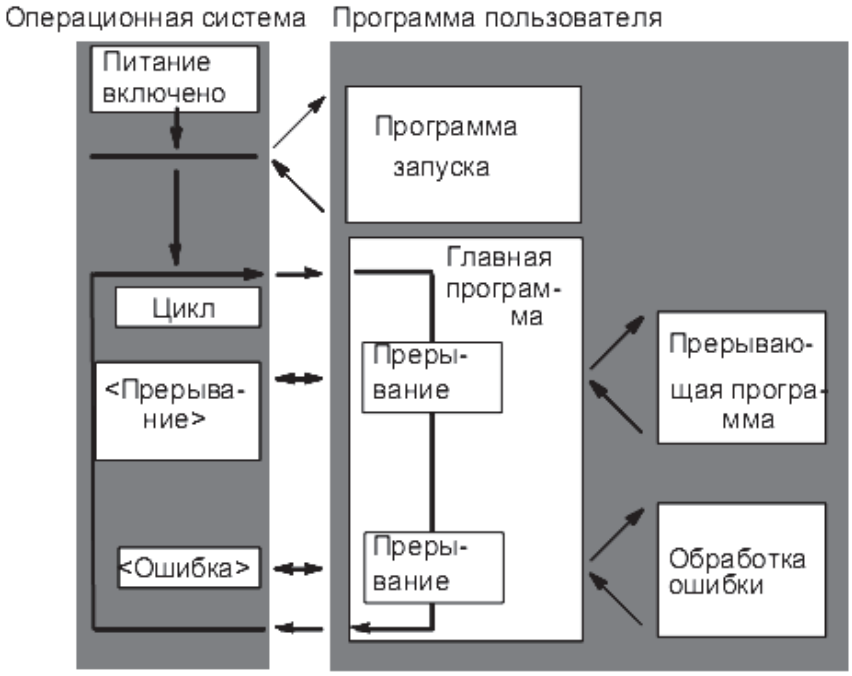


Рис. 3.29. Обработка прерываний в программе STEP7

Для функционирования программы пользователя составляющие ее блоки должны вызываться. Это делается с помощью специальных команд STEP7, вызовов блоков, которые могут быть запрограммированы и запущены только в логических блоках. Порядок и вложение вызовов блоков называется иерархией вызовов.

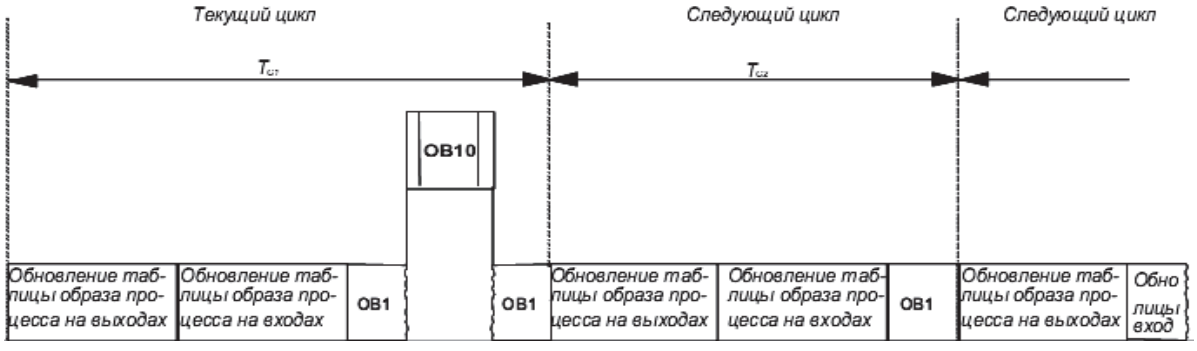


Рис. 3.30. Время выполнения цикла

Время выполнения цикла – это время, необходимое операционной системе для выполнения циклической программы и всех программных секций, прерывающих цикл (например, выполнение других организационных блоков), и системных операций (например, обновления образа процесса). Это время контролируется. Время выполнения цикла (ТС) не одинаково в каждом цикле (рис. 3.30).

В свойствах процессора можно изменить максимальное время цикла, установленное по умолчанию. По умолчанию максимальное время цикла ОВ 1 равно 150 мс (рис. 3.31).

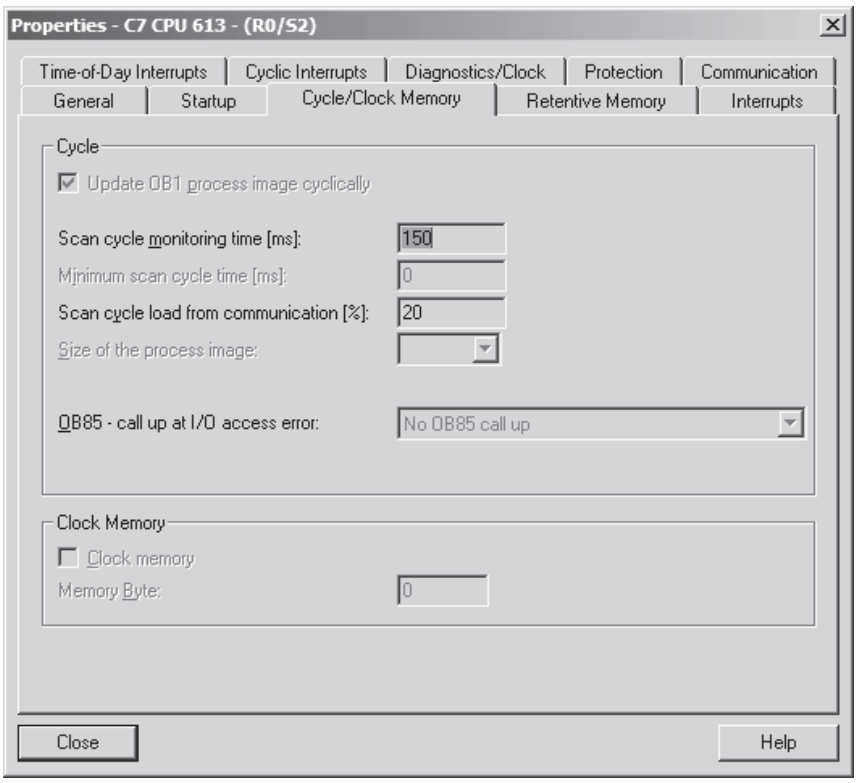


Рис. 3.31. Максимальное время цикла ОВ 1

В этом же окне на вкладке **Cyclic Interrups** можно задать периодичность вызова блока ОВ 35 (рис. 3.32).

Функции (FC) относятся к блокам, которые Вы программируете сами. Функция – это логический блок «с памятью». Временные переменные, принадлежащие FC, хранятся в стеках локальных данных. После того как FC выполнена, эти данные теряются. Чтобы хранить эти данные постоянно, функции могут также использовать разделяемые (глобальные) блоки данных.

Так как FC не имеет собственной памяти, то Вы всегда должны определять для нее фактические параметры. Для локальных данных FC нельзя назначать начальные значения. FC содержит программную секцию,

которая выполняется всегда, когда FC вызывается другим логическим блоком. Функции можно использовать для следующих целей:

- для возврата значения функции в вызывающий блок (например, математические функции);
- выполнения технологической функции (например, отдельная функция управления с битовой логической операцией).

В ***функциональных блоках*** (FB) при назначении параметров используется копия реальных параметров в экземпляре DB. Если параметр ввода не передается или параметр выхода не записывается при вызове FB, то сохраняется старая величина в экземпляре DB (экземпляр DB = память FB), которая и будет использоваться.

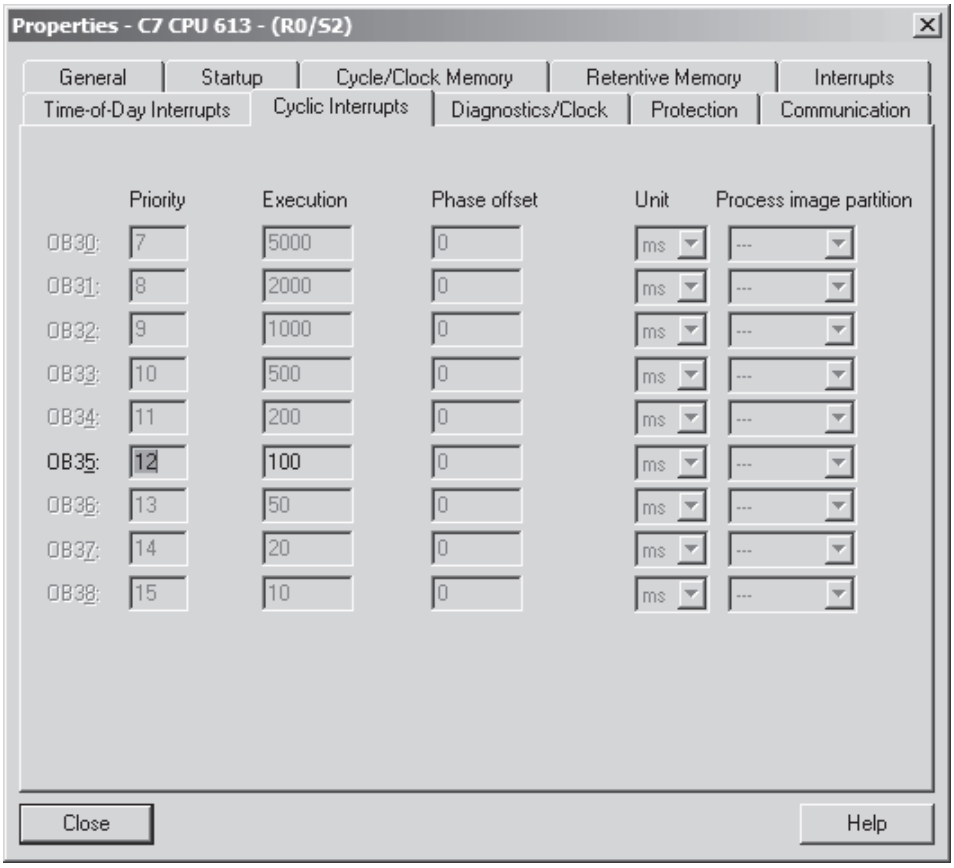


Рис. 3.32. Периодичность вызова блоков цикличных прерываний

Функциональный блок – это логический блок «с памятью». В качестве памяти ему назначается блок данных (экземплярный блок данных). В экземплярном DB сохраняются параметры, передаваемые FB, и статические переменные. Данные, сохраняемые в экземплярном DB, не теряются, когда исполнение FB завершено. Экземплярный блок данных назначается каждому вызову функционального блока, который передает параметры. Можно с помощью одного FB управлять более чем одним устройством

Экземплярный блок данных назначается каждому вызову функционального блока, который передает параметры. Фактические параметры и статические данные FB хранятся в экземплярном DB. Переменные, описанные в FB, определяют структуру экземплярного блока данных. Перед созданием экземплярного блока данных соответствующий FB уже должен существовать. При создании экземплярного блока данных указывается номер FB.

Глобальные блоки данных (DB)

В отличие от логических блоков, блоки данных не содержат команд STEP 7. Они используются для хранения данных пользователя, иными словами, блоки данных содержат переменные данные, с которыми работает программа пользователя. Глобальные блоки данных применяются для хранения пользовательских данных, к которым могут обратиться все остальные блоки.

Каждый FB, FC или OB может читать данные из глобального DB или записывать данные в этот DB. Эти данные сохраняются в DB после выхода из него.

CPU S7 предоставляют заранее запрограммированные блоки, которые можно вызывать в своей пользовательской программе. ***Системный функциональный блок*** (SFB) – это функциональный блок, встроенный в CPU S7. SFB являются частью ОС и не загружаются как часть программы пользователя. Как и FB, SFB – это блоки «с памятью». Для SFB тоже нужно создавать экземплярные блоки данных и загружать их в CPU как часть программы.

Системная функция – это заранее запрограммированная, оттестированная функция, встроенная в CPU S7. SFC являются частью ОС и не загружаются как часть программы. Как и FC, SFC являются блоками «без памяти». CPU S7 предоставляют SFC для следующих функций:

- копирование и блочные функции;
- контроль программы;
- работа с часами и счетчиками рабочего времени;
- передача наборов данных;
- передача событий из CPU всем остальным CPU в мультипроцессорном режиме;
- обработка прерываний по времени и с задержкой;
- обработка синхронных и асинхронных ошибок;
- информация о статических и динамических системных данных;
- обновление образа процесса и обработка битовых массивов;
- адресация модулей;
- децентрализованная периферия;
- связь с помощью глобальных данных;
- связь через неспроектированные соединения;
- генерирование сообщений, относящихся к блокам.

Конфигурация проекта

В начале создания проекта требуется сконфигурировать сеть. Для этого нужно открыть окно NetPro (рис. 3.33) и добавить в нем из предложенного списка сети, к которым подключен контроллер (MPI и ProfibusDP) и стойку для контроллера трехсотой серии.

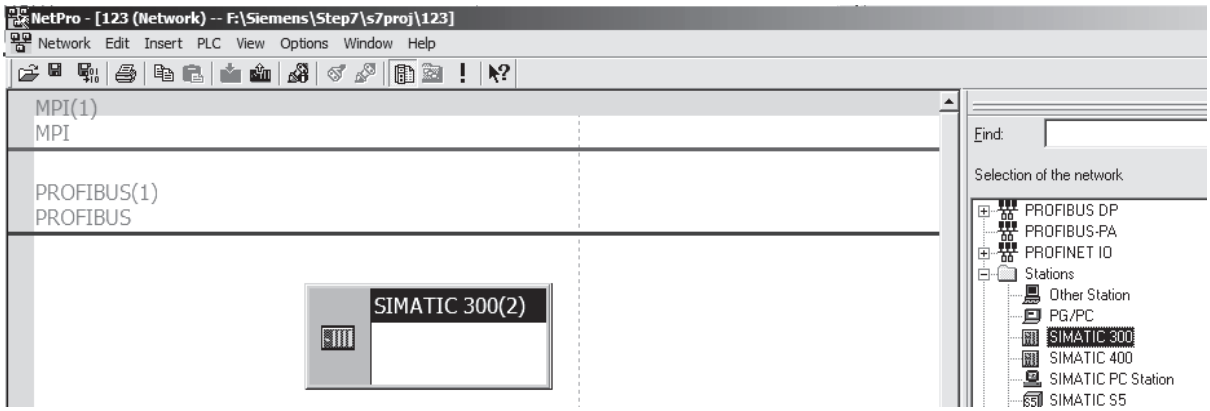


Рис. 3.33. Окно NetPro для настройки соединений

Затем в окне HW Config (рис. 3.34), которое открывается при двойном нажатии на стойку, следует добавить контролер C7-613, коммуникационный модуль CP 342-5 для связи с устройствами сети Profibus DP и два функциональных модуля Siwarex U.

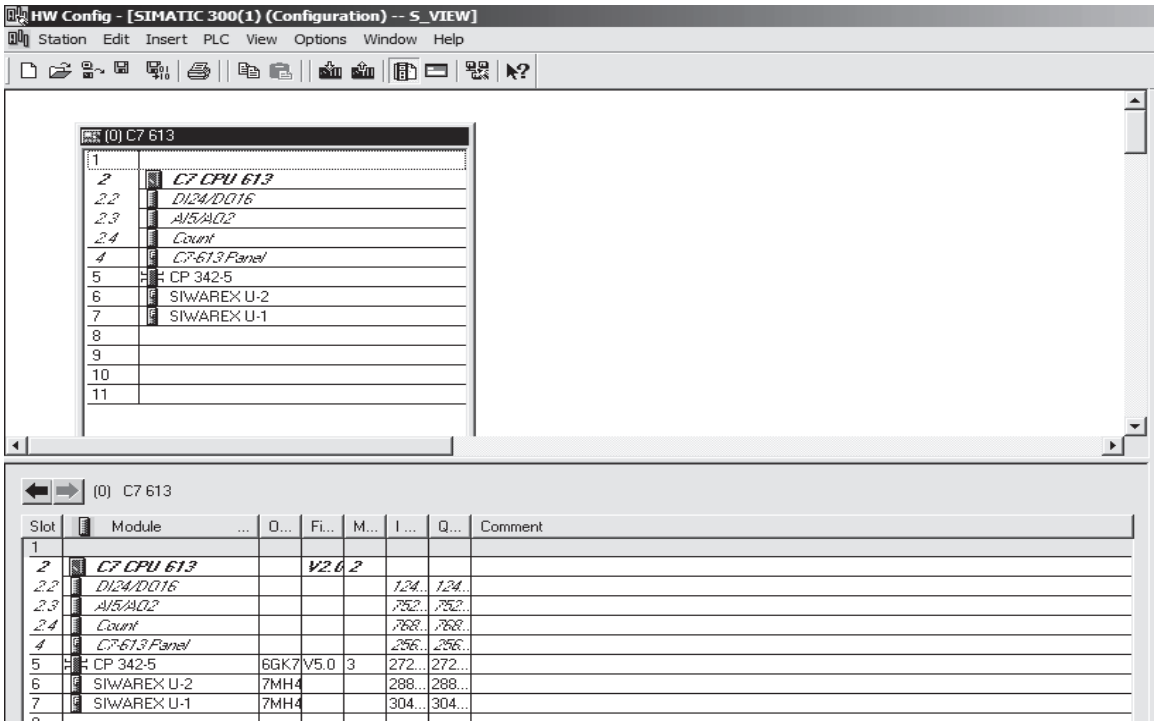


Рис. 3.34. Окно HW Config для настройки аппаратной части системы управления

В свойствах блока **Count** (рис. 3.35) необходимо ввести параметры счетчика.

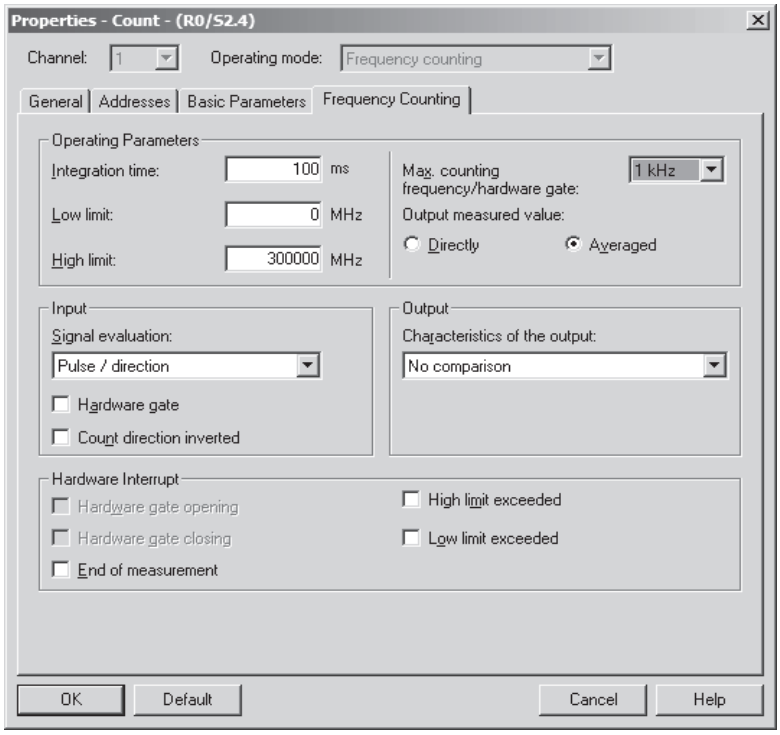


Рис. 3.35. Окно настройки встроенного модуля скоростного счета Count

В свойствах коммуникационного модуля CP 342-5 (рис. 3.36) необходимо задать параметры сети Profibus DP. Скорость соединения 1,5 Мбит/с. Адрес в сети Profibus DP – 1.

