Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Пермский национальный исследовательский
политехнический университет»

М.В. Кавалеров

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Утверждено
Редакционно-издательским советом университета
в качестве учебного пособия

Издательство
Пермского национального исследовательского
политехнического университета
2014

УДК
К12

Рецензенты:

Кавалеров, М.В.
К12 Компьютерные технологии управления в технических системах : учеб. пособие / М.В. Кавалеров. – Пермь : Изд-во Перм. нац. исслед. политехн. ун-та, 2014. – 173 с.

ISBN

В учебном пособии изложены базовые понятия, относящиеся к: системам автоматизации и управления, компьютерным технологиям управления, SCADA-пакетам. Также рассмотрены основы решения задач управления в технических системах и базовые принципы работы со SCADA-пакетами. Пособие является учебным материалом для дисциплины «Компьютерные технологии управления в технических системах».

Ориентировано на студентов магистерской подготовки, а также аспирантов, специализирующихся в области распределенных компьютерных информационно-управляющих систем.

УДК

ISBN

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Дисциплина «Компьютерные технологии управления в технических системах» является частью магистерской программы 22040051.68 «Распределенные компьютерные информационно-управляющие системы» по направлению 220400 «Управление в технических системах».

Трудоемкость дисциплины «Компьютерные технологии управления в технических системах» составляет 180 часов, из них лекции – 6 часов, практические занятия – 18 часов, лабораторные занятия – 16 часов и самостоятельная работа – 136 часов.

Учебное пособие призвано помочь в формировании следующих дисциплинарных профессиональных компетенций магистрантов:

– способен выбирать методы решения задач управления в технических системах с использованием компьютерных технологий (ПК-9-1);

– способен использовать современные SCADA-пакеты при проектировании систем автоматизации и управления (ПК-11-1);

– способен проводить компьютерное моделирование при разработке и отладке программного обеспечения на основе компьютерных технологий управления в технических системах (ПК-22-3).

Учебное пособие разбито на два раздела. Первый раздел посвящен общим вопросам организации систем автоматизации и управления, а также вопросам решения задач управления в технических системах с использованием компьютерных технологий. Во втором разделе рассматриваются вопросы, связанные с применением современных SCADA-пакетов при проектировании систем автоматизации и управления, а также проблематика компьютерного моделирования при разработке и отладке программного обеспечения на основе компьютерных технологий управления в технических системах. После каждого раздела приведены вопросы для самоконтроля.

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

АИС – автоматизированная информационная система.
АСНИ – автоматизированная система научных исследований.
АСУ – автоматизированная система управления.
АСУТП – автоматизированная система управления технологическим процессом.
АЦП – аналого-цифровой преобразователь.
Д – датчик.
ИБ – информационная база.
ИМ – исполнительный механизм.
ИП – измерительный преобразователь.
ИУ – исполнительное устройство.
ЛУВС – локальная управляющая вычислительная сеть.
О – оператор.
ОС – операционная система.
ПК – промышленный компьютер.
ПЛИС – программируемая логическая интегральная схема.
ПЛК – программируемый логический контроллер.
ПО – программное обеспечение.
РВ – реальное время.
САИ – система автоматизации испытаний.
САиУ – система автоматизации и управления.
СРВ – система реального времени.
ТЗ – техническое задание.
УВК – управляющий вычислительный комплекс.
УВМ – управляющая вычислительная машина.
УВО – устройство взаимодействия с оператором.
УОИ – устройство отображения информации.
УСО – устройство связи с объектом.
ЦАП – цифро-аналоговый преобразователь.
ЭИУ – электродвигательное исполнительное устройство.
AI – Analog Input (аналоговый ввод).
AO – Analog Output (аналоговый вывод).
CS – Communication System (коммуникационная система)

DI – Discrete Input (дискретный ввод).

DO – Discrete Output (дискретный вывод).

ERP –Enterprise Resource Planning (планирование ресурсов предприятия).

HMI – Human-Machine Interface (человеко-машинный интерфейс).

MES – Manufacturing Execution System (система управления производственными процессами).

MTU – Master Terminal Unit (главное оконечное устройство)

RTU – Remote Terminal Unit (удаленное оконечное устройство)

SCADA – Supervisory Control And Data Acquisition (диспетчерское управление и сбор данных).

1. РЕШЕНИЕ ЗАДАЧ УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ С ИСПОЛЬЗОВАНИЕМ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

1.1. Системы автоматизации и управления (САиУ), компьютерные технологии

1.1.1. Основные понятия, связанные с системами автоматизации и управления

В мире постоянно происходят события, и между этими событиями можно проследить причинно-следственные связи. В результате различных событий происходят всевозможные манипуляции с веществом, энергией и информацией. Учитывая это, можно дать следующее определение.

Физический процесс – это совокупность взаимосвязанных событий, которые приводят к изменениям, перемещениям и накоплению *вещества*, *энергии* и *информации* [22].

Вообще, понятие «процесс» является многозначным, поэтому в определениях это слово дополняется прилагательными, которые его уточняют. Но часто слово «процесс» может применяться и без таких прилагательных, тогда его смысл понимается из контекста. Здесь в дальнейшем под словом «процесс» будет пониматься именно физический процесс.

С древнейших времен люди как разумные существа воспринимали и перерабатывали информацию об окружающих физических процессах. Они также могли оказывать влияние на эти процессы и, кроме прочего, с помощью орудий труда, которые они изготавливали.

В результате развития науки и техники люди стали способны производить все более сложные орудия труда, все более сложные *технические средства*. Благодаря этому у людей появилось больше возможностей для восприятия сложных процессов и для воздействия на эти процессы.

Развитие орудий труда привело к широкому распространению *механизации*, то есть замене ручных орудий труда на более сложные механические орудия – станки, машины и т.д.

Следующей ступенью развития стала автоматизация.

Автоматизация – это частичное или полное замещение функций контроля и управления, выполняемых человеком, аналогичными функциями, выполняемыми техническими средствами.

Другими словами, механизация обеспечивает замещение или усиление физических действий человека, тогда как автоматизация обеспечивает замещение или усиление сенсорной и умственной активности человека. Именно с сен-

сорной и умственной активностью как раз и связаны функции контроля и управления, выполняемые человеком.

В отношении различных процессов могут формулироваться раз-личные *задачи автоматизации*, которые поясняют, какие конкретные функции контроля и управления должны быть замещены.

Человек воздействует на процесс, пытается управлять им, ориентируясь на некоторые показатели, с помощью которых он может оценить, хорошо или плохо он им управляет. Такие показатели, как правило, числовые, называют *критериями управления*. Примеры критериев управления:

– среднее потребление электроэнергии за сутки в данном здании;

– качество производимой бумаги по совокупности показателей (плотность, белизна и т.д.);

– отклонение от заданных режимов работы двигателя.

Задачи достижения заданных критериев управления будем называть *задачами управления*.

Под *целевым процессом* будем понимать физический процесс, применительно к которому осуществляются автоматизация и управление.

Система автоматизации и управления (САиУ) – это техническая система, реализующая получение информации о целевом процессе, передачу, преобразование и хранение этой информации, а также формирование управляющих воздействий на целевой процесс согласно установленным задачам автоматизации и управления.

Надо заметить, что САиУ – техническая система, а также целевой процесс трактуется как физический процесс, поэтому соответствующие задачи управления – это *задачи управления в технических системах*. Здесь надо сказать, что существуют отдельные теоретические и прикладные области, связанные с решением задач управления в организационных, политических, социально-экономических, медико-биологических системах. То есть область проблем управления шире области управления, связанной с САиУ и задачами управления в технических систем. Но в рамках данного учебного пособия и соответствующей учебной дисциплины мы будем рассматривать именно задачи управления в технических системах.

Важно отметить, что человек может входить в состав САиУ, являться ее частью, например, в качестве оператора системы, контролирующего основные процессы и оказывающего управляющие воздействия на компоненты системы

Человек-оператор – это человек, осуществляющий трудовую деятельность, основу которой составляет взаимодействие с объектом воздействия, машиной и средой на рабочем месте при использовании информационной модели и органов управления [27].

Здесь *информационная модель* – это организованное в соответствии с определенной системой правил отображение состояния предмета труда, технической системы, внешней среды и способов воздействия на них [27].

Схематично взаимодействие САиУ и целевого процесса показано на рис. 1.

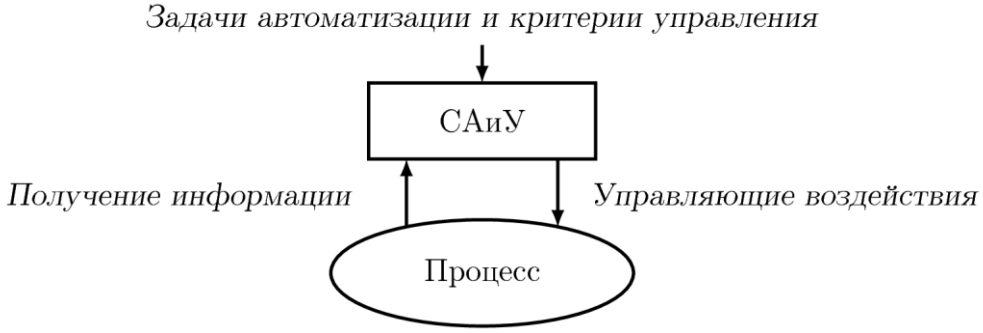


Рис. 1. Взаимодействие САиУ и целевого процесса

Целевой процесс можно представить как процесс переработки вещества, энергии и информации (рис. 2).

В некоторых случаях ввод-вывод одной из трех указанных сущностей (вещества, энергии, информации) может отсутствовать, точнее данная сущность может не рассматриваться в заданном контексте.

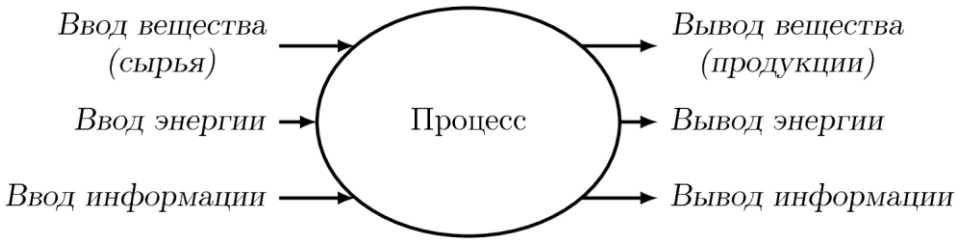


Рис. 2. Переработка вещества, энергии, информации в ходе выполнения процесса

Рассмотрим примеры процессов и САиУ.

Пример 1. *Производство трехслойного гофрированного картона* [10]. На вход технологической линии подаются три непрерывных слоя картона (ввод вещества). В ходе процесса выполняются: нанесение клеящего слоя; склеивание; сушка; поперечная резка. На выходе формируются листы гофрированного картона заданного формата (выход вещества). Для выполнения указанных действий, в частности проката полос картона, требуется ввод энергии (в частности, питание электродвигателей). Для правильного позиционирования картона и правильной резки используются датчики положения. Информация, поступающая от этих датчиков, формирует вывод информации. В качестве ввода информации можно рассматривать команды управления, поступающие на технологическую линию. Например, эти команды определяют скорость проката полос картона, формы листов при резке. Также в качестве ввода инфор-

мации следует рассматривать различные технологические инструкции, требования к ведению процесса и т.д. Кроме того, в качестве вывода информации важно рассматривать также и различные отчеты и прочую документацию, формируемую в ходе данного технологического процесса. В данном примере вывод энергии как таковой является несущественным.

Пример 2. *Автоматизированная система управления стендовыми огневыми испытаниями ракетных двигателей малой тяги* [31]. Ракетные двигатели испытывают на специальных стендах, где имитируются условия эксплуатации, проверяются различные характеристики, чтобы полученные результаты учитывать при разработке этих двигателей. Здесь как такового ввода и вывода вещества не выполняется. Но есть поступление топлива и его сжигание. В данном случае это поступающее топливо можно считать вводом энергии, а повышение температуры и прочие последствия, возникающие в результате сгорания топлива, можно считать проявлением вывода энергии. В данном примере наиболее важным является ввод и вывод информации. Информация, которая выводится в ходе испытаний, сигнализирует о характеристиках двигателей, об особенностях их работы и т.д. Наиболее важная часть этой информации воспринимается в рамках САиУ и потом подвергается обработке и анализу. Часть информации, конечно, может не учитываться и не выявляться имеющимися датчиками. В качестве ввода информации можно рассматривать различные управляющие воздействия, например, направленные на изменения режимов работы двигателей. Также частью ввода информации являются различные предписания, инструкции и прочие документы, регламентирующие порядок испытаний.

Пример 3. *Автоматизированный эколого-аналитический мониторинг источников загрязнения поверхностных вод* [3]. Специализированное речное судно, оснащенное необходимым оборудованием выполняет анализ параметров водной среды. В частности, здесь осуществляется задача автоматизация процесса сбора и обработки информации об окружающей среде, в данном случае водной среды. В данном случае САиУ реализует только получение информации о процессе развития водной среды. Данный процесс характеризуется вводом вещества, в том числе и различных загрязняющих веществ, а также выводом вещества за пределы рассматриваемого пространства. Вводом и выводом энергии в рамках данного процесса можно пренебречь. Вывод информации реализуется в виде различных параметров анализируемой водной среды. При этом ввод информации в целевой процесс не проявлен.

1.1.2. Классификация САиУ по видам процессов

Понятие САиУ является достаточно широким, определяя обширное множество систем. Представляется важным данное множество хотя бы немного детализировать. Для этого выделим в нем подмножества – классы.

Можно пытаться разными способами сделать такое разделение на классы. Например, можно классифицировать САиУ по структуре, по сложности, по типам вычислительных устройствах, по особенностям информационных потоков, по особенностям того или иного вида типового обеспечения, по видам задач автоматизации, по критериям управления и т.д. Примеры классификаций САиУ можно например, найти в [6].

Однако при рассмотрении САиУ с общих позиций наиболее целесообразной выглядит классификация САиУ по видам целевых процессов и особенностям взаимодействия с ними. С этой точки зрения для начала рассмотрим основные классы процессов.

Для начала все процессы разделим на два класса:

– процессы, на которые предполагается воздействовать (*управляемые процессы*);

– процессы, на которые не предполагается воздействовать (*контролируемые процессы*).

Согласно определению физического процесса выделяются три основные сущности: вещество, энергия и информация.

Часто бывает, что проявления вещества и энергии сложно разделить, например, топливо для двигателей можно рассматривать и как вещество, и как носитель энергии. С другой стороны, проявление такой сущности, как информация, более заметно на фоне проявлений вещества и энергии. Поэтому среди управляемых процессов будем отдельно рассматривать три класса процессов:

– процессы, в которых преобладающее значение имеют сущности вещества или энергии;

– процессы, в которых преобладающее значение имеют сущность информации;

– процессы, в которых ни одна из сущностей не имеет преобладающего значения (*комплексные процессы*).

Согласно определению физического процесса можно выделить три аспекта, связанных с веществом, энергией и информацией: изменение, перемещение, накопление. Поэтому в первых двух из вышеуказанных классов можно выделить еще три класса:

– процессы, в которых преобладающее значение имеет аспект *изменения* сущности (имеющей преобладающее значение);

– процессы, в которых преобладающее значение имеет аспект *перемещения* сущности (имеющей преобладающее значение);

– процессы, в которых преобладающее значение имеет аспект *накопления* сущности (имеющей преобладающее значение).

Каждому из выделенных классов процессов поставим в соответствие класс САиУ. Более подробно эти классы САиУ рассмотрим отдельно, а сейчас предложенную классификацию процессов и соответствующих им САиУ схематично представим на рис. 3, при этом на данном рисунке используется аббревиатура АСУ, обозначающая автоматизированную систему управления.

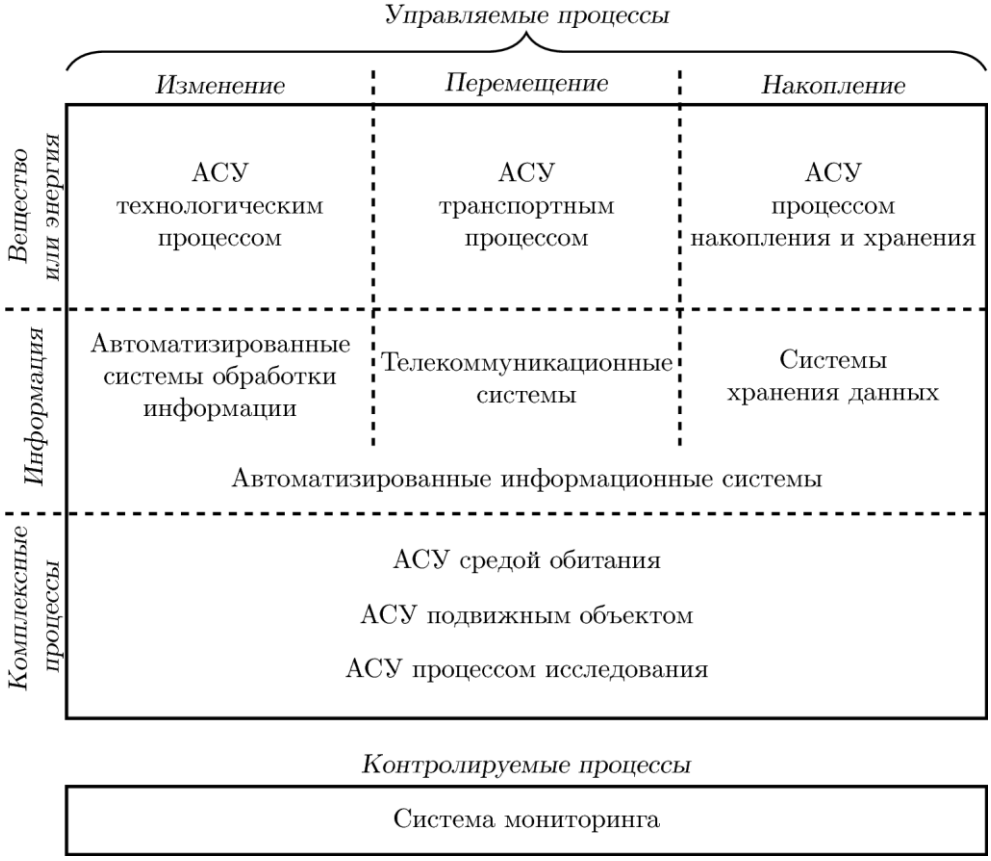


Рис. 3. Основные классы процессов и САиУ

1.1.2.1.Автоматизированная система управления технологическим процессом (АСУТП)

Технологический процесс – это физический процесс, в результате которого выполняется преобразование исходных составляющих в определенную продукцию согласно заданной технологии.

Автоматизированная система управления технологическим процессом (АСУТП) – это САиУ с целевым процессом в виде технологического процесса.

Общая задача автоматизации для всех АСУТП – обеспечение заданного качества продукции на основе успешного протекания целевого технологического процесса.

Выделение разных видов технологических процессов приводит к формированию подклассов АСУТП, характерных для определенных отраслей экономики:

– АСУТП в машиностроении и металлообработке;
– АСУТП в металлургии;
– АСУТП в пищевой промышленности;
– АСУТП в строительстве;
– АСУТП в энергетике;
– АСУТП в нефтехимической промышленности;
– АСУТП в сельском хозяйстве;
– АСУТП в лесном хозяйстве;
– и т. д.

Каждый из указанных подклассов обладает свой спецификой, отображаемой в характерных для данного подкласса задачах автоматизации и критериях управления.

1.1.2.2. Автоматизированная система управления транспортным процессом

Транспортный процесс – это процесс перемещения грузов или пассажиров по заданным маршрутам согласно заданным расписаниям и правилам.

Автоматизированная система управления транспортным процессом (АСУ транспортным процессом) – это САиУ с целевым процессом в виде транспортного процесса.

Общая задача автоматизации для всех АСУ транспортным процессом – обеспечение доставки грузов или пассажиров по заданным маршрутам согласно заданным расписаниям и правилам при условии сохранности грузов и безопасности пассажиров.

Классификация транспортных процессов по видам транспортных средств приводит к формированию подклассов АСУ транспортным процессом:

– на железнодорожном транспорте;
– на автомобильном транспорте;
– на воздушном транспорте;
– на водном транспорте;
– на трубопроводном транспорте;
– и т. д.

Каждый из указанных подклассов обладает свой спецификой, отображаемой в характерных для данного подкласса задачах автоматизации и критериях управления.

1.1.2.3.Автоматизированная система управления процессом накопления и хранения

Процесс накопления и хранения – в данном контексте это процесс, состоящий в накоплении вещества (в частности, изделий) или энергии (в частности, носителей энергии), а также их хранения с целью сохранения неизменными их основных свойств.

Автоматизированная система управления процессом накопления и хранения (АСУ процессом накопления и хранения) – это САиУ с целевым процессом в виде процесса накопления и хранения.

Общая задача автоматизации для всех АСУ процессом накопления и хранения – поддержание требуемых условий хранения, а также обеспечение процессов поступления и выдачи хранимых материалов.

Классификация процессов накопления и хранения по особенностям условий хранения приводит к формированию подклассов АСУ процессом накопления и хранения, например:

– АСУ холодильным складом;
– АСУ складом штучных грузов без температурного режима;
– АСУ музейным хранилищем;
– АСУ библиотечным хранилищем;
– АСУ нефтехранилищем;
– АСУ газохранилищем;
– АСУ топливным хранилищем;
– АСУ зернохранилищем;
– АСУ овощехранилищем;
– и т. д.

Каждый из указанных подклассов обладает свой спецификой, отображаемой в характерных для данного подкласса задачах автоматизации и критериях управления.

1.1.2.4.Автоматизированная информационная система

Информационный процесс – это процесс обработки, перемещения и накопления информации.

Автоматизированная информационная система (АИС) – это САиУ с целевым процессом в виде информационного процесса.

Общая свойство всех АИС состоит в том, что в отличие от ранее рассмотренных классов САиУ, в случае АИС *непосредственное* взаимодействие

с внешним миром, например, с помощью датчиков и исполнительных устройств, сводится к минимуму, и часто ограничивается лишь непосредственным взаимодействием с человеком.

Но это не означает, что АИС не получает информацию от внешнего мира. Эта информация может поступать в АИС от других САиУ по тем или иным информационным каналам. Такое взаимодействие с внешним миром уже является опосредованным (а не непосредственным).

Здесь мы подходим к проблеме выделения границ рассматриваемой САиУ.

Например, если рассмотреть САиУ для сложной транспортной системы, то в целом это будет АСУ транспортным процессом. Но в составе этой САиУ может быть выделена АИС, занимающаяся сложной обработкой многочисленной поступающей информацией, выработкой расписаний, анализом маршрутов и т. д. И здесь возникает вопрос, считать ли эту АИС самостоятельной САиУ или только подсистемой АСУ транспортным процессом?

Ответ на этот вопрос зависит от контекста, от детализации, от целей анализа и рассмотрения системы. В каких-то случаях важно выделить АИС, и например, проектировать ее отдельно, формируя отдельное задание на проектирование. Например, когда надо модернизировать имеющуюся АИС. В каких-то случаях задача проектирования может решаться комплексно, в ходе проектирования всей САиУ, в состав которой входит данная АИС.

Информация может обрабатываться, передаваться, накапливаться. В случае существенного преобладания одного из этих видов действий получается, что АИС сводится к одному из следующих трех подклассов САиУ (см. рис. 3):

– автоматизированная система обработки информации;

– телекоммуникационная система;

– система хранения данных.

В рамках данного учебного пособия и соответствующей учебной дисциплины важно рассмотреть наиболее общие и сложные случаи применения технических средств автоматизации и управления. В случае же АИС мы, как правило, имеем далеко не полный набор различных видов этих технических средств. В частности, как было указано выше, в АИС обычно отсутствуют датчики и исполнительные устройства, то есть технические средства взаимодействия с процессом, ведь АИС имеет дело в первую очередь с информацией. Поэтому в дальнейшем мы не будем останавливаться на специфических особенностях АИС и особенностях процесса разработки таких систем, а будем обращаться к наиболее общим случаям САиУ.

1.1.2.5.Автоматизированная система управления средой обитания

Существуют классы комплексных процессов, где сложно выделить преобладание вещества и энергии или информации. Рассмотрим один из примеров таких процессов.

Процесс обеспечения среды обитания – это процесс, состоящий:

– в изменении того или иного вещества для приведения его к виду, необходимому для поддержания среды обитания, например, очистка воды и воздуха;

– в перемещении вещества к нужным местам среды обитания, например, подведение питьевой воды;

– в накоплении и хранении вещества, например, сохранение запасов продовольствия и питьевой воды;

– в получении энергии, необходимой для обеспечения среды обитания, например, получение электроэнергии;

– в выводе этой энергии в другой форме, например, рассеиваемого тепла;

– в восприятии информации, поступающей изнутри и снаружи среды обитания, что необходимо для правильной адаптации среды обитания к изменяющимся условиям;

– в выработке сигналов о состоянии среды обитания, например, сигнализация об изменениях параметров среды.

Автоматизированная систему управления средой обитания (АСУ средой обитания) – это САиУ с целевым процессом в виде процесса обеспечения среды обитания.

Общая задача автоматизации для всех АСУ средой обитания – поддержание требуемых условий существования среды обитания, в частности обеспечение всеми необходимыми ресурсами, обеспечение безопасности, энергоэкономичности и т. д.

Классификация процессов обеспечения среды обитания по виду среды обитания приводит к формированию подклассов АСУ средой обитания, например:

– система автоматизации здания;

– система «умный дом»;

– АСУ городской инфраструктурой;

– АСУ садово-парковым комплексом;

– АСУ океанариумом;

– и т. д.

Каждый из указанных подклассов обладает свой спецификой, отображаемой в характерных для данного подкласса задачах автоматизации и критериях управления.

Как уже было раньше отмечено, при выделении границ САиУ важное значение имеет контекст и цель рассмотрения системы, и был приведен соответствующий пример. В случае АСУ средой обитания также можно привести подобный пример. Так, в составе системы автоматизации здания можно выделить технологические процессы, например, процесс очистки воды, процесс очистки воздуха. Для каждого из подобных технологических процессов может быть создана своя АСУТП, и их можно рассматривать как отдельно, так и в качестве подсистем общей системы автоматизации здания.

1.1.2.6. Автоматизированная систему управления подвижным объектом

Рассмотрим еще пример комплексного процесса.

Процесс функционирования подвижного объекта – это процесс, состоящий:

– в перемещении вещества в виде самого подвижного объекта, также возможных грузов и пассажиров;

– в получении энергии, необходимой для движения объекта;

– в выводе этой энергии в другой форме, например, в виде кинетической или потенциальной энергии;

– в восприятии информации об окружающей среде для правильного перемещения;

– в выработке сигналов в окружающую среду, например, о своей местоположении.

Автоматизированная систему управления подвижным объектом (АСУ подвижным объектом) – это САиУ с целевым процессом в виде процесса функционирования подвижного объекта.

Классификация процессов функционирования подвижного объекта по виду подвижного объекта приводит к формированию подклассов АСУ подвижным объектом:

– в виде автомобиля;

– в виде морского судна;

– в виде самолета;

– в виде беспилотного самолета;

– в виде робота;

– и т. д.

При обозначении данного класса САиУ используется слово «автоматизированная» (система управления), что предполагает наличие человека в составе данной САиУ. Это справедливо, например, в случае АСУ подвижным объектом в виде самолета, где в составе этой САиУ важную роль играет экипаж самолета.

Однако в случае подвижного объекта в виде беспилотного самолета или робота может оказаться так, что человек не входит в состав данной САиУ. Например, можно рассмотреть реализацию полностью автономных беспилотных самолетов и роботов. Тогда надо говорить об *автоматической* системе управления подвижным объектом.

Но для удобства и краткости будем здесь и в дальнейшем предполагать, что *автоматизированная* система управления в частном случае может сводиться к *автоматической* системе управления (когда роль человека в составе этой системе минимальна или отсутствует).

Может оказаться, что в составе большой АСУ транспортным процессом выделяется множество АСУ отдельными подвижными объектами. Также можно также привести пример роботизированного завода, где множество сборочных операций выполняют роботы. Здесь можно рассматривать отдельные АСУ подвижных объектов в виде роботов, но все эти роботы являются частью большой АСУТП, предназначенной для обеспечения производства некоторой продукции.

1.1.2.7.АСУ процессом исследования

Рассмотрим следующий пример комплексного процесса.

Процесс исследования – это процесс, состоящий:

– в получении информации об объекте исследования;

– в выводе управляющей информации к объекту исследования, например, для переключения режимов работы объекта, условий эксперимента и т. д.;

– во вводе различных форм вещества и энергии, необходимых для проведения исследования;

– в выводе различных форм вещества и энергии, производимых в результате исследования.

Автоматизированная система управления процессом исследования (АСУ процессом исследования) – это САиУ с целевым процессом в виде процесса исследования.

Можно выделить два основных подкласса АСУ процессом исследования:

– автоматизированная система научных исследований;

– система автоматизации испытаний.

Автоматизированная система научных исследований (АСНИ) – это САиУ, обеспечивающая автоматизацию научных экспериментов, а также позволяющая моделировать исследуемые объекты или процессы, когда их изучение другими средствами невозможно или слишком затруднено.

Система автоматизации испытаний (САИ) – это САиУ, обеспечивающая автоматизацию испытаний готовых экземпляров продукции или опытных образцов такой продукции.

При этом важно учитывать, что САИ может являться подсистемой большой АСУТП. Такая САИ, например, обеспечивает проведение испытаний производимой продукции, и на основе этих испытаний принимаются решения о качестве продукции, может корректироваться режим технологического процесса и т. д.

1.1.2.8. Система мониторинга

До этого рассматривались только управляемые процессы, то есть процессы, на которые САиУ может воздействовать. Но существуют также процессы (*контролируемые процессы*), на которые влиять либо нет возможности в данный момент, либо нет самой задачи непосредственного влияния на эти процессы.

Система мониторинга – это САиУ с целевым процессом в виде контролируемого процесса.

Общая задача автоматизации для всех систем мониторинга – это получение необходимой информации о контролируемой процессе как в форме, удобной для непосредственного восприятия человеком, так и в форме, удобной для автоматической обработки этой информации.

Здесь важно указать на то, что система мониторинга часто может быть составной частью более обширной САиУ, являться подсистемой этой САиУ. Бывает, что такую подсистему требуется рассматривать отдельно от общей САиУ, например, отдельно ее проектировать.

1.1.3. Виды обеспечения САиУ

1.1.3.1. Техническое обеспечение

Техническое обеспечение – это комплекс технических средств (hardware), применяемых для функционирования САиУ, например компьютеры, контроллеры, датчики, исполнительные устройства и т. д.

Часто вместо термина «техническое обеспечение» применяют термин «аппаратное обеспечение». Этот вид обеспечения будет подробно рассматриваться в дальнейшем, так как в рамках данного пособия рассматриваются компьютерные технологии в *технических* системах. По сути, все существующие технические средства автоматизации и управления образуют множество, из элементов которого можно формировать техническое обеспечение конкретной САиУ.

1.1.3.2.Программное обеспечение

Программное обеспечение (ПО) – комплекс программ (software), применяемых для функционирования САиУ.

При этом ПО делиться на два вида:

– общее ПО (программы для широкого круга задач, например, операционные системы, текстовые редакторы, архиваторы и т. д.);

– специальное ПО (применяется только для реализации данной САиУ, например: программа, реализующая интерфейс с оператором; программа, формирующая текстовые отчеты и т. д.).

Этот вид обеспечения также будет рассматриваться в дальнейшем более подробно, так как функционирование сложных технических систем, основанных на вычислениях (например, использующих компьютеры или контроллеры) практически невозможно рассматривать без обращения внимания на программное обеспечение.

Часто бывает так, что одни и те же функции могут выполняться как аппаратными, так и программными средствами (или сложной их комбинацией). Поэтому при выборе технических средств важно понимать возможность их замещения программными средствами, и наоборот.

Таким образом, при рассмотрении вопрос управления в технических системах важно уделять внимание совокупности аппаратно-программных средств, то есть рассматривать как техническое, так и программное обеспечение САиУ.

Другие виды обеспечения САиУ рассмотрим здесь более кратко.

1.1.3.3.Математическое обеспечение

Математическое обеспечение – совокупность математических методов, моделей и алгоритмов, используемая при создании САиУ.

Примеры компонентов математического обеспечения:

– методы идентификации управляемого объекта;

– методы теории надежности;

– методы статистического анализа результатов измерений;

– математическая модель управляемого процесса;

– математическая модель поведения человека-оператора;

– алгоритм ПИД-регулирования;

– алгоритм адаптивного управления подсистемой управляемого объекта;

– алгоритм управления, использующий аппарат нечеткой логики;

– цифровая фильтрация поступающих данных измерения;

– обработка измерительной информации с целью компенсации погрешности измерения, например, на основе вычисления «плавающего среднего» по соседним результатам измерений.

1.1.3.4.Информационное обеспечение

Информационное обеспечение – совокупность реализованных решений по объемам, размещению и формам организации информации, циркулирующей в САиУ при ее функционировании.

В составе САиУ имеется информационная база (ИБ), в которой содержится вся необходимая информация для функционирования САиУ. Различают внутримашинную часть ИБ (файлы, базы данных) и внемашинную часть ИБ (например, текстовые документы, и т. д.).

В простом случае внутримашинная часть ИБ организована в виде файлов.

В сложной САиУ внутримашинная часть ИБ может быть организована в виде *базы данных*, которая определяется как совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, и независимая от прикладных программ.

Информационное обеспечение определяет:

– объемы, размещение и формы организации ИБ;

– структуру и содержание нормативно-справочной информации для описания объектов и их свойств (в том числе, программные системы справочной информации);

– структуру и содержание классификаторов (классификация объектов);

– структуру и содержание унифицированных документов (например, отчетных ведомостей, типовых бланков и т. д.);

– протоколы обмена данными между различными устройствами системы.

Как правило, чем сложнее САиУ, тем более существенную роль играет информационное обеспечение.

1.1.3.5.Лингвистическое обеспечение

Лингвистическое обеспечение – совокупность языковых средств для формализации естественного языка, построения и сочетания информационных единиц при общении оперативного персонала со средствами вычислительной техники при функционировании САиУ.

Примеры компонентов лингвистического обеспечения:

– условные обозначения операторского интерфейса (мнемосхемы, текстовые сообщения, цветовые индикаторы);

– сочетание элементов графического интерфейса (меню, диалоговые окна, кнопки, строки ввода, строки состояния и т. д.);

– диалоговые средства для взаимодействия с информационной базой (языки запросов и т. д.);

– средства голосового общения на основе распознавания речи и синтезатора речи;

– языки сценариев (например, короткие программы (скрипты), набираемые оператором для формирования сложной команды).

Пример программы на языке сценариев:

```
ВКЛ нагреватель1
ОТКР вентиль2
ЗАДЕРЖКА 10сек
ЗАКР вентиль2
ЕСЛИ температура3>100 ТО ВКЛ вентилятор1
```

Важно не путать язык сценариев с языками программирования, применяемыми для разработки программного обеспечения в процессе *проектирования* САиУ. Такие языки программирования не входят в состав лингвистического обеспечения. А вот если какой-либо формализованный язык, например язык сценариев, используется человеком-оператором в процессе *функционирования* САиУ, то его можно считать компонентом лингвистического обеспечения.

1.1.3.6.Организационное обеспечение

Организационное обеспечение – совокупность документов, регламентирующих деятельность персонала САиУ в условиях ее функционирования.

Примеры компонентов организационного обеспечения:

– руководство оператора;

– инструкция по эксплуатации;

– инструкция по проведению периодических проверок оборудования.

1.1.3.7.Правовое обеспечение

Правовое обеспечение – совокупность правовых норм, регламентирующих правовые отношения при функционировании САиУ и юридический статус результатов ее функционирования.

1.1.4. Основные виды технических средств автоматизации и управления

На данном этапе необходимо выполнить классификацию технических средств САиУ по функциональному назначению. Но перед этим введем некоторые условные обозначения, которые в дальнейшем будут применяться в схемах.

Прямоугольники с двойными рамками будут обозначать совмещение многих прямоугольников одного вида, а «двойные» стрелки обозначают множество связей. Примеры, поясняющие такие обозначения, показаны на рис. 4. Также используется обозначение в виде двух параллельных линий, с которыми могут стыковаться стрелки, и в итоге это позволяет более компактно обозначить соединение каждого с каждым (см. рис. 4, *е*, 4, *ж*).