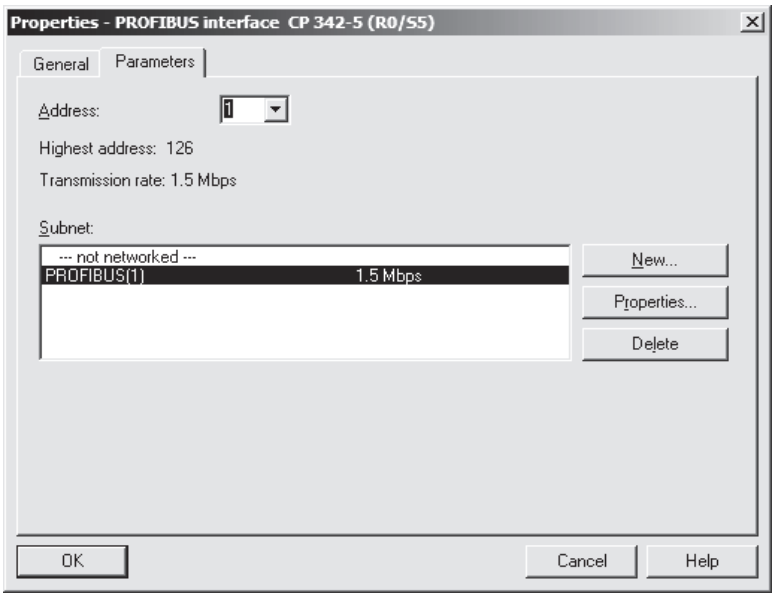


Рис. 3.36. Окно настройки параметров
коммуникационного модуля CP 342-5

После конфигурации аппаратной части и параметров сети окно NetPro (конфигурация сети) примет следующий вид (рис. 3.37).

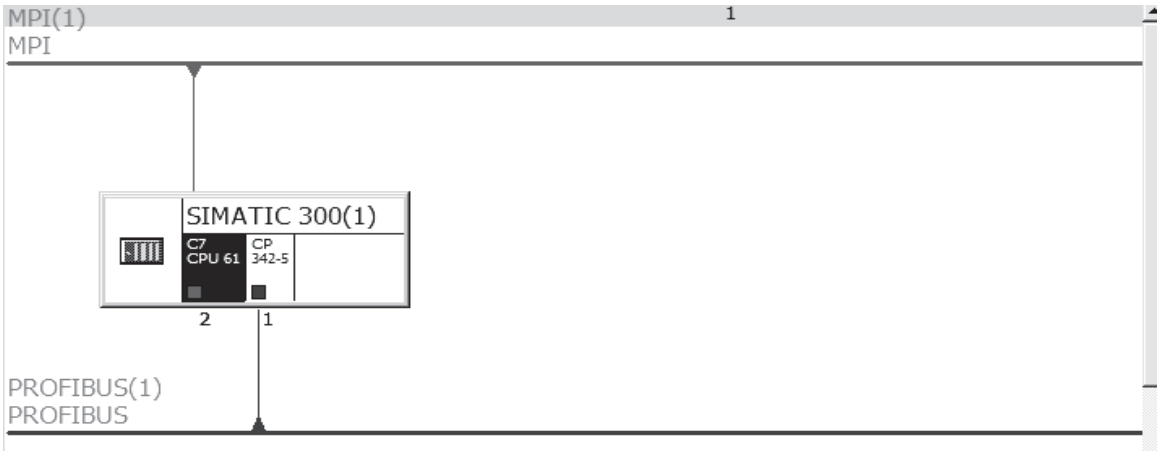


Рис. 3.37. Окно NetPro после настройки аппаратной части и параметров сети

Создание программы управления

Для того чтобы добавить необходимые блоки в программу, необходимо выбрать папку **Blocks** в иерархии проекта и через контекстное меню выбрать необходимый блок (рис. 3.38).

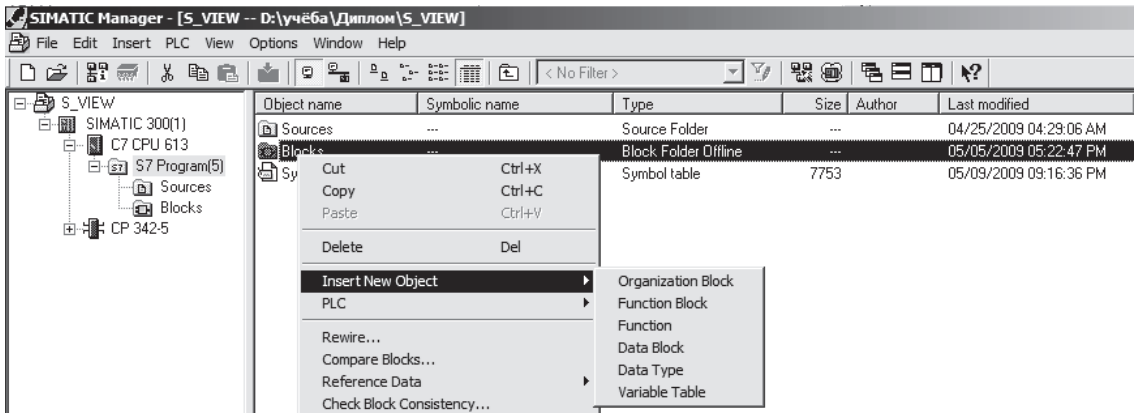
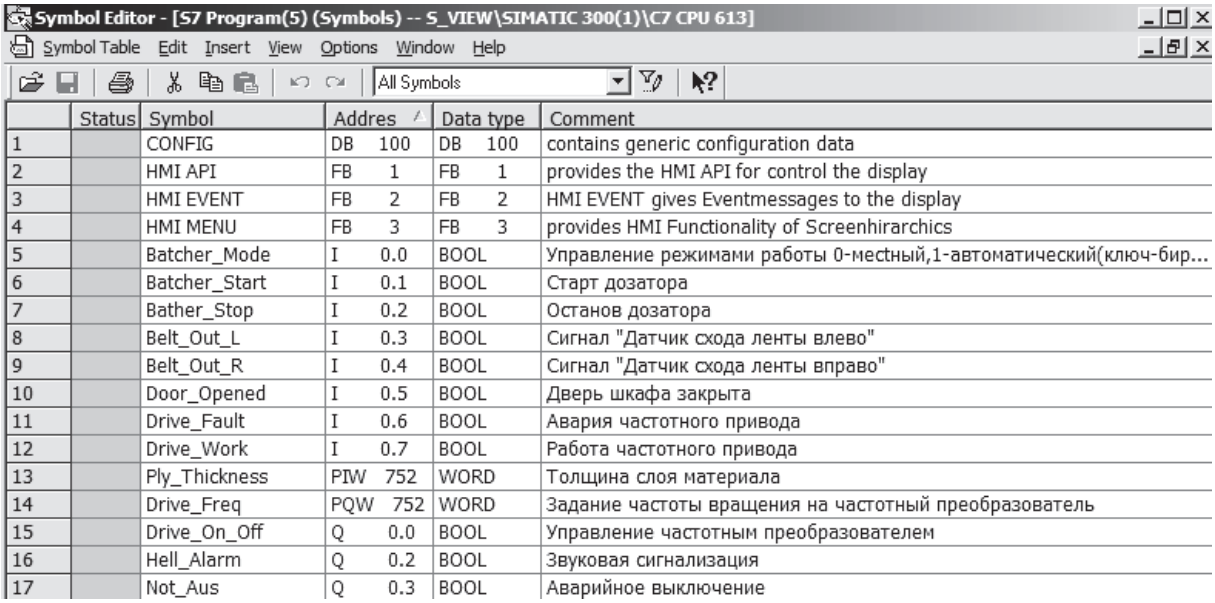


Рис. 3.38. Контекстное меню для добавления блоков в программу

Прежде чем приступить к написанию программы, для удобства присваиваются символьные имена входным и выходным сигналам. Это можно сделать в символьной таблице (рис. 3.39). Также в ней добавляем комментарии для каждого сигнала.



	Status	Symbol	Addres	Data type	Comment
1		CONFIG	DB 100	DB 100	contains generic configuration data
2		HMI API	FB 1	FB 1	provides the HMI API for control the display
3		HMI EVENT	FB 2	FB 2	HMI EVENT gives Eventmessages to the display
4		HMI MENU	FB 3	FB 3	provides HMI Functionality of Screenhirarchics
5		Batcher_Mode	I 0.0	BOOL	Управление режимами работы 0-местный,1-автоматический(ключ-бир...
6		Batcher_Start	I 0.1	BOOL	Старт дозатора
7		Bather_Stop	I 0.2	BOOL	Останов дозатора
8		Belt_Out_L	I 0.3	BOOL	Сигнал "Датчик схода ленты влево"
9		Belt_Out_R	I 0.4	BOOL	Сигнал "Датчик схода ленты вправо"
10		Door_Opened	I 0.5	BOOL	Дверь шкафа закрыта
11		Drive_Fault	I 0.6	BOOL	Авария частотного привода
12		Drive_Work	I 0.7	BOOL	Работа частотного привода
13		Ply_Thickness	PIW 752	WORD	Толщина слоя материала
14		Drive_Freq	PQW 752	WORD	Задание частоты вращения на частотный преобразователь
15		Drive_On_Off	Q 0.0	BOOL	Управление частотным преобразователем
16		Hell_Alarm	Q 0.2	BOOL	Звуковая сигнализация
17		Not_Aus	Q 0.3	BOOL	Аварийное выключение

Рис. 3.39. Символьная таблица

В табл. 3.10 показаны фазы циклической обработки программы.

Таблица 3.10

Фазы циклической обработки программы

Шаг	Последовательность обработки в CPU
1	ОС запускает время контроля цикла
2	CPU записывает значения из таблицы образа процесса на выходах в модули вывода
3	CPU считывает состояния входов модулей ввода и обновляет таблицу образа процесса на входах
4	CPU обрабатывает программу пользователя и исполняет содержащиеся в ней команды
5	В конце цикла ОС выполняет все ждущие своей очереди задачи, например, загрузку и удаление блоков, прием и передачу глобальных данных
6	Наконец, CPU возвращается к началу цикла и перезапускает время контроля цикла

Чтобы в CPU во время циклической обработки программы находился непротиворечивый образ сигналов процесса, CPU обращается не непосредственно к адресным областям входов (I) и выходов (Q) на модулях ввода/вывода, а к области внутренней памяти CPU, содержащей образ входов и выходов.

Организационные блоки образуют интерфейс между ОС и программой пользователя. Они вызываются операционной системой и управляют исполнением программы (циклическим и по прерываниям), а также запуском ПЛК. Организационные блоки определяют порядок, в котором исполняются отдельные программные секции. Исполнение ОВ может быть прервано вызовом другого ОВ. Какому ОВ разрешается прервать другой ОВ, зависит от его приоритета. ОВ с более высоким приоритетом могут прерывать ОВ с более низким приоритетом.

События, которые приводят к вызову ОВ, известны как прерывания.

Табл. 3.11 показывает типы прерываний в STEP 7 и приоритеты соответствующих им организационных блоков.

Таблица 3.11

Использованные в программе организационные блоки

Тип прерывания	Организационный блок	Класс приоритета (по умолчанию)
Главный программный цикл	ОВ 1	1
Циклические прерывания	ОВ 35	12
Асинхронные ошибки	ОВ 82 диагностическое прерывание	26
Запуск	ОВ 100 теплый рестарт	27
Синхронные ошибки	ОВ 121 ошибка программирования ОВ 122 ошибка доступа	Приоритет ОВ, вызвавшего ошибку

Каждый организационный блок имеет стартовую информацию в 20 байтов локальных данных, предоставляемых операционной системой при запуске ОВ. Стартовая информация указывает на событие, вызвавшее запуск ОВ, дату и время запуска ОВ, возникшие ошибки и диагностические события.

Организационный блок ОВ 1 следует добавить в программу в последнюю очередь. Это главный организационный блок, который организует циклическую обработку программы. Циклическая обработка программы – это «стандартный» способ исполнения программы в программируемых логических контроллерах, означающий, что ОС работает в программном цикле и вызывает организационный блок ОВ 1 один раз в каждом цикле главной программы. Поэтому программа пользователя в ОВ 1 исполняется циклически.

Циклическая обработка программы может быть прервана определенными событиями (прерываниями). Если такое событие происходит, то

блок, обрабатываемый в это момент времени, прерывается на границе команды и вызывается другой организационный блок, назначенный соответствующему событию. Как только этот организационный блок завершает свою работу, циклическая программа возобновляется с точки, на которой она была прервана. Это значит, что имеется возможность обрабатывать части программы пользователя, которые не должны обрабатываться циклически, а только когда это необходимо.

Для работы системы весового конвейера помимо OB 1 потребуется:

- OB 35 (организует циклические прерывания и вызывается каждые 100 мс);
- диагностическое прерывание OB 82 (вызывается операционной системой контроллера, когда он обнаруживает ошибку и когда ошибка устраняется. Если не запрограммировать OB 82, то в случае запуска диагностического прерывания CPU переключается в режим STOP);
- стартовый организационный блок OB 100 (вызывается при запуске контроллера и выполняет проверку времени включения. Если оно превосходит допустимое, то CPU переключается в режим STOP);
- блок обработки асинхронных ошибок OB 121 (вызывается в случае обнаружения ошибок программирования. Например, если необходимый блок данных не загружен);
- блок обработки асинхронных ошибок OB 122 (вызывается в случае ошибки доступа к входам/выходам, например, при обращении к несуществующему модулю ввода/вывода).

Также необходимо добавить и запрограммировать программные блоки. В табл. 3.12 представлены используемые функции (FC), функциональные блоки (FB), системные функциональные блоки(SFB) и их назначение.

Таблица 3.12

Программные блоки для работы с конвейером

Блок	Назначение
1	*2*
FC 6	Чтение состояния образа входов, запись их в блоки данных, а также чтение из блоков данных выходных сигналов и запись в образ выходов
FB 11	Синхронизация системных часов с временем главного контроллера управления тракта подачи сыпучих материалов
FB 1	Стандартный блок, поставляемый в комплекте с панелью C7-613. Предназначен для отображения экранов, сообщений, передачи состояния клавиш и светодиодов
FB 2	Стандартный блок, поставляемый в комплекте с панелью C7-613. Предназначен для генерации сообщений на панели

Продолжение табл. 3.12

1	*2*
FB 3	Стандартный блок, поставляемый в комплекте с панелью С7-613. Позволяет запрограммировать интерфейс меню (иерархия экранов) панели
FB 40	Стандартный блок, поставляемый в комплекте с функциональным модулем SiwarexU. С его помощью можно подключать необходимые каналы, диагностировать состояние датчиков и самого модуля. В этом блоке происходит формирование кода АЦП, который соответствует силе деформации тензодатчика
SFB 48	Стандартный системный блок из библиотеки Standart Library в STEP7. С его помощью происходит чтение частоты импульсов с датчика скорости
FB 80	В этом блоке происходит пересчет числа импульсов с датчика скорости в число оборотов вала и рассчитывается линейная скорость ленты конвейера
FB 99	В этом блоке рассчитывается мгновенное значение расхода сыпучего материала
FB 39	Стандартный системный блок из библиотеки PCS7_Library V6.1 в STEP7. Он интегрирует расход материала во времени с заданной частотой квантования
FB 41	Стандартный системный блок из библиотеки Standart Library в STEP7. Он представляет из себя программный ПИД-регулятор

Блоки FB 1, FB 2, FB 3, FC 6, FB 11, FB 41, SFB 48, FB 80 вызываются в организационном блоке OB 1. Блоки FB 39, FB 40, FB 99 вызываются в OB 35.

Описание работы программных блоков:

1) *Чтение состояния входов и установление выходных сигналов в блоке FC 6.*

У функции есть входной параметр IN0. Если его значение TRUE, то при обработке блока осуществляется переход на метку записи значений входных сигналов. Если его значение FALSE, то осуществляется переход на метку записи выходных значений.

2) *Синхронизация времени в блоке FB 11.*

В некоторые моменты времени контроллер верхнего уровня записывает свои текущие дату и время в соответствующие переменные (Year, Month, Day, Hour, Munute, Second), после чего устанавливает в «1» флаг DT_New. Данная программа анализирует флаг DT_New и, если он TRUE, осуществляет следующие действия:

- переменные Year, Month, Day, Hour, Minute, Second преобразуются в формат BCD и записываются в соответствующие поля переменной DT_WinCC;
- вызывается стандартный блок чтения текущих даты и времени SFC1 "READ_CLK", который записывает текущие дату и время в контроллере в переменную DT_PLC;
- вызывается стандартный блок FC 1 "SB_DT_DT", который находит разницу (DT_PLC – DT_WinCC) в формате TIME;
- если найденная разница по модулю больше допустимого отклонения Tolerance_DT или если блок "SB_DT_DT" сбросил флаг BR в «0» (т.е. если разница не укладывается в формат TIME), то вызывается блок установки даты и времени в контроллере SFC0 "SET_CLK", который устанавливает дату и время в контроллере в соответствии с переменной DT_WinCC;
- сбрасывается в «0» флаг DT_New.

3) *Работа блоков FB 1, FB 2, FB 3 будет описана ниже.*

4) *Блок FB 41 осуществляет ПИД-регулирование в данной системе.*

Блок реализует готовый ПИД-регулятор с непрерывным управлением и возможностью ручного воздействия на управляющие сигналы. При назначении параметров есть возможность активировать и деактивировать отдельные функции ПИД-регулятора, чтобы адаптировать его к процессу. На рис. 3.40 представлен внешний вид этого блока с входными и выходными параметрами.

Уставка уровня вводится на входе «Внутренняя уставка» SP_INT в формате числа с плавающей точкой в метрах.

Ввод текущего значения уровня материала осуществляется в переменную «Ввод переменной процесса» PV_IN в формате числа с плавающей точкой.

Для того чтобы было доступно значение PV_IN, необходимо установить значение параметра «Включение переменной процесса периферии» PVPER_ON в FALSE.

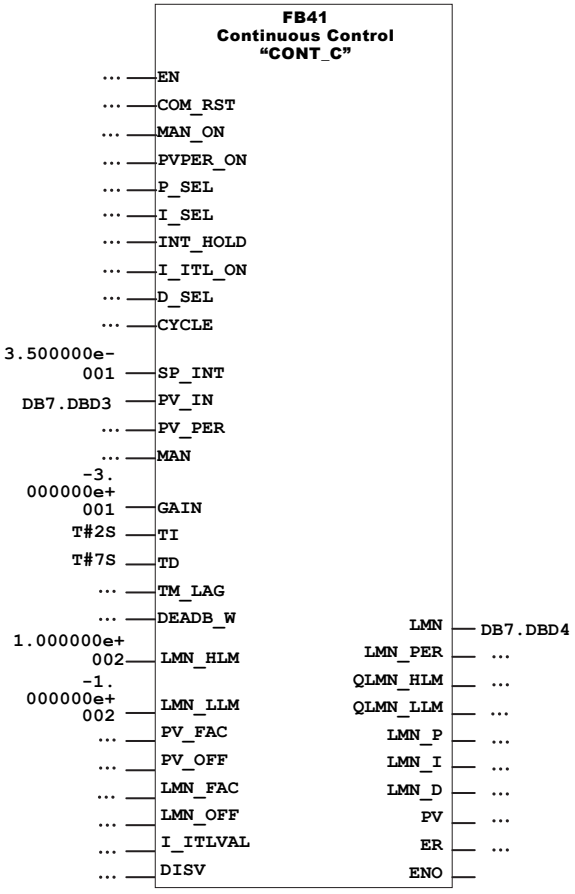


Рис. 3.40. Блок ПИД-регулирования FB 41

Далее происходит формирование сигнала ошибки ER путем вычитания из уставки существующего значения параметра.

Пропорциональная, интегрирующая (INT) и дифференцирующая (DIF) составляющие включены параллельно и могут активироваться и деактивироваться по отдельности. Значение коэффициента пропорциональности вводится в переменную GAIN в формате числа с плавающей точкой (GAIN = 30). Значение времени интегрирования вводится в переменнуюTI в формате TIME (TI = T#2s). Значение времени дифференцирования вводится в переменную TD в формате TIME (TD = T#7s).

Значения переменных P_SEL, I_SEL, D_SEL установлены в TRUE для деблокировки пропорциональной, интегральной и дифференциальной составляющих соответственно.

Значение верхней границы скорости в процентах устанавливается в параметре LMN_HLM, нижняя – в LMN_LLM.

Вход «Коэффициент управляющего воздействия» LMN_FAC умножается на управляющее воздействие. Вход служит для настройки диапазона управляющих воздействий.

На выходе блока LMN формируется значение скорости в формате числа с плавающей точкой.

5) *Стандартный блок SFB 48 считывает количество импульсов с датчика скорости и выдает частоту импульсов* (рис. 3.41).

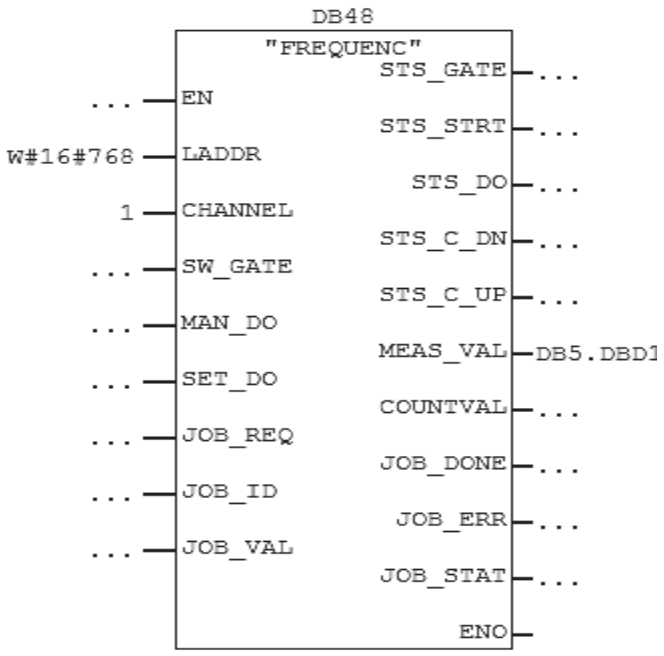


Рис. 3.41. Блок расчета частоты импульсов *SFB 48*

6) *В блоке FB 80 происходит расчет линейной скорости перемещения ленты конвейера.*

В параметр LLADR вводится адрес счетного модуля, который был заранее сконфигурирован в HW Config. В шестнадцатеричной форме адрес 768 будет записан, как W#16#300.

В параметр CHANNEL вводится номер счетного канала. В устройстве С7-613 существует 3 канала для скоростного счета. Используется первый канал.

В параметр MEAS_VAL выводится значение частоты импульсов с датчика скорости.

В параметр MES_FREQ выводится значение частоты импульсов с датчика скорости, рассчитанное блоком SFB 48.

В параметр L_BAR вводится значение длины окружности барабана. Длину окружности барабана рассчитаем по формуле

$$L = 2 \cdot \pi \cdot r, \qquad (3.1)$$

где L – длина окружности барабана конвейера; π – постоянная Пифагора; r – радиус окружности барабана конвейера.

Так как радиус барабана составляет 0,25 м, то длина окружности по формуле (3.1) будет равна

$$L = 2 \cdot 3{,}14 \cdot 0{,}25 = 1{,}57 \text{ м}.$$

В параметр IMP вводится параметр датчика скорости – количество импульсов на один оборот вала. Используемый датчик выдает 100 импульсов на 1 оборот.

В параметр SPEED выводится значение скорости ленты.

Внешний вид блока представлен на рис. 3.42.

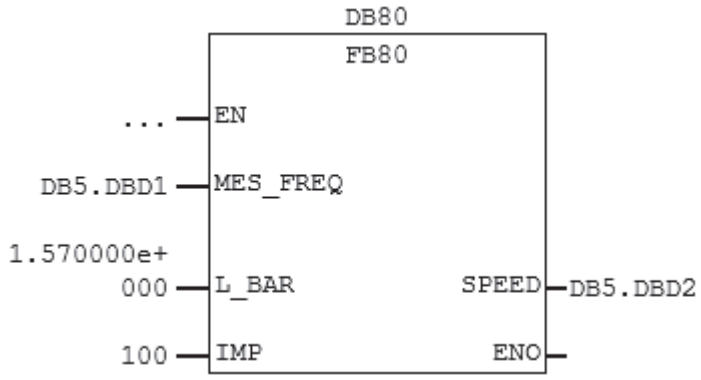


Рис. 3.42. Блок расчета скорости ленты конвейера

7) *Стандартный блок FB 40 для получения кода АЦП веса от модулей Siwarex U.*

Так как модуль Siwarex U двухканальный, то и блок будет вызываться два раза, но с разными параметрами.

В параметр вводится значение адреса модуля Siwarex U. Для двухканального модуля он будет иметь значение 288 или W#16#120 в шестнадцатеричной форме. Адрес одноканального модуля – 304 или W#16#130 в шестнадцатеричной форме.

Параметры CH0_OFF_ON и CH1_OFF_ON предназначены для подключения каналов у модуля.

В параметры CH0_CELL_CHAR и CH1_CELL_CHAR вводится чувствительность тензодатчика («0» – 1 мВ/В, «1» – 2 мВ/В, «2» – 4мВ/В). Требуется ввести значение «1».

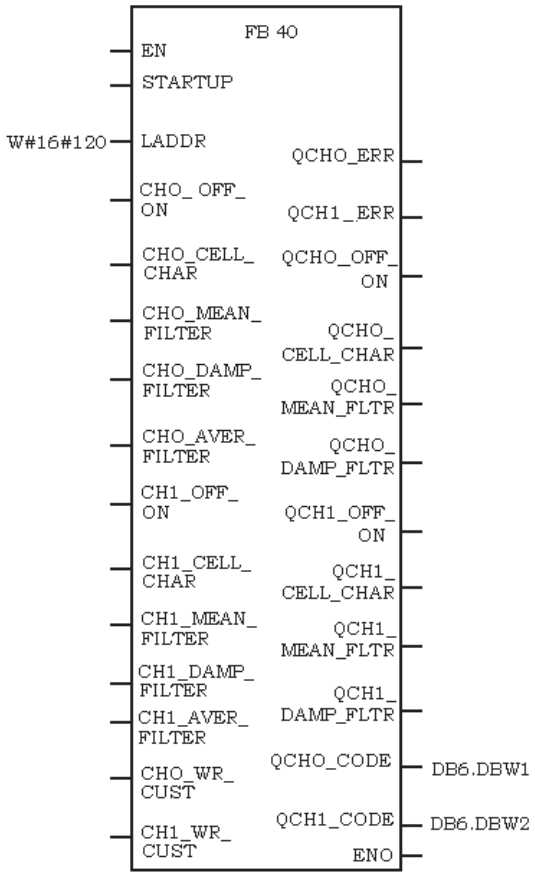


Рис. 3.43. Блок извлечения кода АЦП модулей Siwarex U

В выходных параметрах QCH0_CODE и QCH1_CODE содержится значение кода АЦП, соответствующее значению измеряемого веса.

Чтение и выдача измеренного значения силы (веса) производится блоком непрерывно и безостановочно (каждые 20 мс).

На рис. 3.43 представлен внешний вид блока FB 40.

8) В блоке FB 99 рассчитывается мгновенный расход материала.

В качестве входных параметров CODE_A, CODE_B, CODE_F выступают значения кодов АЦП модулей Siwarex U, полученные при помощи блока FB 40. Для дальнейших расчетов требуется нормализация величины, т.е. преобразование кода АЦП в значение веса. Предел измерения датчика составляет 200 кг. Его значение вводится во входной параметр HL_VES. Величина постоянной нагрузки определяется при помощи утилиты Siwatool U и вводится вручную в качестве входного параметра LL_A, LL_B, LL_F для каждого датчика. АЦП модуля Siwarex U шестнадцатиразрядный. В десятичной форме количество долей составляет 65 535.

Исходя из этого можно составить формулу для получения значения веса при известном коде АЦП для каждого датчика:

$$A = \frac{(HL\_VES - LL\_A)}{65535} \cdot CODE\_A,$$

$$B = \frac{(HL\_VES - LL\_B)}{65535} \cdot CODE\_B,$$
$$F = \frac{(HL\_VES - LL\_F)}{65535} \cdot CODE\_F.$$

Программа для расчета величины А приведена в табл. 3.13.

Таблица 3.13

Программа расчета величины А

Операция	Комментарий
L HL_VES	// загрузка в аккумулятор ACCU1 значения диапазона измерения датчика веса
L LL_A	// загрузка в аккумулятор ACCU1 значения постоянной нагрузки на датчик
-D	// сложение содержимого аккумуляторов с записью результата в ACCU1 для переменных типа DOUBLE
L 65535	// загрузка в ACCU1 константы. При этом старое содержимое перемещается в ACCU2
ITD	// преобразование целого числа (16-бит) типа INTEGER в двойное целое число 32-бита) типа DOUBLE. При этом содержимое ACCU1 сохраняется
/D	// деление содержимого аккумуляторов с записью результата в ACCU1 для переменных типа DOUBLE
L CODE_A	//загрузка в аккумулятор ACCU1 кода АЦП
ITD	// преобразование целого числа (16-бит) типа INTEGER в двойное целое число (32-бита) типа DOUBLE. При этом содержимое ACCU1 сохраняется
*D	// перемножение содержимого аккумуляторов с записью результата в ACCU1 для переменных типа DOUBLE
DTR	// преобразование двойного целое число (32-бита) типа DOUBLE в число с плавающей точкой REAL
T A	// Запись результата в переменную А

Выходным параметром блока будет значение мгновенного расхода материала или производительность конвейера в данный момент времени M_VES. Его значение рассчитывается по формуле [4]:

$$\text{M\_VES} = \left(\frac{A+B}{\cos\alpha}\right) \cdot \sqrt{\frac{\sin\alpha \cdot (A+B) - F \cdot \cos\alpha}{6 \cdot g \cdot L \cdot (A-B)}}, \qquad (3.2)$$

где M_VES – мгновенный расход сыпучего материала, кг/с; A, B, F – масса сыпучего материала, измеренная тензодатчиками, кг; α – угол наклона

весоизмерительного лотка, град; g – ускорение свободного падения, м/с^2; L – длина весоизмерительного лотка, м.

Так как угол наклона и длина лотка – постоянные величины, то в расчетах примем их как константы. Тогда L = 0,5 м; $\cos\alpha$ = 0,866; $\sin\alpha$ = 0,5. Значение ускорения свободного падения также является константой и равно g = 9,8 м/с^2. Тогда формула (3.2) примет вид

$$\text{M\_VES} = \left(\frac{A+B}{0{,}866}\right) \cdot \sqrt{\frac{0{,}5 \cdot (A+B) - F \cdot 0{,}866}{29{,}4 \cdot (A-B)}}.$$

Программа для расчета значения мгновенного расхода представлена в табл. 3.14.

Таблица 3.14

Программа для расчета значения мгновенного расхода

Операция	Комментарий
1	*2*
L A	//загрузка в аккумулятор ACCU1 нормализованного значения веса A
L B	//загрузка в аккумулятор ACCU1 нормализованного значения веса B. При этом старое содержимое перемещается в ACCU2
+R	//сложение содержимого аккумуляторов с записью результата в ACCU1 для переменных типа REAL
L 0.5	//загрузка в ACCU1. При этом старое содержимое перемещается в ACCU2
*R	//перемножение значений аккумуляторов с записью результата операции в ACCU1 для переменных типа REAL
T x_1	//загрузка полученного результата в локальную переменную x_1
L F	//загрузка в аккумулятор ACCU1 нормализованного значения веса F
L 0.866	//загрузка в ACCU1 константы. При этом старое содержимое перемещается в ACCU2
*R	//перемножение значений аккумуляторов с записью результата операции в ACCU1
T x_2	//загрузка полученного результата в локальную переменную x_2
L A	//загрузка в аккумулятор ACCU1 нормализованного значения веса A

Продолжение табл. 3.14

1	*2*
L B	//загрузка в аккумулятор ACCU1 нормализованного значения веса B
-R	//вычитание содержимого аккумуляторов с записью результата в ACCU1 для переменных типа REAL
L 29.4	//загрузка в ACCU1 константы. При этом старое содержимое перемещается в ACCU2
*R	//перемножение значений аккумуляторов с записью результата операции в ACCU1 для переменных типа REAL
T x_3	//загрузка полученного результата в локальную переменную x_1
L x_1	//загрузка в аккумулятор ACCU1 значения x_1
-R	//вычитание содержимого аккумуляторов с записью результата в ACCU1 для переменных типа REAL
L x_3	//загрузка в аккумулятор ACCU1 значения x_3
/R	//деление содержимого аккумуляторов с записью результата в ACCU1 для переменных типа REAL
SQRT	//нахождение корня квадратного из содержимого ACCU1
T x_2	//загрузка полученного результата в локальную переменную x_2
L A	//загрузка в аккумулятор ACCU1 нормализованного значения веса A
L B	//загрузка в аккумулятор ACCU1 нормализованного значения веса B
+R	//сложение содержимого аккумуляторов с записью результата в ACCU1 для переменных типа REAL
L 0.866	//загрузка в ACCU1. При этом старое содержимое перемещается в ACCU2
/R	//деление содержимого аккумуляторов с записью результата в ACCU1 для переменных типа REAL
L x_2	//загрузка в аккумулятор ACCU1 значения x_2
*R	//перемножение значений аккумуляторов с записью результата операции в ACCU1 для переменных типа REAL
T M_VES	//загрузка конечного результата в выходную переменную M_VES

На рис. 3.44 представлен внешний вид блока FB 99.

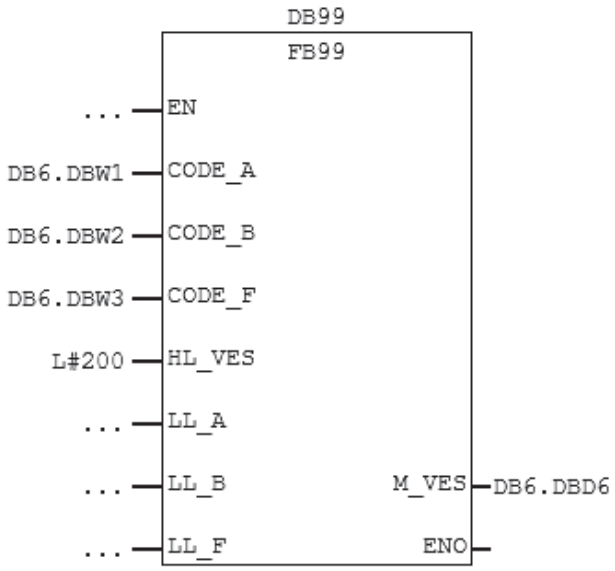


Рис. 3.44. Блок расчета значения мгновенного расхода FB 99

9) *Стандартный системный блок FB 39 интегрирует расход материала во времени.*

Блок вызывается в организационном блоке таймерного прерывания OB 35. Интервал квантования времени задается в параметре SAMPLE_T, время интегрирования задается в параметре TI. В параметрах V_HL и V-LL задаются соответственно верхний и нижний пределы для входной величины U. Внешний вид блока представлен на рис. 3.45.

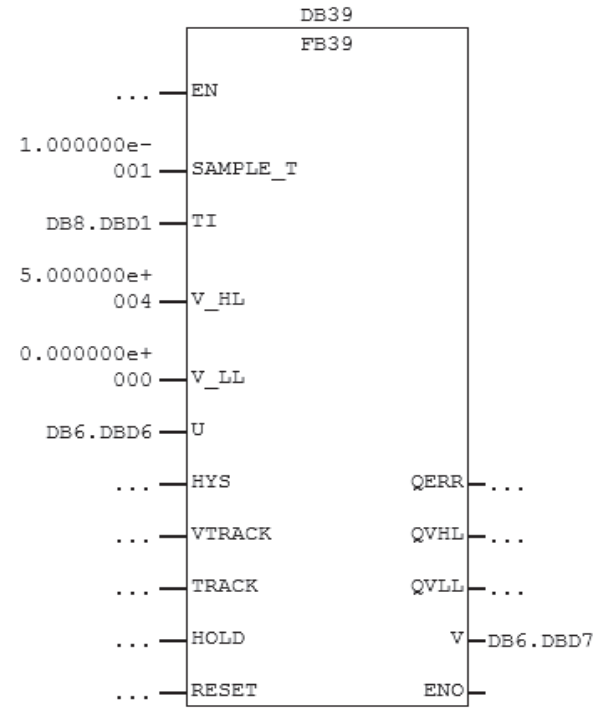


Рис. 3.45. Блок накопления материала

На выходе V блока формируется значение накопленного веса.

Разработка интерфейса пользователя.

Для программирования интерфейса панели используются блоки FB 1 «HMI_API».

1) Блок FB 1 «HMI_API» содержит встроенные функции для работы функции HMI:

- загрузка экранов, сообщений и информации в память встроенного модуля HMI в C7-613;
- отображение экранов;
- отображение сообщений;
- отображение информации;
- передача состояния клавиатуры и светодиодов;
- управление паролями.

Содержимое экранов, сообщений и информации хранится в соответствующих блоках данных: «CONFIG», «SCREEN», «MESSAGE», «INFO». Блоки данных включают всю необходимую информацию для конфигурации экранов, сообщений и информации.

2) Блок FB 2 «HMI_EVENT» нужен для генерации сообщений по вашему желанию. Он управляет входящими сообщениями и их подтверждением. Для каждого сообщения есть:

- текст сообщения;
- дата и время;
- статус сообщения.

В качестве опции вы можете установить блок данных «EVENT_BUFFER» для сохранения буфера событий на карте памяти.

3) Блок FB 3 «HMI_MENU» позволяет запрограммировать интерфейс меню. Для этого необходимо определить иерархию в блоке данных «MENU». Экраны могут меняться с использованием клавиатуры. Когда клавиша нажата, блок оценивает информацию в блоке данных «MENU» и отображает соответствующий экран.

3.6. Система регулирования давления воды во вторичном контуре охлаждения машины непрерывного литья

Концепция вторичного охлаждения предусматривает 13 зон распыления или 25 контуров управления (рис. 3.46). Зона распыления 1, а также зоны 12 и 13 действуют по максимальной ширине заготовки при любой ситуации разливки. В дополнение к этому каждая зона распыления с третьей по пятую разделена на 3 контура управления при максимальной ширине заготовки, а зоны распыления с шестой по одиннадцатую имеют по 2 контура управления. Зона распыления охлаждает один направляющий сегмент ручья по максимальной ширине разливки, в то время как контур управле-

ния – лишь часть максимальной ширины разливки. Зона охлаждения может состоять из одного контура управления либо из трех контуров. Установочные значения системы управления потоком распылительной воды задаются системой автоматизации первого уровня.