При этом понятно, что расшифровка этих обозначений может быть неоднозначной, как, например, в случае рис. 4, *г*, 4, *д*. Однако такая неоднозначность либо является несущественной на принятом уровне обобщений, либо устраняется с помощью информации об особенностях соответствующих компонентов конкретной схемы.

Теперь обратимся к классификации технических средств автоматизации и управления по функциональному назначению. Эта классификация тесно связана с дальнейшей детализацией схемы, представленной на рис. 1. Эта детализация выполняется путем некоторого прояснения внутреннего содержимого прямоугольника САиУ, приведенного на рис 1. В итоге получается обобщенная свернутая схема САиУ, представленная на рис. 5.

На рис. 3 используются следующие обозначения:

– УВМ – управляющая вычислительная машина (например, промышленный компьютер или программируемый логический контроллер);

– УСО – устройство связи с объектом;

– УВО – устройство взаимодействия с оператором;

– УВК – управляющий вычислительный комплекс;

– Д – датчик;

– ИУ – исполнительное устройство;

– О – оператор (диспетчер).

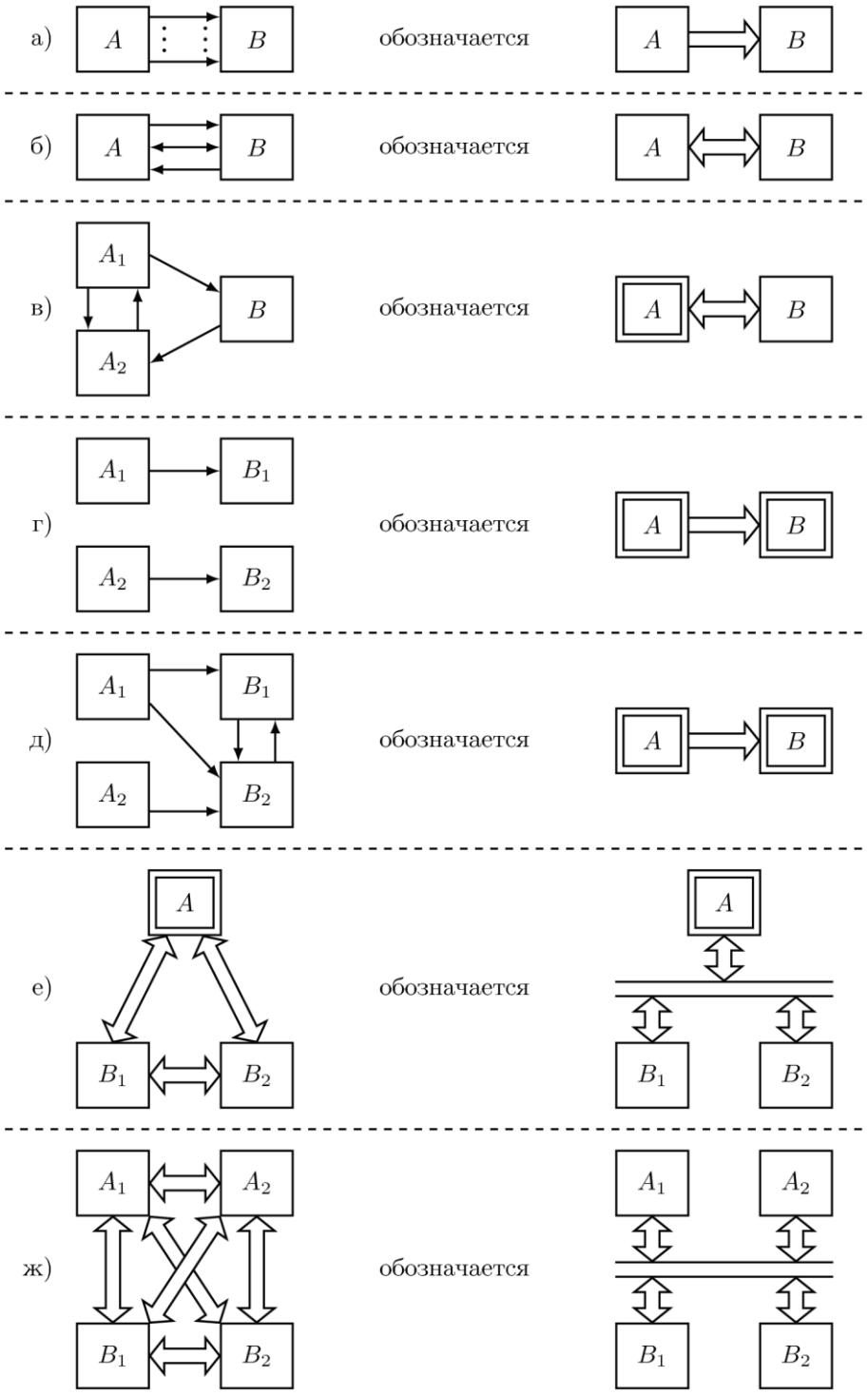


Рис. 4. Примеры принимаемых условных обозначений

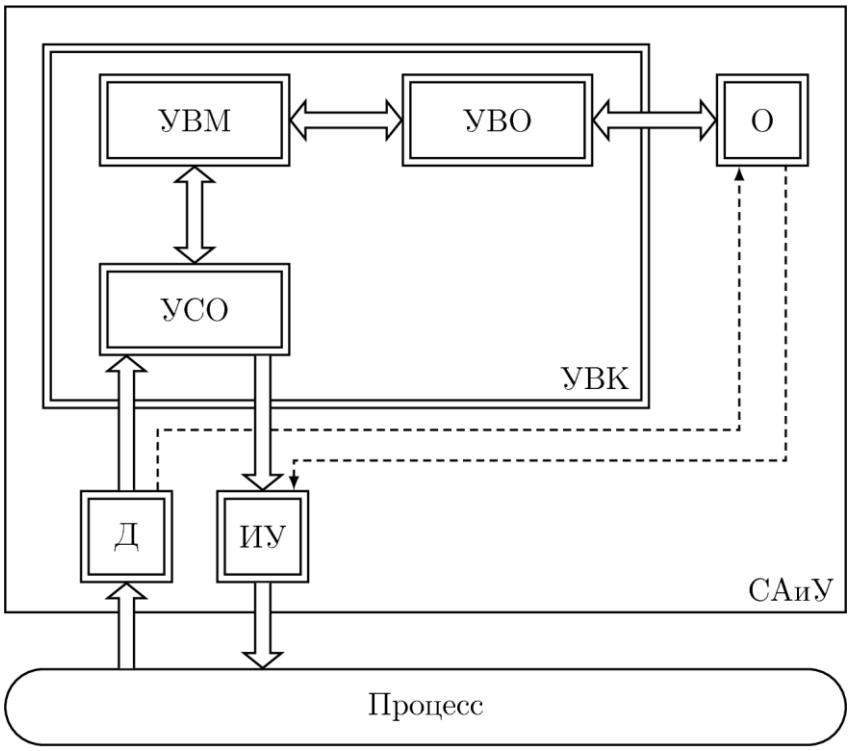


Рис. 5. Обобщенная свернутая схема САиУ

Элементам обобщенной схемы (УВМ, УСО, УВО, Д, ИУ) соответствуют различные классы технических средств.

В составе САиУ можно выделить множество УВМ, которые посредством множества УСО взаимодействуют с датчиками (Д) и ИУ. В свою очередь датчики и ИУ взаимодействуют с целевым процессом. При этом УВМ также взаимодействуют с оператором (О) посредством УВО. В некоторых случаях у оператора имеется возможность непосредственного контроля за показаниями датчиков и непосредственного воздействия на ИУ (показано пунктирными линиями). В дальнейшем для упрощения схем эти пунктирные линии обычно не будут отображены, однако надо понимать, что такие взаимодействия оператора с датчиками и ИУ могут происходить в определенных случаях.

Совокупность УВМ, УСО, УВО образуют УВК, которых также может быть несколько. И эти предполагаемые несколько УВК свернуты в один прямоугольник с двойными линиями, поэтому в названии этой схемы есть слово «свернутая».

Также важную роль в САиУ играют линии связи, которые здесь показаны стрелками, и эти стрелки тоже «свернуты».

Следует отметить, что данная схема представляет наиболее общий случай реализации САиУ. В конкретных САиУ те или иные компоненты данной схемы могут отсутствовать.

Таким образом, на данном этапе уже только из этой обобщенной свернутой схемы можно понять основное назначение классов технических средств.

Основные вычислительные функции САиУ сосредоточиваются в УВМ. Для непосредственного взаимодействия с целевым процессом используются датчики и ИУ. Для связи УВМ с датчиками и ИУ применяются УСО. Для связи УВМ с оператором применяются УВО.

Предполагается, что УВК в своей основе имеет одну УВМ и может иметь несколько УСО и УВО (рис. 6).

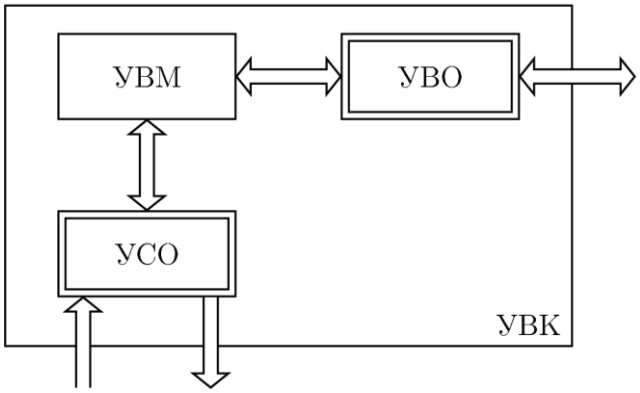


Рис. 6. Обобщенная схема одного УВК

Бывает, что реализуют дублирование технических средств, в том числе и УВМ, для повышения надежности. В этих случаях УВК может содержать несколько физических УВМ, но при этом их можно считать одной логической УВМ, так как несколько физических УВМ используются для дублирования и выполняют одни и те же функции. Кроме того, одной логической УВМ можно считать несколько УВМ, объединенных в единую систему для совместного решения сложных задач путем их распараллеливания или распределения (подобно тому, как это делается в многопроцессорной УВМ).

Естественно, что на рис. 6 представлена обобщенная схема УВК, и в специальных случаях (когда нет взаимодействия с процессом или с оператором) в составе УВК могут отсутствовать УСО или УВО, или даже УВК может состоят только из одной УВМ (рис. 7).

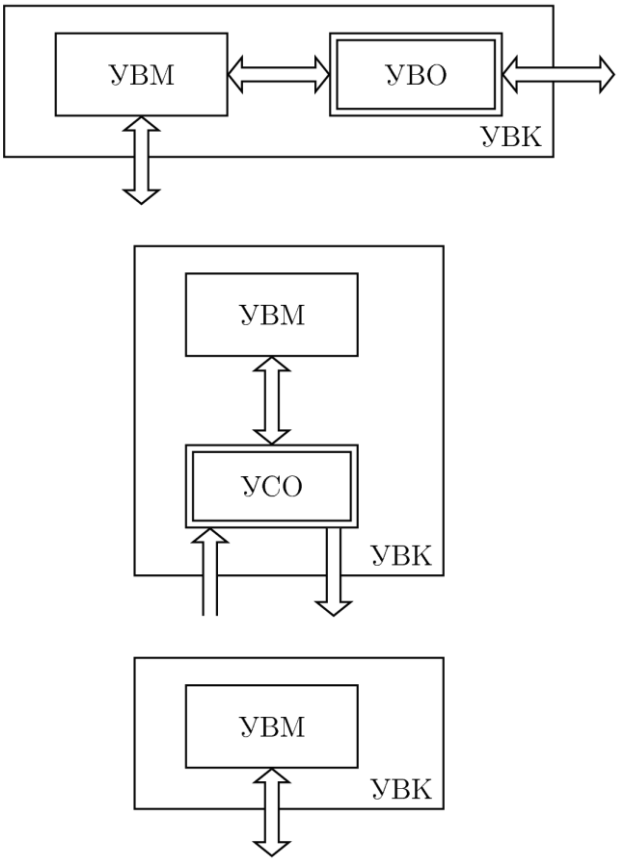


Рис. 7. Специальные варианты УВК

1.1.4.1. Управляющие вычислительные машины (УВМ)

УВМ занимают центральное место в составе САиУ, так как они несут на себе основную нагрузку по обработке информации, с которой имеет дело САиУ.

Слово «вычислительные» означает, что УВМ выполняет вычисления на основе определенных алгоритмов, которые реализуются с помощью тех или иных языков программирования и средств программирования. Поэтому важнейшим компонентом УВМ является программное обеспечение, которое само по себе не относится к техническим средствам (техническому обеспечению), но представляет собой важнейший компонент компьютерных технологий, который будет отдельно подробно рассмотрен в дальнейшем.

Перед тем, как дать определение УВМ, рассмотрим некоторые общие вопросы.

Вычислительное устройство – техническое устройство, которое способно выполнять вычисление согласно заданному алгоритму на основе входных данных и формировать результат вычисления на основе выходных данных этого алгоритма.

При подключении к вычислительному устройству внешней памяти, средств ввода-вывода, другого дополнительного оборудования можно гово-

рить о построении *вычислительной системы* (системы взаимодействующих компонентов для выполнения вычислений). Примерами вычислительных систем являются: компьютеры, программируемые логические контроллеры, мобильные телефоны, смартфоны, суперкомпьютеры.

Часто возникает потребность в изменении алгоритма вычислений.

Появление программируемых вычислительных устройств дало возможность изменять алгоритм вычисления без изменения аппаратного обеспечения. На вход программируемого вычислительного устройства можно подавать данные особого рода, представляющие собой программу, которую можно рассматривать как описание алгоритма вычисления, который должен быть выполнен на данном устройстве. Это может выполняться в особом режиме функционирования этого устройства, в режиме его программирования. По окончанию программирования, данное устройство переводится в обычный режим функционирования и может начинать выполнять заложенную в него новую программу (новый алгоритм вычисления). В случае сложных вычислительных устройств и систем бывает сложно разделить режим программирования и режим обычного функционирования, например, в случае интерактивного взаимодействия с человеком, когда он может менять параметры программы, изменять алгоритм вычислений без остановки работы основной программы.

Совокупность программ, заложенных в вычислительное устройство или систему, образуют *программное обеспечение*.

Можно считать, что УВМ – это вычислительная система, которая используется в составе САиУ.

УВМ обменивается информацией с целевым процессом с помощью датчиков (Д) и ИУ через посредничество УСО, а с человеком-оператором – на основе УВО.

1.1.4.2.Датчики и измерительные преобразователи (ИП)

Датчик – чувствительный элемент, который воспринимает контролируемую физическую величину и преобразует его в сигнал, удобный для последующего измерения, передачи, преобразования, регистрации.

Измерительный прибор – устройство, преобразующее контролируемый параметр в сигнал, удобный для непосредственного восприятия человеком.

Измерительный преобразователь (ИП) – устройство, преобразующее контролируемый параметр в сигнал, предназначенный для последующей передачи, преобразования, регистрации.

Объединение нескольких ИП также является ИП.

В общем случае ИП состоит из двух основных частей: первичного ИП и нормирующего ИП (см. рис. 8).

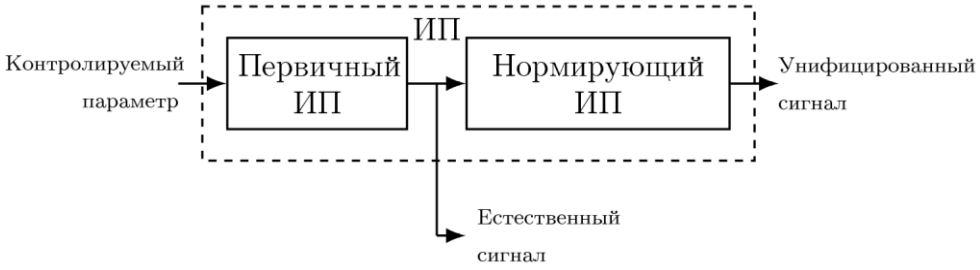


Рис. 8. Обобщенная схема ИП

Первичный ИП содержит датчик и в общем случае формирует сигнал, который называется *естественным*, на основе преобразования физической величины, характеризующей контролируемый параметр.

На рис. 9 представлены примеры первичных ИП.

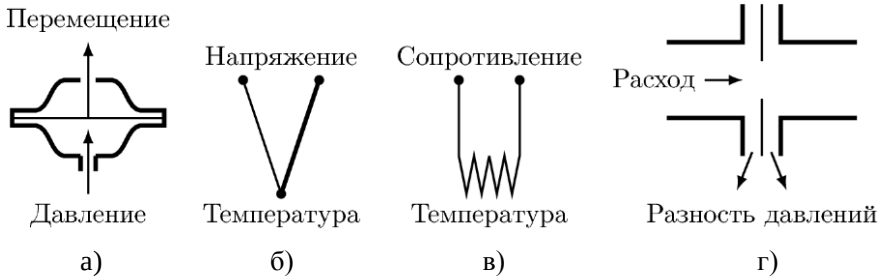


Рис. 9. Примеры первичных ИП
(а – мембранный; б – термопара; в – термосопротивление; г – дроссельный)

Нормирующий ИП служит для получения *унифицированного* сигнала на основе естественного сигнала от первичного ИП. В том числе, нормирующий ИП может формировать *цифровой* сигнал в качестве унифицированного.

Примеры унифицированных сигналов:

– сигналы постоянного тока: 4 – 20 мА, 0 – 10 В;

– сигналы переменного тока: 0 – 2 В, 4 – 8 кГц;

– давление воздуха: 20 – 100 кПа.

В общем случае ИП может, кроме прочего, выполнять масштабирование сигнала, его линеаризацию, фильтрацию помех и другие функции.

Нормирующий ИП может быть реализован отдельно, или объединен конструктивно с первичным ИП в одно устройство

В отдельный класс можно отнести *бинарные датчики* – устройства формирующие сигнал, который может принимать только одно из двух состояний, например «замкнут» или «разомкнут».

Примеры использования бинарных датчиков: контроль открытости двери (окна); контроль открытости заслонки (задвижки); контроль прохождения очередной заготовки на конвейере, подсчет количества людей, прошедших через турникет и т. д.

В простых случаях бинарные датчики, выполняются в виде концевых выключателей и переключателей, коммутирующих электрические сигналы.

Можно указать также и другие способы реализации бинарных датчиков:

– ртутные выключатели (в запаянной стеклянной трубке находится ртуть, которая при изменении наклона трубки под действием силы тяжести переливается в другое положение и, тем самым, замыкает контакты);

- герконы (контакты замыкаются под действием постоянного магнита, то есть механического контакта с датчиком не происходит);

- фотоэлектрические датчики (генератор светового луча и светочувствительный элемент);

- ультразвуковые и инфракрасные датчики для обнаружения объектов на расстоянии от нескольких сантиметров до нескольких метров (работают либо по принципу прерывания сигнала, когда источник и приемник находятся в разных местах, либо по принципу отражения сигнала, когда источник и приемник находятся в одном приборе).

Бинарные датчики также могут использоваться как пороговые датчики (для обнаружения момента, когда аналоговая величина достигает определенного значения и подачи аварийного сигнала).

Также бинарные датчики могут использоваться как индикаторы уровня для определения момента, когда жидкость или сыпучее вещество достигает определенного уровня.

Также см. п. 1.3.4.2, где рассматриваются характеристики ИП, которые необходимо учитывать при выборе ИП.

1.1.4.3.Исполнительные устройства (ИУ)

Исполнительное устройство (ИУ) – это силовое устройство, предназначенное для изменения регулирующего воздействия на объект управления в соответствии с сигналом (блоком командной информации), который подается на вход устройства от командного устройства (регулятора, ручного дистанционного задатчика, УВМ).

В общем случае ИУ состоит из нескольких компонентов (рис. 10).

Регулирующий орган производит регулирующее воздействие на объект управления.

Исполнительный механизм – это блок ИУ, который преобразует входной управляющий сигнал в сигнал, оказывающий воздействие на регулирующий орган.

Блок дистанционного управления принимает команды, передаваемые от командного устройства, декодирует их в сигнал, который воспринимается исполнительным механизмом.

Блок ручного управления обеспечивает управление в ручном режиме для дублирования дистанционного управления. Например, в случае аварийных ситуаций оператор может подойти к ИУ и перевести его в безопасное положение.

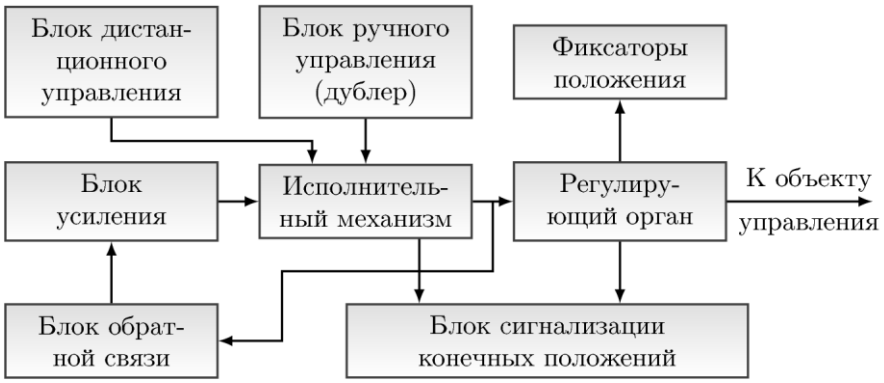


Рис. 10. Обобщенная схема ИУ

Блок сигнализации конечных положений формирует визуальный или акустический сигнал, воспринимаемый человеком, который указывает на то, что исполнительный механизм или регулирующий орган достиг конечного положения.

Фиксаторы положения реализуют фиксацию регулирующего органа (конечных и промежуточных положений), а также могут выполнять роль тормозящей системы.

Блок обратной связи позволяет получать информацию о текущем положении исполнительного механизма. Эта информация может быть использована для корректировки положения исполнительного механизма.

Блок усиления замыкает контур обратной связи, усиливая соответствующий сигнал. Этот контур используется для точного позиционирования регулирующего органа на основе принципов теории автоматического управления.

Особенности ИУ во многом определяются характеристиками исполнительного механизма (ИМ) поэтому важно рассмотреть классификацию ИМ.

Классификация по виду получаемой командной информации:

– аналоговые (например, управляются уровнем напряжения);

– цифровые;

– бинарные (только два сигнала, например, «включить», «отключит»).

Классификация по виду энергии для перестановочного усилия:

– пневматические ИМ (мембранные, поршневые, сильфонные);

– гидравлические (мембранные, поршневые, лопастные);

– электрические ИМ (электродвигательные, электромагнитные).

Классификация по способу перемещения регулирующего органа:

– прямоходные;

– однооборотные;

– многооборотные.

В дальнейшем (см. п. 1.3.4.3) будут подробнее рассмотрены электрические ИМ.

1.1.4.4. Устройства связи с объектом (УСО)

Устройство связи с объектом (УСО) служит для обеспечения связи УВМ с датчиками и ИУ, и в общем случае реализует следующие функции:

– переключение сигналов с помощью коммутаторов;

– гальваническая развязка;

– аналоговая и цифровая фильтрация сигналов;

– усиление аналоговых и дискретных сигналов;

– аналого-цифровое и цифро-аналоговое преобразование сигналов;

– линеаризация и масштабирование сигналов;

– реализация специальных алгоритмов обработки информации (например, алгоритмов сжатия информации);

– хранение измерительной информации в специальных буферных устройствах.

Подробнее УСО будут рассмотрены в п. 1.3.4.4.

1.1.4.5. Устройства взаимодействия с оператором (УВО)

Устройства отображения информации (УОИ) – это технические средства, которые служат для создания информационных визуальных моделей целевого процесса.

УОИ служат для повышения эффективности взаимодействия оператора и технической системы. По мере совершенствования общей технической базы также развивались УОИ. В итоге сейчас можно выделить следующие основные виды УОИ:

1) на основе электромагнитных и электромеханических элементов: показывающие и регистрирующие приборы, стрелочные приборы, различные автоматические графопостроители;

2) панели, табло, мнемосхемы, индикаторы, реализуемые с использованием различных осветительных приборов: ламп накаливания, электролюминесцентных ламп, газоразрядных приборов;

3) панели, табло на основе жидкокристаллических приборов индикации;

4) дисплеи и видеотерминальные средства, в том числе как компоненты компьютерных систем;

5) средства дополненной и виртуальной реальности.

С помощью *органов управления* оператор выполняет действия, формирующие управляющие воздействия на систему, которую он обслуживает.

Можно выделить три основных вида органов управления:

– физические органы управления (тумблеры, механические кнопки, рубильники, переключатели и пр.);

– графические манипуляторы (мышь, трекбол и др.);

– экранные формы, реализуемые в виде элементов графического интерфейса, в том числе с использованием сенсорного экрана (значки, графические кнопки, различные графические аналоги физических органов управления).

1.1.5. Компьютерные технологии управления в технических системах

В широком смысле *технология* – совокупность методов, а также тех или иных ресурсов, предназначенных для эффективного достижения результатов в той или иной области деятельности.

Часто этот термин используется во множественном числе, когда, например, говорят о совокупности технологий в той или иной области деятельности.

Также технологии группируются по областям деятельности. Например, различают машиностроительные технологии, биотехнологии, космические технологии, нанотехнологии и др.

В случае материального производства на основе первичного сырья формируется некая новая материальная продукция согласно тем или иным технологиями производства. Но, как ранее говорилось (см. п. 1.1.1), физический процесс связан с изменением, перемещением и накоплением не только вещества и энергии, но и информации. Поэтому по аналогии с технологиями производства можно говорить об информационных технологиях в случаях, когда исходным «сырьем» и желаемым результатом является информация.

Информационные технологии – это технологии, которые обеспечивают получение новой информации на основе некоторой первичной информации.

Информационные технологии обычно сводятся к методам получения передачи, хранения и преобразования информации, а также к ресурсам (например, вычислительным, коммуникационным), которые обеспечивают выполнение этих методов.

Термин «информационные технологии» в настоящее время широко используется, но это не значит, что информационных технологий не было до появления компьютерной техники. Например, в свое время появление письменности привело к тому, что стало возможным хранить, передавать и обрабатывать большие объемы информации на материальных носителях. Связанные с этим технологии можно считать информационными.

Появление и развитие компьютерной техники привело к бурному развитию разнообразных видов информационных технологий и широкому распростране-

нию термина «информационные технологии». Но это не значит, что информационные технологии тождественны компьютерным технологиям.

Компьютерные технологии – это информационные технологии, основанные на использовании вычислительной техники (компьютеров).

Компьютерные технологии управления в технических системах – это компьютерные технологии, направленные на решение задач управления в технических системах.

Задачи управления в технических системах решаются на основе построения САиУ. Подобные задачи решались и до появления компьютерных технологий, например, для регулирования скорости вращения паровых машин использовались центробежные регуляторы, которые являются примером автоматического регулятора. В данном случае решалась задача управления (скоростью вращения) в технической системе (паровой машине). Но с появлением и развитием цифровой и вычислительной техники эти задачи стали решаться в основном с помощью компьютерных технологий. Поэтому вполне естественно, что учебная дисциплина «Компьютерные технологии управления в технических системах» посвящена темам, связанным с решением задач управления в технических системах на основе средств цифровой и вычислительной техники.

1.2. Архитектуры САиУ

Архитектура системы – это не только набор структурных элементов системы и связей между ними (не просто структура системы), но также еще и модель взаимодействия этих элементов.

Ранее были перечислены основные классы САиУ. С этими классами можно связывать различные варианты архитектур. И есть определенные зависимости между этими классами САиУ и типами архитектур. Но можно также использовать классификацию САиУ по видам типовых архитектур.

1.2.1. Типовые архитектуры САиУ

1.2.1.1. Централизованная архитектура

САиУ, реализованная на основе *централизованной* архитектуры, состоит из одного УВК, включающего в себя одну УВМ и в общем случае множество УСО и УВО (рис. 11).

Вообще, более полное название данной архитектуры – централизованная *одноуровневая* архитектура. Но для краткости слово «одноуровневая» пропускается.

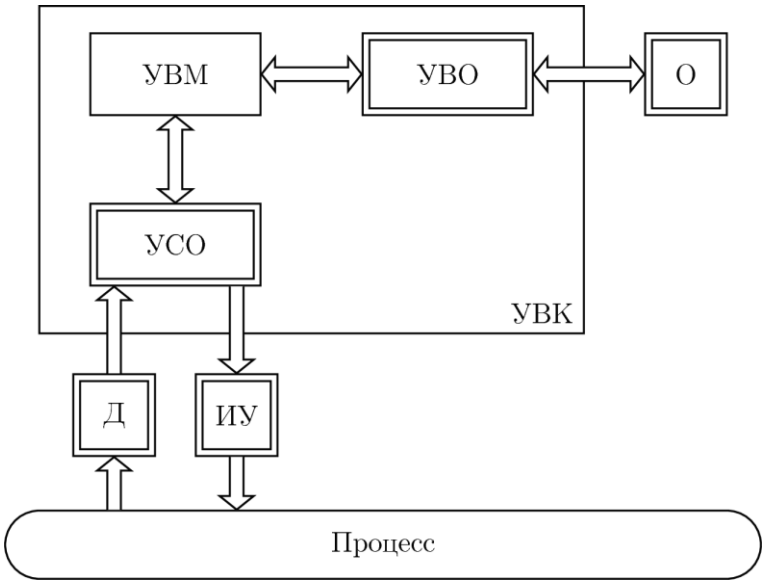


Рис. 11. САиУ централизованной архитектуры (подробный вариант)

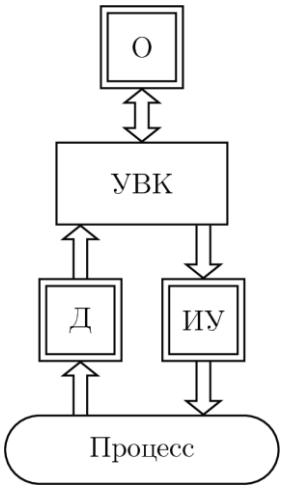


Рис. 12. САиУ централизованной архитектуры

Содержимое УВК показано на рис. 6, поэтому в дальнейшем при отображении схем типовых архитектур содержимое УВК может не раскрываться, чтобы упростить получаемую схему. В частности, тогда централизованную архитектуру САиУ можно представить, как это сделано на рис. 12.

1.2.1.2.Децентрализованная архитектура

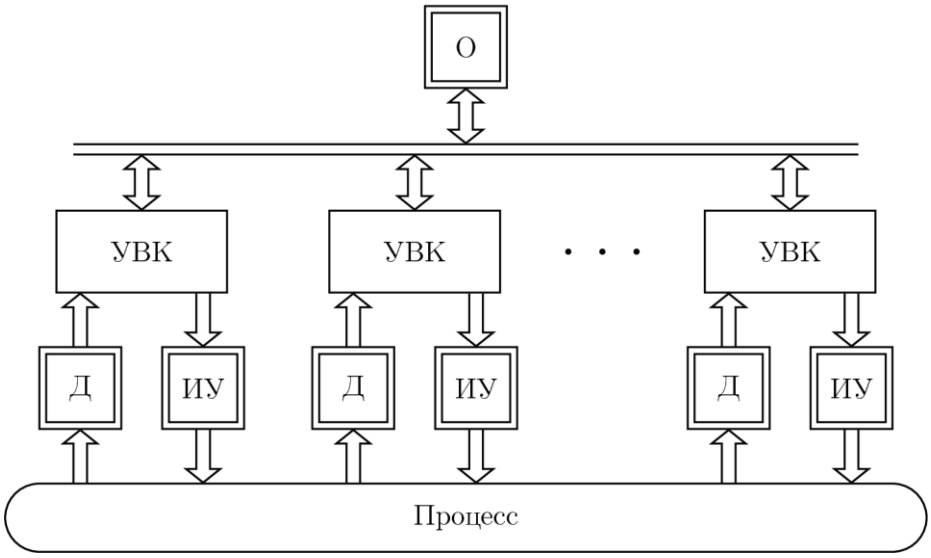


Рис. 13. САиУ децентрализованной архитектуры (обобщенный случай)

САиУ, реализованная на основе *децентрализованной* архитектуры, состоит из нескольких УВК, каждый из которых взаимодействует с процессом с помощью датчиков и ИУ, при этом УВК как будто «распределяются» по процессу, то есть происходит децентрализация (рис. 13).

Здесь надо отметить, что более точное название данной архитектуры – децентрализованная *одноуровневая* архитектура. Но для краткости слово «одноуровневая» пропускается.

1.2.1.3.Многоуровневая архитектура

Два предыдущих вида типовых архитектур, как было отмечено, относятся к одноуровневым.

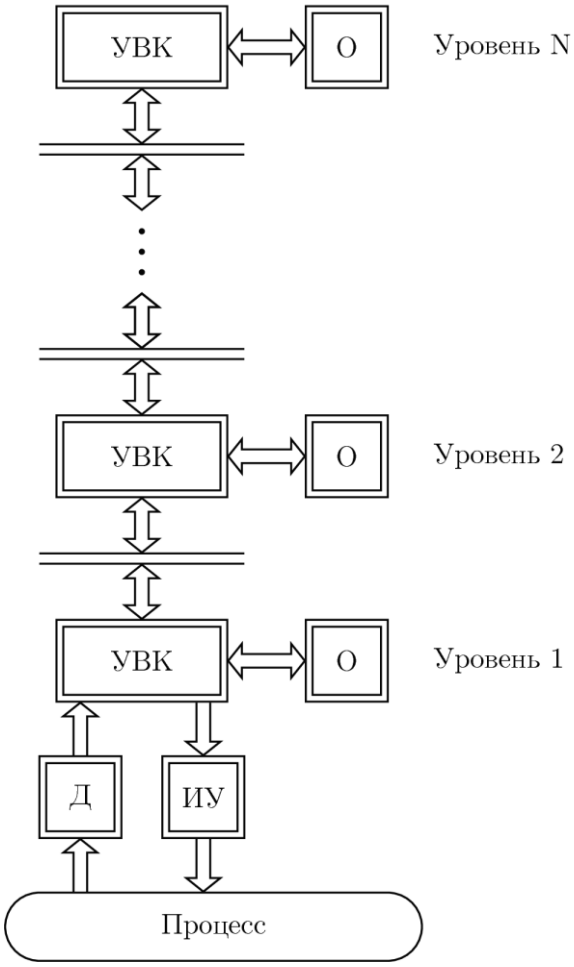


Рис. 14. САиУ многоуровневой архитектуры
(обобщенный случай)

САиУ, реализованная на основе *многоуровневой* архитектуры, состоит из нескольких УВК, при этом некоторые из них взаимодействуют с процессом с помощью датчиков и ИУ, а некоторые такого взаимодействия с процессом не имеют и образуют вышестоящие уровни САиУ (рис. 14).

Конечно, на рис. 14 представлено наиболее общее представление реализации многоуровневой архитектуры. Количество уровней может варьироваться, а также может варьироваться количество УВК на каждом уровне. Применительно к количеству УВК можно говорить о том, что если УВК на уровне есть только один, то такой уровень реализуется согласно принципу централизованной архитектуры, иначе – по принципу децентрализованной архитектуры.

1.2.2. Типовые функции нижних и верхних уровней САиУ

В случае многоуровневой архитектуры САиУ возникает определенное функциональное разделение нижних и верхних уровней системы. Это разделение обусловлено тем, что нижние уровни находятся ближе к целевому процессу (объекту управления), а верхние уровни – ближе к человеку-оператору.

Для упрощения можно уровни выше 1-го объединить в один верхний уровень, а 1-й уровень считать нижним уровнем. Конечно, такой подход будет упрощенным, но для первого приближения и ознакомления с данной темой он вполне подходит.

При таком подходе можно говорить об отдельных функциях нижнего уровня и функциях верхних уровней.

Рассмотрим типовые функции *нижнего* уровня.

1) Непосредственное взаимодействие с объектом (согласование измеряемых параметров, управляющих воздействий).

2) Первичная обработка информации. Сигналы с датчиков могут приходить со значительным уровнем помех, которые надо фильтровать. Кроме того, при измерении возникают погрешности, которые часто требуется компенсировать, например, с помощью повторных измерений и усреднения полученных значений. Таким образом, простейшими вариантами первичной обработки информации будут фильтрация и усреднение.