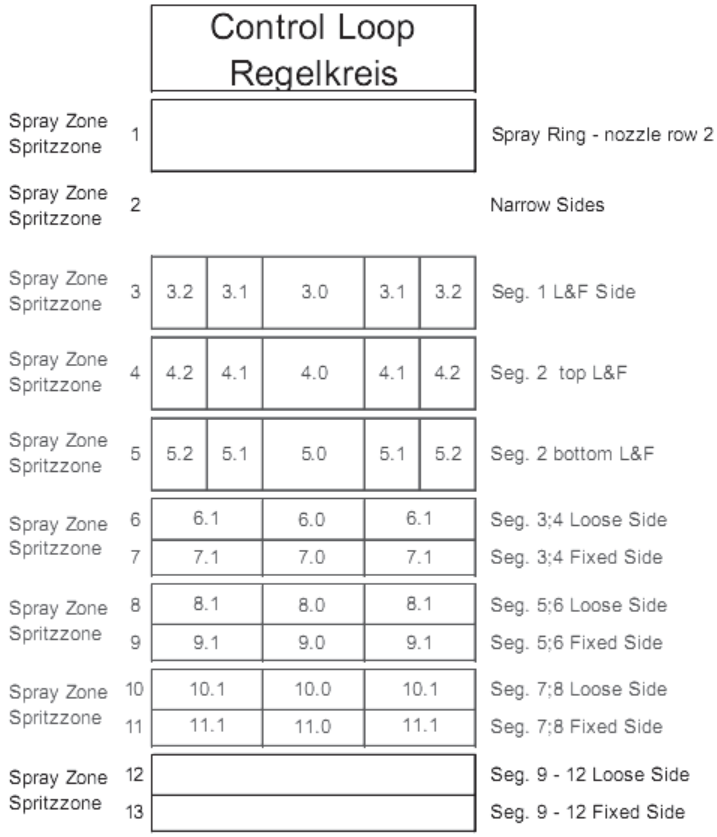


Рис. 3.46. Распределение зон охлаждения и контуров управления

Предусмотрены 3 режима работы:

- Уровень 2 с моделью водообработки по времени. Модель водообработки по времени рассчитывает установочные значения, отправляет их в базу данных уровня 2, который в свою очередь передает их уровню 1.
- Уровень 2 без модели водообработки по времени. Уровень 2 выбирает подходящий номер кривой и передает его уровню 1. Уровень 1 применяет установочные значения из таблицы распыления уровня 1 в соответствии с номером кривой.
- Уровень 1. Оператор вручную выбирает подходящий номер кривой охлаждения. Уровень 1 применяет установочные значения из таблицы в соответствии с этим номером кривой.

На уровне 1 имеется возможность изменить номер таблицы распыления. Когда опорные значения установлены системой базовой автоматизации, показатели воды для всех контрольных контуров сохраняются как функция скорости и качества стали. Зависимость показателя потока рас-

пыляющей воды, л/мин, от скорости разливки, м/мин, тем не менее не нарастает равномерно на всем диапазоне действия сопл.

Калькуляция распыления воды в системе базовой автоматизации производится посредством так называемой «пропорциональной модели скорости». Изменение скорости разливки напрямую изменяет значение протока воды в различных зонах охлаждения. Эта модель не учитывает изменение характеристик ручья во времени. Для выбора образца водораспыления в зависимости от качества и ширины, компьютер машины разливки сообщает номер графика охлаждения системе базовой автоматизации, который соответствует актуальным показателям стали.

Контроль клапана охлаждения и определение опорных значений как функций скорости разливки – это задача системы базовой автоматизации.

В общей сложности 20 графиков на контрольный контур хранится в компьютере разливочной машины и системе базовой автоматизации:

- 10 программ распыления для толщины 200 мм;
- 10 программ распыления для толщины 250 мм.

Каждый из 20 графиков распыления содержит опорные значения для 25 контуров управления в форме кривых охлаждения как функции скорости на данный момент времени. Каждая кривая состоит из 6 опорных точек времени (рис. 3.47).

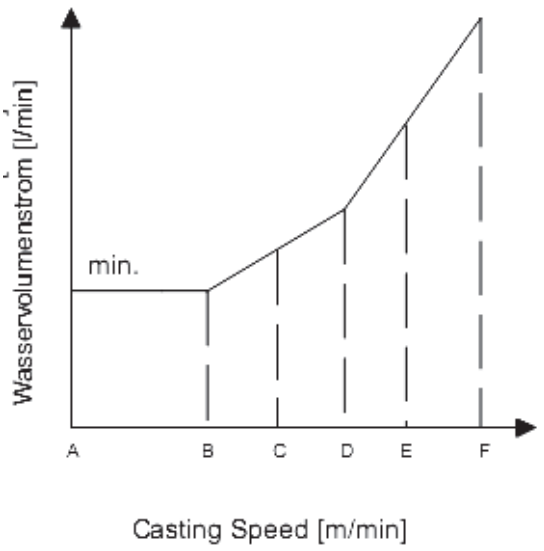


Рис. 3.47. График распыления

Система водоподготовки МНЛЗ предназначена для осуществления управления на современном техническом уровне электрическими приводами и оборудованием насосных станций.

Предусматриваются следующие участки управления:

1) «Насосная яма окалины № 1» – для откачки осветленной воды из приемного резервуара Р1 ямы окалины № 1 и возвращения воды на очистные сооружения и на смыв окалины.

2) «Насосная яма окалины № 2» – для откачки осветленной воды из приемного резервуара Р1 ямы окалины № 2 и возвращения воды на смыв окалины.

3) «Повысительная насосная» – повысительная насосная станция, предназначена для повышения напора воды чистого оборотного цикла и подачи её в зону вторичного охлаждения МНЛЗ № 2.

Все системы управления включают КИП и АСУТП. Каждая система состоит из следующих уровней:

- Нижний уровень – исполнительные устройства, датчики и КИП.
- Уровень 1 – базовая автоматизация технологического процесса. Этот уровень обеспечивает непрерывное измерение и регулирование технологических параметров и управление механизмами.
- Уровень 2 – управление процессами посредством выведения данных на рабочие станции диспетчеров. Этот уровень обеспечивает отображение параметров технологических процессов и управления приводами, интерфейс общения между диспетчером и подсистемой нижнего уровня, хранение архивов, вывод и регистрацию аварийных сообщений.
- Система связи – каналы связи контроллеров с периферийными устройствами и интеллектуальными электрическими приводами (Profibus DP); обмен и передача информации на верхний уровень, передача данных в информационную систему предприятия (INDUSTRIAL ETHERNET).

На первом уровне автоматизации система управления выполняется на базе ПЛК (PLC) SIMATIC S7-400 фирмы Siemens и интеллектуальных электрических приводов с плавными пускателями. АСУТП имеет распределенную структуру сбора данных, на каждом участке объекта устанавливаются станции распределенного ввода/вывода SIMATIC ET-200M.

На втором уровне автоматизации используются персональные компьютеры в промышленном исполнении, для встраивания в 19″ стойки.

Сеть построена на основе технологии Industrial Ethernet.

Система управления, использующая ПЛК SIMATIC S7-400, поддерживает альтернативные варианты решений, в которых узлы ввода/вывода и процессоры пространственно распределены, и сконфигурирована с целью снижения стоимости оборудования и кабельной продукции.

Система управления базируются на следующем:

1) Открытый состав. Основная характеристика открытого состава – модульность. Такая система обеспечивает вертикальное и горизонтальное расширение системы. Вертикальное расширение системы означает, что система поддерживает различное количество ПЛК, количество модулей в ПЛК, т.е. можно добавлять новые либо снимать существующие элементы.

Горизонтальное расширение означает, что в систему могут добавляться новые функции обработки данных.

2) Селективный доступ к данным – означает, что несколько типов пользователей используют систему: диспетчеры, технологи, обслуживающий персонал электрооборудования и приборов и др.

3) Коммуникация (связь) человек – машина (HMI). Развитый HMI обеспечивает пользователям возможность сопровождения технологического процесса через интерактивные показания на операторских станциях.

3.6.1. Подсистема «Повысительная насосная станция»

Повысительная насосная станция предназначена для повышения напора воды чистого оборотного цикла с давлением 0,65 МПа до давления 1,25 МПа и подачи ее на зону вторичного охлаждения.

Решаемые задачи:

1) местное, дистанционное, автоматическое управление механизмами на базе ПЛК и ПК;

2) контроль и сигнализация состояния механизмов;

3) контроль основных технологических параметров:

- ток электродвигателей насосов;
- давление в напорных патрубках насосов и в трубопроводах;
- температура;
- расход;
- уровень в приямке случайных стоков.

В повысительной насосной станции установлены 3 группы насосов. Группа 1 (насосы Н31…Н34) предназначена для подачи воды чистого оборотного цикла с давлением 1,25 МПа потребителю (МНЛЗ № 2) на зону вторичного охлаждения. Рабочие насосы действуют в автоматическом режиме, резервные – запускаются при падении давления на напорных патрубках рабочих. Группа 2 (насосы Н35, Н36) предназначена для откачки случайных стоков из приямка случайных стоков в наружные сети дождевой канализации. Включение и отключение производится по уровню в приямке случайных стоков. Группа 3 (насосы Н37, Н38) предназначена для откачки аварийных стоков при затоплении насосной станции в наружные сети дождевой канализации. Включение и отключение насосов зависит от уровня в приямке случайных стоков.

Работа насосной станции автоматическая, без постоянного присутствия обслуживающего персонала, рабочие и резервные насосы взаимозаменяемы в пределах группы. Первоначальный запуск насосов производится в местном режиме.

3.6.2. Структура сети

Сетевая структура системы управления построена на основе технологии «Profibus DP». ПЛК SIMATIC S7-400 является ведущим устройством для станции распределенного ввода/вывода «ET200-M». Связь контроллера и операторских станций обеспечивается при помощи сети Industrial Ethernet (рис. 3.48).

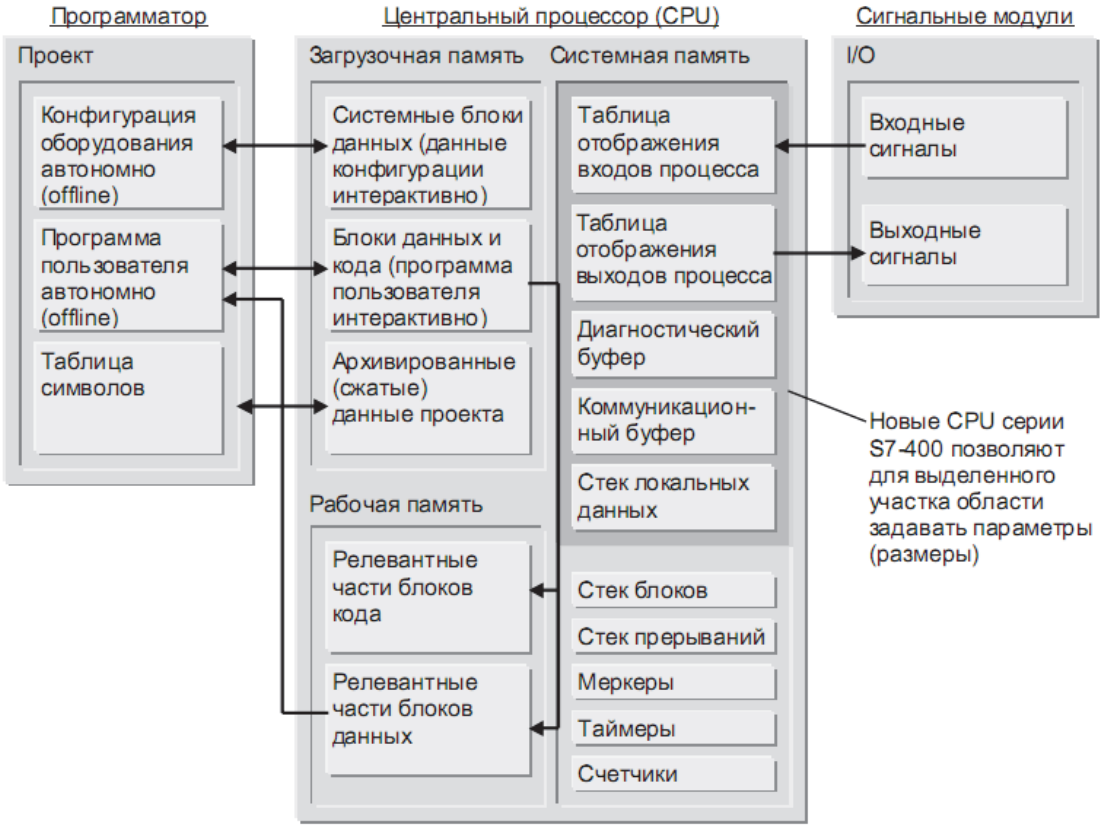


Рис. 3.48. Области памяти CPU

Измерение параметров процесса производится с применением датчиков, которые собирают информацию и придают ей нужную форму, зачастую преобразуя физическую природу измеряемых величин. Управляющие команды передаются к объекту управления органами воздействия (исполнительными механизмами или устройствами привода). Как правило, передача сопровождается изменением физической природы информации и усилением мощности управляющих команд. На базе введенной программы и результатов производимых изменений устройство управления (программируемый контроллер) формирует сигналы воздействия в соответствии с алгоритмом управления объектом.

Для создания пользовательских программ для ПЛК SIMATIC S7-400 служит программный пакет STEP 7. STEP 7 – это базовый пакет программ,

включающий в свой состав весь спектр инструментальных средств, необходимых для программирования и эксплуатации систем управления, построенных на основе программируемых контроллеров SIMATIC S7/C7. Для разработки программ пользователя STEP 7 позволяет использовать следующие способы их представления:

- список инструкций (Statement List – STL). Программы, написанные на STL, занимают минимальный объем в памяти программ контроллеров и обладают наиболее высоким быстродействием;
- диаграммы лестничной логики (Ladder Diagram – LAD). В отечественной литературе этот язык известен как язык релейно-контактных схем;
- язык функциональных блоков (Function Block Control Diagram – FBD) – язык, позволяющий выполнять разработку программы по аналогии с разработкой функциональной схемы устройства управления, создаваемого на основе интегральных логических элементов, счетчиков, таймеров и т.д.

Разработка системы визуализации управления насосами при помощи SCADA-пакета WinCC (рис. 3.49).

WinCC является мощной системой HMI, используемой под управлением Microsoft Windows 2000 и Windows XP. WinCC обменивается информацией как с диспетчером, так и с системой автоматизации.

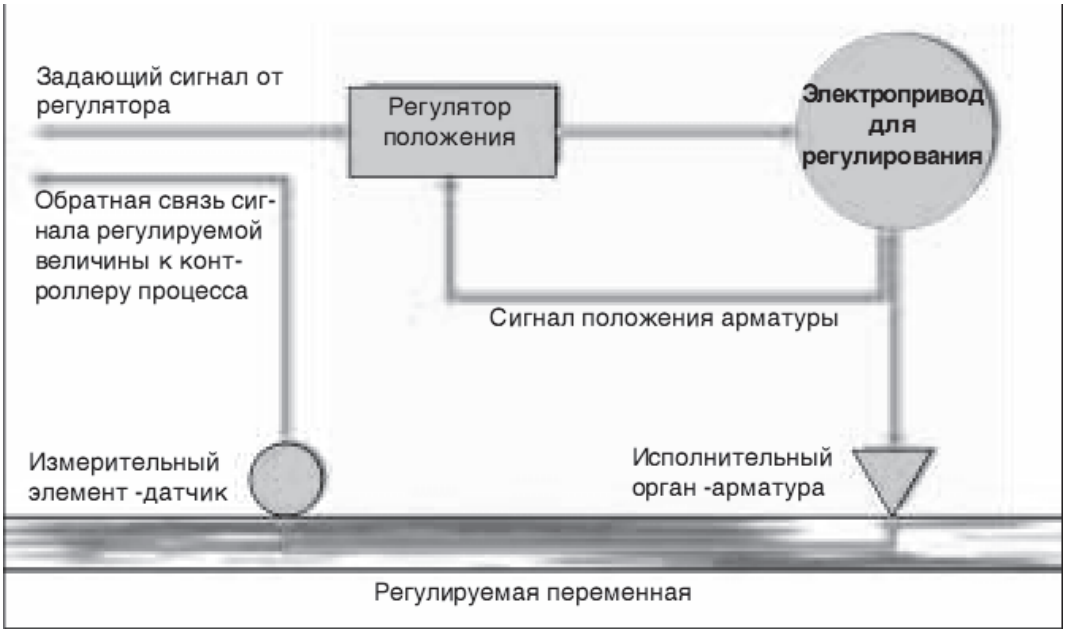


Рис. 3.49. Пример оконных форм WinCC

Для разработки и создания проектов существуют специальные редакторы, к которым можно обращаться с помощью Explorer. Каждый редактор используется для конфигурирования определенной подсистемы WinCC.

Все проектировочные данные хранятся в БД системы проектирования (англ. CS (Configuration System) database).

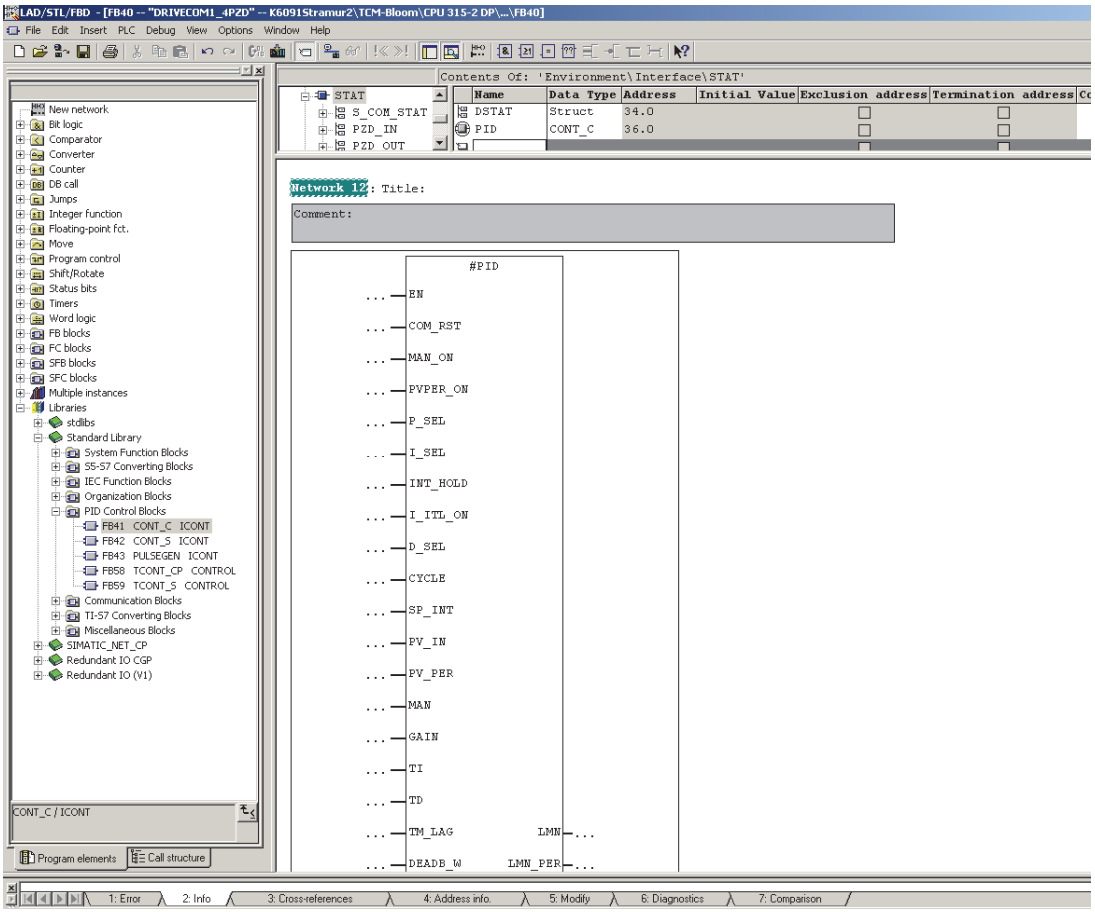
С помощью программного обеспечения системы исполнения диспетчер может контролировать и управлять процессом. В частности, программное обеспечение системы решает следующие задачи:

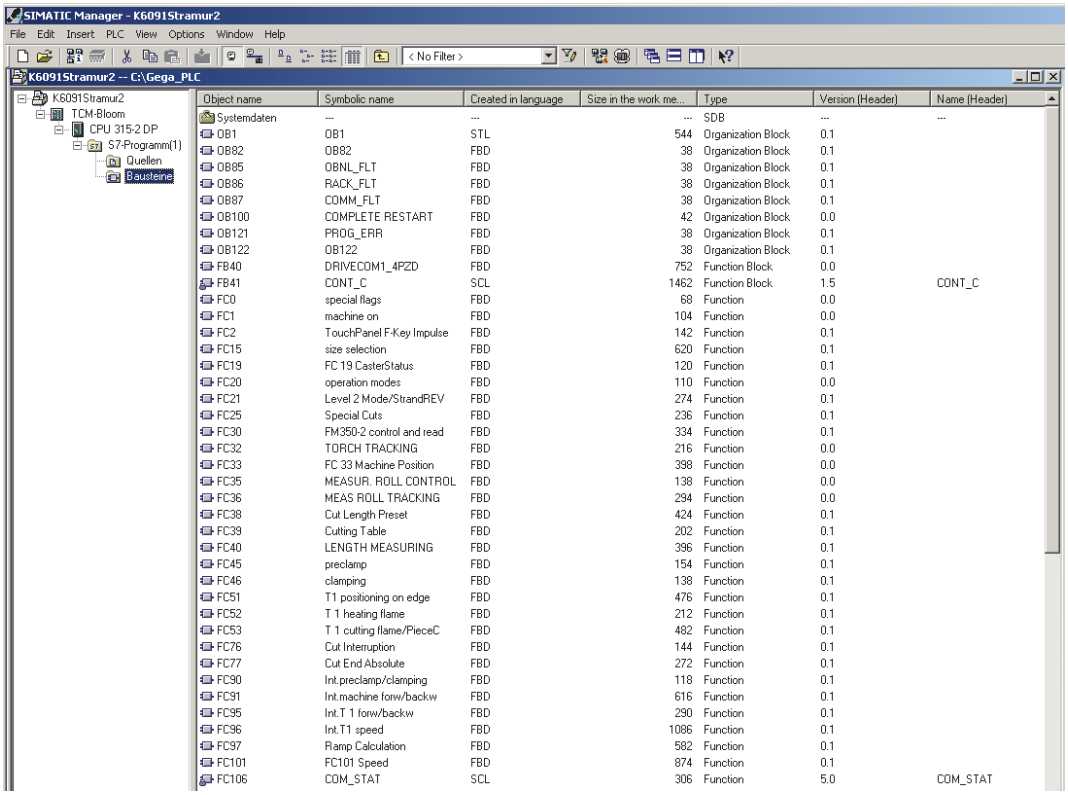
- чтение данных, хранящихся в БД системы проектирования;
- отображение кадров;
- обмен данными с контроллерами;
- архивирование текущих данных режима исполнения – значений процесса и событий системы сообщений;
- управление процессом, например, с помощью определенных значений уставок или активизации/дезактивации.

Программирование ПИД-регулятора.

Для программирования ПИД-регулятора воспользуемся пакетом STEP 7, включающим в себя защищенный от записи блок FB 41, содержащий в себе алгоритм ПИД-регулятора, разработанный немецкой компанией Siemens.

Фрагмент управляющей программы:





Создание СКАДА-системы.

Для привязки значений, хранящихся в блоке данных контроллера, к СКАДА-системе используются так называемые тэги – переменные, хранящие в себе адрес значения технологического процесса.

Работа с проектом визуализации WinCC осуществляется с помощью WinCC Explorer.

Главное окно WinCC Explorer представлено на рис. 3.50.

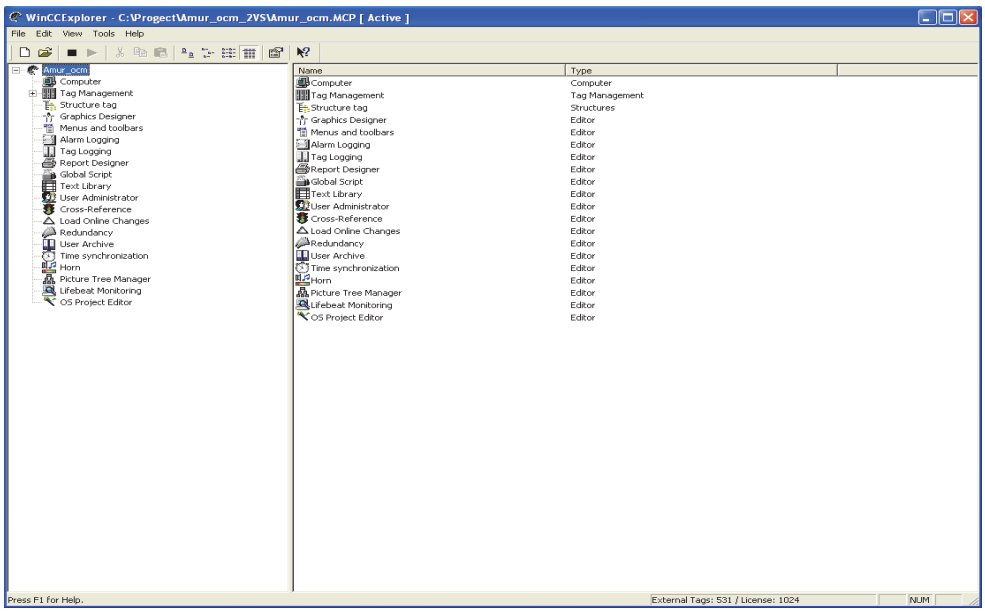


Рис. 3.50. Главное окно WinCC Explorer

В разделе **Tag Management** выбираем раздел **Внешние Тэги** (обращающиеся к блокам данных контроллера). Затем выбираем раздел **Industrial Ethernet**. Это означает, что создаваемые нами тэги будут получать данные из контроллера на основе технологии Industrial Ethernet. Далее выделяем устройство, где хранятся необходимые данные (в нашем случае – контроллер S7-400). В открывшейся корневой структуре все тэги для удобства поиска и хранения разбиты на группы по принадлежности к технологическому подпроцессу (рис. 3.51).

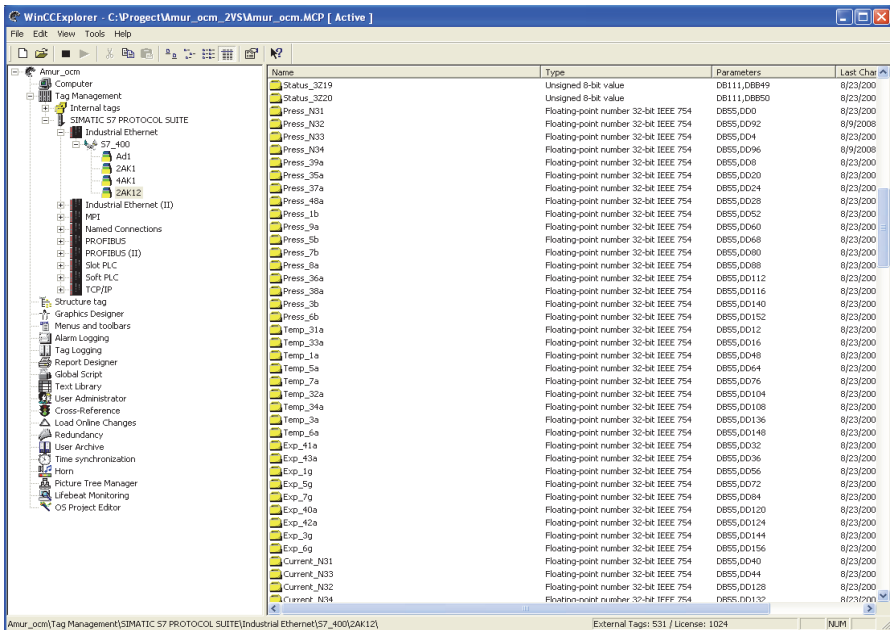


Рис. 3.51. Корневая структура

Нажимаем правую кнопку мышки в открывшемся списке и в контекстном меню выбираем «Создать тэг». Даем тэгу символьное имя, тип данных и адрес в контроллере, по которому тэг должен получать значение (рис. 3.52).

В нашем случае символьное имя давления в трубопроводе «Press_N10». Число является 32-битным с плавающей точкой и находится по адресу DB 35, восьмой байт по счету.

Также, если исходный формат значения не приспособлен для вывода на экран, существует опция внутреннего автоматического преобразования форматов. В нашем случае преобразование происходит в контроллере и здесь этого не требуется.

Главный экран управления повысительной насосной станцией представлен на рис. 3.53. Здесь представлены показания датчиков давления, расхода и температуры, а также состояние и статус насосов, задвижек и пропорциональных клапанов.

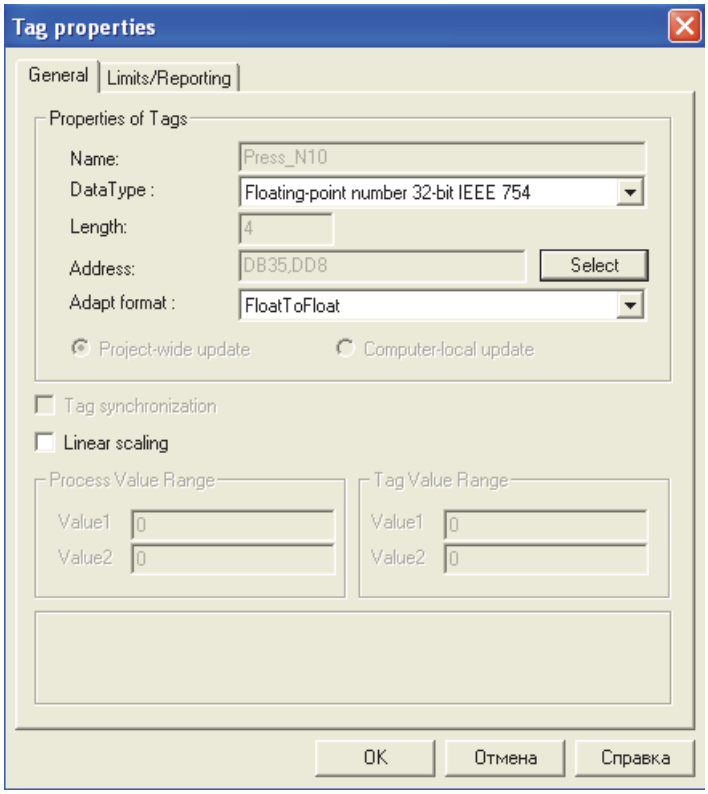


Рис. 3.52. Параметры тэга

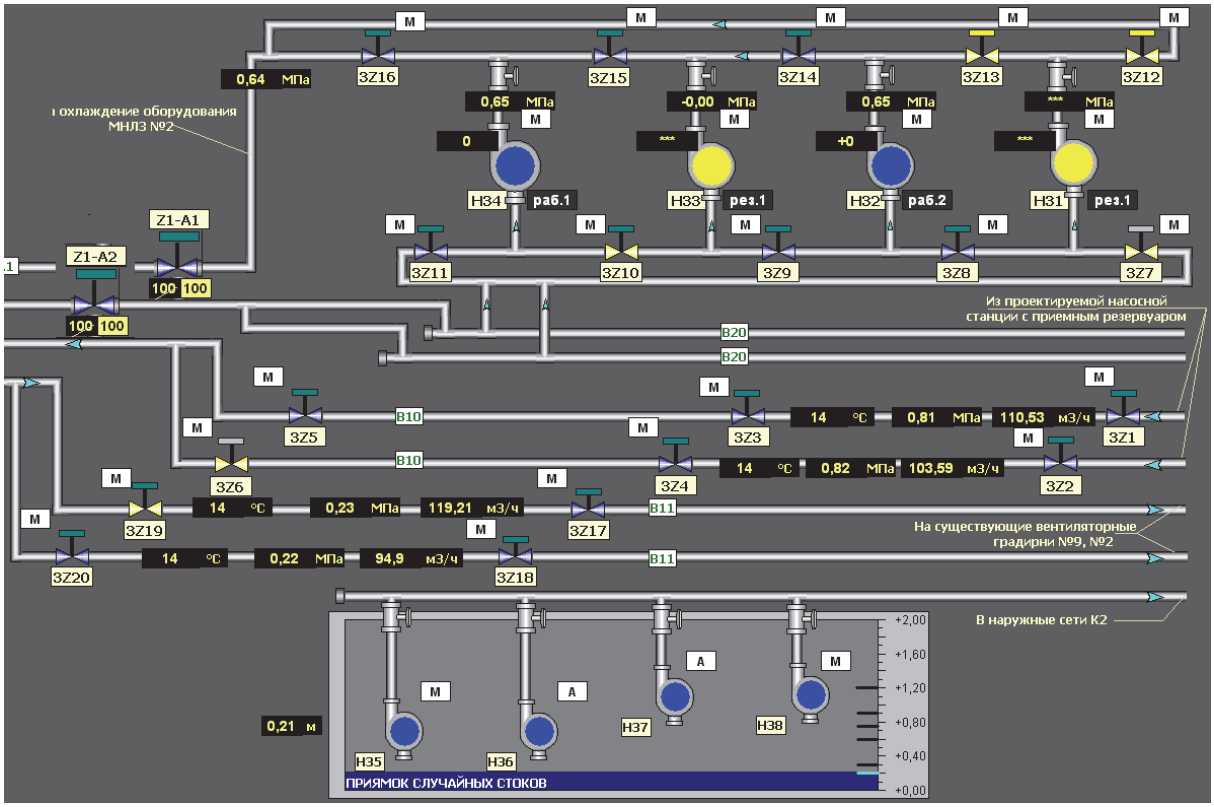


Рис. 3.53. Главный экран управления повысительной насосной станцией

Для архивирования и отображения аналоговых значений во времени, таких как давление, используются тренды – графики зависимости физической величины от времени (рис. 3.54).

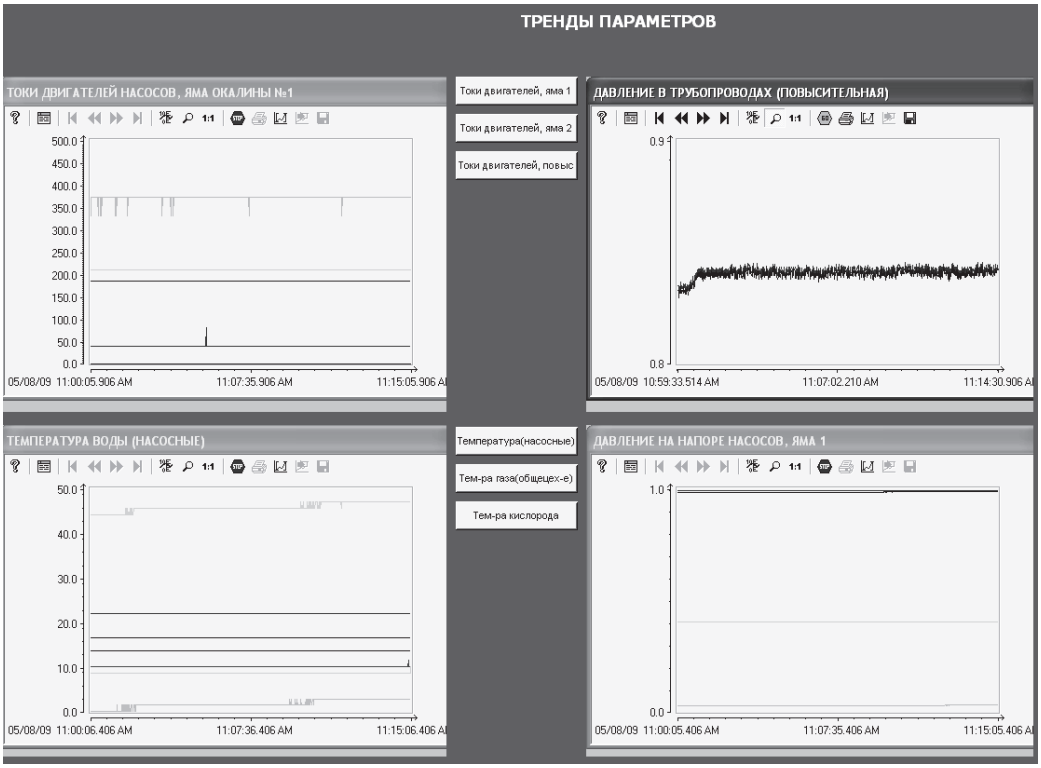
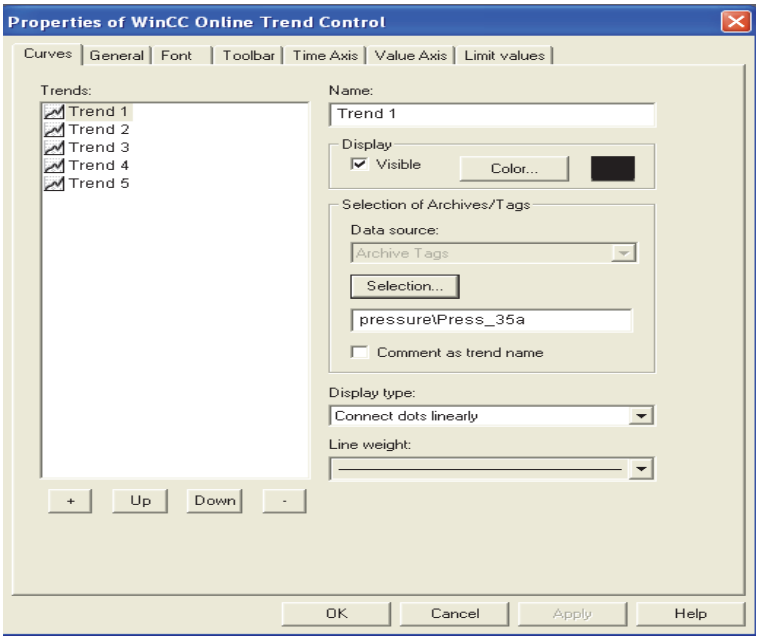
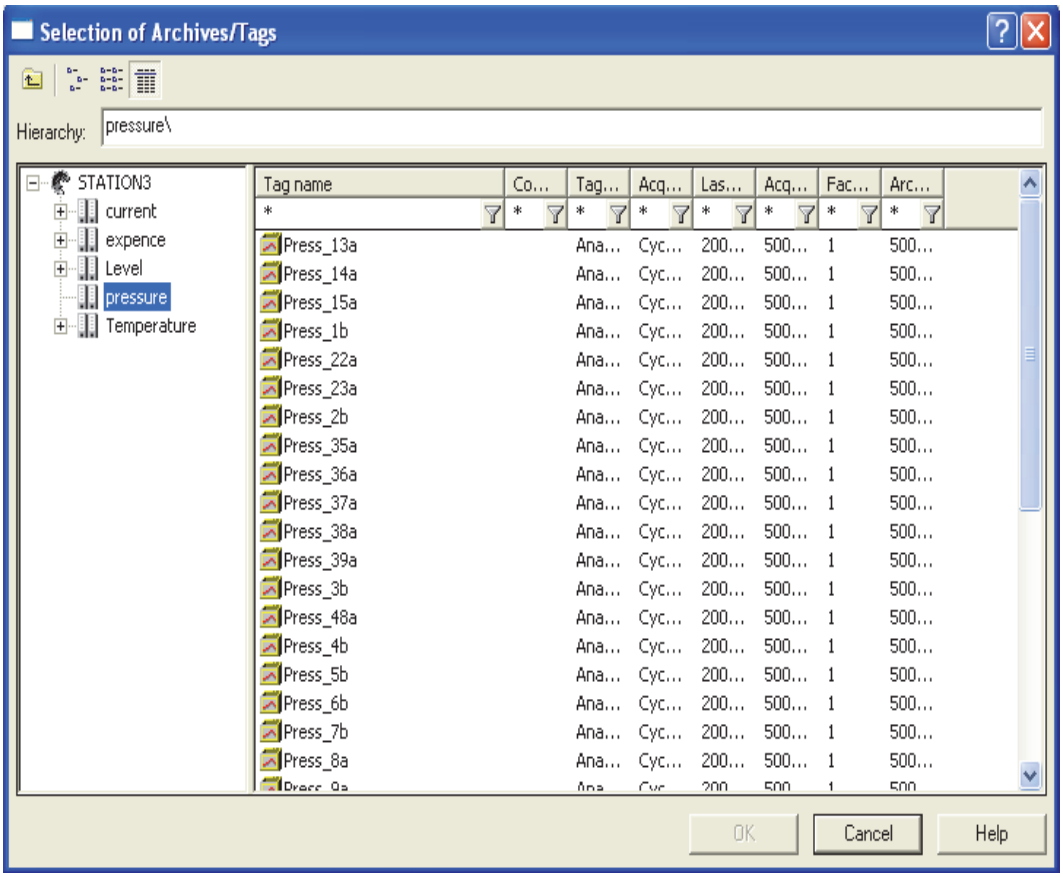
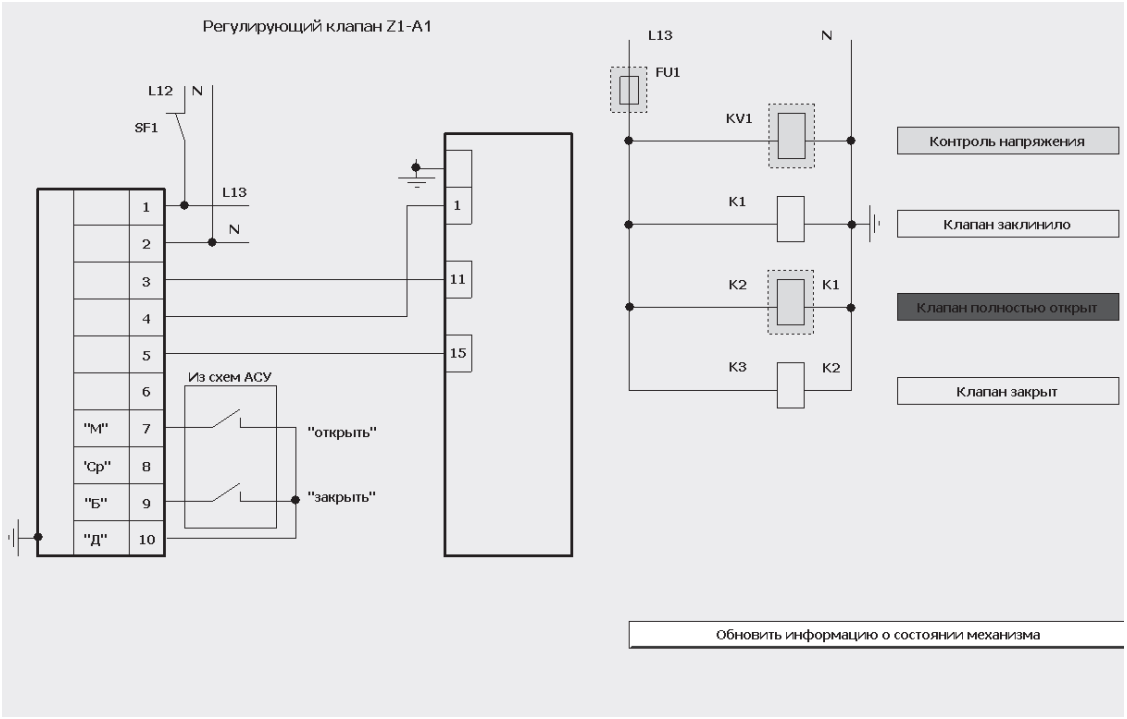