3) Реализация локальных алгоритмов управления. Например, требуется автоматическое регулирование температуры согласно заданному значению на основе включения или отключения нагревателя. Простейшим вариантом будет позиционное регулирование, то есть при превышении заданного значения на некоторую величину нагреватель должен отключаться, а при занижении должен включаться. Алгоритм, обеспечивающий это регулирование должен быть реализован на нижнем уровне. Естественно, локальные алгоритмы управления могут быть гораздо более сложными (см. пример в п. 2.3.1.1).

4) Взаимодействие с верхними уровнями. На верхние уровни передается не вся информация, выделяется только действительно необходимая информация. Например, на нижнем уровне для целей локального управления параметр объекта измеряется с периодом 1 мс, а на верхний уровень новое значение этого параметра передается с периодом в 1 с для отображения на экране монитора с целью сообщения оператору. Тогда на верхние уровни может передаваться, например, одно значение из тысячи или, например, среднее арифметическое тысячи значений. На нижний уровень передаются управляющие воздействия, а именно прямые (включить, отключить и т.д.), а также косвенные (на основе заданий для регуляторов, параметров для локальных алгоритмов управления).

5) Иногда реализуется хранение (архивирование) информации. Архивирование информации имеет смысл, например, когда связь с верхним уровнем производится сеансами, причем с достаточно большим интервалом. Тогда между этими сеансами связи информация должна накапливаться на нижнем уровне, а во время сеанса передаваться на верхний уровень.

Выделим также типовые функции *верхних* уровней.

1) Взаимодействие с операторами (визуализация, звуковые сообщения). При этом используются те или иные средства передачи информации оператору (мониторы, индикаторы, средства мультимедиа и т.д.) и средства ввода информации (клавиатура, мышь и т.д.), а также важно учитывать психофизиологические характеристики оператора.

2) Реализация алгоритмов глобального управления. В отличие от локального управления в данном случае управление осуществляется всем целевым процессом для достижения заданных критериев управления. Например, реализуется энергосбережение в системе автоматизации здания или достижение заданного качества производимой продукции с помощью соответствующей САиУ.

3) Архивирование (хранение) данных. Это необходимо, например, для записи параметров технологического процесса для последующего его анализа, выяснения причин неполадок, определения путей улучшения технологии и т.д.

4) Взаимодействие с другими системами. Например, взаимодействие САиУ цеха с САиУ всего предприятия. Получается, что несколько САиУ интегрируются в единую систему.

5) Взаимодействие с нижним уровнем.

Также важно отметить, что по сравнению с верхними уровнями на нижнем уровне возникают более жесткие требования по своевременности выполнения тех или иных действий (задержки могут, например, вызвать ухудшение качества управления быстротекущими процессами), а также более жесткие требования по надежности (так как во многих случаях приостановка работы верхнего уровня оказывается менее критичной, чем приостановка работы нижнего уровня из-за неполадок).

1.2.3. Пирамида комплексной автоматизации предприятия

В случае сложных САиУ, в частности, для сложных технологических предприятий, могут быть реализованы проекты комплексной автоматизации, которые предполагают автоматизацию не только технологического процесса, но и других процессов более высокого уровня. Одной из моделей для описания таких проектов является пятиуровневая модель, так называемая пирамида комплексной автоматизации предприятия (см. рис. 15).

Рис. 15. Пирамида комплексной автоматизации предприятия

В этой модели нижний уровень выделяет в отдельный слой датчики и ИУ.

Следующий уровень локального управления соответствует нижнему уровню многоуровневой САиУ (см. п. 1.2.2).

Уровень SCADA примерно соответствует верхним уровням многоуровневой САиУ (см. п. 1.2.2). Термин «SCADA», а также родственные ему термины «SCADA-система» и «SCADA-пакет» отдельно и подробно разбираются в п. 1.4.4.

Важно понимать, что первые три нижних уровня пирамиды соответствуют САиУ, которая реализует автоматизацию технологического процесса данного предприятия.

Рассмотрим верхние два уровня и связанные с ними понятия.

MES (Manufacturing Execution System) – система управления производственными процессами, представляющая собой частный случай АИС (см. п. 1.1.2.4) и решающая задачи управления, анализа и оптимизации процесса производства продукции в рамках данного предприятия. На уровне MES решаются задачи управления качеством продукции, совершенствуется технология производства, обеспечивается управление персоналом, техническое обслуживание производственных мощностей.

ERP (Enterprise Resource Planning) – планирование ресурсов предприятия. Это планирование обычно осуществляется на основе соответствующей АИС, которая обеспечивает оптимизацию и планирование ресурсов предприятия для достижения глобальных целей. При этом под ресурсами здесь понимаются не только сырье, технические и финансовые ресурсы, но и людские ресурсы (персонал, работники предприятия). На этом уровне, в частности, решаются логистические и финансовые задачи.

Естественно, что в этой пирамиде все уровни связаны и информация циркулирует между уровнями. Например, информация о текущих показателях технологического процесса с уровня SCADA поступает на уровень MES, где анализируется, что может привести к изменениям условий технологического процесса за счет формирования соответствующих информационных воздействий с уровня MES на уровень SCADA.

1.3. Основные методы решения задач управления в технических системах с использованием компьютерных технологий

САиУ создается для решения поставленных задач управления. Таким образом, можно говорить о том, что методы решения задач управления напрямую связаны с методами проектирования и реализации САиУ. В этом случае САиУ выступает средством решения задач управления. Проблему выбора методов решения задач управления можно в таком случае свести к проблеме выбора вариантов реализации САиУ. Варианты реализации САиУ могут отличаться между собой на разных уровнях детализации. На наиболее высоком уровне абстракции варианты САиУ различаются по видам решений, связанных с архитектурой этих систем. В свою очередь, архитектура конкретной САиУ может наполняться той или иной компьютерной технологией. Если спускаться на еще более детальный уровень, то мы приходим к тому, что компьютерная технология находить свое воплощение в конкретных технических средствах, которые, в свою очередь, предполагают использование тех или иных программных средств. Таким образом, можно выделить три уровня решений задач управления при построении САиУ:

– уровень архитектуры САиУ;

– уровень компьютерных технологий управления;

– уровень технических средств автоматизации и управления.

Естественно, что данная схема, как и любое обобщение оставляет за скобками многие специальные случаи, которые могут не укладываться в данную схему. Но эта схема удобна в качестве смыслового каркаса для анализа процесса решения задач управления в технических системах, и она может послужить хорошей отправной точкой для выбора методов решения задач управления. В дальнейшем, на практике, можно находить специальные и более эффективные решения, начиная рассмотрение проблемы построения САиУ с помощью данной схемы.

В рамках данной схемы можно говорить, что основные методы решения задач управления сводится к решению вопросов, связанных с:

– архитектурой САиУ;

– компьютерными технологиями управления;

– техническими средствами автоматизации и управления.

Ниже будут рассмотрены вопросы, связанные с этими тремя компонентами, но перед этим обратимся к процессу разработки САиУ, в ходе которого как раз и должны находить свое решение задачи управления.

1.3.1. Основные этапы разработки САиУ

При рассмотрении методов решения задач управления важно знать типовую последовательность разработки сложных САиУ. Это необходимо для того, чтобы понимать, как те или иные решения задач управления, сделанные на определенных этапах разработки, могут повлиять на другие этапы разработки. Например, реализация плохо расширяемой архитектуры может повлечь дополнительные издержки на этапах сопровождения в связи с возможными последующими расширениями системы. Эти возможные расширения необходимо учитывать на ранних этапах разработки. Поэтому важно прослеживать, как последствия тех или иных решений могут проявиться на более поздних этапах разработки. Для этого надо четко себе представлять эти этапы.

1.3.1.1. Предпроектная проработка

На этапе предпроектной проработки осуществляются следующие действия.

– Выделение и описание целевого процесса.

– Определение целей внедрения САиУ.

– Формирование задач автоматизации и критериев управления.

– Предварительная оценка алгоритмов решения задач автоматизации, включающая: анализ информационных потоков; оценку расположения датчиков и исполнительных устройств; ориентировочное определение функций оперативного персонала.

– Приближенная оценка состава и стоимости технических средств и предварительная оценка ожидаемого экономического эффекта.

– Составляется *техническое задание (ТЗ)* на разработку САиУ.

ТЗ – это один из важнейших документов при взаимодействии заказчика системы (САиУ) и разработчика.

В общем случае ТЗ может содержать следующие основные пункты:

– общие сведения (плановые сроки окончания работ, наименование и условное обозначение системы);

– назначение и цели создания системы;

– характеристика целевого процесса или объекта управления (условия эксплуатации, характеристики окружающей среды);

– требования к системе (к системе в целом, к функциям, к видам обеспечения);

– состав и содержание работ по созданию системы;

– порядок контроля и приемки системы;

– требования к составу и содержанию работ по подготовке к вводу системы в действие.

Этап предпроектной проработки выполняется либо заказчиком, либо совместно заказчиком и разработчиком.

Если объект является сложным и система реализуется впервые, то разработка ТЗ может быть сложной задачей. Например, может потребоваться длительное исследование объекта. В этом случае данный этап может быть оформлен в виде отдельного договора между заказчиком и разработчиком применительно к решению задачи всестороннего исследования объекта.

1.3.1.2.Предварительная проработка

На этапе предварительной проработки осуществляются следующие действия:

– Оценка характера и объема работ.

– Разработка *технического предложения*, в котором содержатся предложения по реализации видов обеспечения, по тому, как будет реализована система.

– Формирование спецификации аппаратных и программных средств, подлежащих приобретению.

Данный и последующий этапы выполняются разработчиком при возможном взаимодействии с заказчиком.

Например, с заказчиком может обсуждаться техническое предложение. В результате такого обсуждения могут возникать корректировки в архитектуре системе, в совокупности предполагаемых технических средств и т. д. Также может согласовываться спецификация приобретаемых аппаратных и программных средств.

1.3.1.3.Разработка технического проекта

На этапе разработки технического проекта принимаются все принципиальные решения по построению САиУ до момента подключения системы к объекту, в частности:

– выбирают готовые технические средства и решения;

– проектируют нестандартные технические средства;

– разрабатывают функциональные и принципиальные схемы компонентов САиУ;

– проводят физическое и математическое моделирование работы САиУ или ее отдельных компонентов;

– уточняют или разрабатывают алгоритмы управления;

– определяют объемы и формы представления информации;

– частично или полностью разрабатывают программное обеспечение;

– осуществляют отладку части компонентов САиУ на имеющемся оборудовании, на имитаторах, моделях и т. д.

В результате данного этапа разрабатывается комплект документации – *технический проект*.

Иногда формируется *эскизно-технический проект*, в случае когда какие-либо решения могут быть приняты окончательно только после проверки на реальном целевом процессе. Тогда отдельные разделы эскизно-технического проекта могут быть уточнены на дальнейших этапах разработки САиУ.

1.3.1.4. Формирование рабочего проекта

На данном этапе формируется *рабочий проект*, включающий всю необходимую документацию, в частности:

– схемы размещения и соединения технических средств;

– схемы прокладки силовых и сигнальных кабелей;

– чертежи и схемы для нестандартных технических средств;

– документация на программное, математическое, информационное, лингвистическое обеспечение;

– чертежи и схемы строительной части (например, системы электропитания, кондиционирования воздуха, обеспечения режима влажности);

– составляющие организационного обеспечения.

Если внедрение системы приводит к изменениям технологического оборудования на объекте, то в проект должны быть включены чертежи и схемы, которые отражают эти изменения.

Также рассматриваются вопросы электропитания, реализации заземления.

Принимаются решения по обеспечению помехозащищенности компонентов системы, в частности, по экранированию технических средств системы.

Может выполняться более точный расчет экономической эффективности от внедрения САиУ, так как именно на этом этапе становится известным почти весь объем затрат на проектирование системы, установку и наладку.

1.3.1.5. Монтажно-наладочные работы

На этом этапе выполняются работы по монтажу и наладке САиУ, ее стыковке с целевым процессом.

Эти работы выполняются либо после завершения формирования рабочего проекта, либо параллельно с формированием рабочего проекта.

При стыковке системы с объектом могут появиться новые технические требования, которые не были учтены ранее. Поэтому разрабатываемая система должна обладать способностями к частичным изменениям технических средств, программного и других видов обеспечения.

В случае, когда система внедряется для производства, где остановки оборудования нежелательны или невозможны, то тогда время для этого этапа резко сокращается, и возникает потребность в использовании специальных средств, имитирующих целевой процесс.

1.3.1.6.Испытания, опытная эксплуатация, сопровождение

Сначала могут проводиться *предварительные испытания*, чтобы предварительно выяснить работоспособность САиУ, а по итогам этих испытаний решить вопрос о приемке САиУ в *опытную эксплуатацию*. Предварительные испытания проводятся обычно совместно разработчиком и заказчиком.

Результаты принятия в опытную эксплуатацию оформляют соответствующим актом. Основанием для этого акта обычно служат протоколы предварительных испытаний, где отмечается ход проведения предварительных испытаний, основные количественные и качественные показатели работы САиУ.

Длительность опытной эксплуатации зависит от конкретной ситуации. Главное, необходимо детально проверить правильность работы САиУ на реальном целевом процессе.

Затем проводят *приемочные испытания* для того, чтобы выяснить, насколько разработанная САиУ соответствует изначальным требованиям, указанным в ТЗ, а также возможно другим требованиям, например, общим требованиям предприятия или отрасли в случае внедрения АСУТП.

По итогам приемочных испытаний также определяется, возможно или нет осуществить запуск САиУ в обычную (не опытную) эксплуатацию.

Запуск САиУ также может производиться поэтапно, и для каждого этапа могут формироваться соответствующие акты.

В конце успешного завершения приемочных испытаний должен быть составлен протокол этих испытаний, а также подписан акт о вводе САиУ в действие.

После ввода системы в действие некоторое (иногда продолжительное) время может осуществляться *сопровождение* системы силами разработчика (своего рода возможность гарантийного обслуживания с целью быстрого устранения выявляемых неполадок). Сопровождение также может оформляться отдельным договором.

1.3.2. Архитектуры САиУ

Ранее (см. п. 1.2.1) были выделены три типовые архитектуры САиУ:

– централизованная;

– децентрализованная;

– многоуровневая.

Рассмотрим детальнее некоторые аспекты этих архитектур, так как их понимание является важной составляющей для правильного анализа и выбора архитектуры САиУ.

1.3.2.1.Централизованная архитектура

Важно понимать, что централизованная архитектура может быть и значительно *распределенной* в пространстве. Это достигается за счет распределения в пространстве множества УСО. Каждое УСО можно расположить близко к месту подключения соответствующих датчиков и ИУ. Тогда получается, что УВК также распределяется в пространстве за счет УСО, но архитектура при этом остается централизованной из-за наличия *единственной* УВМ в составе УВК (рис. 16).

Пример распределенного УСО (модули ADAM-4000) для реализации подобного варианта архитектуры представлен в п. 1.3.4.4.

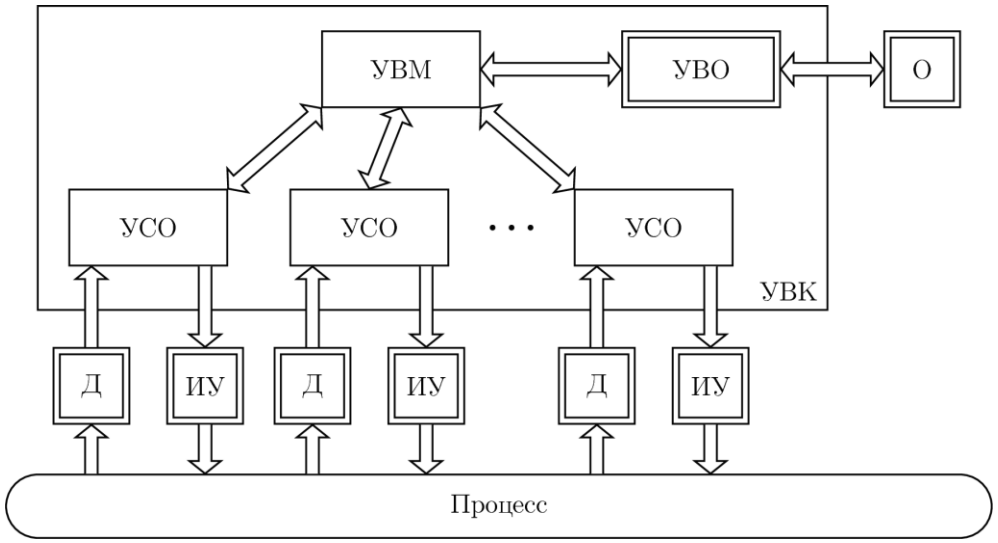


Рис. 16. САиУ централизованной архитектуры (УСО могут значительно распределяться в пространстве, оставаясь в рамках одного УВК)

Преимущества централизованной архитектуры:

– простота взаимодействия компонентов САиУ, так как имеется только одна УВМ, и поэтому отсутствует проблема разделения ресурсов между вычислительными устройствами;

– удобство контроля над системой, так как имеется единый центр управления на основе единственной УВМ.

Недостатки централизованной архитектуры:

– может возникать нехватка вычислительных ресурсов для реализации сложных задач автоматизации из-за наличия только одной УВМ;

– сложность автоматизации процессов, распределенных на большой площади, так как для единственной УВМ надо создавать каналы связи с датчиками и ИУ, удаленными на большие расстояния;

– для повышения надежности системы надо принимать специальные меры для обеспечения стабильной работы единственной УВМ.

Указанные недостатки можно, конечно, попытаться преодолеть:

– установкой мощной УВМ или распараллеливанием вычислительных процессов на нескольких физических УВМ, работающих как единая логическая УВМ;

– использованием УСО распределенного типа, которые, например, подключаются по шинной топологии и размещаются на больших расстояниях от УВМ, тем самым получается УВК, распределенный на большой площади (см. рис. 16);

– реализацией дублирования (резервирования) УВМ для повышения общей надежности системы.

Но в целом централизованная архитектура, в первую очередь, удобна для реализации САиУ, когда задачи автоматизации не очень сложны и целевой процесс не сильно распределен в пространстве.

Как понятно из рис. 11, 12, 16, с учетом наличия одного УВК получается, что рабочее место оператора совмещается с управляющим компьютером, то есть единственная УВМ в составе данной САиУ должна обеспечивать не только взаимодействие с процессом, но и с человеком-оператором (в общем случае с несколькими операторами). Это накладывает дополнительные сложности на процесс разработки рабочего места оператора. В частности, это касается организации вычислительных процессов в рамках одной операционной системы, так как, с одной стороны, требуется обеспечивать жесткое реальное время для взаимодействия с процессом, но при этом с другой – также обеспечивать взаимодействие с человеком, возможные действия которого являются в большой степени недетерминированными.

1.3.2.2.Децентрализованная архитектура

Важно уточнить, что децентрализованная архитектура и «распределение», «распространение» УВК по процессу совсем необязательно связаны со значительным *пространственным* распределением.

Рассмотрим следующий простейший пример. Имеется САиУ децентрализованной архитектуры, обеспечивающая регулирование основных показателей окружающей среды в помещении (в комнате). При этом одно

компактное устройство (УВК) занимается регулированием температуры в нескольких точках. Другое устройство (другой УВК) занимается регулированием влажности. Третье устройство (тоже отдельный УВК) занимается регулированием освещенности. Могут быть и другие подобные устройства (УВК). Каждое из этих устройств обладает собственными средствами ввода измерительной информации и вывода управляющих параметров, что позволяет человеку (оператору), находящемуся в комнате, регулировать основные параметры окружающей среды в данной комнате. Получается, что указанные УВК между собой непосредственно никак не связаны, но целостность САиУ достигается за счет единого процесса (процесса изменения микроклимата в комнате), а также того, что информация от этих УВК поступает к одному оператору, который уже может принимать какие-то решения на основе поступившей информации.

Здесь надо отметить, что на рис. 13 показан наиболее общий способ соединения УВК и операторов. Смысл соответствующего обозначения был раскрыт ранее (см. рис. 4, *е*, 4, *ж*), и он означает возможность взаимодействия каждого с каждым. Поэтому для данного примера приведем рис. 17, который показывает частный случай САиУ децентрализованной архитектуры. В этом случае УВК объединяются посредством одного оператора, но сами УВК при этом не связаны непосредственно друг с другом.

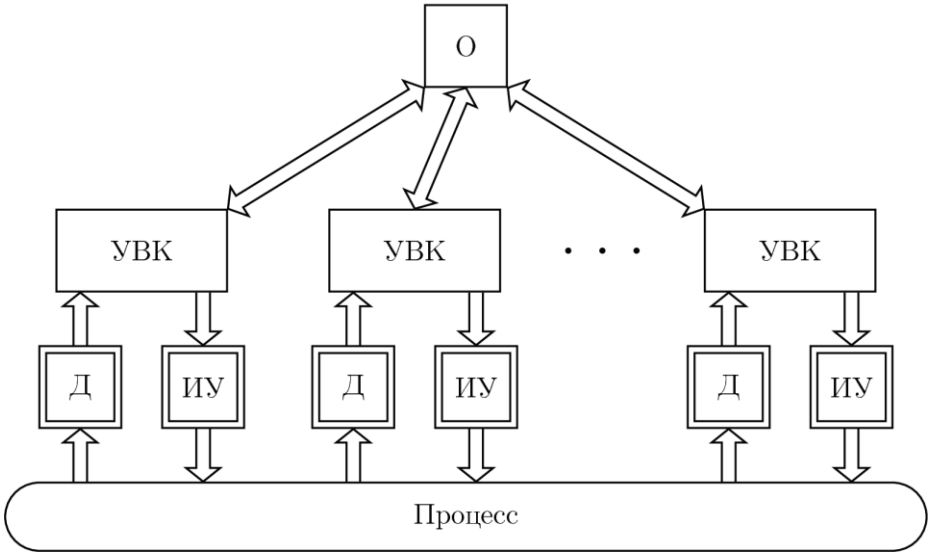


Рис. 17. Частный случай САиУ децентрализованной архитектуры
(объединение УВК через одного оператора)

Теперь рассмотрим другой частный случай децентрализованной архитектуры.

Предположим, что имеется САиУ в виде системы автоматизации здания, в рамках которой выполняется автоматизация нескольких независимых инженерных подсистем (например, освещение, теплоснабжение, кондиционирова-

ние). Для каждой из этих трех подсистем используется отдельный УВК. И за каждой из этих подсистем следит отдельный диспетчер. В данном случае речь идет о САиУ для разных подсистем здания. Но можно говорить и о САиУ всего здания в целом, и такая САиУ будет иметь децентрализованную архитектуру. При этом различные УВК будут соединяться через общий целевой процесс поддержания жизнедеятельности здания. И такой частный случай децентрализованной архитектуры представлен на рис. 18.

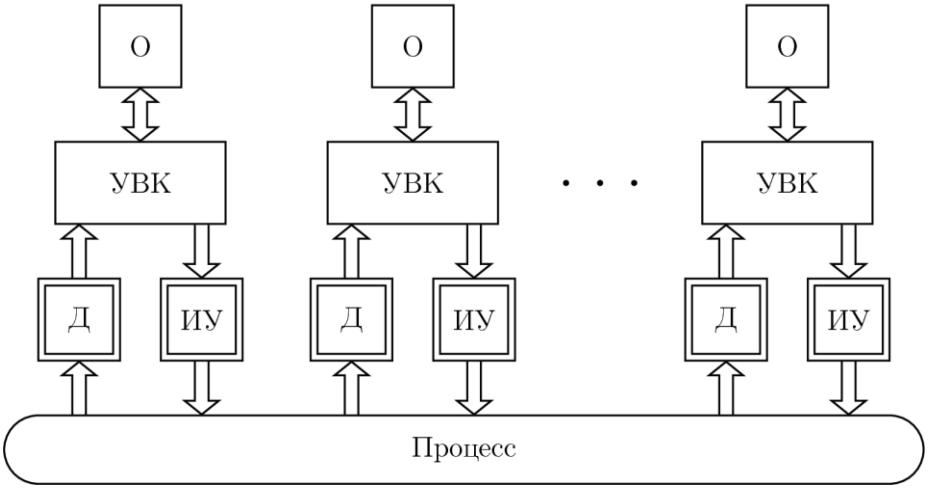


Рис. 18. Частный случай САиУ децентрализованной архитектуры (отдельные операторы для отдельных УВК, объединение УВК через единый целевой процесс)

Другим примером подобной архитектуры может быть большое здание, в котором находится множество комнат и других помещений. В каждом помещении находится отдельный контроллер (УВК), который обеспечивает регулирование основных параметров окружающей среды в данном помещении. Например, он включает нагреватель или систему кондиционирования в данной комнате, если температура вышла за допустимые пределы. В этом примере во всех помещениях здания обеспечивается поддержание нужного микроклимата, то есть обеспечивается поддержание микроклимата во всем здании в целом. Здесь отдельные
УВК взаимодействуют с процессом через датчики и ИУ, но между собой информационно не связаны, так как они работают каждый в своем помещении, не обмениваясь информацией друг с другом. Также в каждом помещении может находиться человек (оператор), который может взаимодействовать с УВК в данном помещении, но не с другими УВК. Перейдя в другое помещение, он становится оператором, который соединяется с другим УВК, но теряет связь с предыдущим УВК. Получается, что целостность данной децентрализованной архитектуры достигается только за счет единого целевого процесса, связанного с обеспечением микроклимата в целом здании.

Конечно, каждый отдельный УВК в помещении можно рассматривать как отдельную САиУ централизованной архитектуры. Но, как уже неоднократно отмечалось, все зависит от того, что мы будем считать целевым процессом. Если мы отдельно хотим рассматривать процессы поддержания микроклимата для каждой комнаты, то тогда мы видим множество САиУ централизованной архитектуры. Если же мы обращаемся к вопросу поддержания микроклимата в целом здании, то тогда уместно говорить о САиУ поддержания микроклимата в здании, и в данном примере эта САиУ оказывается реализованной на основе децентрализованной архитектуры.

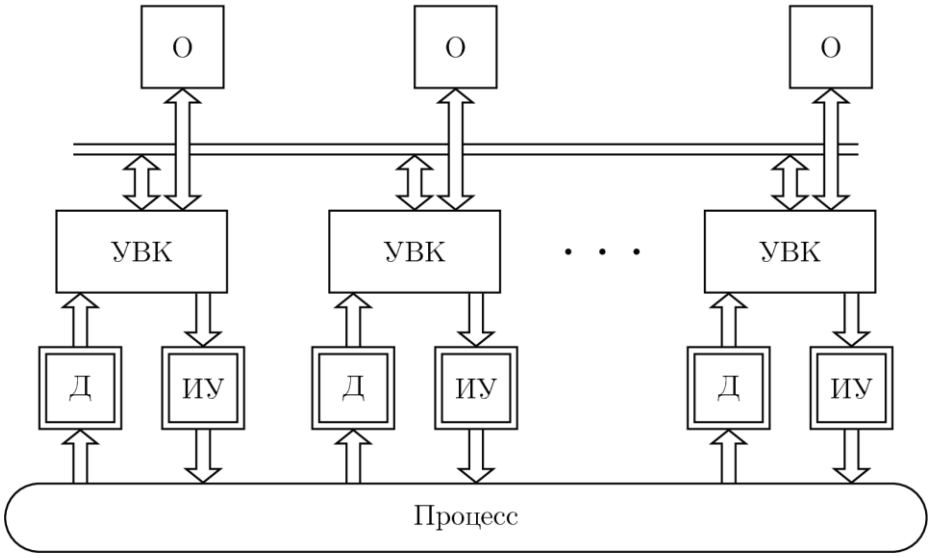


Рис. 19. Частный случай САиУ децентрализованной архитектуры (оператор может подключаться к любому из УВК, соединенных по принципу «каждый с каждым»)

Рассмотрим еще один пример. Пусть имеется САиУ, которая обеспечивает управление микроклиматом квартиры в жилом доме. В каждом отдельном помещении квартиры имеется УВК, регулирующий основные параметры данного помещения. Каждый УВК имеет средства взаимодействия с человеком, как это и было в предыдущих примерах. Отличие состоит в том, что любой УВК может получить информацию с любого другого УВК системы и может послать управляющие воздействия любому УВК системы. В этом случае оказывается, что человек (оператор), находясь в любом помещении квартиры, может управлять параметрами любого другого помещения этой квартиры. Этот пример пояснен на рис. 19, где показан еще один частный случай САиУ децентрализованной архитектуры.

В этом случае все УВК связаны по принципу «каждый с каждым», и к каждому из УВК может подключиться оператор, чтобы с помощью этого УВК управлять всей системой.

Другой вариант децентрализованной архитектуры состоит в том, что только один УВК может обеспечивать взаимодействие с оператором, при этом все УВК соединены по принципу «каждый с каждым» (рис. 20).

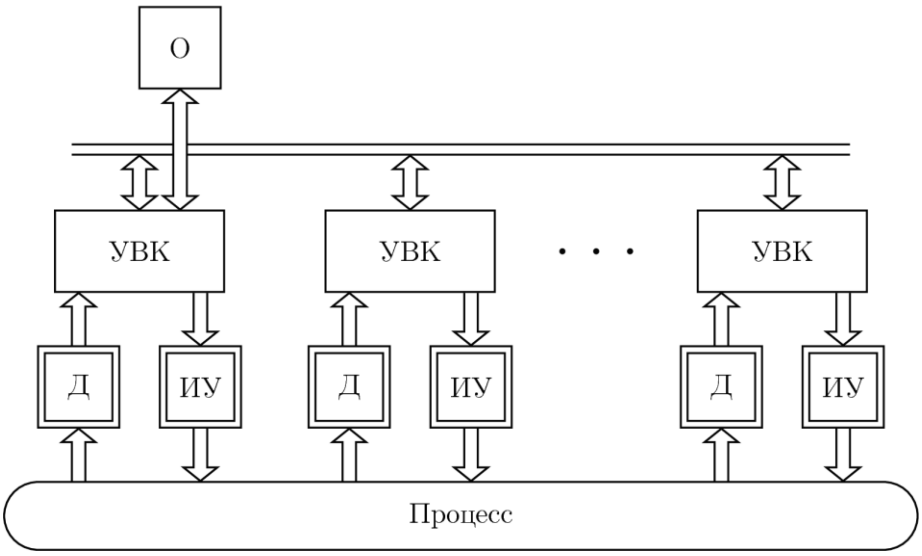


Рис. 20. Частный случай САиУ децентрализованной архитектуры (один УВК для взаимодействия с оператором, и все УВК соединены по принципу «каждый с каждым»)

Такой вариант архитектуры получается, если в только что рассмотренном примере изменить реализацию системы таким образом, что человек управляет всей системой только через один из контроллеров (УВК). Такой контроллер может быть установлен в наиболее удобном месте квартиры, и для удобства взаимодействие с ним может осуществляться, например, с помощью пульта дистанционного управления. В данном примере очень важно, что этот выделенный контроллер также должен обладать возможностью взаимодействовать с процессом, например, регулировать температуру в помещении, где он находится. Иначе получится многоуровневая архитектура, о которой речь пойдет позднее.

Преимущества децентрализованной архитектуры:

– удобство автоматизации процессов, распределенных на большой площади, так как отдельные УВК можно располагать близко к месту подключения датчиков и ИУ;

– общая надежность системы может повышаться за счет децентрализации функций автоматизации, так как выход из строя отдельного УВК может не привести к полному выходу из строя всей системы;

– суммарную вычислительную мощность можно наращивать за счет выделения отдельных УВК для отдельных задач, например, для реализации отдельных контуров управления.

Недостатки децентрализованной архитектуры:

– могут возникать сложности в реализации взаимодействия отдельных УВК, когда это предусмотрено архитектурой, как, например, показано на рис. 19;

– возможны сложности реализации общего управления системой, как, например, в случае, показанном на рис. 18;

– может возникать дублирование функций у различных УВК, в частности, функции взаимодействия с оператором, как в примере, показанном на рис. 19, когда оператор должен иметь возможность непосредственного взаимодействия с любым УВК.

Указанные недостатки, конечно, отчасти компенсируются соответственно:

– использованием специальных технологий построения децентрализованной архитектуры взаимодействующих УВК, например, на основе соответствующих сетевых технологий (см. п. 1.3.3.2);

– применением метода, при котором отдельные УВК могут брать на себя функции единого центра, как это, например, делается в случае, указанном на рис. 19;

– удешевлением технических средств, в том числе вычислительных ресурсов и средств взаимодействия с человеком.

Но в целом децентрализованная архитектура, в первую очередь, удобна для реализации САиУ либо когда функции единого центра управления не столь сложны, либо когда целевой процесс не столь сильно распределен в пространстве и оператор может выполнять роль такого единого центра управления, взаимодействуя сразу со всеми УВК по схеме, показанной на рис. 17.

1.3.2.3.Многоуровневая архитектура

Рассмотрим частные случаи многоуровневой архитектуры.

Один из самых простых вариантов многоуровневой архитектуры представлен на рис. 21.

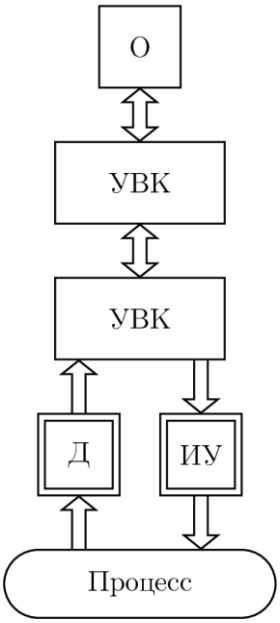


Рис. 21. САиУ многоуровневой архитектуры
(один из простейших вариантов)

На практике обычно данный вариант архитектуры проявляется в том, что УВК верхнего уровня реализуется на основе компьютера, который обеспечивает взаимодействие с оператором, а также взаимодействие с управляющим контроллером (УВК нижнего уровня). Контроллер берет на себя функции взаимодействия с объектом, а также функции выполнения алгоритмов локального управления. Этот вариант во многом подобен централизованной архитектуре (см. рис. 12). Но основное отличие от нее в том, что функции единственного УВК (в централизованной архитектуре) здесь распределяются между двумя УВК, один из которых ближе к процессу, другой ближе к оператору. И в этом случае удобно разделить жесткое реальное время, обеспечиваемое на нижнем уровне, и мягкое реальное время, обеспечиваемое на верхнем уровне.

В случае, когда на нижнем уровне требуется децентрализация, требуется много УВК, тогда часто используется следующий вариант архитектуры (рис. 22).

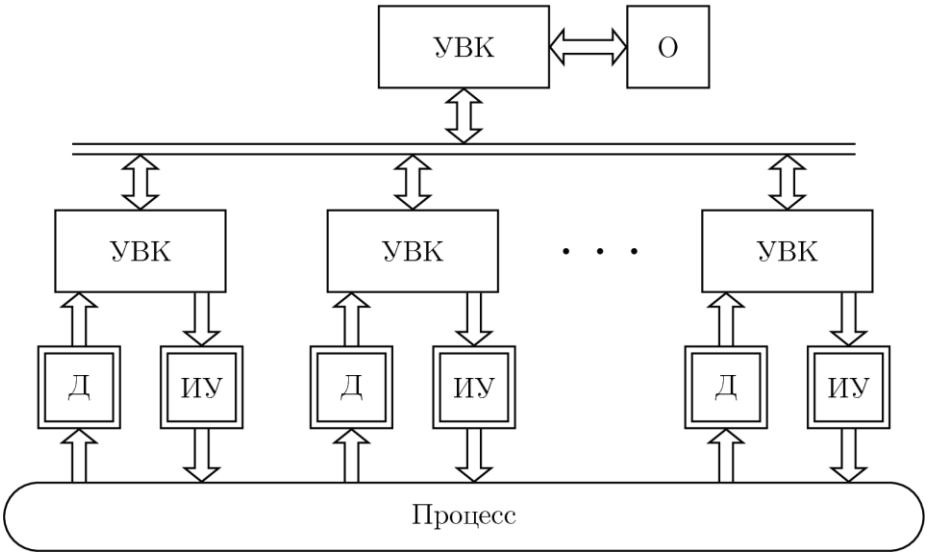


Рис. 22. САиУ многоуровневой архитектуры
(два уровня, нижний уровень децентрализован)

Этот вариант архитектуры соответствует примерам, когда имеется один управляющий компьютер, который взаимодействует с оператором, образуя верхний уровень. Одновременно он обеспечивает управление множеством контроллеров, распределенных по процессу. Получается, что нижний уровень децентрализован, а верхний уровень играет роль единого центра управления, который взаимодействует с оператором. Этот вариант многоуровневой архитектуры (см. рис. 22) отличается от варианта децентрализованной архитектуры, который представлен на рис. 20. Основной отличительный признак многоуровневости состоит в том, что имеется УВК, который не взаимодействует с процессом посредством датчиков и ИУ. Этот УВК находится на одном из верхних
уровней. В данном примере он находится на втором уровне и взаимодействует с оператором. В случае же децентрализованной архитектуры оказывается, что УВК взаимодействует как с оператором, так и с датчиками и ИУ.