Рис. 3.54. Тренды

Для создания трендов воспользуемся встроенными средствами пакета WinCC:




Selection of Archives/Tags
Hierarchy:
pressure\
STATION3
current
expence
Level
pressure
Temperature
Tag name
Press_13a
Press_14a
Press_15a
Press_1b
Press_22a
Press_23a
Press_2b
Press_35a
Press_36a
Press_37a
Press_38a
Press_39a
Press_3b
Press_48a
Press_4b
Press_5b
Press_6b
Press_7b
Press_8a
OK
Cancel
Help


Регулирующий клапан Z1-A1
L12
N
SF1
L13
N
1
2
3
4
5
6
"М"
7
'Ср"
8
"Б"
9
"Д"
10
Из схем АСУ
"открыть"
"закрыть"
1
11
15
L13
N
FU1
KV1
K1
K2
K1
K3
K2
Контроль напряжения
Клапан заклинило
Клапан полностью открыт
Клапан закрыт
Обновить информацию о состоянии механизма

3.7. Пример программирования системы управления котлоагрегатом

Характеристика технологического процесса.

Микроконтроллер управляет исходящими сигналами в зависимости от анализа входящих сигналов горелки, дополнительными более простыми устройствами автоматики, подогревателем топлива и другими механизмами котлоагрегата.

Процесс управления механизмами состоит из нескольких частей:

- во-первых, нужно организовать постоянный уровень воды в резервуаре парообразования, для этого регулируют подачу воды по уровню в резервуаре, на выходе из резервуара снимают температуру и расход пара, по расходу пара регулируют подачу топлива в камеру сгорания котлоагрегата, для обеспечения более эффективного снятия тепла включают насос принудительной циркуляции воды;
- во-вторых, для организации контура подготовки мазута (в который входит аварийная отсечка подачи мазута, насос мазута, электрический тен и датчик температуры мазута) при достижении заданной температуры мазута разрешают подачу топлива в горелку;
- в-третьих, необходимо регулирование разрежения перед дымососом.

Редукционно-охлаждающее устройство отвечает за подачу пара конечным пользователям с заданными параметрами температуры и давления пара, регулирование давления производится с помощью изменения подачи пара в систему, температура регулируется с помощью подачи холодной воды в систему.

Технические данные горелки.

Основными узлами горелок типа ГМ являются форсуночный узел, газовая часть и воздухонаправляющее устройство.

В форсуночный узел горелок входят паромеханическая форсунка и устройство с захлопками для установки сменной форсунки без останова котла. Основная форсунка устанавливается по оси горелки, сменная – под небольшим углом к оси горелки. Сменная форсунка включается на короткое время, необходимое для чистки или замены основной форсунки.

Газовая часть горелок периферийного типа состоит из кольцевого коллектора с однорядно-однокалиброванной системой газовыдающих отверстий и газоподводящей трубы. Внутри коллектора установлена кольцевая диафрагма, служащая для обеспечения равномерного распределения газа по отверстиям.

Воздухонаправляющее устройство горелок типа ГМ состоит из воздушного короба, осевого завихрителя воздуха и конусного стабилизатора. Лопатки осевого завихрителя профильные, установлены под углом 45° к оси горелки. Небольшая часть воздуха проходит через дырчатый лист (диффузор) для охлаждения форсунки. Однако имеется ряд отличий в конструкциях воздухонаправляющих устройств горелок типа ГМ.

Двухступенчатая мазутная горелка типа «Г200 М2К-М» является автоматической горелкой, распыляющей топливо под высоким давлением.

Для нормальной работы горелки необходимо статическое давление, измеряемое у входа в топливную камеру, не превышающее отчетное, измеряемое у входа в топливную камеру, не превышающее отчетное в диаграмме *Q*/*h* (Г200 М2К-М-11) о максимальном расходе топлива горелки.

Работа горелки.

Мазут в рабочей цистерне необходимо подогреть электрическими тенами (50…60 °С).

Закрываются клапана подачи мазута и газа, открывается клапан подачи воздуха для продувки системы на заданный промежуток времени.

Открывается клапан подачи мазута или газа с одновременной подачей напряжения на высоковольтную систему зажигания топлива до поступления сигнала от фотодатчика, при превышении установленного времени ожидания сигнала с фотодатчика закрываются все краны.

Идет регулирование подачи топлива.

Алгоритм работы контроллера.

Для начала работы программы нужно выбрать вид топлива (газ, мазут).

При выборе газа программа выполняет следующие действия в строгой последовательности:

1) включается насос принудительной циркуляции воды;

2) включается дымоотсос;

3) открывается клапан подачи воздуха на 100 % для продува системы на 15…20 с;

4) закрывается клапан подачи воздуха до установленного значения процента его закрытия;

5) открывается клапан подачи газа с одновременной подачей напряжения на высоковольтную систему розжига;

6) проверяется сигнал с фотодатчика на наличия пламени в камере сгорания;

7) при отсутствии сигнала с фотодатчика по достижении установленного времени, клапана подачи газа и воздуха и отсечка газа закрываются и выводится соответствующее сообщение о сбое, при поступлении сигнала о присутствии пламени в камере сгорания, начинается регулирование подачи газа в зависимости от расхода пара на выходе из камеры парообразования, воздух регулируется в соответствии с расходом газа в установленном соотношении.

При выборе мазута:

1) включается цикл подготовки мазута, подается напряжение на электрический тен в резервуаре с мазутом и камере быстрой подготовки мазута;

2) при достижении установленной температуры мазута открывается клапан подачи воздуха на 100 % для продува системы на 15…20 с;

3) закрывается клапан подачи воздуха до установленного процента закрытия;

4) открываются отсечка мазута и клапан подачи мазута с одновременной подачей напряжения на высоковольтную систему розжига;

5) проверяется сигнал с фотодатчика на наличие пламени в камере сгорания;

6) при отсутствии сигнала с фотодатчика по достижении установленного времени клапан подачи мазута, отсечка мазута и клапан воздуха закрываются и выводится соответствующее сообщение о сбое, при поступлении сигнала о присутствии пламени в камере сгорания начинается регулирование подачи мазута в зависимости от расхода пара на выходе из камеры парообразования, воздух регулируется в соответствии с расходом мазута в установленном соотношении.

Регулирование разрежения осуществляется с помощью управляемого клапана перед вентилятором дымоотсоса до заданной величины разряжения.

В редукционно-охлаждающей установке регулирование давления осуществляется с помощью управляемого клапана на входе в паропровод, регулирование температуры – с помощью управляемого клапана подачи холодной воды в паропровод.

Для организации задачи управления выбран контролер семейства Siemens 416-2DP. Электропитание обеспечивает блок питания PS 407 10A.

Связь с полевыми устройствами обеспечивают модули ввода/вывода, связь с модулями обеспечивается по протоколу Profibus DP через станцию распределенной периферии ET 200M.

Для связи с полевыми устройствами можно рекомендовать следующие модули:

1) Модули аналогового входа – три AI8x12Bit – восьмиканальные модули аналогового входа, к которым подключены следующие аналоговые сигналы:

- Температура на выходе из камеры парообразования.
- Уровень воды в камере парообразования.
- Давление в камере парообразования.
- Процент открытия клапана подачи воды в камеру парообразования.
- Расход пара на выходе из камеры парообразования.
- Температура мазута в контуре подготовки мазута.
- Процент открытия клапана отходящих газов перед дымоотсосом.
- Разряжение перед дымоотсосом.
- Разряжение 1 в камере сгорания.
- Разряжение 2 в камере сгорания.
- Разряжение 3 в камере сгорания.

- Расход газа.
- Расход воздуха.
- Расход мазута.
- Процент открытия клапана подачи газа.
- Процент открытия клапана подачи воздуха.
- Процент открытия клапана подачи мазута.
- Температура в камере сгорания.
- Давление в РОУ.
- Температура в РОУ.
- Процент открытия клапана подачи пара в паропровод.
- Процент открытия клапана подачи холодной воды в РОУ.

2) Дискретные модули входа/выхода – DI32xDC24V – 32-канальный модуль дискретных входов, к которому подключены следующие сигналы:

- Насос принудительной циркуляции воды включен.
- Клапан подачи воды в камеру парообразования открыт.
- Клапан подачи воды в камеру парообразования закрыт.
- Насос подачи мазута включен.
- Вентилятор дымоотсоса включен.
- Клапан отходящих газов перед дымоотсосом открыт.
- Клапан отходящих газов перед дымоотсосом закрыт.
- Клапан пара в РОУ открыт.
- Клапан пара в РОУ закрыт.
- Клапан воды в РОУ открыт.
- Клапан воды в РОУ закрыт.
- Фотодатчик наличия пламя в камере сгорания.
- Клапан подачи газа в горелку открыт.
- Клапан подачи газа в горелку закрыт.
- Клапан подачи мазута в горелку открыт.
- Клапан подачи мазута в горелку закрыт.
- Клапан подачи воздуха в горелку открыт.
- Клапан подачи воздуха в горелку закрыт.

3) DO32xDC24V/0.5A – 32-канальный модуль дискретных выходов, к которому подключены следующие сигналы:

- Клапан подачи воды в камеру парообразования открыть.
- Клапан подачи воды в камеру парообразования закрыть.
- Включить насос принудительной циркуляции воды.
- Закрыть/открыть отсечку мазута.
- Включить электрический тэн подготовки мазута.
- Включить насос мазута.
- Включить дымоотсос.
- Клапан отходящих газов перед газоотсосом открыть.
- Клапан отходящих газов перед газоотсосом закрыть.

- Запустить высоковольтную систему розжига.
- Клапан подачи газа открыть.
- Клапан подачи газа закрыть.
- Клапан подачи воздуха открыть.
- Клапан подачи воздуха закрыть.
- Клапан подачи мазута открыть.
- Клапан подачи мазута закрыть.
- Клапан пара в РОУ открыть.
- Клапан пара в РОУ закрыть.
- Клапан воды в РОУ открыть.
- Клапан воды в РОУ закрыть.

На рис. 3.55 представлено графическое изображение конфигурации оборудования принятое в средстве разработки проектов автоматизации STEP 7 v 5.4.

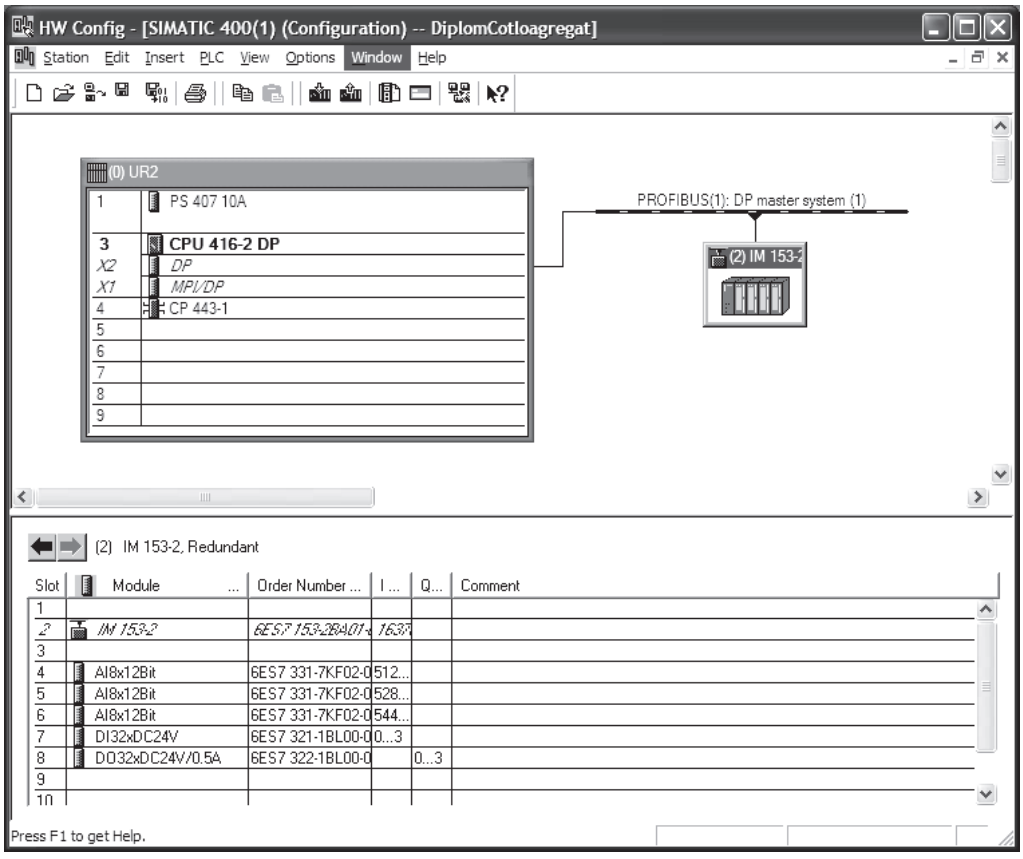


Рис. 3.55. Конфигурация оборудования проекта автоматизации котлоагрегата

Связь с системой визуализации.

Система визуализации реализована на IBM совместимой рабочей станции с установленной программой SIMATIC WinCC 6.0. Связь контроллера с программой визуализации осуществляется по средствам Industrial

Ethernet через коммуникационный процессор CP443-1, сеть MPI нужна для первоначальной загрузки конфигурации в контроллер.

На рис. 3.56 представлено графическое изображение конфигурации сети, принятое в средстве разработки проектов автоматизации STEP 7 v 5.4.

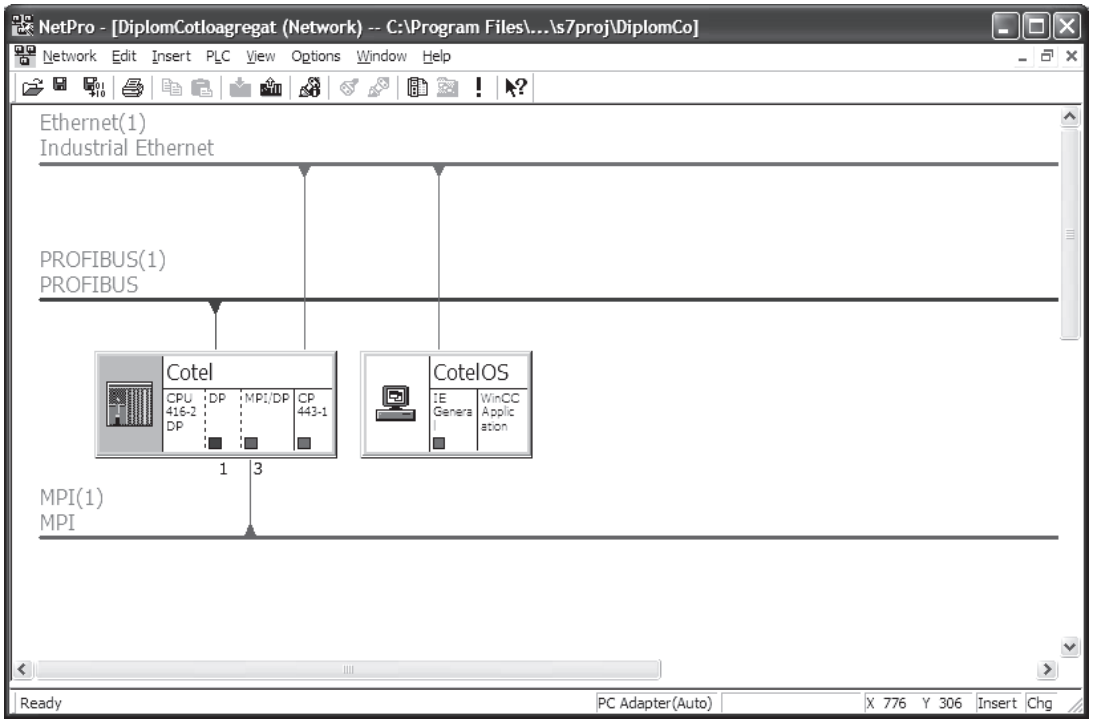


Рис. 3.56. Конфигурация сети проекта автоматизации котлоагрегата

Для реализации программы был выбран язык программирования контроллеров STL, а также графическое представление структуры программы в CFC-схемах.