Преимущества многоуровневой архитектуры:

– удобство автоматизации процессов, распределенных на большой площади, так как отдельные УВК можно располагать близко к месту подключения датчиков и ИУ, что достигается децентрализацией нижнего уровня;

– удобство контроля над системой, так как на верхнем уровне можно реализовать единый центр управления, например, на основе единственной УВМ;

– общая надежность системы может повышаться за счет децентрализации функций автоматизации, так как выход из строя отдельного УВК (например, на нижнем уровне) может оказаться не столь критичным и не привести к полному выходу из строя всей системы;

– суммарную вычислительную мощность можно наращивать за счет выделения отдельных УВК для отдельных задач, например, для реализации отдельных контуров управления, а также за счет распределения вычислительных ресурсов и информационных потоков по уровням.

Недостатки многоуровневой архитектуры:

– могут возникать сложности в реализации взаимодействия отдельных УВК на одном уровне;

– могут возникать сложности в реализации взаимодействия различных уровней, в распределении и согласовании информационных потоков.

Указанные недостатки стараются компенсировать применением готовых решений и технологий организации многоуровневых САиУ, например, на основе комплексных сетевых технологий (см. п. 1.3.3.2).

В целом многоуровневая архитектура сочетает в себе преимущества централизованной и децентрализованной архитектуры, но это влечет за собой несколько повышенную сложность реализации этого вида архитектуры САиУ.

В рамках данной архитектуры взаимодействие с человеком-оператором реализуются на верхних уровнях на основе УВК, которые не взаимодействуют с процессом непосредственно. Но при этом к оператору поступает информация от нижних уровней, а также формируются управляющие воздействия на нижние уровни, которые в свою очередь непосредственно взаимодействуют с целевым процессом.

1.3.2.4.Взаимодействие с человеком-оператором

При анализе и выборе архитектуры САиУ важно учитывать необходимость взаимодействия с человеком-оператором (или отсутствие такой необходимости). Здесь важно, что рассматривается именно человек, выполняющий роль оператора (диспетчера), а не человек-пользователь или человек-объект.

Для реализации указанного взаимодействия в составе САиУ организуется рабочее место оператора, которое занимает часть УВК. А именно к рабочему месту оператора относятся УВО, имеющиеся в составе УВК, а также обычно УВМ, находящаяся в составе УВК (рис. 23).

Конечно, во многих случаях в качестве рабочего места оператора используется один из специальных вариантов организации УВК, представленных на рис. 7, а именно вариант УВК, в котором есть УВМ и УВО, но нет УСО (рис. 24).

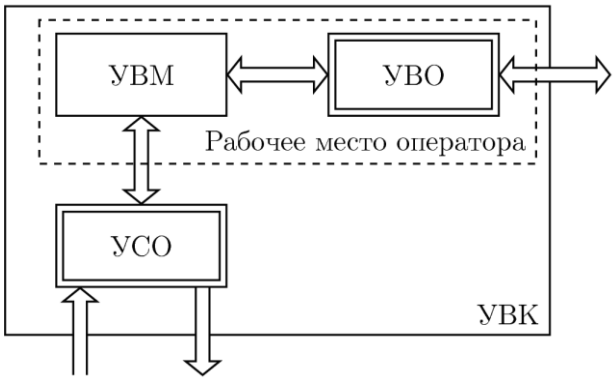


Рис. 23. Рабочее место оператора в составе обобщенного УВК

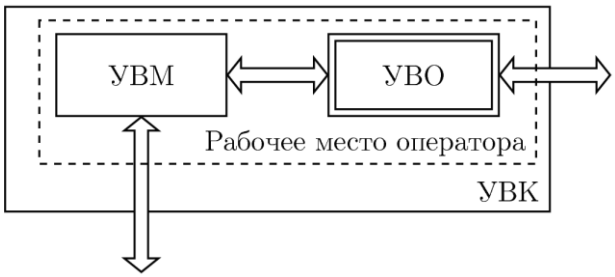


Рис. 24.Специализированный вариант организации
рабочего места оператора в виде УВК

Также применительно к архитектуре САиУ важно учитывать *назначение и основные функции* рабочего места оператора.

Можно констатировать, что в первую очередь, рабочее место оператора предназначено для обеспечения интерфейса между человеком и техническими средствами, на основе которых реализуется САиУ. Этот интерфейс должен обеспечивать стыковку информационных потоков, идущих от технических средств к оператору (восприятие информации оператором) и от человека к техническим средствам (формирование оператором управляющих воздействий).

К функциям рабочего места оператора можно отнести:

– получение информации о текущем состоянии целевого процесса;

– формирование управляющих воздействий с целью оказания влияния на состояние целевого процесса;

– протоколирование и архивирование информации о состоянии целевого процесса, а также о формируемых управляющих воздействиях;

– предоставление оператору инструментальных средств для анализа данных (для дополнительной обработки информации о состоянии целевого процесса с целью получения и использования полученных результатов в ходе принятия решений об управляющих воздействиях);

– автоматический анализ текущей ситуации и формирование сообщений и подсказок оператору в проблемных ситуациях (в пределах выполнения функций экспертной системы).

Конечно, в реальных системах не все из этих функции могут быть воплощены. Также в отдельных случаях могут быть реализованы функции, не приведенные в данном перечне.

Также важно учитывать *базовые требования* к рабочему месту оператора:

1) Функциональные требования. Эти требования определяются соответствующими требованиями, указанными в техническом задании проектируемой САиУ. Они определяются перечнем функций контроля и управления, которые должен выполнять оператор конкретной системы.

2) Требования реального времени. Эти требования регламентируют: допустимые задержки при передачи информации и управляющих воздействий; периодичность вывода информации и выполнения необходимых действий; а также другие параметры, определяющие отношения событий, происходящих на рабочем месте, к событиям, происходящим в остальной системе, и к событиям, связанным с целевым процессом.

3) Эргономические требования. Эти требования, в частности, устанавливаются государственными стандартами по эргономике, например, см. ГОСТ 12.2.049-80. Более подробная информация об эргономических требованиях к устройствам взаимодействия с оператором приводится в работе [28].

При выборе архитектуры САиУ важно обеспечивать эффективное разделение функций между человеком-оператором и остальной системой (машиной). Сложность человеко-машинного взаимодействия определяется разнородностью комплексных систем физической и биологической природы. В ходе технического прогресса интеллектуальность технических систем возрастает, при этом роль человека может уменьшаться. Но можно предположить, что даже при очень высокой степени автоматизации, все равно, на самых высоких уровнях системы будет звено «человек», и получаемая замкнутая система будет иметь свойства системы «человек-машина» [27].

В работе [27] выделяются следующие базовые различия, присущие основным элементам человеко-машинной системы:

1) на человека следует возлагать следующие функции:

– распознавание ситуации в целом, особенно при неполной информации (распознавание ситуации в целом машиной – это сложнейшая научная и техническая задача, и на сегодняшнем этапе развития с этой задачей человек справляется гораздо успешнее);

– обобщение отдельных фактов в единую систему;

– решение задач, для которых нет четкого алгоритма ее решения;

– решение задач, в которых требуется высокая адаптивность;

– решение задач с высокой ответственностью в случае возникновения ошибки;

2) машине следует поручать:

– выполнение математических расчетов;

– выполнение однообразных действий, выполняемых по четкому алгоритму;

– хранение и представление больших объемов однородной информации;

– решение задач для частных случаев на основе общих правил (дедуктивный вывод);

– выполнений действий, требующих высокой скорости реакции.

Указанные рекомендации не следует воспринимать как буквальное руководство к действию, а скорее именно как иллюстрацию различий между человеком и машиной, и при проектировании конкретного рабочего места надо учитывать множество разных факторов и выполнять более тонкий анализ этих факторов [27].

Также при выборе архитектуры САиУ необходимо учитывать, какая информация должна выдаваться оператору, в каких объемах, когда и с какой частотой, то есть необходимо собрать информацию об информационных потоках, предполагаемых для выдачи оператору. В качестве базисных можно использовать оценки количества параметров о целевом процессе, информация о которых должна предоставляться оператору. При этом важно разделять параметры:

– по важности (например, особо можно выделить параметры аварийных ситуаций);

– цикличности (например, некоторые параметры должны обновляться на экране периодически, а некоторые должны обновляться только при возникновении некоторых событий);

– принадлежности к различным подсистемам целевого процесса (это важно для последующей группировки параметров в ходе отображения оператору, например, для группировки на различных экранах или окнах);

– форме представления (числовая форма, текстовая форма, в виде цветовых индикаторов и т.д.).

Похожим образом необходимо собрать информацию о видах управляющих воздействий, которые может формировать оператор для изменения состояния целевого процесса. В частности, управляющие воздействия важно разделять:

– по ответственности, учитывая возможные последствия (наиболее ответственные управляющие воздействия в ходе проектирования рабочего места должны быть защищены системой подтверждений, разграничения доступа и т.д.);

– форме представления информации о воздействии (например, бинарная форма в виде «включить/отключить» или числовая форма в виде команды, содержащей информацию о степени закрытия некоторого вентиля);

– принадлежности различным подсистемам целевого процесса (это важно для последующей группировки органов управления, например, для группировки на различных экранах или окнах пользовательского интерфейса).

При сборе и анализе информации об информационных потоках и необходимых управляющих воздействиях важно учитывать вышеупомянутое разделение функций между человеком и машиной. Например, информация о части параметров целевого процесса может быть скрыта от внимания человека, но тогда эти параметры должны анализироваться машиной согласно определенным правилам и алгоритмам, на основе который машина вырабатывает сообщения оператору, привлекая его внимание к важным изменениям этих параметров.

1.3.3. Компьютерные технологии управления

Компьютерные технологии в своем развитии порождают много интересных областей, которые находят свое применении при построении САиУ. В качестве примеров таких компьютерных технологий можно привести:

– технологии программно-технических комплексов [7];

– технологии компьютерного (технического) зрения [25];

– технологии интеллектуальных сенсоров [5];

– технологии иммерсивных сред [27].

Эти и другие новые технологии находят свою реализацию в конкретных примерах САиУ. Важно иметь о них представление и учитывать при анализ и выборе компьютерных технологий САиУ.

Но среди наиболее распространенных групп технологий стоит, пожалуй, в первую очередь выделить программные и сетевые технологии, а также технологии систем реального времени и технологии взаимодействия с человеком-оператором.

Вопрос программных технологий во многом сводится к вопросу разработки и использования ПО в САиУ. Этот вопрос подробно будет рассматриваться отдельно (см. п. 1.4, п. 2). Поэтому здесь обратимся к технологиям реального времени, сетевым технологиям, а также технологиям взаимодействия с человеком-оператором.

1.3.3.1. Технологии систем реального времени

При разработке ПО для САиУ важно учитывать так называемые требования *реального времени.*

Система реального времени (СРВ) – система, у которой правильность функционирования зависит не только от логического результата вычислений, но и от физического времени, когда эти результаты формируются.

В целом, САиУ является СРВ, поэтому отдельно можно говорить о *технологиях систем реального времени*. Эти технологии позволяют так организовать ПО, чтобы обеспечить (заранее гарантировать) необходимые требования реального времени.

САиУ взаимодействует со средой на основе информации, поступающей от различных датчиков. Требуется, чтобы модель состояния среды, формируемая для себя системой управления, была адекватной текущему состоянию среды. Иначе воздействие системы управления на эту среду может оказаться вредным или даже разрушительным. Поэтому система управления должна периодически отслеживать текущее состояние среды и обрабатывать новую информацию для формирования возможных управляющий воздействий.

Требования своевременности получения, обработки информации и формирования управляющий воздействий возникают из-за физического (реального) взаимодействия системы управления и среды, в которой она функционирует. Другими словами, реальная природа взаимодействия системы управления и среды порождает требования реального масштаба времени, то есть своевременности функционирования системы управления. Например, если сборочному роботу не будет своевременно выдана команда об остановке или начале нового действия, то это может привести к серьезным повреждениям объекта сборки.

Развитие вычислительной техники и средств автоматизации обеспечивает увеличение количества разрабатываемых систем управления и их усложнение. Увеличиваются объемы обрабатываемой информации, выполняется автоматизация все более сложных объектов управления, разрабатываются все более сложные технические устройства, реализующие все более сложное взаимодействие с окружающей средой: роботы, бортовые и встроенные системы. Все это приводит к усложнению требований реального времени (РВ), к возникновению противоречащих требований предсказуемости и адаптивности СРВ, что и должно решаться в рамках так называемой технологии СРВ. Технология СРВ является одной из ключевых технологий в области сложных технических систем, приобретающей все большую актуальность в повседневной жизни.

СРВ должна выполнять определенные действия для внешних взаимодействий, а также для изменения своего внутреннего состояния. Действия, которые должна выполнять система, задаются на основе спецификации совокупности выполняемых *задач РВ*. При этом выполнение определенной задачи соответствует выполнению системой определенных действий. Такой подход

реализует декомпозицию системы как совокупности взаимодействующих задач, что обеспечивает упрощение анализа, верификации и реализации системы.

Повторяемость и цикличность при выполнении задач обычно отражается на основе представления задачи как последовательности *запросов*. Тогда выполнение задачи предполагает выполнение последовательности запросов.

Различают *периодические* задачи, регулярность формирования запросов которых заранее известна, а также *апериодические* задачи, моменты появления запросов которых заранее неизвестны.

Рассмотрим параметры задач РВ.

Период определяет интервал времени между появлением запросов периодической задачи. Для апериодической задачи имеет смысл указывать только *минимальный* период формирования запросов, так как моменты появления запросов заранее неизвестны, но может быть известен некоторый минимальный интервал между запросами. Апериодическая задача, для которой известен минимальный период формирования запросов, обычно называются *спорадической* задачей.

Для задачи может быть известна *максимальная длительность* запросов, формируемых этой задачей. Тогда длительность любого запроса, формируемого этой задачей не больше максимальной длительности.

Ограничение РВ для задачи определяется на основе *относительного крайнего срока*, который равен интервалу времени между временем появления и крайним сроком для каждого запроса, формируемого этой задачей.

Таким образом, к основным параметрам задач РВ относятся: период (только для периодических задач); минимальный период (только для спорадических задач); максимальная длительность запросов; относительный крайний срок.

Своевременность функционирования СРВ определяется своевременностью выполнения отдельных задач в составе СРВ. Поэтому требования РВ для СРВ предполагают формулирование требований РВ для каждой из задач в составе СРВ. Эти требования РВ удобнее всего определять в виде некоторых условий, так называемых *ограничений РВ*.

Можно говорить о том, что основное отличие СРВ от других систем, определяемых на основе совокупностей задач, – это наличие ограничений РВ для выполняемых задач.

В зависимости от необходимости выполнения ограничений РВ обычно выделяют следующие два основных типа задач РВ.

1) Задачи *жесткого* реального времени (ЖРВ). Ограничения ЖРВ этих задач должны всегда соблюдаться. Не должно допускаться ни одного нарушения ограничения ЖРВ.

2) Задачи мягкого реального времени (МРВ). Ограничения МРВ этих задач могут иногда нарушаться, при этом следует стремиться к оптимальному значению некоторой функции качества, значение которой зависит от количества нарушенных ограничений РВ. Выполнение запроса, сформированного задачей МРВ, может оставаться актуальным даже после нарушения ограничения РВ этого запроса.

Пример задачи ЖРВ. Получен сигнал о превышении температуры. Необходимо сформировать сигнал отключения нагревателя не позднее, чем через 20 миллисекунд.

Пример задачи МРВ. Необходимо обновлять измеряемые параметры в окне пульта оператора (на экране монитора) не реже, чем через 1 секунду. Возможны незначительные и редкие задержки (до 3 секунд.).

В ходе разработки программного обеспечения СРВ ограничения РВ учитываются при реализации *подсистемы планирования (диспетчеризации)*.

В общем случае *планирование* – это распределение ресурсов (памяти, времени доступа к процессору, к устройствам ввода-вывода) между запросами различных задач, направленное на соблюдение ограничений РВ. Распределение ресурсов, в частности предоставление процессорного времени, осуществляется на основе определенного *алгоритма планирования*.

В дальнейшем, в качестве простого примера под планированием будет пониматься только распределение ресурсов процессорного времени, другими словами распределение вычислительных ресурсов.

Планирования является более эффективным, если при прочих равных условиях, в частности при тех же аппаратных средствах, программное обеспечение и вся система в целом могут функционировать в условиях более жестких ограничений РВ.

Подсистема планирования для СРВ используется в составе применяемой операционной системы РВ или специально разрабатывается с учетом особенностей конкретной системы.

Таким образом, при на этапе анализа и выбора компьютерных технологий управления важно уделить пристальное внимание вопросам, связанным с функционирование системы в реальном масштабе времени, а также методам планирования задач РВ, обеспечивающим соблюдение требований РВ.

1.3.3.2. Сетевые технологии

Сетевые информационные технологии в настоящее время являются важнейшей составляющей инфраструктурных компонентов во многих областях нашей жизни. В частности, они играют существенную роль и в составе САиУ. Если обратиться, например, к рис. 22, то можно заметить, что на этом рисунке большой процент площади занимают стрелки и линии связи, которые как раз

во многом и реализуются с помощью сетевых технологий, в частности, находящих свое воплощение в виде локальных управляющих вычислительных сетей (ЛУВС).

Локальная управляющая вычислительная сеть (ЛУВС) – это локальная вычислительная сеть, которая связывает компоненты САиУ: датчики, ИУ, УСО, УВМ, УВО.

ЛУВС может обеспечивать:

– связь датчиков и ИУ с УВК, в частности с контроллерами;

– подключение распределенных УВО, в частности средств визуализации;

– взаимодействие нескольких УВК, в частности нескольких пультов оператора или нескольких контроллеров;

– передачу данных от датчиков к ИУ без участия УВК, в частности возможно создание контуров управления без участия УВК на основе «интеллектуальных» датчиков и ИУ;

– питание датчиков и ИУ;

– удаленную диагностику и настройку оборудования.

На физическом уровне различия между ЛУВС определяются особенностями среды передачи данных, которая может представлять собой:

– коаксиальный кабель;

– витую пару;

– оптоволокно;

– тот или иной вид радиосвязи;

– другой вид среды (оптическая связь в «свободной» среде, акустическая и гидроакустическая связь, инфракрасная связь, механическая связь);

Также различия между ЛУВС определяются особенностями соответствующих сетевых интерфейсов, к которым, например, относятся: RS-232C, RS-485, RJ-11 RJ-45, 8P8C, BNC.

ЛУВС также различаются по используемым методам доступа к моноканалу (единственному каналу связи). При этом можно выделить:

– метод опроса (Master-Slave или Задатчик-Исполнитель) (детерминированный метод);

– метод передачи маркера (детерминированный метод);

– метод множественного доступа с контролем несущей и обнаружением столкновений (коллизий), который имеет сокращение МДКН/ОС или CSMA/CD (недетерминированный метод).

Метод опроса состоит в том, что в сети имеется один главный узел (сервер, Master, Задатчик), и только он может инициировать передачу данных, обращаясь, опрашивая другие узлы (станции, Slaves, Исполнители) и получая в ответ необходимые данные. Пример: подключение модулей серии ADAM-4000, где компьютер (Задатчик) может обра-щаться к модулям ADAM (к Ис-

полнителям) либо за измерительной информацией, либо для того, чтобы передать управляющую информацию (см. п. 1.3.4.4).

Преимущества:

– простота реализации;

– детерминированность, так как можно четко определять временные интервалы взаимодействия с каждым узлом.

Главный недостаток:

– для своевременного отслеживания наступления события, которое распознает подчиненный узел, необходимо его часто опрашивать, что может приводить к проблемам избыточной загруженности моноканала.

Метод передачи маркера (или маркерный доступ) состоит в том, что все узлы в сети являются равноправными, при этом в канале передается специальный пакет данных (маркер), который может помечаться свободным или занятым. Как только узел получает маркер, помеченный как свободный, он может пометить его как занятый и начать обращение к любому узлу сети (узел становится инициативным, на время становится Задатчиком). Через определенное время, которое может заранее определяться, инициативный узел «освобождает» маркер. После этого следующий узел в сети может «занять» этот маркер и стать инициативным.

Преимущества:

– детерминированность, так как можно четко определять временные интервалы активности каждого узла, установления связей между узлами и т. д.;

– каждый узел может быть инициативным, тем самым решается проблема отслеживания наступления события, свойственная методу опроса.

Главный недостаток:

– могут иметь значение задержки, связанные с ожиданием маркера.

Метод множественного доступа с контролем несущей и обнаружением столкновений (МДКН/ОС или CSMA/CD) состоит в том, что все узлы в сети являются равноправными и инициативными, при этом каждый из узлов проверяет канал, перед тем как начать обращение к другому узлу по своей инициативе. Если канал свободен, то узел начинает обращение. Может получится так, что это начнут делать сразу несколько узлов, что приведет к коллизии (столкновению). Эти узлы распознают такую ситуацию, тем самым, осуществляется контроль несущей и обнаружение столкновений.

После этого каждый из узлов прекращает передачу на случайный промежуток времени, а затем возобновляет передачу. В итоге один из узлов раньше других занимает моноканал.

Преимущества:

– каждый узел может быть инициативным, тем самым решается проблема отслеживания наступления события, свойственная методу опроса;

– нет задержек ожидания маркера, свойственных методу передачи маркера.

Главный недостаток:

– недетерминированность, так как заранее невозможно точно определить интервалы активности каждого узла, установления связей между узлами и т. д.

Рассмотрим несколько примеров ЛУВС: Industrial Ethernet; PROFIBUS; LonWorks; CAN; KNX.

Industrial Ethernet основан на методе доступа МДКН/ОС (CSMA/CD) и получил широкое распространение в большей степени для подключения компьютеров и УВО на верхних уровнях САиУ. Из-за недетерминированности используемого метода реже применяется на нижних уровнях САиУ, особенно при подключении датчиков и ИУ. Некоторые основные отличия от обычного Ethernet состоят в следующем:

– имеются специальные стандарты на кабели и разъемы для обеспечения повышенной защищенности в промышленных условиях;

– имеются специальные стандарты для связи с подвижными объектами с помощью гибких кабелей, беспроводной связи и т. д.;

– обеспечивается возможность частой передачи небольших порций информации, что вообще характерно для ЛУВС на нижнем уровне.

PROFIBUS (PROcess FIeld BUS) – широко распространенная (особенно в Европе) промышленная сеть. Для доступа к моноканалу применяется метод передачи маркера, который передается между подмножеством всех узлов (между ведущими узлами). В частности, когда это подмножество состоит только из одного узла, то метод доступа сводится к методу опроса. Тем самым, обеспечивается гибкость и детерминированность.

На физическом уровне может использоваться:

– экранированная витая пара по стандарту RS-485;

– оптоволоконный кабель;

– инфракрасный канал связи.

Выделяют несколько разновидностей PROFIBUS:

– PROFIBUS-DP (ведущие узлы – это обычно контроллеры, а ведомые – УСО, датчики, ИУ; при этом скорость передачи возможна до 12 Мбит/с);

– PROFIBUS-PA (во многом аналогична PROFIBUS-DP, но с ориентацией на взрывоопасные зоны, при этом скорость передачи понижена до десятков Кбит/с);

– PROFIBUS-FMS (ориентирована на верхний уровень САиУ для высокоскоростной связи компонентов).

LonWorks – промышленная сеть, которая часто применяется в системах автоматизации зданий (освещение, отопление, вентиляция и т. д.).

На физическом уровне может использоваться: витая пара; линии электропитания. Характеризуется объединением в сеть большого количество устройств (до нескольких тысяч), каждое из которых может выполнять свою функцию, например, функцию датчика и ИУ. При этом возможно создание контуров управления без участия УВК на основе «интеллектуальных» датчиков и ИУ.

CAN (Controller–area network) – стандарт, разработанный для обеспечения взаимодействия датчиков, ИУ и микроконтроллерных устройств между собой в составе подвижного объекта, в частности, транспортного средства, например, автомобиля.

На близких расстояниях (до 40 м) обеспечивается скорость передачи до 1Мбит/с. Большим преимуществом является возможность работы в режиме жесткого реального времени.

Этому способствует особый метода доступа, который основан на том, что разрешение коллизий осуществляется за счет того, что преимущество получает узел с наибольшим приоритетом. Приоритеты могут быть заранее назначены, что обеспечивает детерминированность.

KNX – стандарт, широко применяемый для автоматизации зданий, является объединением стандартов EIB, EHS, BatiBUS.

На физическом уровне может использоваться:

– витая пара;

– линии силового электропитания;

– сеть EIB на основе Ethernet;

– радиосвязь;

– инфракрасная связь.

Используется метод МДКН/ОС (CSMA/CD). Различают следующие группы устройств:

– входные модули (sensors) (кнопки включения, датчики температуры, влажности и т. д.);

– выходные модули (actuators) (управление электрооборудованием, информационные дисплеи, управление жалюзи и т.д.);

– контроллеры (для реализации сложных функций управления);

– дополнительные устройства (блоки питания, повторители, конвертеры интерфейсов и т. д.).

Эти рассмотренные примеры сетевых технологий показывают лишь небольшую часть возможных реализаций ЛУВС. Поэтому при анализе и выборе данного вида компьютерных технологий необходимо прорабатывать большое количество вариантов реализации ЛУВС с учетом предполагаемой архитектуры САиУ.

Дополнительную информацию по сетевым технологиям в САиУ можно найти в [7], [17], [30].

1.3.3.3. Технологии взаимодействия с человеком-оператором

Видеотерминальные средства организации рабочего места оператора – одна из наиболее широко используемых компьютерных технологий взаимодействия с человеком оператором.

При использовании этой технологии особое значение приобретает разработка программного обеспечения, реализующего человеко-машинный интерфейс (англ. *HMI – Human-Machine Interface*), в виде так называемого пользовательского интерфейса.

В пользовательском интерфейсе выделяют три основные части [28]:

1) визуальное оформление, которое предоставляет информацию оператору;

2) функциональные возможности системы, которые включают возможности для эффективного функционирования оператора;

3) техники взаимодействия оператора с системой.

В целом пользовательский интерфейс – это не только красивая динамическая «картинка», отображающая числа, текст, мнемосхему и другую визуальную информацию, но это также набор средств для формирования управляющих воздействий.

Программное обеспечение для реализации HMI может быть реализовано с помощью универсальных средств программирования, а также с помощью специализированных средств программирования, так называемых SCADA-пакетов, которые будут в дальнейшем будут рассматриваться подробно.

Развитие аппаратных и программных средств приводит к дальнейшей эволюции пользовательских интерфейсов, а именно к формированию *иммерсивных* (погружающих) сред [26, 27], использующих технологии виртуальной и дополненной реальности. Естественно, что к проектированию и разработке таких иммерсивных сред предъявляются повышенные требования к аппаратно-программному обеспечению.

В целом же следует отметить, что в настоящее время чаще всего компьютерные технологии взаимодействия с человеком-оператором предполагают наличие аппаратного обеспечения (УВМ, УВО), а также программного обеспечения, реализующего пользовательский интерфейс, функционирующего на УВМ и использующего имеющиеся УВО.

Важным элементом технологии организации визуальной информации, особенно в составе видеотерминальных средств, являются мнемосхемы.

Мнемосхемы – это условные графические изображения, отражающие состояние целевого процесса (объекта управления) и соответствующего оборудования.

Например, на экране рабочего места оператора может быть выведена мнемосхема технологической линии производства некоторой продукции. Эта мнемосхема должна отображать основные сведения о состоянии этой технологической линии, ее основных компонентов.

Применительно к мнемосхемам необходимо выделить несколько основных требований по их организации:

– мнемосхема не должна содержать элементы, которые являются лишними, и предоставляемая информация должна быть четкой и по возможности однозначной;

– на мнемосхемах желательно употреблять типовые условные обозначения или, в крайнем случае, специализированные условные обозначения, расшифровка которых должна даваться в явном виде или должна пониматься интуитивно;

– в составе мнемосхем необходимо выделять активные элементы, состояние которых можно изменить, например, с помощью указателя мыши, нажатия на сенсорный экран или с помощью клавиш.

1.3.4. Технические средства автоматизации и управления

Здесь рассмотрим подробнее основные виды технических средств автоматизации и управления, так как понимание их специфики является очень важным при решении задач анализа и выбора этих средств.

Сейчас обратимся к некоторым базовым вопросам, более детальное рассмотрение будет отнесено на практические занятия и самостоятельную работу. Но для успешного прохождения указанных этапов важно получить базовое представление о технических средствах автоматизации и управления.

1.3.4.1. Управляющие вычислительные машины (УВМ)

В качестве УВМ в САиУ в первую очередь следует рассматривать программируемые логические контроллеры (ПЛК) и промышленные компьютеры (ПК).

Промышленные компьютеры (ПК) – совместимы с обычными коммерческими («офисными») компьютерами, но адаптированы к жестким условиям функционирования. Для этого используются специальные средства защиты. ПК обладают повышенной надежностью. Различия между ПК могут быть выражены в конструктивном исполнении, в архитектуре, в схемотехнике. Они могут иметь в своем составе специальные платы ввода-вывода, которые обеспечивают сопряжение с объектом, например, АЦП, ЦАП, частотный и дис-

кретный ввод-вывод и т.д. В качестве операционных систем (ОС) применяются как спе-циализированные ОС, так и адаптированные к применению в САиУ различные коммерческие ОС.

Программируемые логические контроллеры (ПЛК) – это вычислительные устройства, специализированные для применения в составе САиУ. Могут иметь специализированную вычислительную структуру, специализированную ОС, модули ввода-вывода для сопряжения с объектом. Также обычно обладают средствами защиты и высокой надежностью. В большинстве случаев поддерживают стандарт на языки про-граммирования IEC 61131-3 [19], [16].

ПК обладают следующими преимуществами по сравнению с ПЛК.

– Являются более универсальными. В частности, обычно предоставляют возможность применения широко распространенных ОС и средств программирования.

– Более широкие возможности для хранения, накопления и обработки больших массивов данных.

– Потенциально имеют лучшие возможности для визуализации необходимой информации.

ПЛК обладают следующими преимуществами по сравнению с ПК.

– Легче реализуются специализированные конфигурации под конкретные задачи с целью уменьшения стоимости.

– Потенциально имеют лучшие возможности для создания распределенных конфигураций.

– Имеют удобные специализированные средства программирования и не требуют «тонкой» настройки системного программного обеспечения.

Дополнительную информацию по теме УВМ можно найти в [30], [6].

1.3.4.2.Датчики и измерительные преобразователи (ИП)

Датчики и измерительные преобразователи (ИП) являются важнейшим компонентом САиУ, который обеспечивает получение информации о целевом процессе (см. п. 1.1.4.2). Естественно, что при анализе и выборе датчиков следует в первую очередь обращать внимание на то, какие параметры требуется измерять, и какие датчики служат для измерения этих видов параметров. Помимо этого, важно рассматривать следующие вопросы:

– погрешности измерения при использовании данного вида датчиков;

– статические и динамические характеристики данного вида датчиков;

– размеры, энергопотребление, помехозащищенность, надежность и другие вопросы, связанные с монтажом, компоновкой, эксплуатацией датчиков и измерительных преобразователей в составе САиУ.

Рассмотрим подробнее вопрос погрешностей.

Различают абсолютную, относительную и приведенную погрешности.

Абсолютная погрешность – это значение

$$X - E,$$

где X – измеренное значение; E – истинное значение измеряемой величины.

Недостаток абсолютной погрешности: величина зависит от единицы измерения.

Относительная погрешность – это значение

$$\frac{X - E}{X} \cdot 100\%.$$

Недостаток относительной погрешности: величина зависит от текущего измеренного значения (в частности может возникать деление на 0).

Приведенная погрешность – это значение

$$\frac{X - E}{X_{max} - X_{min}} \cdot 100\%$$

где X_{min}, X_{max} – определяют диапазон $[X_{min}, X_{max}]$ возможных изменений измеряемого значения.

Также различают случайную и систематическую погрешности.

Случайная погрешность – это случайные отклонения измеренного значения (X) от истинного значения (E).

Случайную погрешность можно пытаться уменьшать:

– за счет устранения или ослабления причин, ее вызывающих (например, удаление источника помех, экранирование);

– на основе статистической обработки результатов измерений (например, на основе усреднения по нескольким повторным измерениям).

Систематическая погрешность – это различия между измеренным (X) и истинным (E) значением, которые повторяются при каждом измерении (систематически) и меняют мало (в отличие от случайной погрешности). Систематическая погрешность чаще всего проявляется именно в постоянных смещениях показаний.

Для устранения систематической погрешность обычно применяют калибровку ИП.

Также различают основную и дополнительную погрешности.

Основная погрешность – это погрешность, которая проявляется при нормальных условиях эксплуатации (заранее оговоренных производителем прибора).

Дополнительная погрешность – это погрешность, которая возникает при выходе за пределы нормальных условий эксплуатации. Она обычно выражается в процентах в зависимости от того параметра, который характеризует выход за пределы нормальных условий эксплуатации. Например, может быть так, что погрешность увеличивается на 0.5% при каждом повышении температуры на 10 градусов.

Показатели погрешностей можно считать частным случаем статических характеристик ИП, то есть таких характеристик, которые не меняются или слабо меняются в ходе функционирования датчиков.

Рассмотрим другие виды статических характеристик.

Градуировочная характеристика – это зависимость выходного значения ИП от значения контролируемого параметра (рис. 25).

Рис. 25. Пример градуировочной характеристики

Линейность градуировочной характеристики определяется близостью к прямой линии.

Важность линейности определяется тем, что при линейной зависимости легче определить контролируемый параметр, исходя из выходного значения ИП.

Если градуировочная характеристика является нелинейной, то тогда для получения контролируемого параметра надо знать хотя бы примерно функциональную (в данном случае нелинейную) зависимость между контролируемым параметром и выходным значением ИП.

Пусть эта функциональная зависимость выражается аналитически в виде $y = f(x)$, где x – контролируемый параметр; y – выходное значение ИП. Тогда для получения контролируемого параметра надо определить обратную функцию, такую что $x = f^{-1}(y)$.

Например, в случае градуировочной характеристики вида $y = x^2$ можно определять контролируемый параметр (x) по известному выходному значению ИП (y) на основе соотношения $x = \sqrt{y}$.

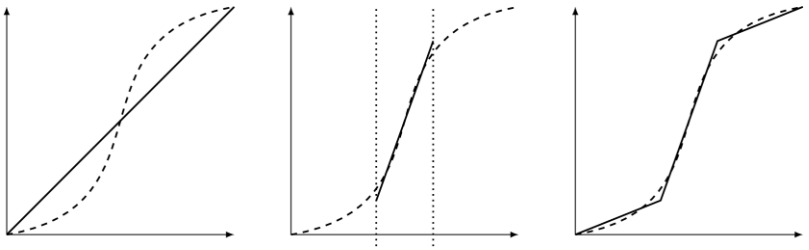
Другой способ определения значения контролируемого параметра состоит в *линеаризации* градуировочной характеристики.

Простейший вариант (рис. 26, *а*) состоит в том, чтобы в качестве узловых точек линии взять точки на концах диапазона значения контролируемого параметра. Но при этом на средних участках может возникать слишком большое расхождение, которое может приводить к возникновению погрешности.

Другой вариант (см. рис. 26, *б*) состоит в том, чтобы выделить «наиболее линейный» участок и наложить линии на него. В этом случае придется

уменьшить диапазон измеряемых значений, а это может оказаться недопустимым.

Поэтому при сильной нелинейности градуировочной характеристики можно попробовать ее заменить ломаной линией на основе кусочно-линейной апроксимации (см. рис. 26, *в*). Также возможен вариант кусочно-линейной интерполяции.



а) б) в)

Рис. 26. Пример градуировочной характеристики

При этом надо находить баланс между погрешностью и количеством отрезков.

Конечно, кроме кусочно-линейной аппроксимации и интерполяции можно применять другие методы, например те или иные интерполяционные полиномы.

Линеаризация может осуществляться средствами ИП, но может понадобиться выполнять дополнительную линеаризацию (например, если ИП не предоставляет такой возможности). В этом случае, линеаризация может выполняться как аппаратно (например, на основе преобразователя электрического сигнала), так и программно (за счет цифровой обработки сигнала).

Порог чувствительности – это минимальное изменение контролируемого параметра, при котором изменяется выходное значение ИП.

Например, в случае резистивного датчика положения, реализованного на основе проволочной обмотки, порог чувствительности примерно равен расстоянию между соседними витками.

Дрейф – это отклонение выходной величины ИП при постоянном значении контролируемого параметра в течение длительного времени.

Наличие дрейфа надо проверять при различных значениях контролируемого параметра.

Повторяемость – это отклонение между несколькими последовательными измерениями при одном и том же значении контролируемого параметра *за небольшой промежуток времени* (пока не появился дрейф).

Воспроизводимость – это величина, аналогичная повторяемости, но определяется за более длительный промежуток времени (до нескольких меся-

цев), при этом в остальное время ИП должен функционировать и использоваться по назначению.

Теперь обратимся к *динамическим* характеристикам ИП.

Пусть значение контролируемого параметра изменилось скачком. Тогда выходное значение ИП должно тоже измениться. Но это произойдет не мгновенно. В общем случае возникает переходный процесс. Параметры этого переходного процесса и отражают динамические характеристики ИП. Среди них можно выделить:

– время установления (время переходного процесса);

– максимальное превышение значения в статическом режиме (перерегулирование);

– время достижения первого максимума;

– время прохождения зоны нечувствительности (определяется моментом, когда выходная величина начинает изменяться).

Также в качестве динамических характеристик ИП могут использоваться:

– передаточная функция ИП;

– амплитудно-частотная и фазо-частотная характеристики ИП.

При выборе ИП, особенно для быстропротекающих процессов, важно оценивать значимость его динамических характеристик. Например, может оказаться, что контролируемый параметр меняется так быстро, что датчик не успевает за ним из-за своих динамических характеристик.

Все эти характеристики ИП важно знать и понимать в контексте решения задач выбора технических средств автоматизации и управления. Так, например, если ставится задача для имеющихся датчиков подобрать подходящее УСО, то разработчик должен хорошо понимать смысл различных характеристик этих датчиков, чтобы принять правильное решение по поводу УСО.

Дополнительную информацию о датчиках можно найти в [21], [9].

1.3.4.3.Исполнительные устройства (ИУ)

Рассмотрим подробнее ИУ (см. также п. 1.1.4.3), основанные на различных видах *электрических* ИМ, так как, пожалуй, именно такие ИУ в настоящее время наиболее распространены в САиУ. Важно понимать основные компоненты таких ИУ, чтобы правильно оценивать и выбирать ИУ, а также другие виды технических средств (например, УСО, которые взаимодействуют с ИУ) при проектировании САиУ.

Одна из распространенных задач в автоматизации – это точное позиционирование *сервомеханизма*, например инструмента станка или манипулятора робота. Для решения подобных задач создают *сервосистемы* управления позиционированием (рис. 27), а также сервосистемы управления скоростью.

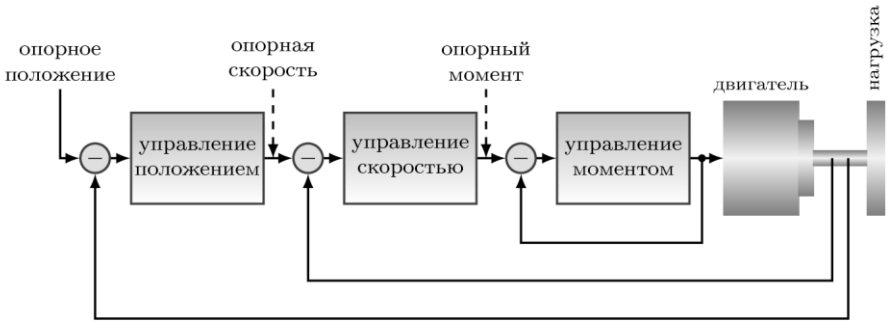


Рис. 27. Структура сервосистемы управления позиционированием

В основе сервосистемы находится электрический *двигатель*, который имеет на своем валу некоторую *нагрузку*.

Для вращения нагрузки требуется создавать момент вращения. С этой целью необходим блок *управления моментом*.

Чтобы точно управлять моментом необходимо уметь оценивать текущий момент. Это можно делать, например, измеряя ток ротора двигателя постоянного тока. Тогда можно реализовать *контур автоматического управления моментом* согласно заданному опорному значению момента.

Но в большинстве случаев надо управлять не моментом, а например, скоростью вращения. Один из примеров – управление скоростью вращения компакт-диска. Тогда потребуется блок *управления скоростью*.

Чтобы точно управлять скоростью необходимо уметь измерять текущую скорость, например, с помощью тахометра на валу двигателя. Тогда можно реализовать *контур автоматического управления скоростью* согласно заданному опорному значению скорости.

Также во многих схемах важно управлять позиционированием. Например, продольным позиционированием лазерного датчика устройства чтения компакт-дисков. Тогда потребуется блок *управления положением*.

Чтобы точно управлять положением необходимо уметь измерять текущее положение, например, с помощью дифференциального трансформаторного датчика перемещений. Тогда можно реализовать *контур автоматического управления положением* согласно заданному опорному значению положения.

В случае, когда требуется управление скоростью, то тогда, естественно, в структуре сервосистемы исключается контур управления положением, при этом получается *структура сервосистемы управления скоростью*. Если же контур управления положением остается, то тогда получается структура сервосистемы управления позиционированием (см. рис. 27).

Вообще ИУ, основанные на электродвигательных ИМ, являются одними из наиболее распространенных.

Они могут быть реализованы на основе:

– двигателей постоянного тока;

– асинхронных двигателей;

– синхронных двигателей;

– шаговых двигателей.

При выборе технических средств автоматизации и управления необходимо обращать внимание на ***характеристики электродвигательных ИУ (ЭИУ)***. К таким характеристикам, в частности, относятся:

– *Номинальный момент ИМ*. Должен достигаться при любых допустимых условиях эксплуатации.

– *Время полного хода*. Определяется допустимым временем перестановки регулирующего органа от начала до конца.

– *Выбег*. Это перемещение выходного органа ИМ после выключение двигателя. Желательно, чтобы выбег не приводил к выходу из зоны нечувствительности регулятора. Согласно ГОСТ 7192–80Е, выбег не должен быть больше: 1% для ЭИУ с временем полного хода 10 с; 0.5% для ЭИУ с временем 25 с; 0.25% для ЭИУ с временем 63 с и более.

– *Люфт*. Возникает из-за свободного хода выходного органа при неподвижном вале электродвигателя по причине зазоров в зацеплениях различных узлов. Согласно ГОСТ 7192–80Е, люфт не должен превышать: 1 градуса для однооборотных ЭИУ с номинальной нагрузкой 40 Н·м и менее; 0.75 градусов для однооборотных ЭИУ с нагрузкой более 40 Н·м; 3 градуса для многооборотных ЭИУ; 0.2 мм для прямоходных ЭИУ с нагрузкой до 1000 Н; 0.5 мм для прямоходных ЭИУ с нагрузкой свыше 1000 Н.

– *Гистерезис*. Возникает между положением выходного органа и сигналом датчика положения из-за люфта механической передачи и вариаций показания датчика.

Отдельный вопрос – это ***управление ИМ с помощью компьютера***. Проблема в том, что мощность выходного порта компьютера обычно очень мала (порядка 100 мВт). Этот сигнал надо усиливать, чтобы управлять большинством ИМ. С этой целью используются *управляемые выключатели*. Рассмотрим основные варианты реализации управляемых выключателей.

1) *Электромеханическое реле*. Когда через обмотку реле протекает ток, он создает магнитное поле, которое перемещает якорь, переключающий контакты. Эти контакты коммутируют токи, которые могут быть намного больше, чем токи управляющие реле. Тем самым, достигается усиление сигнала. Для подключения реле к компьютеру обычно используются промежуточные транзисторные усилители, так как мощности выходного порта компьютера, как правило, не хватает для управления реле.

Преимущества: простота конструкции, надежность.

Недостатки: относительно низкое быстродействие; дребезжание контактов

2) *Твердотельные полупроводниковые приборы*. Лишены основных недостатков реле. Можно выделить следующие варианты реализации:

– интегральные схемы с транзисторным выходом (для малой и средней мощности);

– пороговые или полевые МОП-транзисторы (для большей мощности);

– тиристоры (для еще большей мощности).

Дополнительную информацию по теме ИУ можно найти в [2], [4], [13], [24].

1.3.4.4. Устройства связи с объектом (УСО)

Можно выделить следующие виды УСО.

Централизованные УСО (пространственно обычно располагаются рядом с УВМ).

Среди централизованных УСО можно выделить:

– УСО с *последовательным* вводом аналоговых сигналов.

– УСО с *параллельным* вводом аналоговых сигналов.

Распределенные УСО (пространственно могут располагаться на значительных расстояниях от УВМ).

Рассмотрим УСО с ***последовательным*** вводом аналоговых сигналов (рис. 28).

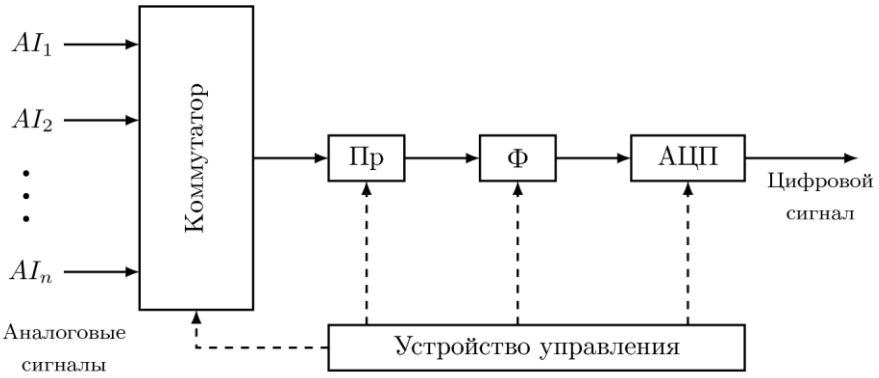


Рис. 28. Структурная схема УСО с последовательным вводом аналоговых сигналов

В таких УСО аналоговые сигналы последовательно поступают на аналого-цифровой преобразователь (АЦП) за счет коммутатора. Сигнал с устройства управления приводит к тому, что коммутатор подключает один из входных сигналов (AI_i) на вход измерительного канала, состоящего из преобразователя (Пр), фильтра (Ф) и АЦП. Преобразователь (Пр) может выполнять: усиление; линеаризацию; масштабирование; специальные алгоритмы обработки сигнала. Параметры преобразования задаются устройством управления.

Фильтр (Ф) выполняет фильтрацию сигнала (например, ослабление высокочастотных помех). Параметры фильтрации могут задаваться устройством управления. АЦП формирует цифровой сигнал для выбранного аналогового сигнала (AI_i). Устройство управления может выдавать команду для запуска аналого-цифрового преобразования, а также может изменять некоторые параметры этого преобразования.

Преимущество данной схемы состоит в экономии аппаратной части (один Пр, один Ф, один АЦП).

Недостаток состоит в том, что время измерения всех сигналов зависит от их числа (n), что может приводит к существенным задержкам. Например, если на измерение каждого аналогового сигнала тратиться 30 мкс, то для последовательного измерения 10-ти таких сигналов потребуется 300 мкс.

В качестве примера УСО такого типа рассмотрим карту ввода-вывода 5700 Octagon Micro PC (рис. 29).

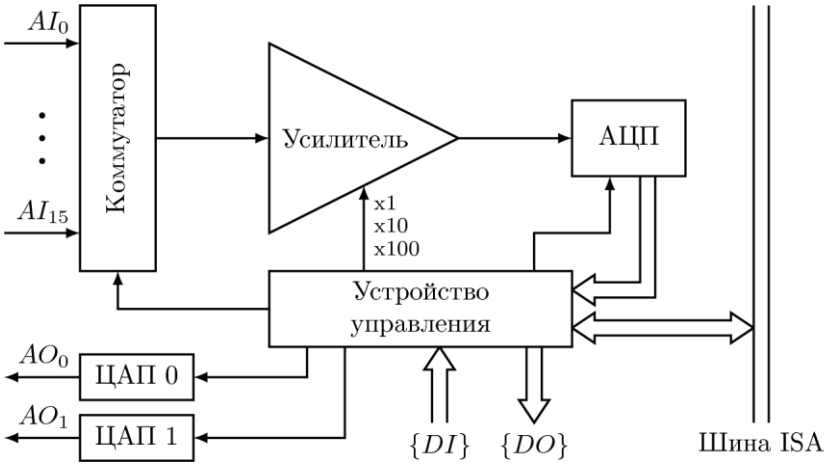


Рис. 29. Структурная схема карты ввода-вывода 5700 Octagon Micro PC

Данная карта ввода-вывода присоединяется к компьютеру на основе шины ISA. У нее имеется 16 коммутируемых каналов ввода аналоговых сигналов (AI_0 …, AI_{15}). Устройство управления, находящееся в составе платы, выбирает канал с помощью коммутатора. При коммутации каждого из этих каналов сигнал может быть усилен. Один из трех коэффициентов усиления (x1, x10 ,x100) устанавливается с помощью устройства управления.

Также устройство управления может выполнять настройку аналого-цифрового преобразователя (АЦП). В частности, может быть задана автокомпенсация нуля, а также автокалибровка. Кроме того, устройство управления выдает команду начала аналого-цифрового преобразования. АЦП является 12-ти разрядным (плюс еще один знаковый разряд). Диапазон входных значений АЦП: [-5,+5]В. Время одного преобразования составляет примерно 15

мкс. При включении автокомпенсации нуля это время увеличивается примерно в 2 раза.

Цифровой код, полученный после аналого-цифрового преобразования, поступает в соответствующий регистр устройства управления. Этот код посредством шины ISA может быть прочитан для дальнейшей его обработки управляющим компьютером.

Также плата имеет два аналоговых канала вывода, реализованных на основе двух независимых цифро-аналоговых преобразователей (ЦАП). Диапазоны выходных сигналов (AO_0, AO_1) задаются специальными перемычками на плате. Можно устанавливать следующие диапазоны: [0,+5]В, [0,+10]В, [-5,+5]В.

Кроме того, предоставляется возможность для работы с дискретными каналами ввода-вывода $\{DI\}$, $\{DO\}$.

Плата ввода-вывода обладает хорошими эксплуатационными характеристиками. В частности, она имеет: рабочий температурный диапазон от -40°С до +85°С; устойчивость к вибрациям до 5g и к ударам до 20g.

В качестве входных аналоговых сигналов могут использоваться сигналы от ИП, например, от термопар. Выходные сигналы с этой карты ввода-вывода могут поступать на управляющие входы ИУ. Например, сигналы $\{DO\}$ могут использоваться для управления шаговым двигателем (см. п. 1.3.4.3).

Другой вид централизованных УСО – это УСО с ***параллельным*** вводом аналоговых сигналов (рис. 30).

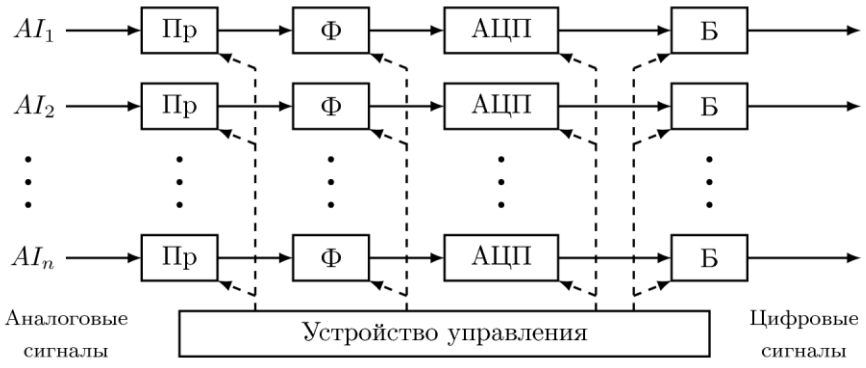


Рис. 30. Структурная схема УСО с параллельным вводом аналоговых сигналов

Для каждого из входных сигналов (AI_i) имеется свой измерительный канал, состоящий из преобразователя (Пр), фильтра (Ф), АЦП и буферного устройства (Б). Устройство управления задает параметры для каждого Пр, Ф, АЦП, а также выдает команды для запуска каждого АЦП. Полученные цифровые сигналы накапливаются в буферных устройствах (Б). По команде устройства управления выполняется считывание цифровых сигналов из этих буферных устройств.

Преимущество данной схемы состоит в уменьшении задержек при измерении n сигналов, так как для каждого из сигналов есть свой измерительный канал.

Недостаток состоит в увеличении аппаратной части (n измерительных каналов).

В качестве примера УСО такого типа рассмотрим систему, реализованную на основе универсальной карты ввода-вывода UNIO96 и модулей ввода-вывода GrayHill (рис. 31).

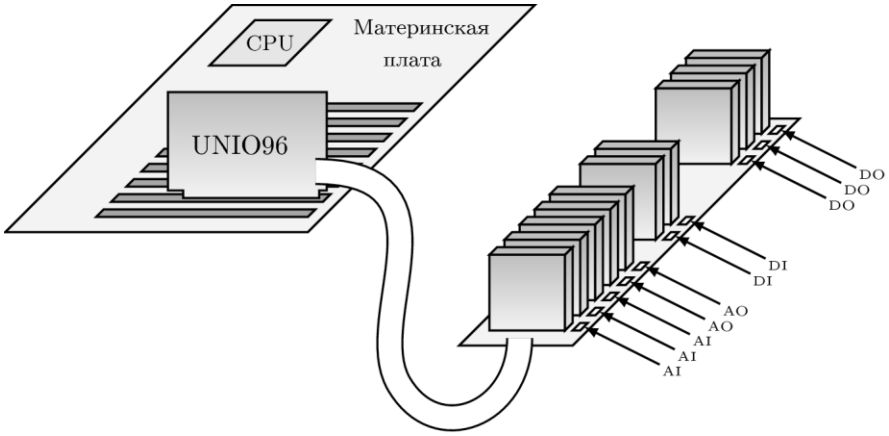


Рис. 31. Пример УСО с параллельным вводом аналоговых сигналов (система на основе платы UNIO96 и модулей GrayHill)

В данном случае на материнской плате компьютера в один из слотов ISA устанавливается универсальная карта ввода-вывода UNIO96. Она имеет 4 программируемые логические матрицы (ПЛМ), каждая из которых может работать независимо. С помощью специального шлейфа к плате UNIO96 подключается выносная панель. В данном примере к плате UNIO96 подключена только одна панель, но может быть подключено до 4-х подобных панелей (по числу ПЛМ на плате UNIO96). На этой панели располагаются модули GrayHill для взаимодействия с каналами *AI, AO, DI, DO*. Каждый модуль для *AI* имеет АЦП, а также может осуществлять масштабирование и линеаризацию сигнала. Каждый модуль для *AO* имеет ЦАП. Кроме того, модули реализуют гальваническую развязку. Данный пример можно отнести к УСО с параллельным вводом аналоговых сигналов, так как для каждого аналогового входа имеется отдельный АЦП в соответствующем модуле GrayHill. В случае данного примера УСО обеспечивается гальваническая развязка для каждого модуля аналогового ввода. Кроме того, обеспечивается гибкость конфигурации, так как имеется возможность подбора необходимых модулей ввода-вывода.

В случае ***распределенных*** УСО сложно привести обобщенную структурную схему, которая могла бы охватить многие виды таких УСО и при этом не

быть слишком абстрактной. Поэтому рассмотрим конкретный пример УСО данного типа.

В качестве примера будут рассматриваться модули Advantech ADAM-4000. Для передачи данных у этих УСО используется RS-485 с максимальной скоростью в 19200 бит/с и максимальным расстоянием 1200 м. До 256 модулей может использоваться на один последовательный порт компьютера, при этом предполагается, что имеется до 32 модулей в сегменте, а сегменты разделяются повторителями. Модули питаются постоянным напряжением в 24В. Могут работать при температуре окружающей среды от -10° С до +70° С и влажности от 5% до 95%.

Различают модули *AI* (для приема аналоговых сигналов), модули *AO* (для вывода аналоговых сигналов), модули *DI* (для приема дискретных сигналов), модули *DO* (для вывода дискретных сигналов).

Могут использоваться различные топологии подключения этих модулей.

При *шинной* топологии к УВМ подключается модуль-конвертер (К) ADAM-4520, который обеспечивает стыковку RS-232 от УВМ и RS-485, к которому подключаются остальные модули (М), находясь на одной шине (см. рис. 32).

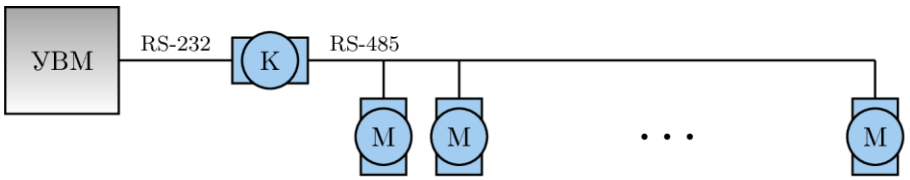


Рис. 32. Шинная топология подключения модулей ADAM-4000

При *радиальной* топологии используются модули-повторители (П) ADAM-4510, каждый из которых формирует отдельную шину с модулями (М) (см. рис. 33). Каждая такая шина формирует отдельный «радиус» для реализации этой топологии.

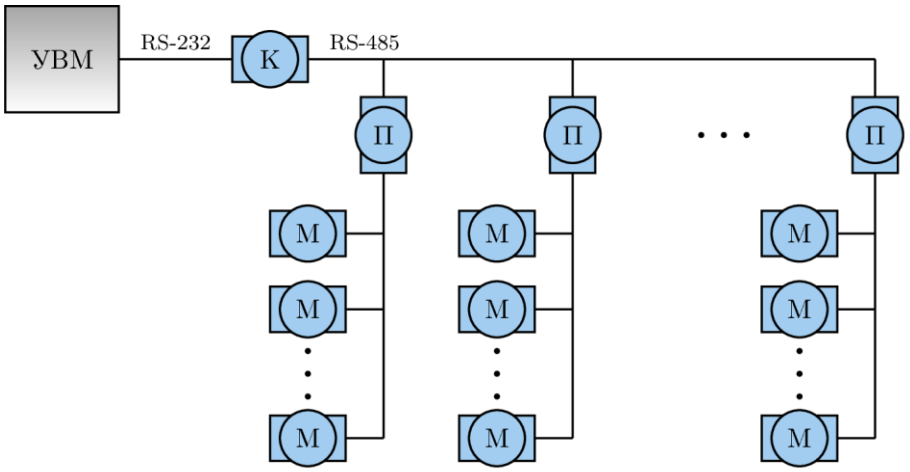


Рис. 33. Радиальная топология подключения модулей ADAM-4000

Древовидная (или смешанная топологии) реализуется за счет различных вариантов расположения повторителей (П), к которым подключаются шины с модулями (М) (см. рис. 34).

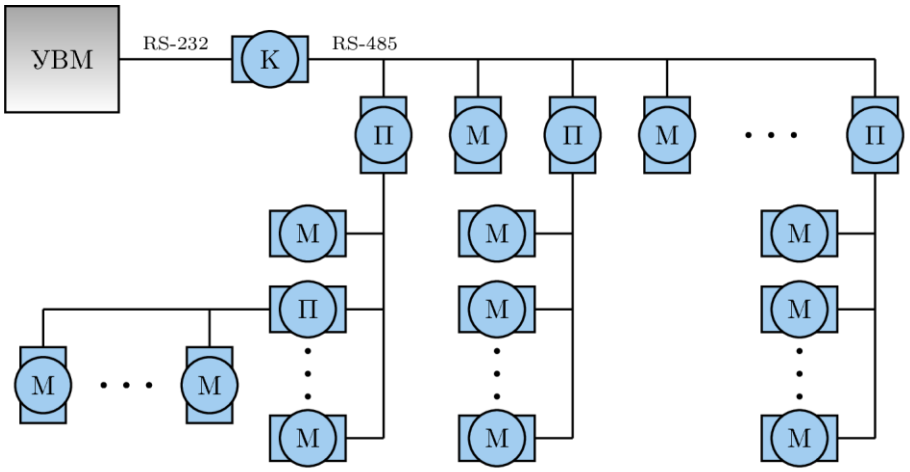


Рис. 34. Древовидная топология подключения модулей ADAM-4000

На примере различных топологий сетей взаимодействия модулей ADAM-4000 легко увидеть наличие сложных взаимосвязей между проблемами выбора архитектуры САиУ, компьютерных технологий и технических средств (в данном случае УСО).

Дополнительную информацию по теме УСО можно найти в [14], [26].

1.3.4.5. Устройства взаимодействия с оператором (УВО)

При выборе дисплеев, сенсорных экранов и других устройств отображения информации необходимо учитывать не только требования информационной совместимости, надежности, эксплуатационных качеств, но и эргономические требования [27], [28].

Среди различных видов УВО отдельно надо упомянуть *операторские панели*, которые часто используются в составе программно-технических комплексов автоматизации [7]. Операторские панели представляют собой совмещение в едином конструктиве дисплея и компьютера, обычно с некоторым набором функциональных клавиш. То есть это тоже может рассматриваться как объединение УВМ и УВО (см. п. 1.3.2.4). Операторские панели удобно располагать вне диспетчерского зала, в непосредственной близости от объекта управления. С помощью этих панелей можно прямо на объекте оценить количественные показатели процесса, а также оказать некоторые управляющие воздействия.

Вообще, операторские панели можно считать удаленной частью рабочего места оператора, например, в рамках архитектуры, изображенной на рис. 22,

либо одним из нескольких рабочих мест оператора, например, в рамках архитектур, изображенных на рис. 18, 19.

Хотя наибольшей пропускной способностью у человека выделяется визуальный канал получения информации, надо учитывать и другие варианты восприятия информации о внешнем мире. Поэтому оператору может предоставляться информация о целевом процессе не только в форме визуальной информации. Другие формы определяются особенностями сенсорной системы человека, а именно органами его чувств.

Конечно, здесь надо отметить важность звуковой информации. Особенно это касается организации различных оповещений, например, об аварийных и нештатных ситуациях. И здесь выделяют звуковые сообщения:

– речевые, содержащие более детальную информацию о событии;

– предупредительно-звуковые, которые не содержат детальной информацию о событии, но привлекают внимание оператора к событию и побуждают его к получению детальной информации, например, в визуально-текстовой форме.

Помимо визуальной и звуковой информации также определенное значение может придаваться тактильной информации. Особенно это касается информации обратной связи о срабатывании органов управления.

Выбор органов управления для конкретного рабочего места оператора определяется экспертным путем и зависит от особенностей управляющих алгоритмов, а также от традиций проектирования, принятых в данной разрабатывающей организации [28].

1.4. Разработка программного обеспечения систем автоматизации и управления

1.4.1. Специфика программного обеспечения САиУ

Основные отличительные особенности специального (см. определение в п. 1.1.3.2) программного обеспечения САиУ состоят в следующем.

– *Высокая вероятность отсутствия ошибок в работе* («надежность»). Обеспечивается тестированием, верификацией. Большую роль здесь играет технология программирования.

– *Открытость*. В большинстве случаев специальное ПО для САиУ не является отдельным товаром в отрыве от всей САиУ, поэтому часто нет смысла защищать это ПО от копирования и скрывать исходные тексты от заказчика. Как раз наоборот, открытость ПО с точки зрения заказчика упрощает процесс его возможных модификаций и, в целом, процесс его сопровождения, в том числе, и другими исполнителями работ.

– *Модульность*. Возможность изменения отдельных подсистем (модулей) с наименьшими затратами. Например, путем обеспечения минимального взаимодействия, взаимовлияния между модулями. Здесь встают вопросы декомпозиции, построения архитектуры ПО.

Необходимость в модульности, как и в открытости, возникает, во многом, из-за возможных проблем при сопровождении и дальнейшей модификации системы.

– *Выполнение требований реального времени* (см. п. 1.3.3.1). Эти требования должны обеспечивать на системном уровне за счет обеспечения взаимодействия задач реального времени и за счет их эффективного планирования. Кроме того, эти требования должны выполняться и на уровне отдельных задач за счет, например, анализа времени их выполнения, а также последующей оптимизации их выполнения с точки зрения характеристик реального времени.

– *Существенные различия между ПО верхних и нижних уровней САиУ*. Нижние уровни САиУ характеризуются близостью к целевому процессу, что обуславливает, как правило, более жесткие требования к реальному времени, а также специфичность УВМ. Например, на нижних уровнях в качестве УВМ часто используются не промышленные компьютеры, а ПЛК (см. п. 1.3.4.1), которые предъявляют особые требования к ПО и средствам его разработки. Верхние же уровни САиУ характеризуются близостью к человеку-оператору, что предполагает, как правило, использование программных средств реализации тех или иных вариантов человеко-машинного интерфейса (см. п. 1.3.3.3).

1.4.2. Разработка программного обеспечения нижних уровней САиУ

На нижних уровнях САиУ, как правило, разработка программного обеспечения осуществляется в рамках следующих подходов:

– разработка ПО для встроенных систем;

– разработка ПО для ПЛК.

При разработке ПО для ***встроенных систем*** многое определяется особенностями вычислительной системы, например, типом процессора, наличием или отсутствием специальных средств программирования.

В конечном итоге для разработки ПО определяется тот или иной язык программирования и соответствующие инструментальные средства. Среди языков программирования обычно выделяют два класса языков:

– языки программирования низкого уровня (ассемблер и другие языки, близкие к программированию в машинных кодах);

– языки программирования высокого уровня (языки с достаточным уровнем абстракции, например, Си, Си++, Паскаль, Java, Python и др.).

Низкоуровневые языки программирования обычно используются в тех случаях, когда требуется дополнительная оптимизация ресурсов (процессорного времени, памяти), или когда нет возможности использования языков высокого уровня. В остальных случаях чаще всего используются высокоуровневые языки программирования.

Особо следует отметить язык программирования Си, который является достаточно низкоуровневым, хотя его и следует отнести к классу языков высокого уровня. В частности, это проявляется в том, что он является, пожалуй, самым популярным языком системного программирования. Например, этот язык используется для разработки ядра Linux. Также с помощью него обычно создаются драйверы устройств, что часто бывает необходимым в случае разработки нового технического устройства или при подключении нестандартного устройства. В частности, язык Си также используется в качестве базового в монографии [38], в которой излагаются основные особенности разработки программного обеспечения для систем реального времени, что актуально именно применительно к нижним уровням САиУ. В учебном пособии [18], где рассматриваются особенности разработки программного обеспечения встроенных вычислительных систем, также в качестве базового используется язык программирования Си.

При разработке ПО для ***ПЛК*** многое определяется особенностями конкретного ПЛК и средств программирования, предназначенных для данного ПЛК. Унификация процесса разработки ПО для ПЛК становится возможной благодаря международному стандарту IEC 61131-3 [19], [16].

Этот стандарт обеспечивает единство средств и методов программирования для различных видов ПЛК.

Стандарт определяет следующие языки.

– IL (instruction list или язык списка команд). Язык реализует набор машинных команд. Этот язык напоминает Ассемблер.

– FBD (function block diagram или язык схем функциональных блоков). Язык основан на построении схем соединения функциональных блоков для реализации логических функций. Например, логические И, ИЛИ реализуются в виде соответствующих функциональных блоков AND, OR.

– LD (ladder diagram или язык релейных схем). Язык основан на имитации релейных схем для реализации логических функций. Например, логическое И реализуется в виде последовательного соединения ключей, а логическое ИЛИ – в виде параллельного соединения.

– ST (structured text или язык структурированного текста). Язык программирования с синтаксисом, похожим на синтаксис языка Pascal.

– SFC (sequential function chart или язык функциональных карт). Используется для анализа задач управления методом «сверху вниз» и описания

управляющих последовательностей. По своему внешнему виду немного напоминает блок-схемы алгоритмов.

Благодаря этому стандарту, становится возможным разрабатывать ПО, которое не очень сильно зависит от конкретного типа ПЛК. В идеальных случаях ПО, разработанное для одного ПЛК, может почти без изменений переносится на ПЛК другого производителя. Но в реальности может возникать необходимость в существенной доработке или модификации этого ПО, учитывая особенности конкретного ПЛК.

В рамках учебных дисциплин «Системное программное обеспечение управляющих систем реального времени», а также «Проектирование встроенных управляющих систем реального времени», которые является частью магистерской программы 22040051.68 «Распределенные компьютерные информационно-управляющие системы», будут подробнее рассматриваться вопросы, связанные с разработкой ПО нижних уровней САиУ. В рамках данного учебного пособия более пристальное внимание уделим разработке ПО верхних уровней САиУ, и этому вопросу будет посвящена существенная часть материала оставшейся части пособия.

1.4.3. Основные классы инструментальных средств разработки программного обеспечения верхних уровней САиУ

Можно выделить следующие два основных класса инструментальных средств разработки программного обеспечения САиУ:

– универсальные среды программирования;

– SCADA-пакеты.

Универсальные среды программирования характеризуются следующими основными признаками:

– они могут применяться для разработки программного обеспечения различных видов, а не только программного обеспечения САиУ;

– в их основе – язык программирования достаточно широкого спектра применения, например: C++, *Java, Python.*

Можно выделить следующие основные особенности SCADA-пакетов:

– они предназначены для разработки программного обеспечения в составе автоматизированных систем управления техническими объектами и в первую очередь – программного обеспечения верхних уровней таких систем, в частности, как подсистемы взаимодействия с оператором-диспетчером;

– базовые средства разработки основаны на широком использовании графических средств, например, настройка различных параметров и формирование связей с использованием графического интерфейса;

– разработанное программное обеспечение обычно выполняется (интерпретируется) в специальной среде, а не напрямую, как исполняемый файл в рамках операционной системы.

1.4.4. О терминах «SCADA-система» и «SCADA-пакет»

Термин «SCADA-система» известен достаточно широко, но с его применением связана проблема, вызванная его существенной неоднозначностью. Рассмотрим подробнее эту проблему.

Аббревиатуру SCADA (*Supervisory Control And Data Acquisition*) можно перевести как «диспетчерское управление и сбор данных». И здесь мы обращаемся к проблематике архитектур систем автоматизации и управления. Концепция SCADA предполагает наличие некоторого главного узла MTU (*Master Terminal Unit*) и множества подконтрольных удаленных узлов RTU (*Remote Terminal Unit*), которые все вместе объединяются с помощью коммуникационной системы CS (*Communication System*) в одну систему автоматизации [15], [29], [32], [33], [34], [35], [36], [37]. При этом MTU характеризуется наличием развитых средств человеко-машинного интерфейса HMI (*Human Machine Interface*), обычно реализованного с помощью вычислительной техники и соответствующих средств взаимодействия с человеком, а также выполняет еще целый ряд специфических функций. В качестве RTU могут выступать как универсальные вычислительные устройства, например программируемые логические контроллеры, так и специальные устройства связи с объектом, которые могут быть конструктивно и функционально объединены с датчиками и исполнительными устройствами.

Отдельный вопрос – это программное обеспечение тех систем автоматизации, которые проектируются по принципам SCADA. Существует достаточно много специализированных средств реализации программного обеспечения в рамках концепции SCADA. В качестве примеров можно отметить: *TRACE MODE, InTouch, iFIX, CitectSCADA* и др.

Подобные программные средства часто называют «SCADA-системами». Однако этот термин обладает неоднозначностью, так как его можно интерпретировать не только как «программную систему для архитектуры SCADA», но и как «систему автоматизации, реализованную согласно архитектуре SCADA».

Действительно, употребление во втором смысле очень часто встречается в англоязычной литературе, см. например [32], [33], [37]. Также и в рускоязычной литературе встречается аналогичное понимание данного термина, например, в работах [15], [20], [29].

В целом можно сказать, что сложилась следующая ситуация. В русском языке термином «SCADA-система» чаще называют именно программный про-

дукт, используемый для реализации ПО систем автоматизации и управления. В английском же языке подобные программные продукты обычно именуются как «SCADA software» (SCADA ПО), «automation software» (ПО для автоматизации), «SCADA suite» (SCADA-набор), «SCADA package» (SCADA-пакет) и т.д. Такое различие достаточно заметно, например, при просмотре русскоязычных и англоязычных официальных сайтов производителей подобных программных продуктов.

При этом в англоязычных научных статьях (см., например, [34], [35], [36]) термин «SCADA system» обычно используется применительно к архитектуре системы автоматизации. Например, в работе [35] на первом рисунке изображена обобщенная топология SCADA-системы, и далее описываются компоненты SCADA-системы, под которыми понимаются: датчики и исполнительные устройства; RTU; MTU.