Выполнение программы начинается с нажатия кнопки «Пуск» в системе визуализации. После нажатия кнопки проверяется, выбрано ли топливо (по умолчанию топливо не выбрано, для того чтобы избежать человеческого фактора), после выбора топлива осуществляется проверка, какое топливо выбрано, т.к. выбор происходит из двух видов топлива. Поскольку перед этим была проверка на наличие выбора, достаточно проверить, выбран ли один из видов топлива.

Регулирование технологических параметров.

Регулирование основных параметров осуществляется по алгоритму ПИД-регулятора, программную реализацию этого алгоритма берем из библиотеки PCS 7 (регулятор с импульсным управлением, CTRL_S). В этом регуляторе предусмотрены ручной режим и автоматический, можно использовать это, чтобы включать и выключать контур регулирования. Также предусмотрены внутренняя и внешняя уставки, значения внутренних уставок будут приняты за значения по умолчанию.

Блок-схема выполнения программы представлена на рис. 3.57.

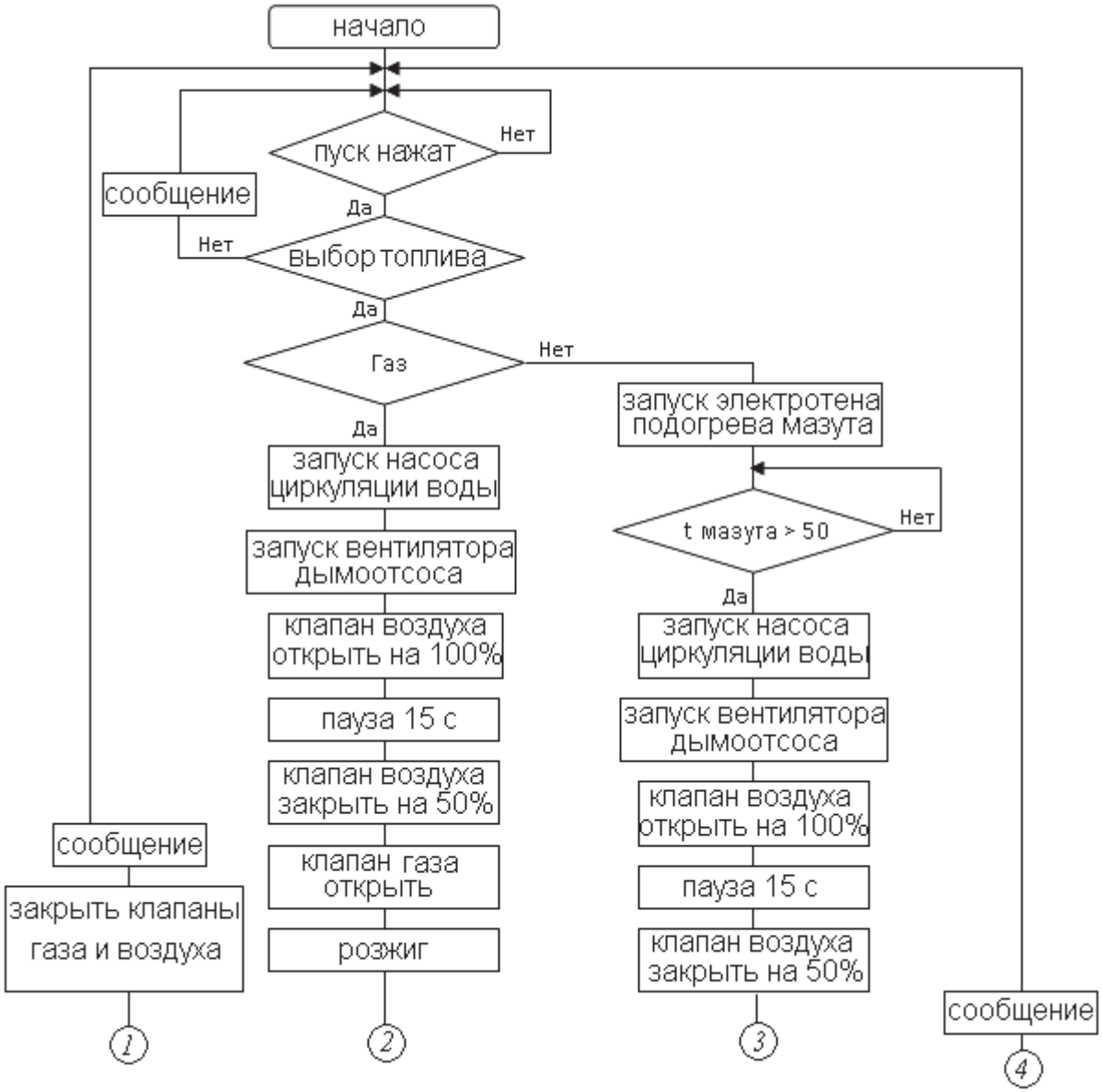


Рис. 3.57. Блок-схема выполнения программы (начало)

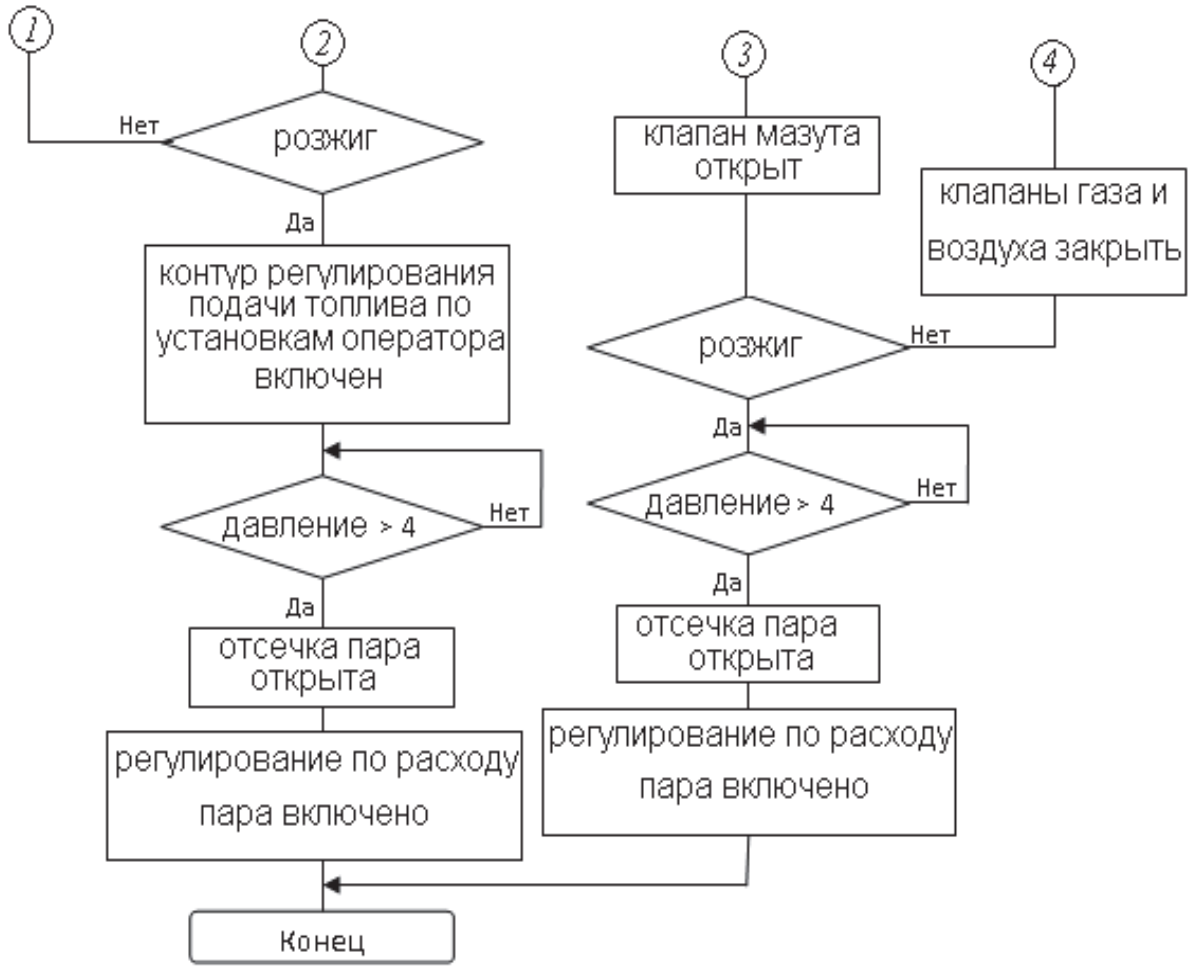


Рис. 3.57. Блок-схема выполнения программы (продолжение)

На рис. 3.58 приведен пример CFC-схемы контура регулирования уровня воды в камере парообразования. Таким же образом организованы контуры регулирования разряжения перед дымоотсосом, давления в РОУ, температуры пара в РОУ и процент открытия клапана подачи воздуха.

Блок CTRL_S – это ПИД-регулятор пошагового управления для систем управления процессами, в которых используются исполнительные элементы интегрального действия (например, клапаны с приводом от двигателя). Клапаны управляются посредством двух двоичных управляющих сигналов.

Принцип работы шагового регулятора основан на комбинации ПИД-алгоритма дискретного регулятора и устройства позиционирования. Во время работы непрерывный управляющий сигнал преобразуется в последовательность управляющих импульсов. Для активизации или деактивизации отдельных функций ПИД-алгоритма может быть выполнена **настройка параметров** для его адаптации к процессу:

1) Режимы работы: ручной, автоматический или следящий режим.

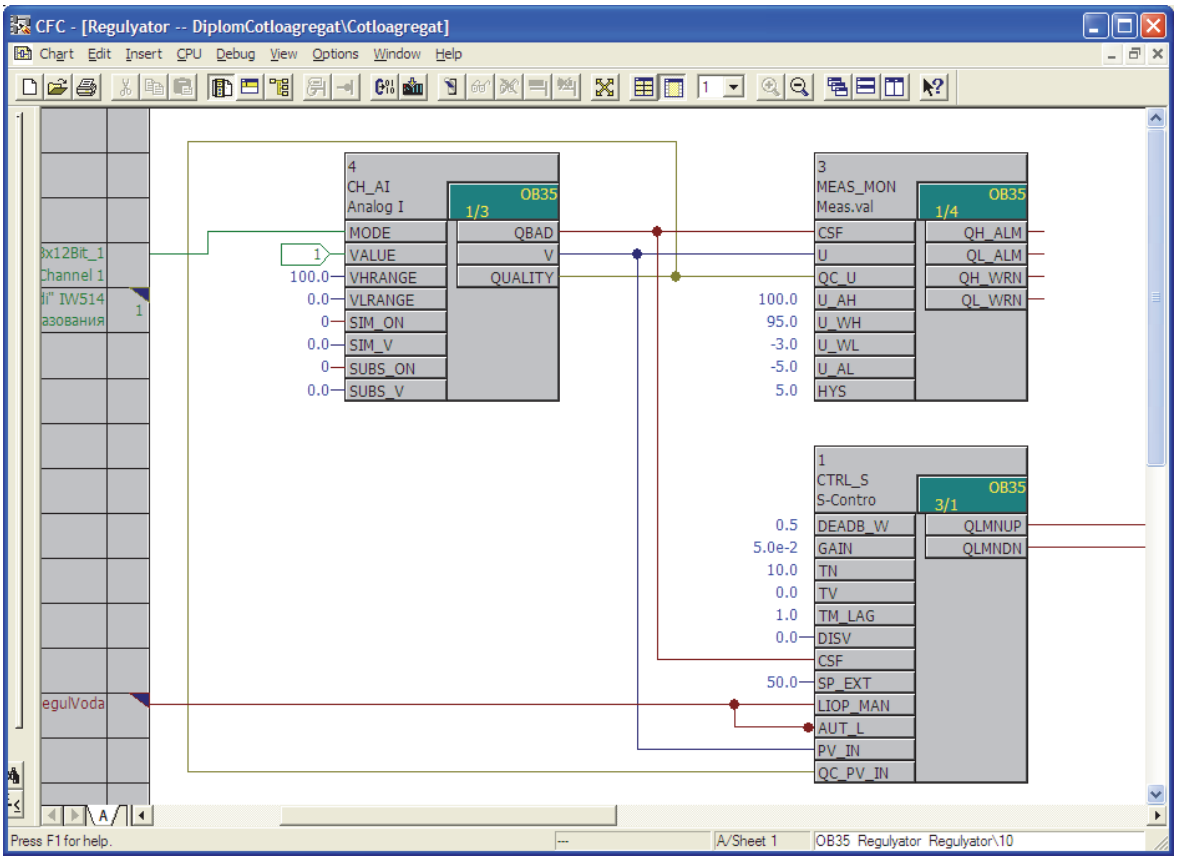


Рис. 3.58. CFC-схема организации регулирования уровня воды в камере парообразования

2) Мониторинг сигналов относительно граничных значений: переменной процесса (process variable) и ошибки рассогласования (error signal), а также формирование сообщений об ошибках посредством блока ALARM8_P.

3) Возмущающее воздействие.

4) Отслеживание заданного значения (Setpoint tracking: SP = PV_IN).

5) Установка диапазона для заданного значения и регулируемой величины (физическая нормализация).

6) Компоненты пропорционального управления, интегрального управления и дифференциального управления могут быть индивидуально активизированы и деактивизированы.

7) Элементы пропорционального управления и дифференциального управления могут быть установлены в цепи обратной связи.

8) Настройка рабочей точки для пропорционального и пропорционально-дифференциального компонентов регулятора.

9) Позиционирующее устройство прямого действия учитывает следующие возможности применения:

- Управление с сигналом обратной связи по положению.

- Управление без сигнала обратной связи по положению.
- Непосредственная настройка сигнала вручную или с помощью подключаемых сигналов.
- Подавление управляющих сигналов сигналами состояния от двигателя (защита двигателя) или клапана (сигналы концевых выключателей).
- Уменьшение числа управляющих импульсов с помощью настраиваемого порогового фильтра.

В данном случае использованы следующие входы/выходы данной функции:

1) DEADB_W – Ширина периода времени нечувствительности:
- GAIN – Коэффициент усиления;
- TN – Время изодрома в секундах;
- TV – Постоянная времени дифференциатора в секундах.

2) AUT_L – Подключаемый вход для ручного/автоматического режима (0 – ручной, 1 – автоматический).

Так как уровень воды в камере парообразования имеет очень малую инертность к регулированию, настройки регулятора выбираем согласно рекомендациям в технической документации к клапану (для быстрого регулирования).

Аналогично настраивают регуляторы давления в РОУ, температуры пара в РОУ, процента открытия клапана подачи воздуха.

Для регулирования разряжения в камере сгорания, т.к. инертность значительно больше, настраивают регулятор согласно рекомендациям в технической документации к клапану (для медленного регулирования).

Для регулирования параметров клапанов подачи топлива, т.к. подача воздуха осуществляется в соотношении с реальным расходом газа или мазута, в схему регулирования включается блок, в котором исходя из реального расхода ведущего топлива и соотношения топлива и воздуха вычисляется уставка на расход воздуха.

На рис. 3.59 представлен пример CFC-схемы регулирования газа и воздуха в соответствии с реальным значением расхода газа и установленным соотношением воздух/топливо.

Программа регулирования газ/воздух по установленному значению расхода газа:

SOOT – функция вычисления уставки расхода воздуха

Входы/выходы функции:

Gaz – реальное значение расхода газа

Soot – установленное соотношение газ/воздух

Vozd – реальное значение расхода воздуха

V_SP – уставка на расход воздуха

R_Soot – Реальное соотношение газ/воздух

Код функции SOOT:

```
L    #Gaz
L    #Soot
*R
T    #V_SP
L    #Vozd
L    #Gaz
/R
T    #R_Soot
```

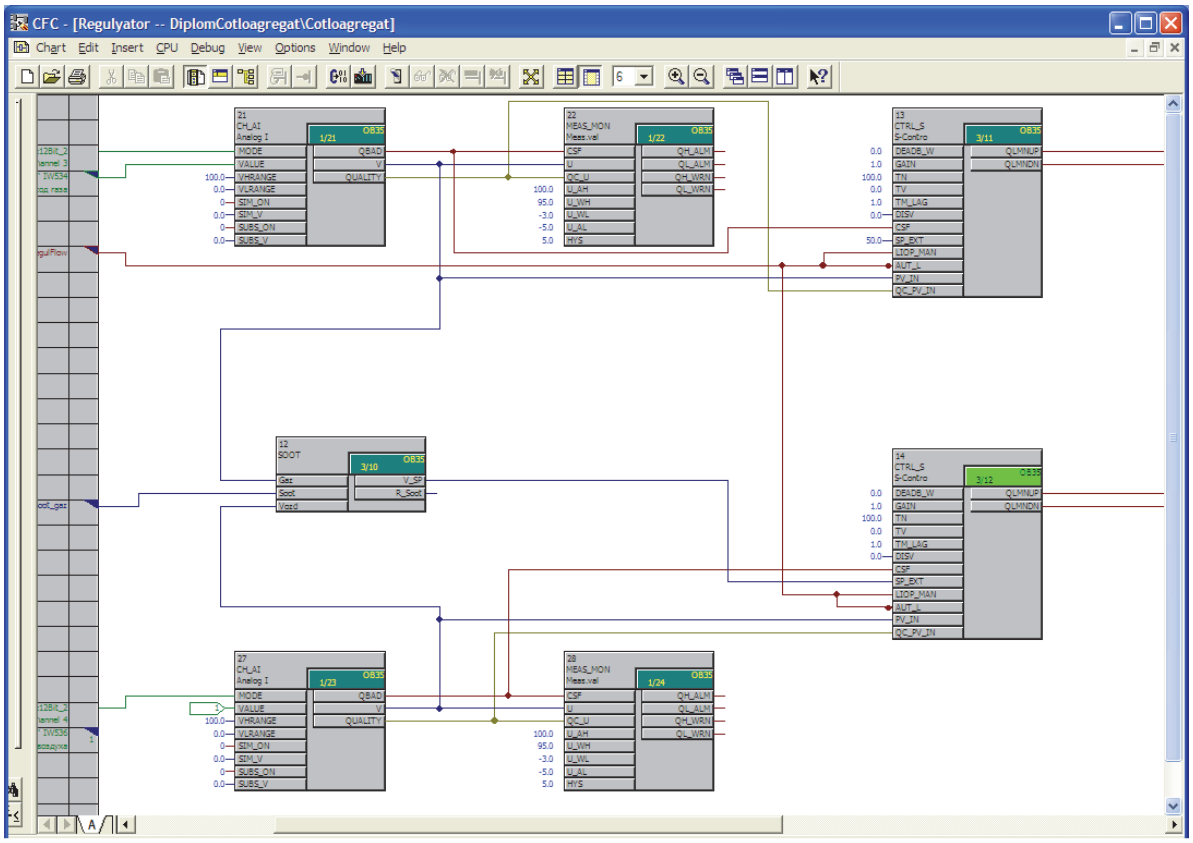


Рис. 3.59. CFC-схема контура регулирования газ/воздух
по установленному значению расхода газа

По такой же схеме организованно регулирование мазут/воздух по расходу мазута, газ/воздух по расходу пара, мазут/воздух по расходу пара.