В работах на русском языке встречаются различные случаи употребления термина «SCADA-система», и в целом интересно посмотреть, как проявляется существенная неоднозначность данного термина.

Во многих источниках (см., например, [1], [26]) термин «SCADA-система» используется только применительно к соответствующим программным продуктам.

В работе [12] уже в заголовке дано указание на рассмотрение SCADA-системы как программного инструмента.

В работе [6] в основном используются термины «система SCADA» и «система типа SCADA», но именно для указания на архитектурные особенности системы, а не на программный продукт.

В работах [15], [20], [29] под SCADA-системами понимается определенный класс систем автоматизации и управления, обладающий соответствующими архитектурными особенностями. И это достаточно хорошо соотносится с англоязычной литературой [32], [33], [34], [35], [36], [37]. При этом в статье [29] используется отдельный термин «программные продукты класса SCADA/HMI». В статье [20] под SCADA-системой понимается комплекс аппаратного и программного обеспечения для реализации нужных функций применительно к контролируемой энергосистеме. В работе [15] есть отдельный раздел с названием «SCADA-системы», где излагается «архитектурное» понимание этого термина.

Здесь приводится малое число примеров, так как основная цель – это не анализ особенностей применения данного термина в большом массиве статей, книг, интернет-ресурсов и т.д., а целью является лишь указание на проблему различных толкований одного и того же термина.

Конечно, при понимании двойственной природы термина «SCADA-система» обычно не составляет труда разобраться, в каком смысле употребля-

ется этот термин в том или ином контексте. Однако у человека, который не очень хорошо знаком с данной тематикой, подобная ситуация может вызывать вопросы. Поэтому предлагается следующий подход. Вместо термина «SCADA-система» использовать отдельный термин для программных продуктов и отдельный термин для описания архитектурных особенностей системы.

Можно порекомендовать для обозначения программных продуктов использовать термин «SCADA-пакет», тем более что этот термин довольно часто используется как в русскоязычных текстах (см., например, [23]), так и в англоязычных (см., например, [32], где применяется термин «SCADA package»).

В свою очередь, для указания на архитектурные особенности системы автоматизации можно посоветовать пользоваться, например, термином «система с архитектурой SCADA», а в каких-то случаях – просто «архитектура SCADA».

Именно этот подход представляется более разумным, так как он позволяет четко разграничить понятия и использовать отдельные термины без риска неправильной трактовки. При этом, конечно, желательно объяснять термин «SCADA-системы» и отмечать его существенную неоднозначность. Поскольку этот термин используется повсеместно и широко, то читатель должен понимать особенности его использования в зависимости от контекста.

Еще одним преимуществом предложенного подхода является уменьшение рисков неверного перевода с английского на русский (и обратно). Например, при прочтении названия книги [37] человек, привыкший использовать термин «SCADA-система» в смысле программного продукта, может ошибочно подумать, что эта книга посвящена информационной безопасности именно программных продуктов. Вероятность подобной ошибки будет гораздо меньше у человека, который оперирует терминами «SCADA-пакет» и «система с архитектурой SCADA», и если он знает про неоднозначность термина «SCADA-система».

Также применение данного подхода позволяет давать более четкие наименования. Например, в работе [8] используется словосочетание «гибридный SCADA-пакет», и сразу становится понятно, что речь идет о программном пакете, обладающем какими-то гибридными свойствами. Но если бы использовалось словосочетание «гибридная SCADA-система», то это порождало бы неоднозначность. Например, мог бы возникать вопрос: здесь подразумевается гибридный программный пакет или система автоматизации с какой-то гибридной архитектурой?

Таким образом, в качестве итога можно сформулировать следующую рекомендацию. Вместо неоднозначного термина «SCADA-система» лучше использовать либо «SCADA-пакет» (в случае указания на программный про-

дукт), либо «система с архитектурой SCADA» (в случае указания на систему автоматизации, а также особенности построения ее программно-аппаратного комплекса). Кроме того, надо понимать неоднозначность термина «SCADA-система» и уточнять его для себя, учитывая контекст, в котором этот термин используется.

1.4.5. Организация и основные функции современных SCADA-пакетов

1.4.5.1. Выбор примера SCADA-пакета

Во многом все SCADA-пакеты похожи по своим базовым принципам построения и базовым компонентам. Конечно, есть и существенные отличия между разными SCADA-пакетами, но есть и общие черты, которые и проявляются в общности базовых компонентов.

Базовые принципы построения и базовые компоненты SCADA-пакетов удобно рассмотреть на примере достаточно простого SCADA-пакета, который называется *Advantech Genie v2.0*. В дальнейшем для краткости он будет называться просто *Genie*.

Надо отметить, что этот SCADA-пакет был выпущен уже очень давно, однако он очень простой в освоении, имеет очень простой интерфейс. Поэтому он наиболее удобен для первоначального знакомства с основными особенностями SCADA-пакетов и принципами работы с ними.

Важно то, что, разобравшись с базовыми принципами построения и базовыми компонентами на примере *Genie*, гораздо легче приступать к работе с современными SCADA-пакетами, которые, как правило, имеют более сложный и разветвленный интерфейс, что обусловливается тем, что они предоставляют гораздо больше функций и средств для работы.

Еще следует отметить, что в дальнейшем будет более подробно рассматриваться современный и сложный SCADA-пакет TRACE MODE. И в ходе знакомства с ним будут упоминаться базовые принципы построения и базовые компоненты SCADA-пакета, рассмотренные ранее на примере *Genie*. При этом можно будет проследить соответствия, то есть будет увидеть, как тот или иной аспект, представленный в TRACE MODE, находит свое соответствие в более простом SCADA-пакете *Genie*.

Также читателю рекомендуется выполнить приведенные в пособии примеры на другом сложном SCADA-пакете, например, CitectSCADA [11].

Такой подход достаточно удобен, так как в *Genie* многие базовые компоненты реализуются более наглядно («графически»), чем в TRACE MODE и CitectSCADA. И здесь надо подчеркнуть, что это не является недостатком TRACE MODE и CitectSCADA. Просто эти пакеты являются более сложными и предназначены для работы с более значительным количеством объектов и

связей, а большое количество объектов и связей удобнее представлять не графически, а таблично, в виде списков. И такой вариант представления более сложен при начальном знакомстве. Вот поэтому для первого знакомства с базовыми принципами построения и базовыми компонентами SCADA-пакетов был выбран простой SCADA-пакет *Advantech Genie*. При этом в дальнейшем будет использоваться демонстрационная версия этого SCADA-пакета, называемая *Advantech Genie Starter 2.0*. Как уже было ранее отмечено, для простоты в дальнейшем вместо этого длинного названия будет использоваться краткое название *Genie*.

1.4.5.2.Принцип организации SCADA-пакета на основе двух базовых модулей

SCADA-пакет *Genie*, как и, пожалуй, большинство SCADA-пакетов, работает в условиях операционной системы семейства *Windows*. После установки *Genie* в меню программ операционной системы формируется совокупность ярлыков, среди которых надо выделить два основных:

– «Advantech Genie Starter»;

– «Advantech Genie Starter Runtime».

Первый из них (*Advantech Genie Starter*) запускает программу-редактор для создания и редактирования алгоритмической основы и графического интерфейса, которые будут записаны в результирующий файл.

Второй, *Advantech Genie Starter Runtime*, или просто *Runtime*, служит для запуска программы-интерпретатора, которая «выполняет» тот файл, который был создан программой-редактором.

По сути, это два базовых модуля почти любого SCADA-пакета:

– программа-редактор;

– программа-интерпретатор.

У этих двух модулей могут быть разные названия в разных SCADA-пакетах, однако они, как правило, в них присутствуют.

При этом надо отметить, что программа-редактор также может выполнять определенные функции программы-интерпретатора, в частности, запускать разрабатываемую систему, что важно, например, для промежуточных проверок того, что получается при создании системы. И эта возможность программы-редактора часто используется во многих SCADA-пакетах.

Но вот в SCADA-пакете *Genie* эти две программы жестко разделены. Программа-редактор формирует файл с расширением «gni», то есть название файла вида «filename.gni», а программа-интерпретатор «выполняет» (интерпретирует) этот файл. На данном этапе первого знакомства со SCADA-пакетом это даже лучше, так как можно более четко увидеть различие между этими двумя базовыми модулями SCADA-пакета.

1.4.5.3.Особенности применения современных SCADA-пакетов при проектировании систем автоматизации и управления

SCADA-пакеты являются одним из инструментальных средств *разработки* программного обеспечения при проектировании САиУ. Но также важно, что SCADA-пакет предоставляет средства для *выполнения* программного обеспечения в ходе последующего функционирования САиУ, и эту особенность необходимо также учитывать при проектировании САиУ.

Схематично процесс создания результирующего программного обеспечения с помощью SCADA-пакета представлен в общем виде на рис. 35.

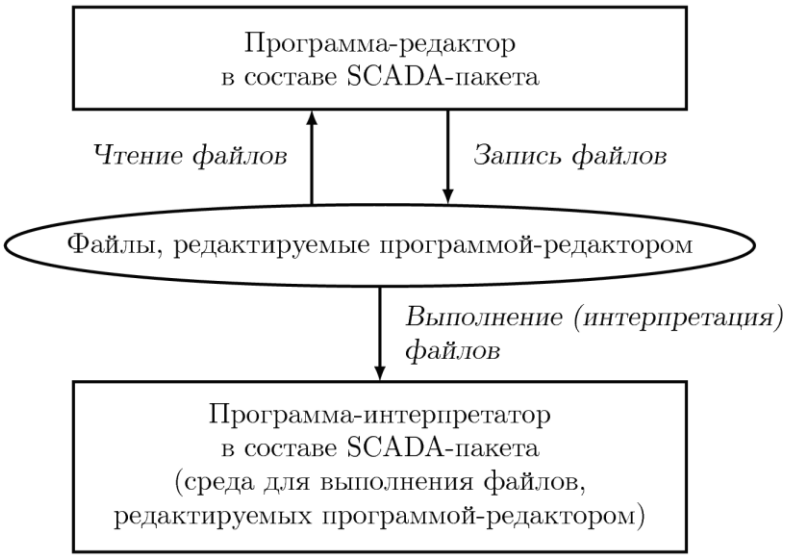


Рис. 35. Процесс создания программного обеспечения
с помощью SCADA-пакета

Из этой схемы видно, что программ-редактор (в случае *Genie* это программа *Advantech Genie Starter*) формирует один или несколько файлов. Так, в случае SCADA-пакета *Genie* формирует один файл с расширением «gni». Эти файлы (или единственный файл) поступают на вход программы-интерпретатора (в случае *Genie* это программа *Advantech Genie Starter Runtime*).

Программа-интерпретатор выполняет (интерпретирует) эти файлы, формируя, в частности, необходимый графический интерфейс с пользователем и выполняя необходимые алгоритмы. «Описание» графического интерфейса и этих алгоритмов как раз и находится в файлах, формируемых программой-редактором.

Таким образом, программа-интерпретатор обеспечивает среду для выполнения файлов, создаваемых с помощью программы-редактора.

Все дело в том, что сами по себе файлы, создаваемые программой-редактором, не могут выполняться в рамках операционной системы, так как они не являются исполняемыми файлами для этой операционной системы.

Вообще, подавляющее большинство SCADA-пакетов (и, в частности, *Genie*) с помощью программы-редакторы обеспечивают создание одного или нескольких файлов, которые не являются исполняемыми файлами для операционной системы.

Например, в рамках операционных систем семейства *Windows* исполняемыми файлами считаются файлы с расширением «exe». Эти файлы можно запустить, нажав на них, например, с помощью мыши. Так вот, результатом работы программы-редактора SCADA-пакета *Genie* являются не файлы с расширением «exe», а файлы с расширением «gni» (как и было указано выше), и эти файлы (с расширением «gni») нельзя просто так запустить в рамках операционной системы *Windows*. Эти файлы должны «действовать совместно» с программой-интерпретатором, и в такой связке они и обеспечивают работу программного обеспечения, создаваемого с помощью SCADA-пакета.

Таким образом, программа-редактор в составе SCADA-пакета формирует один или несколько неисполняемых файлов. Эти файлы исполняются не операционной системой, а интерпретируются программой-интерпретатором в составе SCADA-пакета. Сама программа-интерпретатор является исполняемым файлом, и она загружает один или несколько файлов, полученных на выходе программы-редактора, и после этого интерпретирует их (читает их и выполняет необходимые действия). Поэтому и можно говорить, что программа-интерпретатор формирует среду для выполнения файлов, сформированных с помощью программы-редактора в составе SCADA-пакета.

И здесь возникает следующий вопрос: что же считать программным обеспечением, создаваемым с помощью SCADA-пакета?

Казалось бы, проще всего было бы ответить, что с помощью программы-редактора создаются файлы, которые и надо считать программным обеспечением, создаваемым с помощью SCADA-пакета. И при таком подходе программа-интерпретатор считается средой, в которой данное программное обеспечение выполняется. Таким образом, программа-интерпретатор является своего рода *дополнением* для операционной системы, которое позволяет выполнять программное обеспечение, создаваемое с помощью данного SCADA-пакета. Это очень похоже на то, как понимается программное обеспечение в случае так называемых интерпретируемых языков.

С другой стороны, надо понимать, что сами по себе файлы, создаваемые с помощью SCADA-пакета, не могут выполняться в операционной системе. Для их выполнения (интерпретации) нужна программа-интерпретатор, которая либо входит в состав SCADA-пакета, либо отдельно приобретается как дополнение к

SCADA-пакету для обеспечения среды, позволяющей выполнить указанные файлы.

Поэтому иногда бывает удобно в качестве программного обеспечения, создаваемого с помощью SCADA-пакета, рассматривать сочетание программы-интерпретатора и файлов, формируемых программой-редактором.

Например, когда ставится задача разработать программное обеспечение рабочего места оператора в составе САиУ, и когда необходимо обеспечить не только разработку этого программного обеспечения, но и его функционирование и сопровождение, то тогда проще говорить о разрабатываемом программном обеспечении как о связке программы-интерпретатора и файлов, создаваемых с помощью программы-редактора. Действительно, ведь в этом случае требуется не только разработать (создать с помощью программы-редактора нужные файлы), но и обеспечить функционирование программного обеспечения, то есть обеспечить выполнение файлов с помощью программы-интерпретатора, которая должна быть установлена и настроена на рабочем месте оператора САиУ.

Но в дальнейшем, чтобы не было разночтений, мы будем понимать, по умолчанию, под разработкой программного обеспечения с помощью SCADA-пакета именно разработку файлов с помощью программы-редактора в составе SCADA-пакета.

1.4.5.4. Функции SCADA-пакетов

Кроме функции исполнения (интерпретации) разработанного ПО, о чем было сказано выше, современные SCADA-пакеты обычно реализуют следующие основные функции:

– разработка пользовательского графического интерфейса, в том числе, с использование средств анимации;

– реализация взаимодействия с типовыми УСО;

– реализация алгоритмов управления;

– разграничение доступа пользователей разных уровней;

– формирование отчетов событий и архивов параметров управляемого процесса;

– разработка программ на некотором встроенном (обычно интерпретируемом) языке.

Наличие развитых средств программирования на собственном встроенном языке позволяет также осуществлять такую функцию, как моделирование объектов управления.

Контрольные вопросы

1. Что такое САиУ и каким образом происходит взаимодействие САиУ с целевым процессом? Что может быть целевым процессом в случае САиУ? Приведите примеры целевых процессов.

2. Как можно классифицировать САиУ по видам целевых процессов?

3. Какие существуют виды обеспечения САиУ?

4. Перечислите основные виды технических средств автоматизации и управления и приведите примеры для каждого из этих видов.

5. Что понимается под компьютерными технологиями управления в технических системах?

6. Какие можно выделить типовые архитектуры САиУ? Каковы преимущества и недостатки этих архитектур?

7. Как различаются нижние и верхние уровни САиУ по выполняемым функциям?

8. Каковы основные этапы разработки САиУ?

9. Какова специфика программного обеспечения САиУ?

10. В чем состоит неоднозначность термина «SCADA-система»?

11. Каковы основные принципы организации SCADA-пакетов? Какие базовые функции выполняют современные SCADA-пакеты?

2. ПРИМЕНЕНИЕ СОВРЕМЕННЫХ SCADA-ПАКЕТОВ ПРИ ПРОЕКТИРОВАНИИ СИСТЕМ АВТОМАТИЗАЦИИ И УПРАВЛЕНИЯ

2.1. РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА С ПОМОЩЬЮ SCADA-ПАКЕТОВ

2.1.1. Общие принципы разработки пользовательского интерфейса с помощью SCADA-пакетов

Обычно в рамках SCADA-пакетов пользовательский интерфейс разрабатывается на основе типовых блоков, элементов графического интерфейса, а также функциональных блоков (или их аналогов), реализующих алгоритмическую основу интерфейса.

Проще всего понять базовые принципы разработки пользовательского интерфейса на конкретных примерах.

2.1.2. Запуск демонстрационного примера в SCADA-пакете Genie

Чтобы лучше понять, как с помощью SCADA-пакетов реализуется пользовательский интерфейс, осуществим загрузку и запуск демонстрационного примера в SCADA-пакете *Genie*.

Начнем работу со SCADA-пакетом *Genie*, используя программу-редактор, запустив ее с помощью ярлыка «Advantech Genie Starter». При этом откроется окно программы-редактора. Изначально в окне программы-редактора ничего нет, так как еще не создан файл, который будет редактироваться, и не открыт ни один готовый файл.

Откроем файл, реализующий демонстрационный пример, который входит в состав SCADA-пакета. Этот пример был немного подкорректирован для лучшего отображения. Здесь мы рассматриваем уже готовый файл, чтобы сразу увидеть работающий пример пользовательского интерфейса со многими элементами. А потом уже перейдем к рассмотрению процесса создания работающего примера пользовательского интерфейса, используя готовые элементы в составе SCADA-пакета *Genie*.

После открытия этого файла, используя меню «File=>Open» или кнопку  , в окне программы-редактора появляется несколько квадратных элементов, соединенных множеством стрелок (рис. 36).

Среди этих прямоугольников можно выделить блоки с названиями сигналов: «Синусоида», «Треугольник», «Случайный сигнал». Эти блоки являются генераторами сигналов различной формы, например, в форме треугольника.

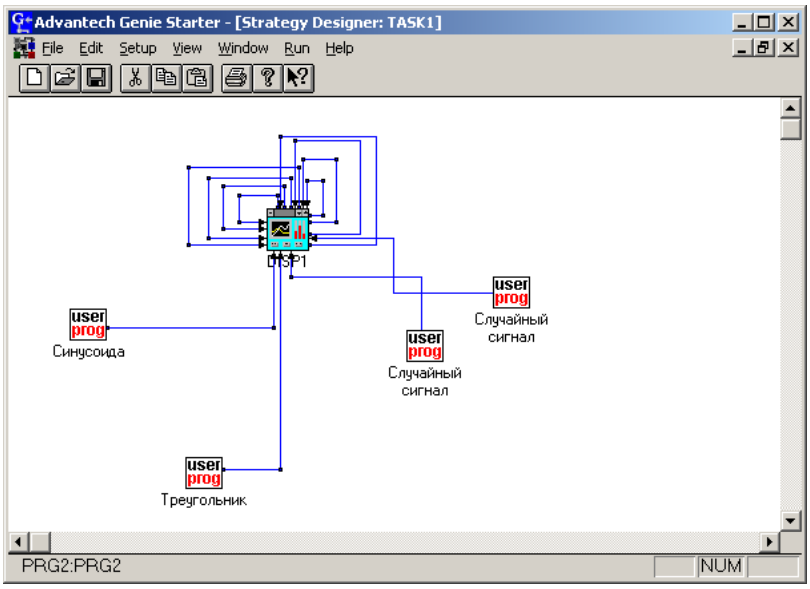


Рис. 36. Окно программы-редактора *Genie* при открытии демонстрационного файла (окно «конструктора стратегии»)

От этих блоков идут стрелки, направленные к блоку с названием «DISP1», который реализует одно из окон графического интерфейса. В данном примере графический интерфейс содержит только одно окно, поэтому и блок «DISP1» только один.

Все эти блоки со стрелками расположены в поле так называемого конструктора стратегии (*Strategy Designer*). Данное название является специфичным для SCADA-пакета *Genie*, но в других SCADA-пакетах также имеются подобные модули внутри программы-редактора, только они носят другие названия. При изучении SCADA-пакета TRACE MODE это будет отмечено.

Подобные модули в составе программы-редактора служат для реализации алгоритмической основы программного обеспечения, разрабатываемого с использованием данного SCADA-пакета. Хотя надо отметить, что словосочетание «алгоритмическая основа» в данном контексте вбирает в себя не только собственно построение алгоритмов, но и организацию структур данных в явном или неявном виде. Просто для краткости здесь используется именно такой термин, тем более что реализация алгоритма предполагает учет особенностей имеющихся структур данных или конструирование таких структур.

Теперь если в данном «конструкторе стратегии» нажать на этот блок «DISP1» (см. рис. 21), то произойдет переход в так называемый конструктор экрана (*Screen Designer*), внешний вид которого представлен на рис. 37.

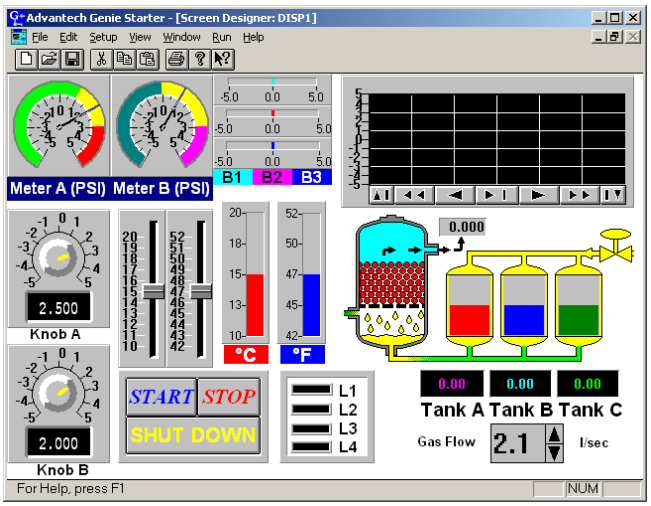


Рис. 37. Окно конструктора экрана в составе программы-редактора *Genie*

Здесь можно перемещать отдельные графические элементы (стрелочные индикаторы, кнопки и т.д.), размещая их нужным образом и выравнивая по сетке.

Также можно настраивать свойства каждого элемента. Настройку свойств отдельных элементов рассмотрим позднее, когда обратимся к простому примеру, реализуемому в SCADA-пакете *Genie*.

А сейчас предположим, что данный файл создан и отредактирован с помощью программы-редактора, и теперь требуется проверить его работу.

Для этого надо запустить данный файл с помощью программы-интерпретатора. Этот запуск осуществляется с помощью нажатия меню «Run». При этом автоматически запускается программа-интерпретатор, и в ней запускается файл, созданный с помощью программы-редактора (рис. 38).

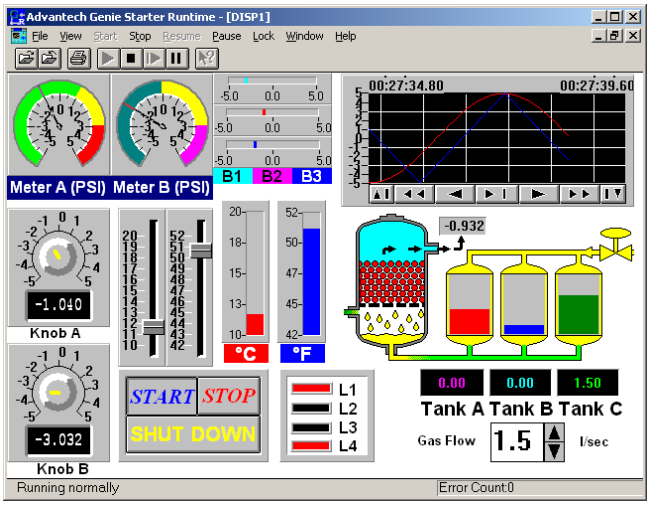


Рис. 38. Окно программы-интерпретатора SCADA-пакета *Genie*

В открывшемся окне программы-интерпретатора, называемой *Advantech Genie Starter Runtime*, отображается графический интерфейс, ранее созданный

в программе-редакторе. Но при этом элементы графического интерфейса теперь начинают «работать»: стрелочные индикаторы начинают отслеживать изменяющиеся значения (стрелка все время перемещается автоматически); на графике начинают строиться линии сигналов «Треугольник» и «Синусоида» и т.д.

Здесь важно отметить, что ранее в конструкторе стратегии (там, где отображались блоки со стрелками) можно было видеть названия блоков «Треугольник», «Синусоида», «Случайный сигнал» (см. рис. 36). Эти блоки как раз и предоставляют значения на графические элементы. Например, «Случайный сигнал» поставляет значения для стрелочного индикатора, а блоки «Треугольник» и «Синусоида» – для графика.

Стоит отметить, что данный пример является демонстрационным, поэтому вместо реальных сигналов с датчиков здесь имитируются сигналы с помощью специальных небольших программ, реализованных в блоках «user prog» 

. Эти блоки мы рассмотрим позднее.

В верхней части окна программы-интерпретатора можно видеть панель кнопок для управления процессом выполнения (рис. 39).

Рис. 39. Управляющие кнопки программы-интерпретатора SCADA-пакета *Genie*

С помощью этих кнопок можно запускать процесс выполнения, останавливать, возобновлять, приостанавливать. Это базовые управляющие действия, которые можно производить в ходе выполнения программного обеспечения, созданного с помощью SCADA-пакет *Genie*.

Таким образом, мы рассмотрели базовые принципы построения и основные компоненты SCADA-пакета *Genie* на основе готового демонстрационного примера.

Теперь можно перейти к ознакомлению с основными функциями, реализуемыми в рамках SCADA-пакетов, опять-таки на примере SCADA-пакета *Genie*. С этой целью осуществим разработку простого примера программы с помощью SCADA-пакета *Genie*. И в ходе этой разработки будем знакомиться с основными принципами разработки пользовательского интерфейса.

2.1.3. Разработка пользовательского интерфейса с помощью SCADA-пакета Genie

Разработаем простой пример, в котором будет обеспечиваться вывод синусоидального сигнала на график а затем можно будет сделать аналогичный пример в более сложном SCADA-пакете. Это позволит лучше понять

принципы и способы разработки пользовательского интерфейса, и насколько они похожи и отличаются в различных SCADA-пакетах.

Как уже было сказано ранее, в качестве примера используется SCADA-пакет *Genie*, и именно на основе данного SCADA-пакета мы и будем рассматривать процесс формирования и вывода сигнала на график. В ходе этого процесса можно будет познакомиться с некоторыми базовыми компонентами *Genie*, которые в той или иной мере характерны для большинства SCADA-пакетов, а также можно будет лучше усвоить базовые принципы построения SCADA-пакета.

Для начала сделаем именно этот простой пример, а потом будем его постепенно усложнять, добавляя новые элементы и функции, знакомясь тем самым с отдельными компонентами SCADA-пакета и с базовыми принципами работы со SCADA-пакетом.

Для создания нового файла нажимаем соответствующую команду меню «File=>New» или нажимаем кнопку  . И это создает новый файл в программе-редакторе. Этот файл пока еще ничем «не наполнен», поэтому имеется только пустое поле конструктора-стратегии.

Для начала надо добавить новый элемент, который будет генерировать случайно изменяющийся сигнал. Для этого нам понадобится панель готовых блоков для конструктора стратегии (рис. 40).

Рис. 40. Панель готовых блоков для конструктора стратегии SCADA-пакета *Genie*

На этой панели расположены готовые блоки для реализации алгоритмической основы с помощью конструктора стратегии. Это готовые блоки, которые можно будет соединять между собой с помощью стрелок,
и в результате будет получаться некоторая структура связанных между собой блоков, подобная той, что мы наблюдали ранее при рассмотрении демонстрационного примера (см. рис. 36).

Здесь важно отметить, что в каждом SCADA-пакете имеются подобные готовые наборы компонентов для создания алгоритмической основы. Обычно их количество намного больше, чем в простом SCADA-пакете *Genie*, а также

они могут быть представлены в различных формах. Это можно будет увидеть позднее при рассмотрении SCADA-пакета TRACE MODE.

Теперь можно установить блок для формирования сигнала. Как уже было отмечено, будет использоваться имитатор сигнала. Но если бы понадобилось подключение реального сигнала, получаемого от какого-нибудь датчика, то можно было бы установить соответствующий блок, например «AI»  . Для этого надо нажать соответствующую кнопку  . В итоге в поле редактирования будет установлен этот блок. И если нажать на него, то откроется диалоговое окно настроек (рис. 41).

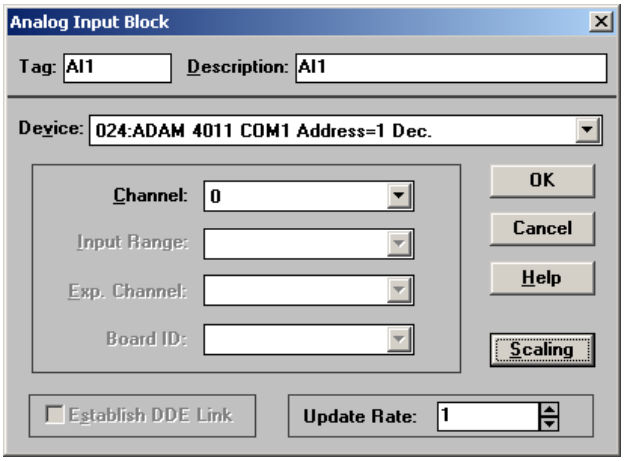


Рис. 41. Окно настроек блока «AI»

В этом окне в поле «Device» можно выбирать нужное устройство, для выбранного устройства устанавливать необходимые настройки, например, номер канала, к которому подключен нужный датчик.

Подобным же образом происходит получение информации с датчиков в большинстве SCADA-пакетов, то есть имеется некоторый компонент (он может называться, например, «блок», «канал», «тег»), и этот компонент служит для получения информации с некоторого реального устройства, или имеется возможность настроить данный компонент для подключения к такому устройству.

Но, как уже было упомянуто, в данном примере мы будем использовать имитатор источника сигнала. Поэтому вместо блока «AI»  будем использовать блок, который позволит реализовать имитатор сигнала. В качестве такого блока будет использоваться уже ранее упомянутый блок «user prog»  . Для установки этого блока в поле редактирования необходимо на панели готовых блоков (см. рис. 40) нажать соответствующую кнопку  .

После этого достаточно нажать на поле редактирования и в него установится этот блок (рис. 42).

Теперь можно открыть его настройки, дважды нажав на него, при этом откроется окно настроек (рис. 43).

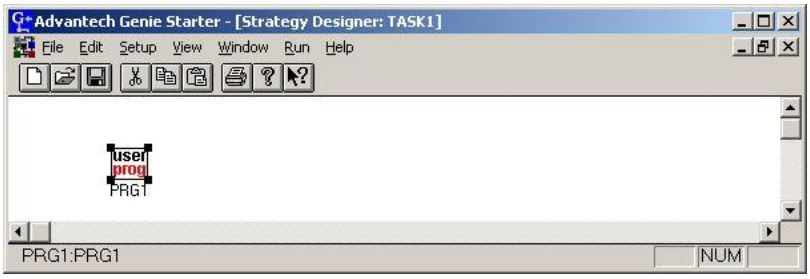


Рис. 42. Установка блока «user prog» в окно редактирования конструктора стратегии *Genie*

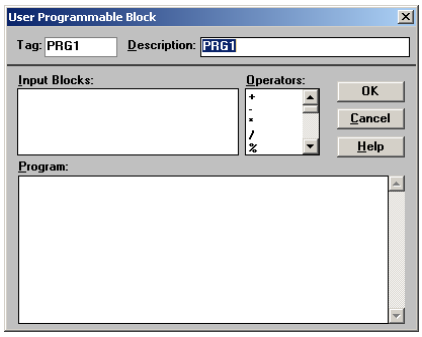


Рис. 43. Окно настроек блока «user prog»

В поле редактирования *Program* задается текст программы, которая будет выполняться на каждом цикле выполнения программы-интерпретатора. Здесь надо отметить, что программа-интерпретатор SCADA-пакета обычно работает на основе некоторых заранее установленных циклов. Эта цикличность задается параметрами периода цикла (в самом простом случае). В зависимости от SCADA-пакета используются различные способы задания цикличности выполнения.

Надо установить необходимое значение цикличности выполнения. Для этого временно закроем окно настроек блока «user prog».

В случае SCADA-пакет *Genie* цикличность определяется значением периода выполнения задач, который устанавливается с помощью меню «Setup=>Task». Надо выбрать этот пункт меню. При этом активизируется соответствующее окно настроек.

В этом окне можно задать так называемый период сканирования (Scan Period). Он как раз и определяет цикличность опроса задач (блоков) в ходе работы программы-интерпретатора, выполняющей файл, созданный программой-редактором.

По умолчанию задается период, равный 1 секунде. Но это слишком большой период, мы его уменьшим, чтобы процессы все протекали немного быстрее. В частности, чтобы быстрее строился график синусоидального сигнала, который в итоге мы и должны получить. Изменим значение этого периода с 1 секунды на 100 миллисекунд (рис. 44).

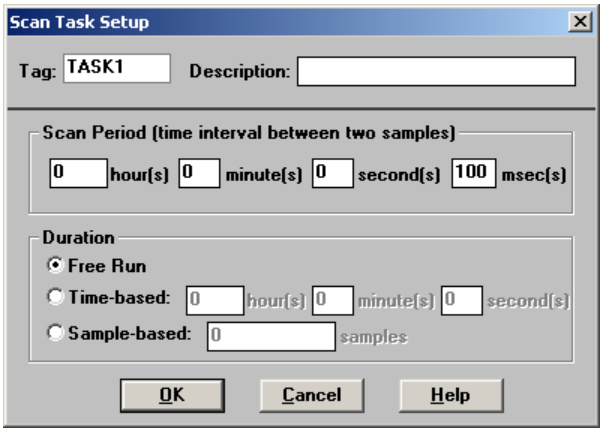


Рис. 44. Новое значение периода сканирования задач в соответствующем окне настроек

Теперь имеем период в 10 раз меньше, поэтому процессы при выполнении программы-интерпретатора будут протекать в 10 раз быстрее.

Но вернемся к настройкам блока «user prog» (см. рис. 43). Необходимо заполнить поле *Program*, а именно в этом поле надо указать программу для генерирования синусоидального сигнала. Программа в блоке «user prog» пишется на языке, подобном языку программирования Си. Текст программы для генерирования синусоидального сигнала будет выглядеть следующим образом:

```
q0=q0+0.1;
output 5*sin (q0);
```

Здесь используется переменная *q0* в качестве счетчика времени. Такое название для переменной выбрано для того, чтобы не перекрывать названия для переменных с более распространенными наименованиями, например, такими, как *x*, *y*, *i*, *j* и т.д. Дело в том, что при написании подобных программ в данном SCADA-пакете *Genie* надо учитывать, что имена переменных являются глобальными, то есть если в одном из блоков «user prog» уже используется переменная *x*, то обращение к переменной с таким же именем в другом блоке «user prog» приводит к тому, что, по сути, происходит обращение к одной и той же переменной. Другими словами, переменная *x* доступна из всех блоков одновременно. Поэтому если в указанной программе использовать вместо переменной *q0* переменную с именем *x*, то это означает, что в других блоках «user prog» для *независимой* переменной надо будет использовать другое наименование. Кроме того, имеется ограничение на наименование пере-

менных в программах рассматриваемого SCADA-пакета. Имя переменной может состоять из одной маленькой латинской буквы (от *a* до *z*) или из буквы и цифры от 1 до 9 (то есть от *a1*, …, *z1* до *a9*, …, *z9*). Поэтому для переменной в указанной программе используется буква *q* как не самое распространенное имя для переменной, а также после нее используется цифра 0, так как в других программах также могут использоваться подобные переменные с этой буквой, например, *q1*, *q2* и т.д. (это можно будет увидеть на примере других подобных программ).

Сама же программа довольно простая. В первой строке происходит увеличение переменной *q0* на 1. Здесь важно понимать, что данная программа, как и все программы в блоках «user prog», выполняются на каждом цикле программы-интерпретатора, то есть каждые 100 миллисекунд в нашем случае (см. рис. 44). Поэтому получается, что каждые 100 миллисекунд происходит увеличение переменной *q0* на 1. При этом при старте значение этой переменной равно нулю. Значит, через 1 секунду после старта эта переменная будет иметь значение 10, через 2 секунды – значение 20, и т.д. Таким образом, переменная *q0* служит в качестве счетчика времени.