Контуры регулирования котлоагрегата:

- Уровня воды в камере парообразования.
- Разряжения перед дымоотсосом.
- Давления в РОУ.
- Температуры пара в РОУ.
- Процента открытия клапана подачи воздуха в горелку.
- Расхода подачи газа/воздуха в горелку.
- Расхода подачи мазута/воздуха в горелку.

- Подачи газа/воздуха в зависимости от расхода пара на выходе из камеры парообразования.
- Подачи мазута/воздуха в зависимости от расхода пара на выходе из камеры парообразования.

Программная реализация алгоритма управления.

Реализация основного алгоритма написана в функции FB 1, которая исполняется циклически.

Для хранения управляющих битов, данных и связи с уровнем визуализации создается блок данных DB 1.

Список данных, хранящихся в DB 1 (Data):

RunRegulVoda BOOL Включить/выключить контур регулирования уровня воды

RunRegulDimosos BOOL Включить/выключить контур регулирования отходящих газов

RunRegulPresROU BOOL Включить/выключить контур регулирования давление в РОУ

RunRegulVodaROU BOOL Включить/выключить контур регулирования подачи воды в РОУ

RunRegulVozduxProc BOOL Включить/выключить контур регулирования процента открытия клапана подачи воздуха

SP_ProcVozd REAL Уставка на процент открытия клапана подачи воздуха

SP_Soot_gaz REAL Уставка на соотношение газ/воздух

Основной алгоритм:

RunRegulFlowGazVozd BOOL Включить/выключить контур регулирования газа и воздуха по уставкам расхода

RunRegulFlowMazVozd BOOL Включить/выключить контур регулирования мазута и воздуха по уставкам расхода

RunRegulParGazVozd BOOL Включить/выключить контур регулирования газа и воздуха по уставкам расхода пара

RunRegulParMazVozd BOOL Включить/выключить контур регулирования мазута и воздуха по уставкам расхода пара

StartBut BOOL Нажатие кнопки Старт

Start BOOL Признак того, что старт был нажат

ChoseGaz BOOL Выбор топлива газ

ChoseMaz BOOL Выбор топлива мазут

Chose BOOL Выбор топлива сделан

MesChoseEr BOOL Ошибка Топливо не выбрано

SP_Soot_maz BOOL Соотношение мазут/воздух

ErFlame BOOL Сообщение Нет пламени

PresPar REAL Давление пара

Текст функции FB 1 (Cotel):

```
      A     "Data".Start        // проверка установлен ли признак запуска
      JC    Next                // если установлен, то переход на метку Next
      A     "Data".StartBut  // проверка нажата ли кнопка «Пуск»
      JCN   End                 // если нет, то переход на метку End
      S     "Data".Start        // установка признака запуска
Next: A     "Data".Chose   // проверка сделан ли выбор топлива
      JCN   End1              // если нет, то переход на End1
      A     "Data".ChoseGaz // проверка выбран ли газ
      JCN   Maz                 // если нет, то переход на метку Maz
      S     "RunVoda"         // запуск насоса циркуляции
      S     "RunDim"          // запуск дымоотсоса
          L     100                 // запись 100 в первый регистр
      T     "Data".SP_ProcVozd // установка значения первого регистра в блок данных, откуда он попадет в регулятор
      S     "Data".RunRegulVozduxProc // Запуск контура регулирования процента открытия клапана подачи воздуха
      S     #temp // установка значения temp в 1
      CALL  "TON" , DB2  // вызов функции задержки на 15 секунд
      IN:=#temp
      PT:=T#15S
      Q :=#q
      ET:=#temp1
      A     #q // проверка прошло ли время
      JCN   End // если не прошло, переход на метку End
      L     50      // запись 50 в первый регистр
      T     "Data".SP_ProcVozd  // установка значения первого регистра в блок данных, откуда он попадет в регулятор
      CALL  "TON" , DB3 // вызов функции задержки на 5 секунд
      IN:=#temp
      PT:=T#5S
      Q :=#q
      ET:=#temp1
      A     #q    // проверка прошло ли время
      JCN   End  // если не прошло, переход на метку End
      R     "Data".RunRegulVozduxProc // выключение контура регулирования процента открытия клапана подачи воздуха
      S     "Data".RunRegulFlowGazVozd // Запуск контура регулирования подачи газ/воздух в зависимости от установленного значения расхода газа
      S     "Flame" // Запуск системы розжига
      CALL  "TON" , DB4 // вызов функции задержки на 5 секунд
```

```
IN:=#temp
PT:=T#5S
Q :=#q
ET:=#temp1
A    #q          // проверка прошло ли время
JCN  End      // если не прошло, переход на метку End
A    "Foto"   // Проверка сигнал с фотодатчика
JCN  End2   // если нет, то переход на метку End2
L    4      // запись 4 в первый регистр
L    "Data".PresPar // запись во второй регистр значения давления в камере парообразования
<R    // проверка больше ли значение во втором регистре по сравнению с первым
JCN  End // если нет, то переход на End
S    "OpenPar" // Отсечка пара
R    "Data".RunRegulFlowGazVozd // Выключение регулирования по расходу газа
S    "Data".RunRegulParGazVozd // Включаем регулирование по расходу пара
JC   end /переход на End
Maz: S    "RunVoda"          // запуск насоса циркуляции
     S    "RunDim"           // запуск дымоотсоса
          L    100                  // запись 100 в первый регистр
     T    "Data".SP_ProcVozd // установка значения первого регистра в блок данных, откуда он попадет в регулятор
     S    "Data".RunRegulVozduxProc // Запуск контура регулирования процента открытия клапана подачи воздуха
     S    #temp // установка значения temp в 1
CALL "TON" , DB2  // вызов функции задержки на 15 секунд
IN:=#temp
PT:=T#15S
Q :=#q
ET:=#temp1
A    #q // проверка прошло ли время
JCN  End // если не прошло, переход на метку End
L    50      // запись 50 в первый регистр
T    "Data".SP_ProcVozd  // установка значения первого регистра в блок данных, откуда он попадет в регулятор
CALL "TON" , DB3 // вызов функции задержки на 5 секунд
IN:=#temp
PT:=T#5S
Q :=#q
```

```
      ET:=#temp1
      A    #q    // проверка прошло ли время
      JCN   End  // если не прошло, переход на метку End
      R      "Data".RunRegulVozduxProc // выключение контура регулирования процента открытия клапана подачи воздуха
      S      "Data".RunRegulFlowMazVozd // включение контура регулирования подачи мазута в зависимости от уставки расхода мазута
      S    "Flame" // розжиг
          CALL  "TON" , DB4 // вызов функции задержки на 5 секунд
      IN:=#temp
      PT:=T#5S
      Q :=#q
      ET:=#temp1
      A    #q         // проверка прошло ли время
      JCN   End      // если не прошло, переход на метку End
      A    "Foto"   // Проверка сигнала с фотодатчика
      JCN   End2   // если нет, то переход на метку End2
      L    4      // запись 4 в первый регистр
      L     "Data".PresPar // запись во второй регистр значения давления в камере парообразования
      <R    // проверка больше ли значение во втором регистре по сравнению с первым
      JCN   End // если нет то переход на End
      S    "OpenPar" // Отсечка пара
      R     "Data".RunRegulFlowMazVozd // Выключение регулирования по расходу мазута
      S      "Data".RunRegulParMazVozd // Включение регулирования по расходу пара
      JC    End /переход на End
End2: S     "GazDn"          // Закрыть газ
      S    "VDn"              // Закрыть воздух
          S  "MazDn"              // закрыть мазут
          R    "Data".Start        // снять признак старта
      S    "Data".ErFlame     // ошибка розжига
      JC    End                // переход на метку End
End1: S     "Data".MesChoseEr  //ошибка выбора топлива
      R    "Data".Start           // снятие признака старта
End:  NOP   0       // конец алгоритма
```

Разработка визуализации по средствам WinCC 6.0

Разработка технологического окна котлоагрегата.

В основном окне визуализации показаны все механизмы котлоагрегата, выведены все технологические параметры и основные графические компоненты регулирования. Регулирование процента открытия клапана воздуха имеет характер внутреннего регулирования с заранее известными параметрами, поэтому вывод его на основной экран не является обязательным, а ход выполнения команд можно проследить по изменению расхода воздуха.

На рис. 3.60 изображено главное технологическое окно котлоагрегата.

Для ввода параметров регулирования организованы окна для каждого регулятора, таким же образом организован ввод параметров нормализации и соотношения газ/воздух, мазут/воздух. При поступлении сигналов обратной связи с механизмами соответствующий графический объект меняет цвет.

Для визуализации состояния механизмов приняты следующие цвета:

- красный – ошибка работы механизма;
- серый – механизм не запушен;
- зеленый – механизм в работе.

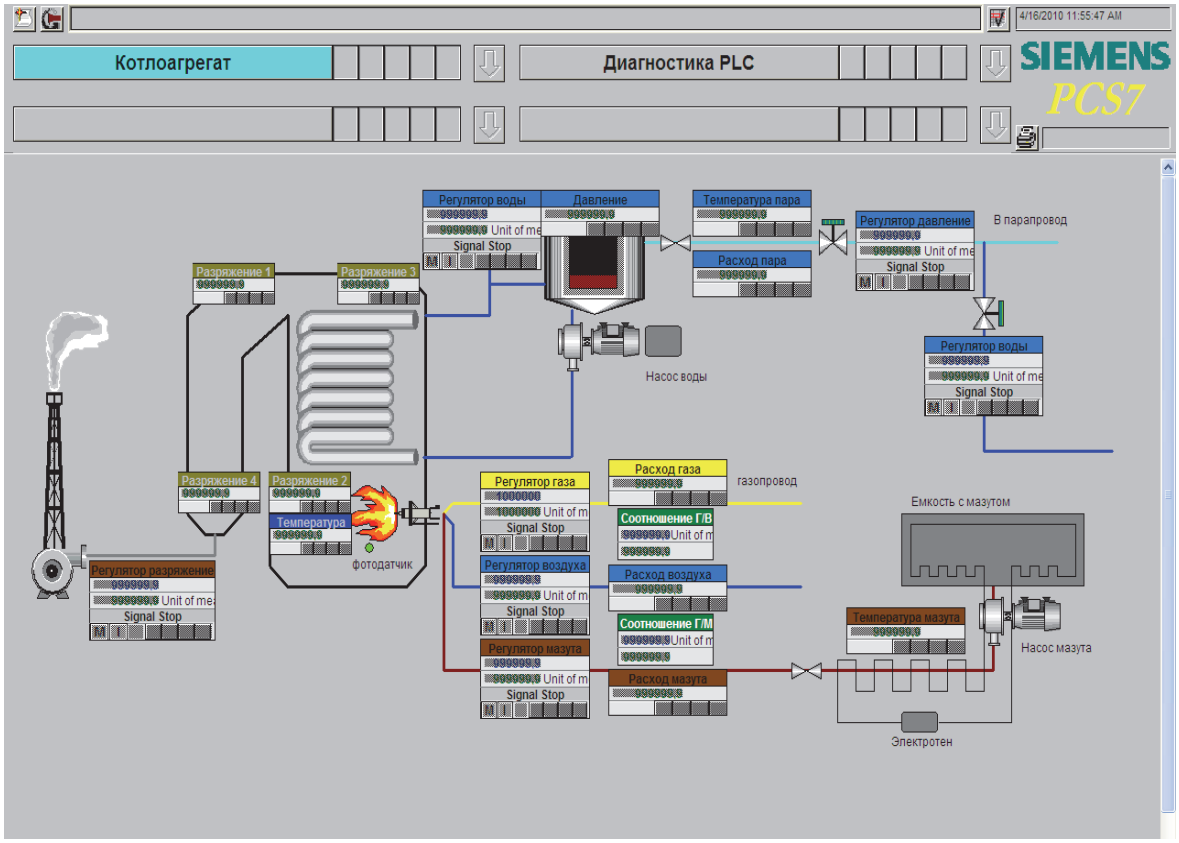


Рис. 3.60. Главное технологическое окно котлоагрегата

На рис. 3.61 – 3.63 изображены окна настройки параметров регулирования и ввода соотношений.

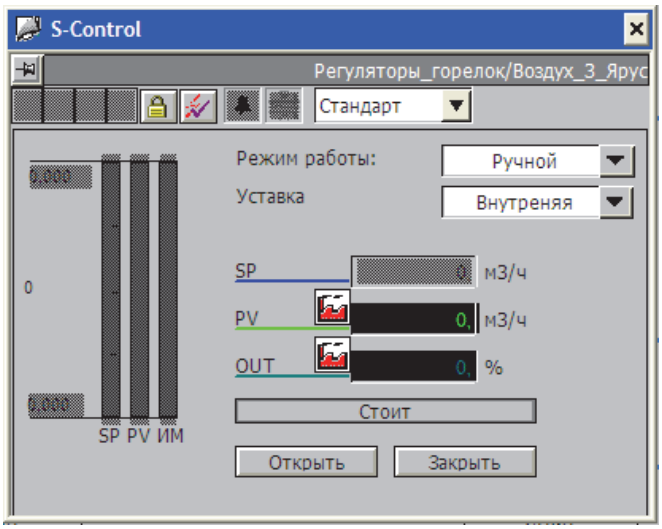


Рис. 3.61. Окно регулятора

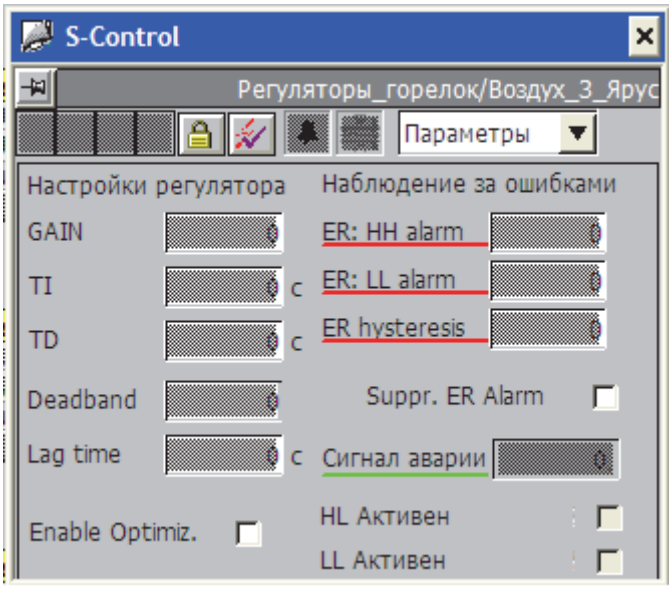


Рис. 3.62. Окно ввода настройки регулятора

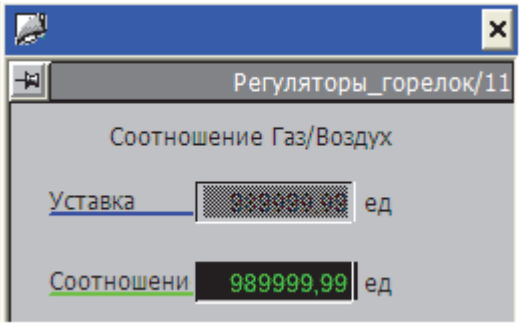


Рис. 3.63. Окно ввода соотношения

ЗАКЛЮЧЕНИЕ

«Информационное обеспечение систем управления» как дисциплина для студентов направлений «Автоматизация и управление», «Управление в технических системах» и других родственных технических специализаций имеет главную специфическую особенность – принадлежность к системам реального времени. Именно эта особенность диктует необходимость организовывать процессы получения, обработки, хранения данных и формирования собственно управлений таким образом, чтобы обеспечивать необходимые законы автоматического регулирования в пределах выделенных для этого временных интервалов. Информационные технологии, основанные на использовании различных языков, в частности с регулярной грамматикой (языков релейных и функциональных схем, логических формул, символических кодов) и проблемно-ориентированных языков программирования, составляют креативную часть информационного обеспечения систем управления. Аппаратная часть технических систем, программная реализация информационных процессов существенным образом влияют на способы обмена данными, их хранения и обработки в реальном времени. Наиболее наглядно это отражается в организации информационного обеспечения систем коммутации электрооборудования и средств промышленной автоматизации и микропроцессорной техники.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Водовозов, В. М. Практическое введение в информационные системы / В. М. Водовозов, В. О. Осипов, А. К. Пожидаев. – СПб. : ГЭТУ, 1995. – 209 с.

2. Годин, В. В. Управление информационными ресурсами: 17-модульная программа для менеджеров «Управление развитием организации». Модуль 17 / В. В. Годин, И. К. Корнеев. – М. : ИНФРА-М, 2000. – 352 с.

3. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания. – Введ. 1992-01-01. – М. : Изд-во стандартов, 1991.

4. РД 50-680-88. Методические указания. Автоматизированные системы. Основные положения. – Введ. 1990-01-01. – М. : Государственный комитет СССР по стандартам, 1991.

5. Лисков, Б. Использование абстракций и спецификаций при разработке программ / Б. Лисков, Дж. Гатэг ; пер. с англ. – М. : Мир, 1989. – 424 с.

6. Петров, В. Н. Информационные системы / В. Н. Петров. – СПб. : Питер, 2002. – 688 с.

7. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем : учеб. / А. М. Вендров. – М. : Финансы и статистика, 2000. – 352 с.

8. Суздорф, В. И. Системы автоматизированного проектирования систем автоматического управления : учеб.-методическое пособие / В. И. Суздорф. – Комсомольск-на-Амуре : ГОУВПО «КнАГТУ», 2004. – 102 с.

9. Голенищев, Э. П. Информационное обеспечение систем управления / Э. П. Голенищев, И. В. Клименко. – Ростов н/Д : Феникс, 2003. – 352 с.

10. Джексон, Г. Проектирование реляционных баз данных для использования с микроЭВМ / Г. Джексон. – М. : Мир, 1991. – 242 с.

11. Дейт, К. Введение в системы баз данных / К. Дейт. – М. : Наука, 1980. – 187 с.

12. Выплавка и разливка стали в ЭСПЦ-2 : технологическая инструкция 103-ЭС-388-98 / ОАО «Кузнецкий металлургический комбинат». – Новокузнецк, 1998.

13. Мешков, А. С. Формирование характеристик систем массового электропривода / А. С. Мешков, В. И. Суздорф // Ученые записки Комсомольского-на-Амуре гос. техн. ун-та. Науки о природе и технике. – 2010. – № III-1(3). – С. 57-61.

Учебное издание

Суздорф Виктор Иванович

ПРОЕКТИРОВАНИЕ СИСТЕМ АВТОМАТИЗАЦИИ И УПРАВЛЕНИЯ

Учебное пособие

Научный редактор – доктор технических наук,
профессор В. А. Соловьев

Редактор Т. Н. Карпова

Подписано в печать 13.03.2014.
Формат 60 × 84 1/16. Бумага 60 г/м^2. Ризограф EZ570E.
Усл. печ. л. 11,97. Уч.-изд. л. 11,68. Тираж 500 экз. Заказ 26140.

Редакционно-издательский отдел
Федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Комсомольский-на-Амуре государственный технический университет»
681013, Комсомольск-на-Амуре, пр. Ленина, 27.

Полиграфическая лаборатория
Федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Комсомольский-на-Амуре государственный технический университет»
681013, г. Комсомольск-на-Амуре, пр. Ленина