Во второй строке программы сначала вычисляется выражение `5*sin (q0)`, которое и определяет значение синусоидального сигнала с амплитудой, равной 5. Здесь значение функции синуса меняется в зависимости от значения переменной *q0*, то есть с течением времени. Поэтому и формируется синусоидальный сигнал. Но надо помнить, что значение синуса не превышает 1 по абсолютному значению, поэтому и надо умножать синус на 5, чтобы получить синусоидальный сигнал, значение которого находится в диапазоне от –5 до 5.

Результат формулы `5*sin (q0)` выводится на выход блока «user prog» с помощью специального оператора `output`. Это позволяет с помощью стрелки в конструкторе стратегии передавать значение синусоидального сигнала другим блокам. Попробуем это и сделать в нашем примере.

Итак, размещаем указанную программу в поле «Program», и дополнительно поменяем название данного блока с «PRG1» на «Синусоида» (рис. 45).

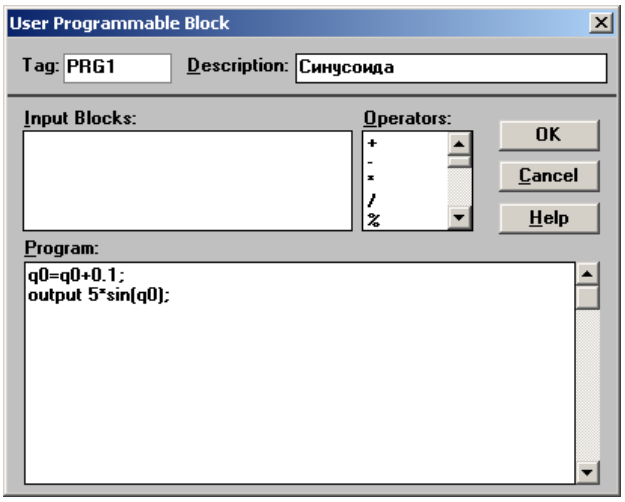


Рис. 45. Добавление текста программы в настройки блока типа «user prog»

Теперь достаточно нажать кнопку «OK». При этом произойдет проверка синтаксической правильности программы. Если нет синтаксических ошибок, то окно просто закроется.

Для примера можно намеренно внести ошибку в текст программы и проверить, что получится. Например, можно убрать точку с запятой в конце первой строки и после этого нажать кнопку «OK». В результате появится окно, сигнализирующее о синтаксической ошибке и содержащее текст «Syntax error».

После возвращения точки с запятой в конец первой строки и нажатия кнопки «OK» окно с программой блока «user prog» закроется. Теперь данный блок будет вырабатывать синусоидальный сигнал.

Требуется вывести синусоидальный сигнал на график, то есть отобразить его в графическом интерфейсе. Для этого надо в поле редактирования поместить специальный блок графического интерфейса «DISP»  . Этот блок мы уже ранее встречали. Для установки данного блока надо в панели готовых блоков (см. рис. 40) нажать соответствующую кнопку  . После этого можно нажать на свободное место в поле редактирования рядом с блоком «user prog» (который теперь называется «Синусоида»), и в результате там установится блок типа «DISP» (рис. 46).

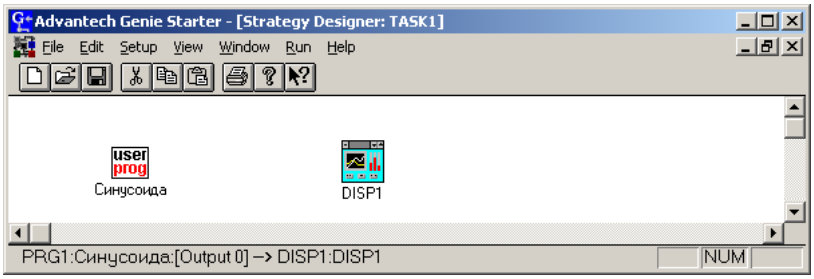


Рис. 46. Установка блока типа «DISP» в поле редактирования

Поскольку блок типа «DISP» всего один в данном случае, поэтому он получает имя «DISP1» (см. рис. 46). Аналогичным образом единственный блок «user prog» сначала получил имя «PRG1» (см. рис. 42), а потом мы его поменяли на «Синусоида».

Теперь надо соединить блоки «user prog» и «DISP», то есть надо «нарисовать» стрелку от блока «Синусоида» к блоку «DISP1». Для этого надо в панели готовых блоков (см. рис. 40) нажать соответствующую кнопку  . После этого курсор мыши приобретает особый вид, похожий на катушку ниток  . Теперь надо нажать сначала на блок «Синусоида», при этом появится окно выбора одного из 8 выходов блока типа «user prog» (рис. 47).

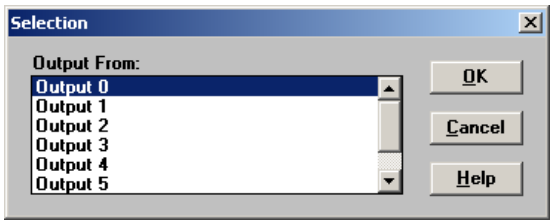


Рис. 47. Окно выбора одного из выходов блока типа «user prog»

У блока «user prog» (то есть и у данного блока «Синусоида») имеется 8 выходов, то есть такой блок может формировать 8 разных выходных сигналов. В программе данного блока используется строка `output 5*sin (q0),` а это значит, что номер выхода не задан, поэтому (по умолчанию) выходной сигнал будет формироваться на 0-м выходе (Output 0). Таким образом в окне выбора выхода (см. рис. 47) надо выбрать именно позицию «Output 0».

После этого выбора в поле редактирования начнет формироваться линия соединения (стрелка). Можно изменять направление этой линии, указывая точки перегиба на поле редактирования. В итоге надо нажать на выходной блок (куда будет указывать стрелка). Этим блоком является блок «DISP1», поэтому надо нажать на него. В результате сформируется стрелка, направленная от блока «Синусоида» к блоку «DISP1» (рис. 48).

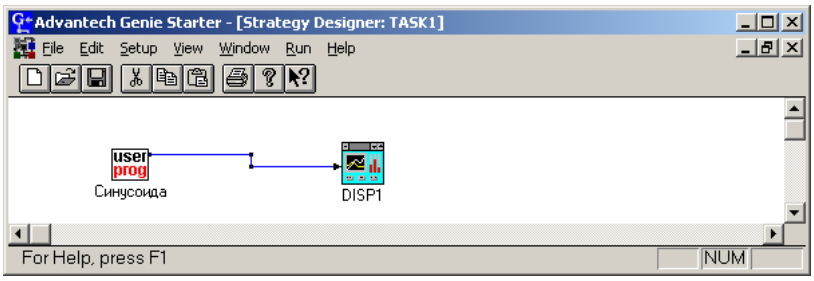


Рис. 48. Установление соединения между блоками

Это соединение означает, что синусоидальный сигнал, формируемый на выходе с номером 0 блока «Синусоида», поступает на вход блока «DISP1». В свою очередь блок «DISP1» служит для создания графического интерфейса. Именно там необходимо расположить график, на который будет выводиться указанный синусоидальный сигнал. Для этого надо дважды нажать на блок «DISP1», в результате произойдет переход в редактор графического интерфейса. В поле этого редактора пока еще ничего нет, оно пустое, так как никакого графического элемента туда еще не добавлено.

Но при этом появляется панель готовых блоков графического интерфейса (рис. 49).

На этой панели расположены готовые блоки для реализации графического интерфейса с помощью конструктора экрана. Это готовые блоки, которые можно будет располагать на окне конструктора экрана, и в результате можно получить графический интерфейс, подобный тому, что мы наблюдали ранее при рассмотрении демонстрационного примера (см. рис. 37).

Рис. 49. Панель готовых блоков для конструктора
экрана SCADA-пакета *Genie*

Здесь важно отметить, что в каждом SCADA-пакете имеются подобные готовые наборы компонентов для создания графического интерфейса. Обычно их количество намного больше, чем в простом SCADA-пакете *Genie*. Это можно будет увидеть позднее при рассмотрении SCADA-пакета TRACE MODE.

Теперь надо расположить в конструкторе экрана график, на который будет выводиться синусоидальный сигнал. Для этого надо на панели готовых блоков графического интерфейса нажать соответствующую кнопку  .

После этого достаточно нажать на свободное место в поле конструктора экрана, и тем самым будет установлен элемент графического интерфейса «График» («Trend»). По умолчанию, график имеет маленькие размеры, но его можно увеличить, потянув за угол, в итоге его можно растянуть на большую часть поля редактора экрана, а также немного сместить. Однако на нем ничего не выводится (рис. 50).

Если попробовать запустить программу (через меню Run), то появится ошибка, сообщающая о том, что график не сконфигурирован правильно («Unconfigured Block: XT Graph»).

Для того чтобы настроить график, надо дважды нажать на элемент графического интерфейса «График» (см. рис. 50). Это открывает окно настроек данного элемента графического интерфейса. В этом окне имеется поле входных данных «Input from», где указывается единственный элемент, а именно «Синусоида», так как именно от этого блока с помощью стрелки передается синусоидальный сигнал на блок «DISP1». Казалось бы, все правильно. Но в чем же тогда проблема, почему появляется сообщение об ошибке? Это связано с тем, что данный элемент списка в поле «Input from» не выбран. Чтобы его выбрать, надо дважды нажать на него, и перед его названием появится символ «*» (рис. 51).

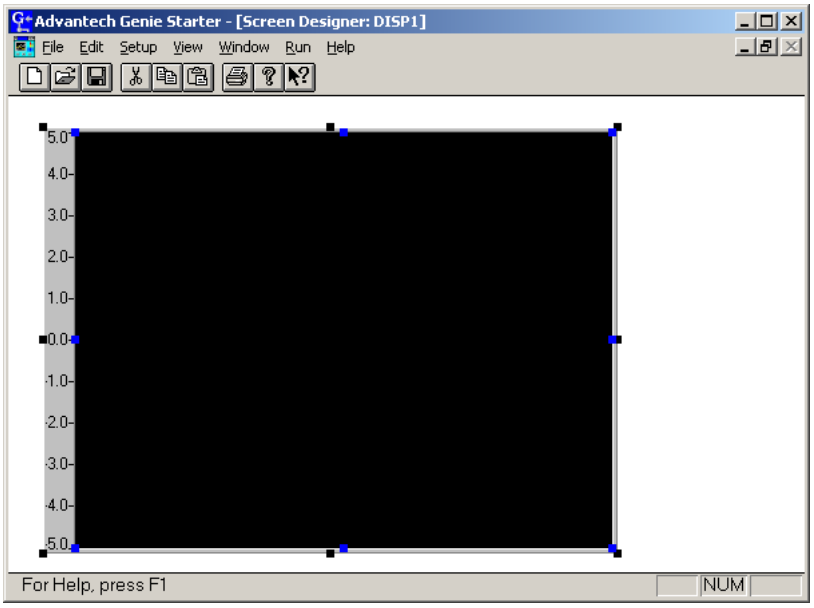


Рис. 50. Элемент «График» после изменения размеров и перемещения

Этот символ указывает на то, что данный элемент выбран для графика. Необходимость в таком выборе возникает по той причине, что в блок «DISP1» может поступать много сигналов (стрелок от других блоков), но не все они должны быть выведены на график. Однако они все будут отображены в этом поле «Input from». Поэтому и надо иметь возможность выбрать один или несколько элементов в данном поле.

Кроме того, можно выбрать цвет линии графика в поле «Trace Color», а также фона графика. Так, можно для линии графика выбрать, например, зеленый цвет (Green) (см. рис. 51).

Кроме этого в окне настроек элемента графического интерфейса «График» также имеется много дополнительных возможностей для настройки

параметров. Например, можно настроить масштаб графика. Но сейчас мы ничего больше менять не будем.

Теперь после этих настроек попробуем запустить сделанную программу с помощью программы-интерпретатора. Для этого, как и ранее, воспользуемся пунктом меню Run. Теперь программа запускается без ошибок, и мы можем наблюдать строящийся график синусоидального сигнала (рис. 52).

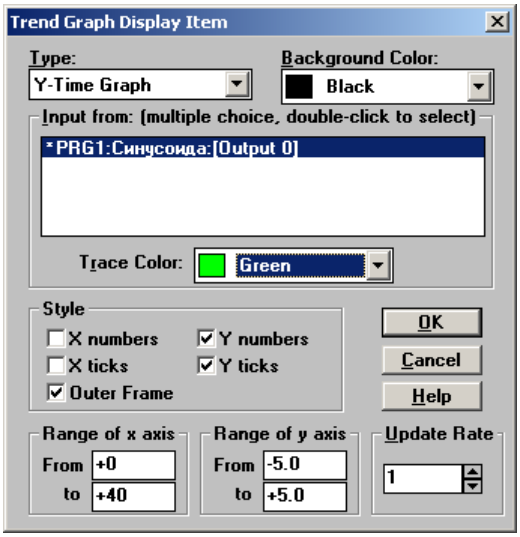


Рис. 51. Изменение настроек элемента графического интерфейса «График»

Теперь дополним разработанную программу следующим образом. Предположим, что получаемый сигнал надо сравнить с некоторым аварийным порогом, и если сигнал превысит этот порог, то надо выработать аварийный сигнал, например, на основе цветового или текстового индикатора. Но первым решением этой задачи будет задача формирования самого управляемого аварийного порога.

Что значит «управляемого»? Это значит, что оператор может в ходе процесса (во время функционирования системы) изменять значение этого порога.

Изменять значение некоторого параметра можно с помощью разных элементов графического интерфейса. Одним из удобных элементов является так называемый «Ползунок» («Slider»). В SCADA-пакете *Genie* тоже есть такой элемент, и его можно добавить в поле конструктора экрана с помощью готовых блоков графического интерфейса (см. рис. 49), где надо нажать соответствующую кнопку  .

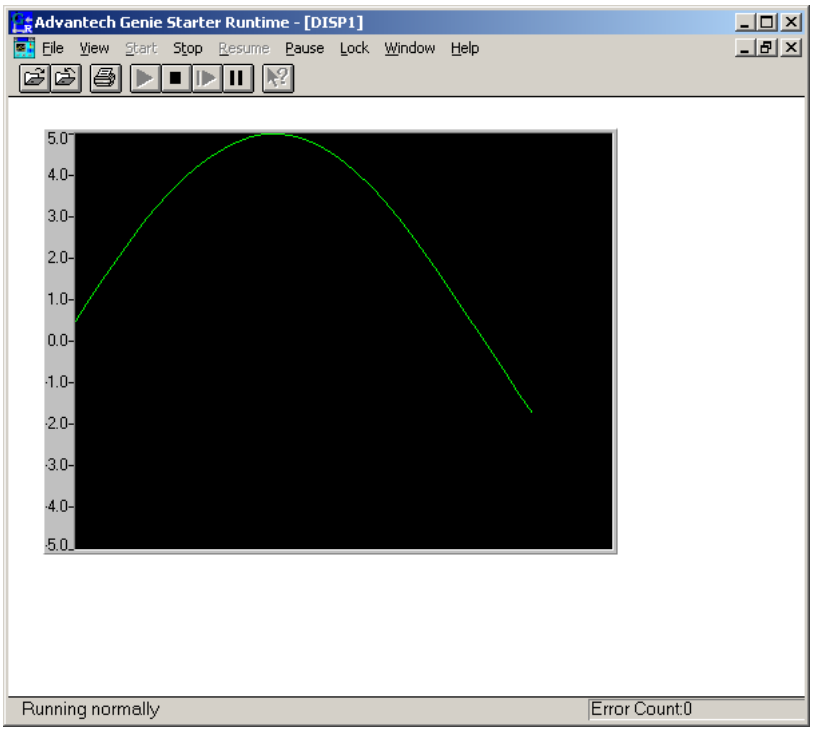


Рис. 52. График синусоидального сигнала

После этого надо нажать на свободное место в поле конструктора стратегии, и там добавится элемент графического интерфейса «Ползунок». Его можно немного увеличить в размерах и выровнять с элементом графического интерфейса «График» (рис. 53).

Зайдем в настройки этого элемента графического интерфейса, нажав дважды на него, откроется окно настроек (рис. 54), в котором в поле «Description» установим новое название данного элемента, например, «В. порог» (что означает «Верхний порог» – мы будем использовать аварийный порог в качестве верхнего порога).

В настройках элемента графического интерфейса «Ползунок» можно поменять некоторые параметры, например, верхнюю и нижнюю отметки («Tics Start», «Tics End»). Но в данном случае это не требуется, так как они точно соответствуют пределам элемента графического интерфейса «График».

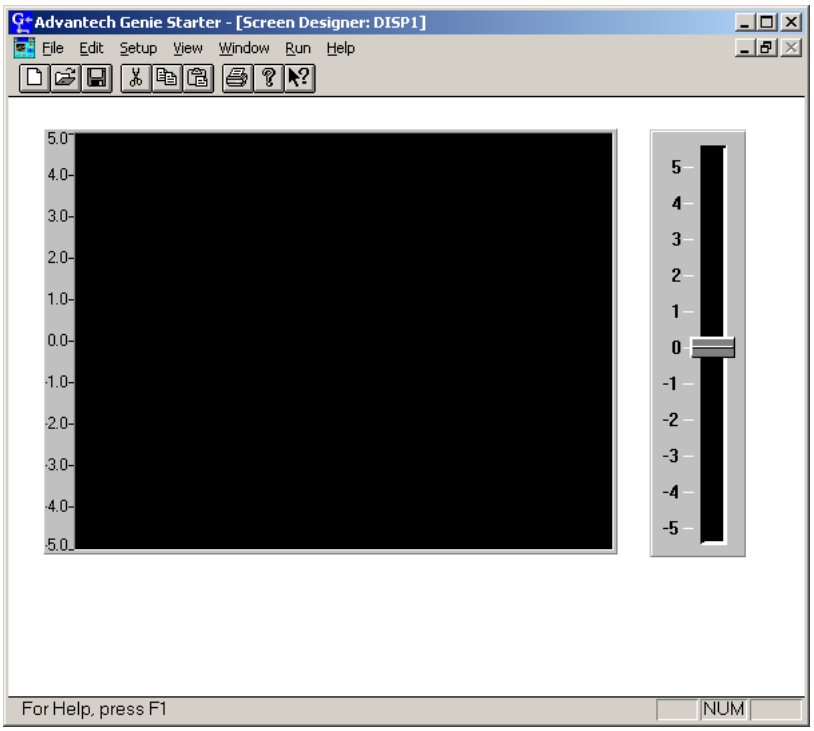


Рис. 53. Добавление элемента графического интерфейса «Ползунок»

Если сейчас запустить программу, то окажется, что «Ползунок» можно перемещать вверх-вниз, но на графике ничего не отражается. Хотелось бы, чтобы на графике также отображалась линия аварийного порога, которым можно управлять с помощью интерфейса «Ползунок».

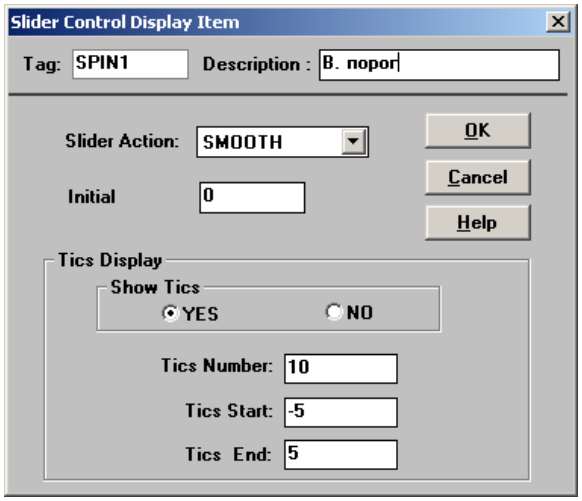


Рис. 54. Окно настроек элемента графического интерфейса «Ползунок»

Для этого надо перейти в конструктор стратегии, например, с помощью меню «Window=>Strategy Designer». И в окне конструктора стратегии надо установить такую стрелку, чтобы она выходила из блока «DISP1» и обратно направлялась в этот же блок. При создании этой стрелки нажимаем на исходный блок, которым является блок «DISP1», и при этом появляется окно выбо-

ра элемента графического интерфейса, от которого должна исходить стрелка (рис. 55).

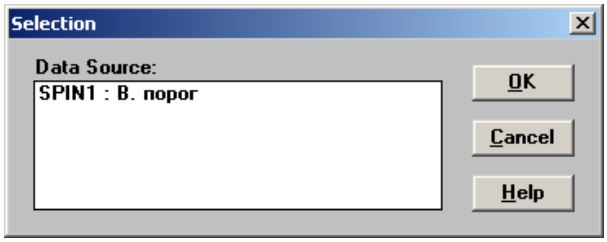


Рис. 55. Окно выбора элемента графического интерфейса

В этом окне в данный момент имеется единственная строка, соответствующая новому элементу графического интерфейса «Ползунок». Это единственный элемент графического интерфейса, который может выдавать данные и из которого может исходить стрелка. Очевидно, элемент графического интерфейса «График» не может выдавать данные. Выбираем в окне эту единственную строку и устанавливаем стрелку, как было описано выше. Теперь она исходит из блока «DISP1», а потом возвращается обратно в этот блок (рис. 56).

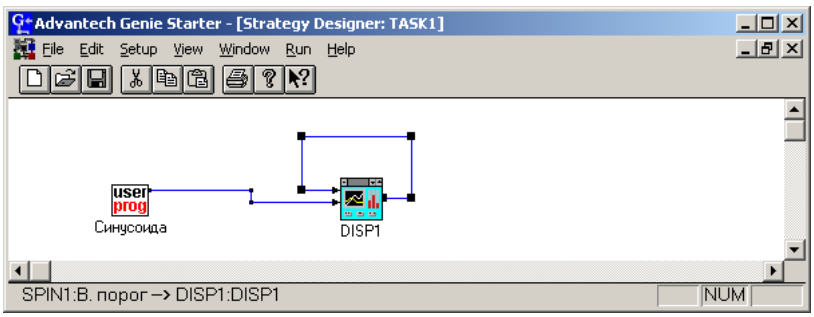


Рис. 56. Установка соединения

Для чего было сделано это соединение? Оно позволяет направить с помощью этой стрелки данные от элемента графического интерфейса «Ползунок» в элемент графического интерфейса «График». И теперь, если открыть окно настроек элемента графического интерфейса «График», то мы увидим, что в списке «Input from» появляется второй элемент. Выберем этот элемент (в начале соответствующей строки должен появиться символ «*») и зададим ему цвет, например красный (Red) (рис. 57).

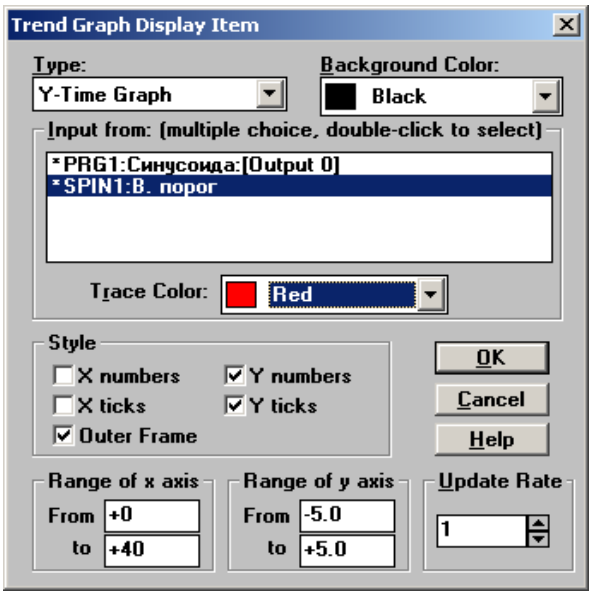


Рис. 57. Настройка сигнала от элемента графического интерфейса «Ползунок» с названием «B. порог»

Теперь можно запустить полученную систему, и теперь изменения аварийного порога, выполняемые с помощью элемента графического интерфейса «Ползунок», отображаются на графике (рис. 58).

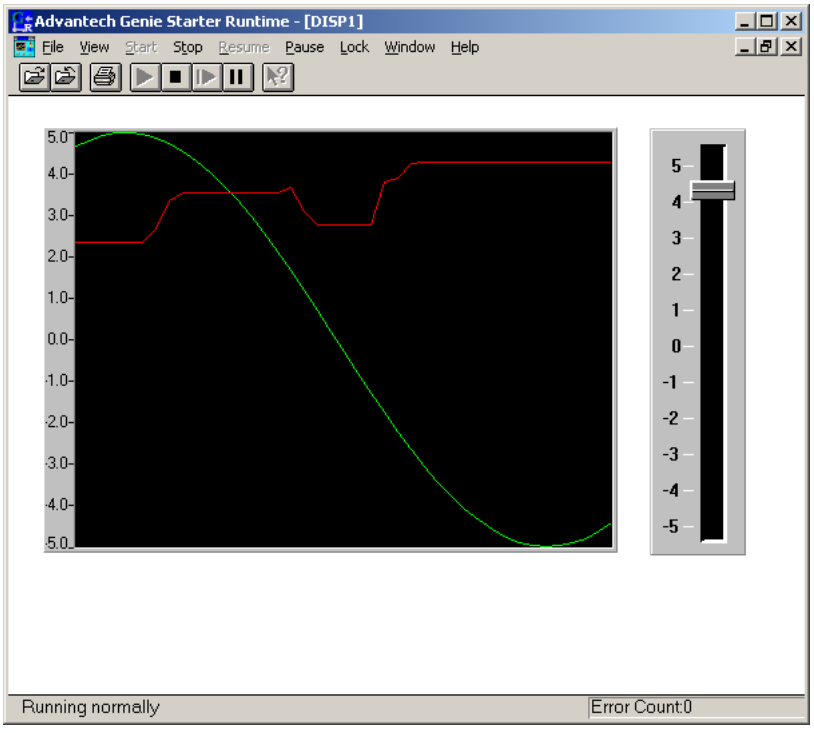


Рис. 58. Отображение на графике изменения аварийного порога

В итоге мы реализовали еще один из типовых компонентов пользовательского интерфейса. Но пока сделан лишь вывод на график, а нам нужно сделать так, чтобы автоматически формировалась аварийная индикация о

превышении верхнего порога (данного аварийного порога), то есть выполнялось сравнение. Эту функцию сравнения можно отнести к алгоритмической основе разрабатываемой программы. Поэтому для ее реализации надо переключиться в конструктор стратегии, где и создается эта алгоритмическая основа.

И здесь в конструкторе стратегии в поле редактирования надо добавить специальный блок, который будет реализовывать сравнение. Для этого надо в панели готовых блоков (см. рис. 40) нажать соответствующую кнопку  . И после нажатия на поле редактирования туда добавится блок оператора «Оператор вычисления»  .

По умолчанию устанавливается именно оператор «Плюс», что и видно, если открыть окно настроек этого блока (рис. 59).

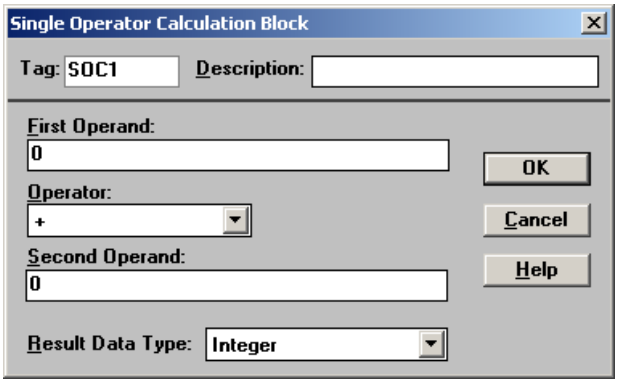


Рис. 59. Окно настроек блока «Оператор вычисления»

В этом окне настроек в поле «Operator» надо поменять знак «+» на знак «>» (знак «больше»). Тогда в поле редактирования изменится внешний вид этого блока  .

И сейчас этот блок готов реализовывать сравнение. Но для этого к нему надо «подвести» с помощью стрелок эти сравниваемые значения, а также вывести результат сравнения (тоже с помощью стрелки) в блок «DISP1». Поэтому устанавливаем стрелку от блока «Синусоида», стрелку от блока «DISP1», а также выводим стрелку из блока сравнения в блок «DISP1» (рис. 60).

Здесь надо отметить, что при установке стрелок из блоков «Синусоида» и «DISP1» будут возникать окна выбора (см. рис. 47 и 48), и в этом случае надо сделать действия, аналогичные ранее выполненным при возникновении этих окон. Также при подведении стрелок к блоку сравнения будет возникать окно выбора операнда (рис. 61).

При подведении стрелки от блока «Синусоида» надо выбирать операнд 1 (Operand 1), а при подведении стрелки от блока «DISP1» – операнд 2 (Operand 2). Можно мысленно представлять, что между этими

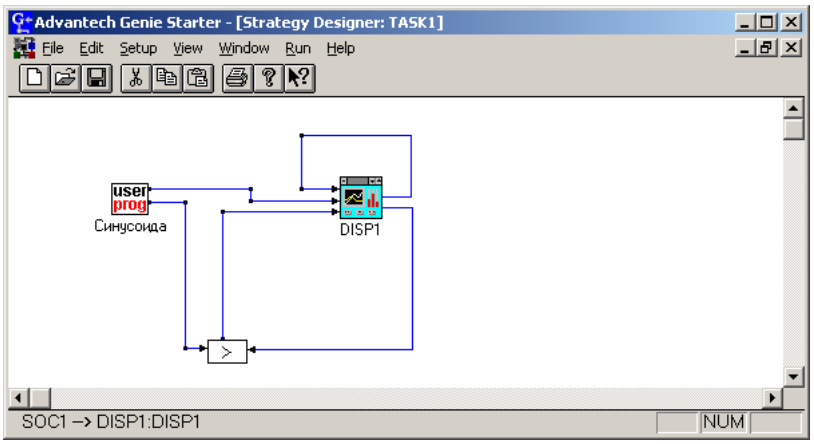


Рис. 60. Подключение блока «Оператор вычисления»

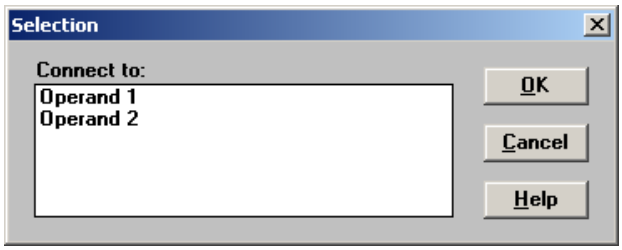


Рис. 61. Окно выбора операнда для блока
«Оператор вычисления»

двумя операндами располагается знак оператора вычисления, в данном случае – знак «>».

Итак, теперь у нас в блок «DISP1» поступает значение, формируемое после выполнения оператора сравнения, за счет стрелки, направляемой от блока сравнения (см. рис. 60). Этот сигнал может принимать два значения: 0 и 1. Значение 0 означает, что синусоидальный сигнал меньше значения аварийного порога, а значение 1 – наоборот.

Теперь надо реализовать индикацию этого сигнала с помощью элемента графического интерфейса «Индикатор». Для этого после перехода в конструктор экрана надо на панели готовых блоков графического интерфейса нажать соответствующую кнопку  , установить элемент графического интерфейса «Индикатор» в поле редактирования и немного увеличить его размеры (рис. 62).

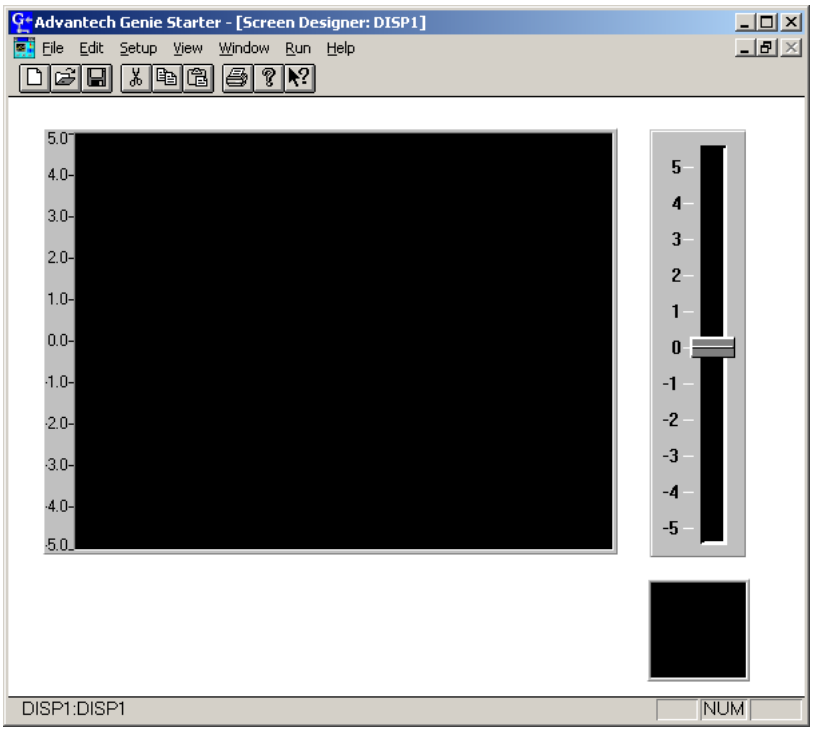


Рис. 62. Добавление индикатора сигнала о превышении порога

Сейчас он просто представляет собой черный прямоугольник. Надо изменить его параметры. Для этого открываем его окно настроек и изменяем там некоторые параметры (рис. 63).

При этом в поле «Input from» указываем, откуда поступает сигнал для индикатора, а именно указываем «SOC1», ведь так называется блок сравнения (см. рис. 59, поле «Tag»). Также в поле «Color for OFF (0) state» устанавливаем зеленый цвет (Green).

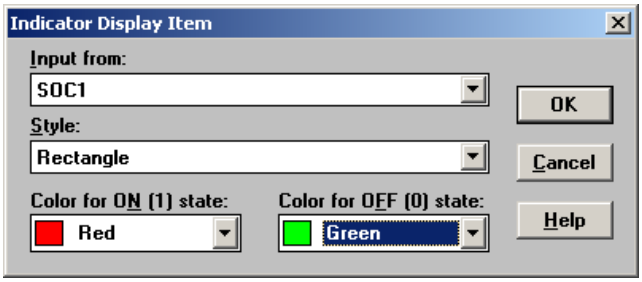


Рис. 63. Окно настроек элемента графического интерфейса «Индикатор»

Теперь запускаем программу и видим, что в случаях, когда синусоидальный сигнал превышает аварийный порог, индикатор становится красным (рис. 64), в других же случаях он остается зеленым.

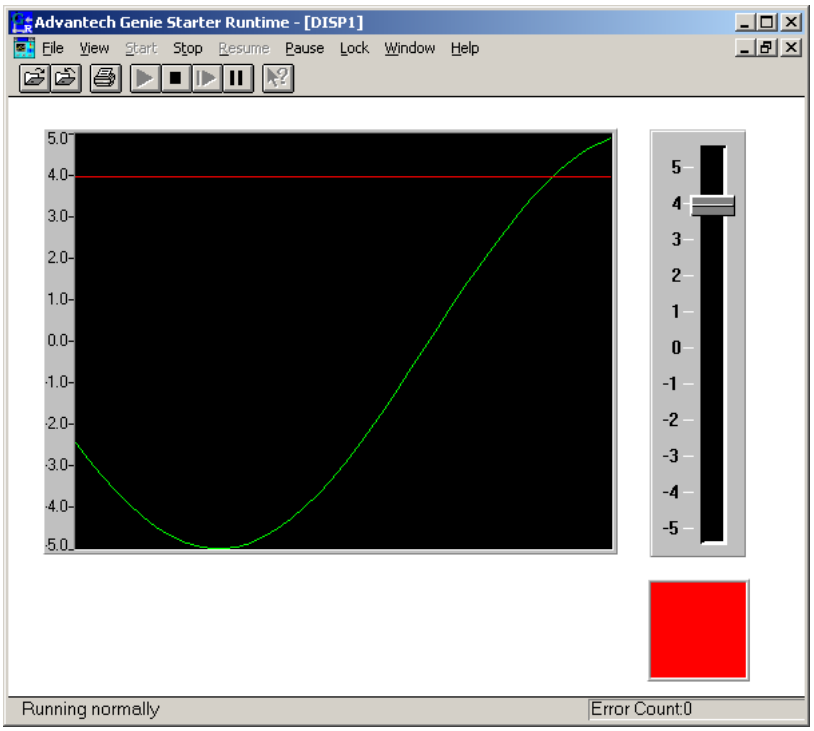


Рис. 64. Изменение цвета индикатора на красный
при превышении аварийного порога

Примерно похожим образом можно реализовать индикацию в виде текстовых сообщений с помощью элемента графического интерфейса «Условный текст» («Conditional Text»). Для этого после перехода в конструктор экрана надо на панели готовых блоков графического интерфейса нажать соответствующую кнопку 

, установить элемент графического интерфейса в поле редактирования и немного увеличить его размеры. После этого в окне настроек этого элемента можно изменить нужные параметры (рис. 65).

После размещения и настройки данного элемента графического интерфейса можно запустить программу и убедиться, что в случаях, когда синусоидальный сигнал превышает аварийный порог, на индикаторе выводится текст «Превышение аварийного порога» (рис. 66), в других же случаях на этом текстовом индикаторе выводится текст «Сигнал в норме» (рис. 67).

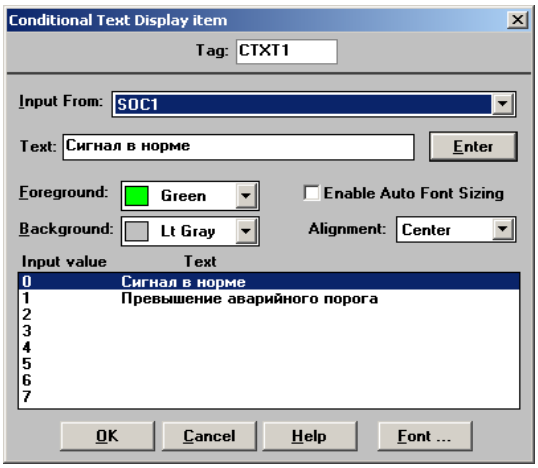


Рис. 65. Окно настроек элемента графического интерфейса «Условный текст»

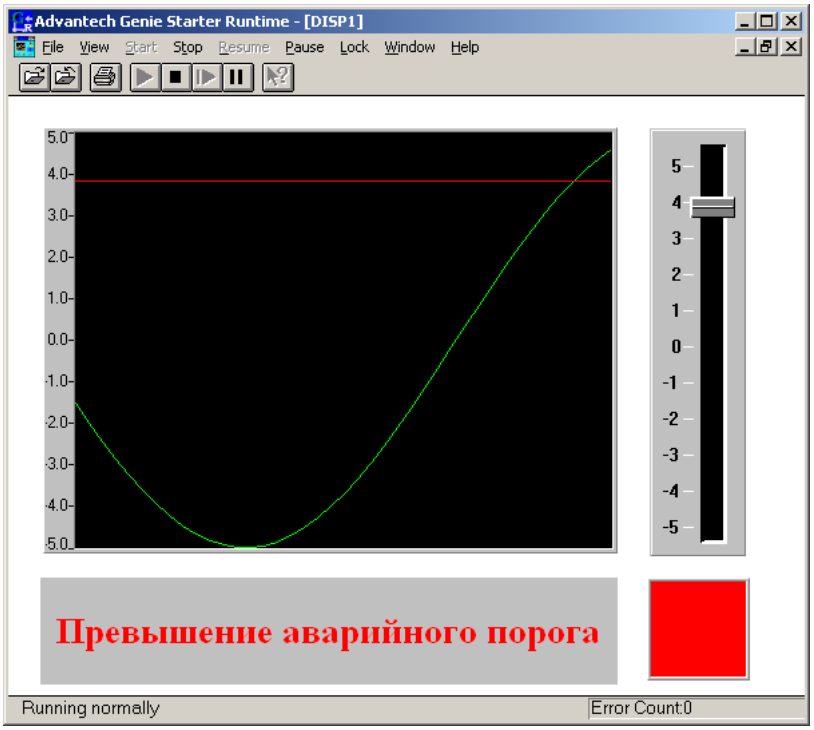


Рис. 66. Вид индикатора с текстом «Превышение аварийного порога», при превышении аварийного порога синусоидальным сигналом

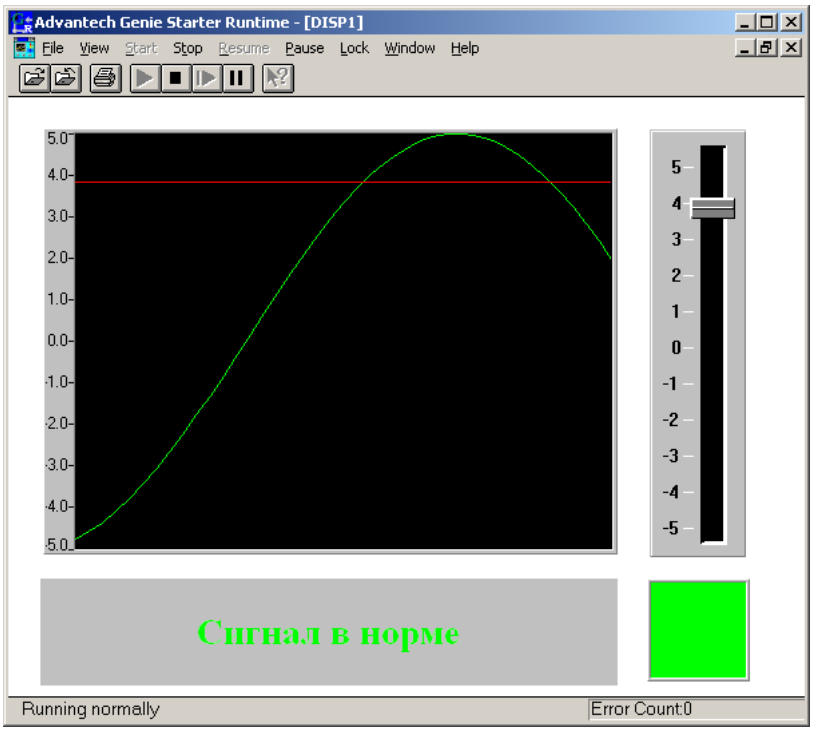


Рис. 67. Вид индикатора с текстом «Сигнал в норме» при превышении синусоидальным сигналом аварийного порога

В рассматриваемом примере пока реализовано лишь одно окно пользовательского интерфейса. Но в усложненных программах обычно требуется реализация более чем одного окна. Поэтому в SCADA-пакетах обычно имеются возможности и функции реализации многооконных программ, а также функции управления этими окнами.

Рассмотрим на примере SCADA-пакета *Genie* принцип реализации многооконной программы.

Для этого проще всего продолжить выполнение предыдущего примера следующим образом. Добавим еще два вида сигнала к уже имеющемуся синусоидальному сигналу. А именно добавим сигнал вида «треугольник» и сигнал, случайно изменяющийся. Это все можно сделать по аналогии с тем, как это было сделано в случае блока «Синусоида», только
теперь добавляются два других блока вида «user prog», и они будут отличаться названиями и программами внутри этих блоков.

Для реализации сигнала типа «треугольник» можно использовать программу следующего вида:

```
if (q1==0) q1=0.5;
if (q2+q1>5) q1=-0.5;
if (q2+q1<-5) q1=0.5;
q2=q2+q1;
output q2;
```

Не вдаваясь в подробности текста программы, отметим, что она реализует периодический сигнал типа «треугольник», изменяющийся в диапазоне от –5 до 5.

Для реализации случайно изменяющегося сигнала можно использовать следующую программу:

```
q3=(rnd(0)%101-50)/100.0;
q4=q4+q3;
if (q4>5.0) q4=5.0;
if (q4<-5.0) q4=-5.0;
output q4;
```

Эти программы размещаем в блоки «user prog», которые назовем «Треугольник» и «Случайный сигнал».

Также рядом с каждым из этих блоков установим блок вида «DISP», куда выведем с помощью стрелки соответствующий сигнал. Таким образом будут созданы два дополнительных блока графического интерфейса: «DISP2», «DISP3» (рис. 68).

Каждый из этих новых блоков вида «DISP» формирует новое окно графического интерфейса, то есть при нажатии на каждый из этих блоков в конструкторе экрана открывается новое чистое поле редактирования.

В каждом этом новом поле редактирования установим элемент графического интерфейса «График» и подключим к этому элементу сигнал от соответствующего блока («Треугольник» или «Случайный сигнал»).

Тогда при запуске программы сначала показывается первое окно с синусоидальным сигналом. Но в верхней части этого окна появляется ряд кнопок с цифрами *1*, *2*, *3*, которые позволяют переключать окна.

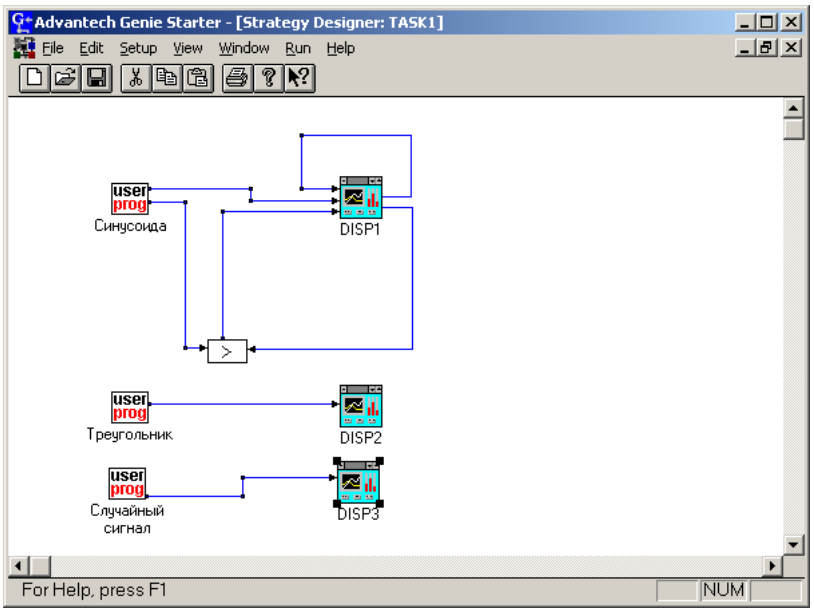


Рис. 68. Установка дополнительных блоков для вывода сигналов в дополнительные окна

В нашем примере было сделано три окна, поэтому и появились только три кнопки. При нажатии на одну из этих кнопок происходит активизация из трех окон. Например, можно активизировать окно с сигналом типа «треугольник» (рис. 69) или окно со случайным сигналом (рис. 70).

По аналогии с тем, как это было сделано в случае синусоидального сигнала, для каждого из двух новых сигналов можно организовать сравнение сигнала с некоторым пороговым значением и вывод сообщения о переходе через это значение, а также дополнительно усложнить рассматриваемый пример. Справочная информация по работе с пакетом *Genie* v2.0 дается в приложении в конце пособия.

Как было уже отмечено, SCADA-пакет *Genie* v2.0 является очень простым в использовании, поэтому на его примере был продемонстрирован процесс разработки пользовательского интерфейса с помощью SCADA-пакетов.

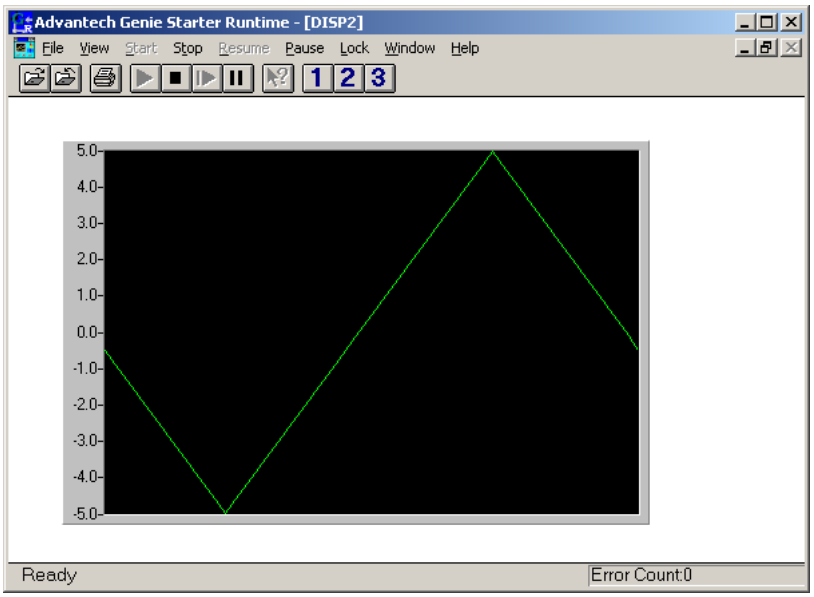


Рис. 69. График сигнала вида «треугольник»

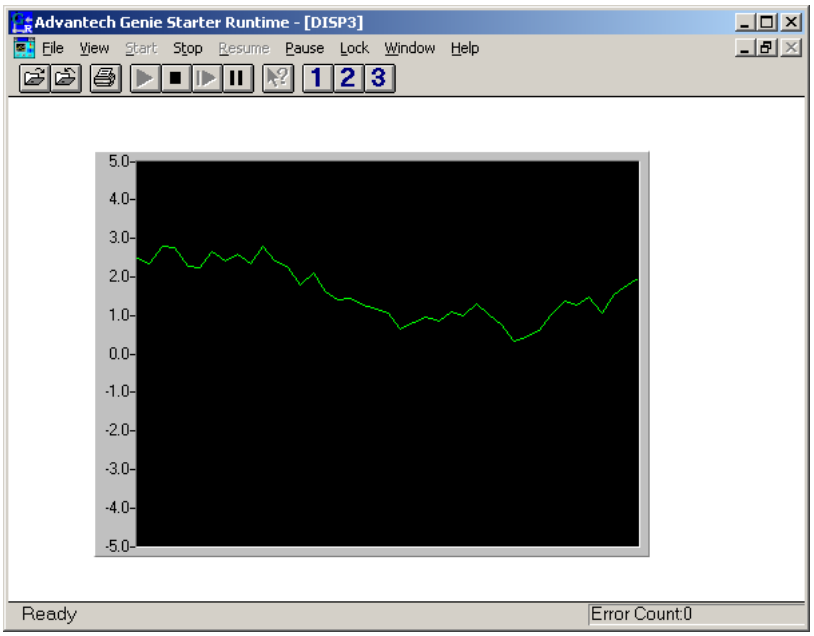


Рис. 70. График случайного сигнала

2.1.4. Разработка пользовательского интерфейса с помощью SCADA-пакета TRACE MODE

Основные особенности работы со SCADA-пакетом TRACE MODE будем рассматривать на основе построения простого примера программы, который аналогичен примеру, ранее выполненному с помощью SCADA-пакета *Genie*.

Это довольно удобно, так как позволяет сравнить между собой два разных SCADA-пакета. И кроме того, надо учитывать, что SCADA-пакет TRACE MODE является более сложным, поэтому, ранее рассмотрев пример для очень простого SCADA-пакета *Genie*, теперь будет проще рассмотреть аналогичный пример для TRACE MODE.

Для начала запустим TRACE MODE, именно так называемую интегрированную среду разработки (рис. 71).

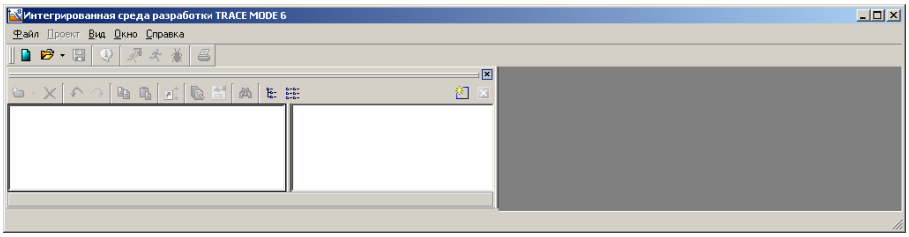


Рис. 71. Окно интегрированной среды разработки
после запуска TRACE MODE

Как уже было ранее отмечено, почти все SCADA-пакеты имеют в своем составе два основных модуля: программу-редактор и программу-интерпретатор. Нетрудно догадаться, упомянутая интегрированная среда разработка является одним из этих модулей, а именно программой-редактором. Поэтому для краткости и единообразия в дальнейшем интегрированная среда разработки SCADA-пакета TRACE MODE будет называться программой-редактором.

Сразу после старта (см. рис. 71) в этой программе-редакторе ничего не загружено. Надо создать или открыть какой-нибудь проект.

И вот уже на этом этапе можно отметить одно из отличий TRACE MODE от *Genie*. В начале работы с *Genie* создавался новый *файл*, а при начале работы с TRACE MODE надо создавать новый *проект*. Можно считать, что в SCADA-пакете *Genie* проект состоит из одного файла, при этом в случае TRACE MODE проект представляет собой совокупность файлов. Уже одно это иллюстрирует то, что SCADA-пакет TRACE MODE существенно сложнее ранее рассмотренного SCADA-пакета *Genie*.

Поэтому создадим новый проект, используя меню «Файл=>Новый» или соответствующую кнопку на панели инструментов  . При этом появится окно выбора типа проекта (рис. 72).

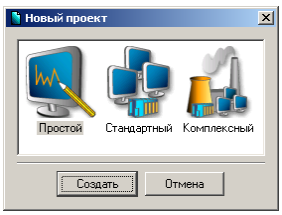


Рис. 72. Окно выбора типа проекта при создании нового проекта в SCADA-пакете TRACE MODE

Кроме того, что в TRACE MODE в отличие от *Genie* надо создавать проект, а не файл, в случае TRACE MODE, как показывает рис. 72, существует несколько видов проектов.

Естественно, что для нашего простого примера мы выбираем тип «Простой» и нажимаем кнопку «Создать».

При этом создается новый проект и окно программы-редактора принимает вид, представленный на рис. 73.

На рисунке видно, что в левой части этого окна возникает система вложенных элементов, похожая на древовидную систему вложенных папок и файлов (как, например, в программе «Проводник» операционной системы семейства *Windows*). Правее расположены два компонента: группа «Каналы», а также элемент с названием «Экран#1:1».

Если дважды нажать на элемент «Экран#1:1», то правее откроется поле графического редактора, а также добавятся дополнительные панели инструментов в верхней части окна (рис. 74).

Нетрудно догадаться, что этот элемент «Экран#1:1» во многом аналогичен ранее рассмотренному блоку вида «DISP»  , используемому в SCADA-пакете *Genie*. При этом панели инструментов, которые добавились в верхней части окна, примерно соответствуют панели готовых блоков SCADA-пакета *Genie* (см. рис. 40).

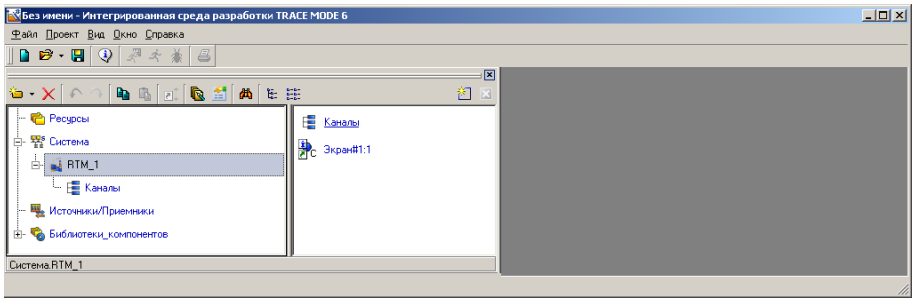


Рис. 73. Окно программы-редактора SCADA-пакета TRACE MODE сразу после создания нового проекта

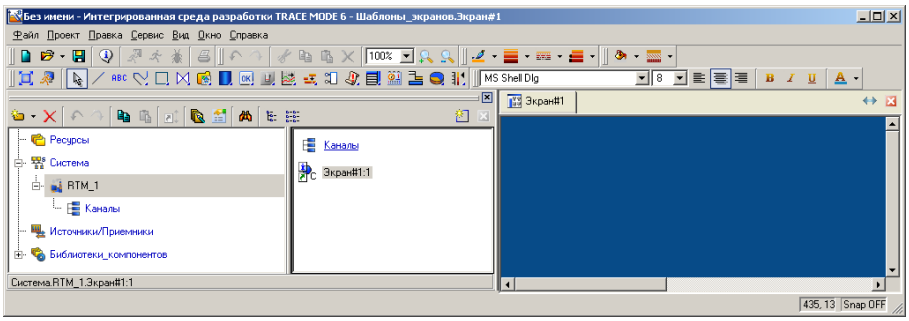


Рис. 74. Активация окна графического редактора

Поле графического редактора, связанное с элементов «Экран#1:1», пока пустое, так как ни одного графического элемента туда не установлено. Но мы будет действовать в том же порядке, что и при реализации этого же примера ранее с помощью SCADA-пакета *Genie*. А именно для начала надо добавить имитатор синусоидального сигнала.

Ранее в SCADA-пакете *Genie* для создания имитаторов сигналов использовался блок типа «user prog»  , в котором можно было размещать программы (см. рис. 45). В SCADA-пакете TRACE MODE, конечно, тоже имеются возможности написания программ. Более того, в TRACE MODE гораздо больше возможностей и способов написания программ. И в данном простом примере мы рассмотрим только один из этих способов.

Нажав правой кнопкой на элемент «RTM_1», создадим компонент «Программа» (рис. 75).

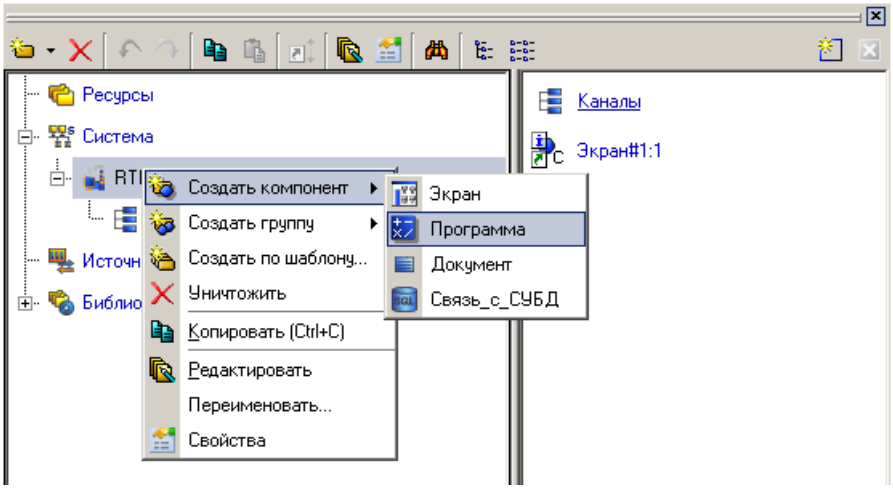


Рис. 75. Создание компонента типа «Программа»

К имеющимся компонентам добавится компонент «Программа#1», отображаемый как «Программа#1:2» (рис. 76).

Теперь дважды нажмем на компонент «Программа#1:2», что приведет к открытию поля редактирования этого компонента (рис. 77).

В этом поле имеется таблица со столбцом «Структура программ». В этом столбце нажмем на строку «Программа#1». Это приведет к открытию окна выбора языка программирования (рис. 78).

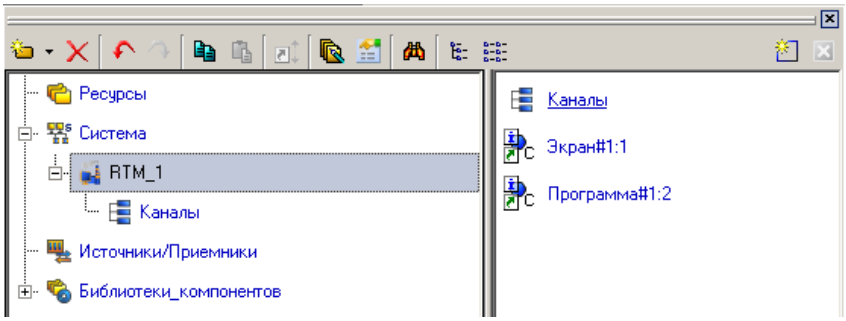


Рис. 76. Добавление компонента типа «Программа»

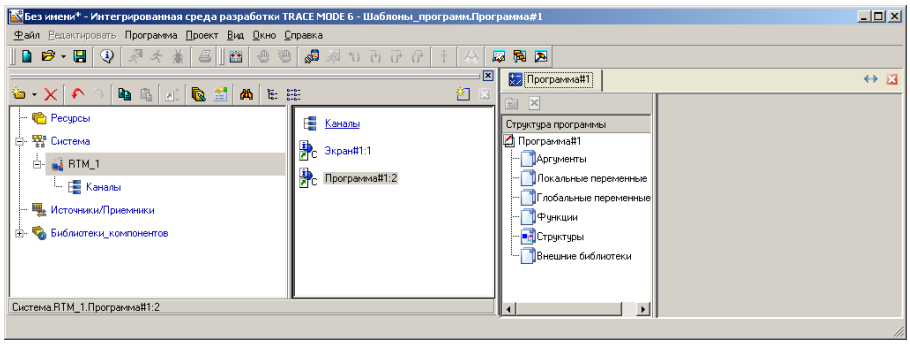


Рис. 77. Поля редактирования компонента типа «Программа#1»

Выбираем пункт «ST программа» нажатием кнопки «Принять». В итоге правее в поле редактирования компонента «Программа#1» откроется поле для редактирования текста программы (рис. 79).

В этом поле уже имеется пустой шаблон программы (PROGRAM … END_PROGRAM). Внутрь этого шаблона добавим тот же текст программы, который ранее использовался для реализации синусоидального сигнала в примере для SCADA-пакета *Genie*. При этом поле редактирования компонента «Программа#1» приобретет вид, представленный на рис. 80.

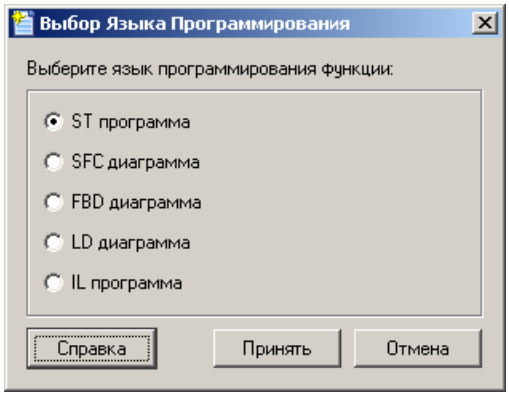


Рис. 78. Окно выбора языка программирования

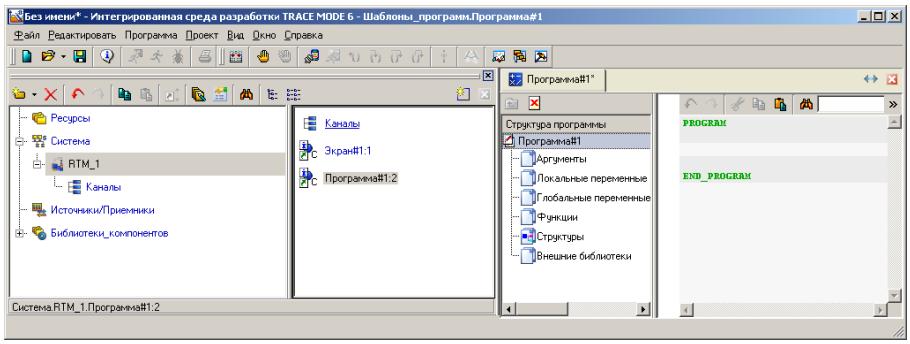


Рис. 79. Открытие поля редактирования текста программы

Казалось бы, все сделано, программа синусоидального сигнала реализована. Но на самом деле еще необходимо объявить глобальную переменную `q0`. Здесь проявляется различие между TRACE MODE и *Genie*. В случае SCADA-

пакета *Genie* переменные создаются автоматически, и все они являются глобальными. В случае же SCADA-пакета TRACE MODE надо явным образом объявлять переменную.

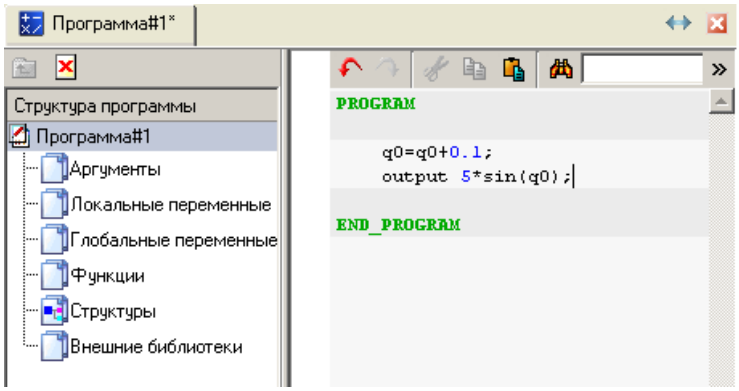


Рис. 80. Поле редактирования компонента «Программа#1» после добавления текста программы, реализующей синусоидальный сигнал

Чтобы убедиться в этом, попробуем откомпилировать данную программу. Тем самым мы проверяем правильность текста программы.

Для компиляции надо нажать клавишу F7 или нажать соответствующую кнопку на панели инструментов  .

Результаты компиляции отобразятся в специальном окне (рис. 81).

При этом выводятся сообщения о неизвестной переменной `q0`, а также о переменной `output`. Сравните эти сообщения об ошибках с сообщением о синтаксической ошибке в случае SCADA-пакета *Genie* (см. рис. 34). В случае SCADA-пакета TRACE MODE они, конечно, гораздо более подробны, но в принципе здесь работает примерно один и тот же механизм.

Учитывая результат компиляции (рис. 81), нетрудно придти к выводу о необходимости объявления переменной `q0`.

Здесь надо отметить, что можно было бы сначала объявить глобальную переменную `q0`, а уже потом добавлять текст программы. Но в данном примере используется другой порядок для того, чтобы проще было увидеть отличие от SCADA-пакета *Genie*.

Для этого нажмем на строку «Глобальные переменные» в столбце «Структура программы», что приведет к открытию таблицы глобальных переменных (см. рис. 82).

На этом рисунке представлен вид окна TRACE MODE в случае закрытия области «Навигатор проекта». Эту область можно закрывать и открывать с помощью меню «Вид=>Навигатор проекта».

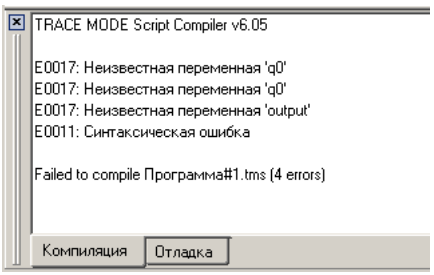


Рис. 81. Окно с результатами компиляции при отсутствии объявления переменной `q0`

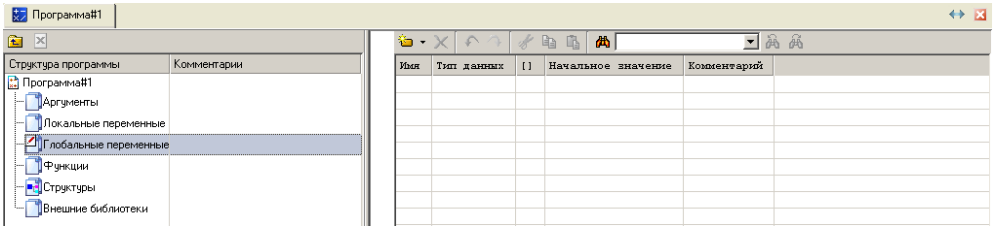


Рис. 82. Открытие таблицы глобальных переменных

В открывшейся таблице глобальных переменных создадим переменную `q0`. Для этого нажмем на значок  в левой верхней части этой таблицы.

Это приведет к созданию строки в таблице глобальных переменных (рис. 83).

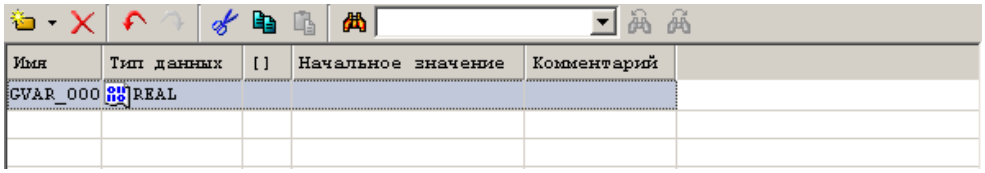


Рис. 83. Создание строки в таблице глобальных переменных

Тем самым создается переменная с именем «GVAR_000» и типом данных «REAL».

В этой строке таблицы изменим имя «GVAR_000» на «q0» и установим начальное значение, равное нулю (рис. 84).

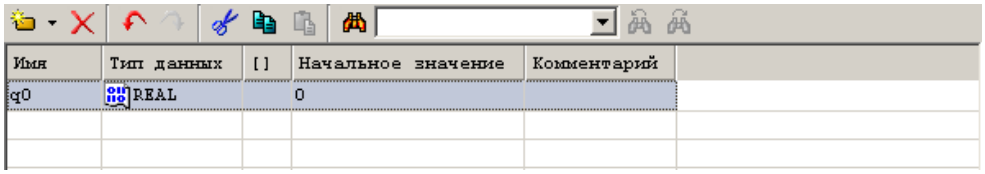


Рис. 84. Настройка глобальной переменной «q0»

Теперь можно снова откомпилировать программу, нажав клавишу F7 или нажав соответствующую кнопку на панели инструментов  . При этом выводится окно с сообщением о результатах компиляции (рис. 85).

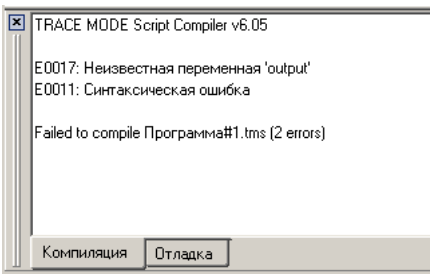


Рис. 85. Окно с результатами компиляции после объявления переменной q0

Как можно заметить, по сравнению с сообщениями о предыдущей попытке компиляции (см. рис. 81) теперь исчезли сообщения о неизвестной переменной `q0`. Однако осталось сообщение о неизвестной переменной `output`.

Это происходит по причине того, что текущая программа представляет собой текст, полностью скопированный из программы для SCADA-пакета *Genie*. Но как нетрудно догадаться, разные SCADA-пакеты по своему могут интерпретировать один и тот же текст программы. В общем случае могут использоваться разные языки программирования. В данном случае языки программирования SCADA-пакетов TRACE MODE и *Genie* являются похожими, но полностью не совпадают.

В частности, в случае *Genie* результат формулы `5*sin (q0)` выводится на выход с помощью специального оператора `output`. Значит, SCADA-пакет *Genie* воспринимает слово «output» как специальный оператор и «выполняет» (интерпретирует) его действие. А SCADA-пакет TRACE MODE «не знает» такого специального слова и пытается его интерпретировать как некоторую переменную, которая на самом деле не объявлена.

Как исправить программу, и что надо сделать, чтобы устранить данную ошибку компиляции (см. рис. 85)? Один из вариантов – это добавить знак «равно» между `output` и `5*sin (q0)`. Тогда текст программы примем вид, представленный на рис. 86.

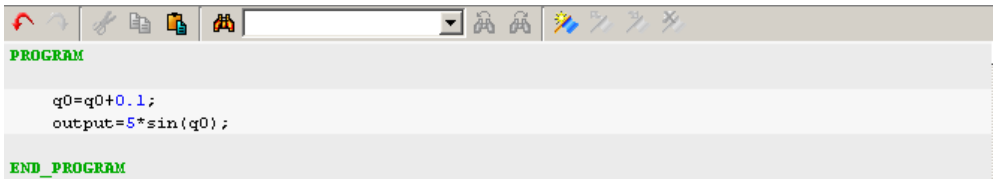


Рис. 86. Поле редактирования компонента «Программа#1» после исправления программы синусоидального сигнала

Здесь надо отметить, что для возврата обратно в редактор текста программы достаточно нажать на элемент «Программа#1» в столбце «Структура программы».

При компиляции этой исправленной программы также будет выводиться сообщение о неизвестной переменной `output`. Это происходит по той при-

чине, что данная переменная не была объявлена (подобно тому, как это было сделано в случае с переменной `q0`).

Но в отличие от переменной `q0` данная переменная должна быть выходной переменной компонента «Программа#1». Подобно тому, как в случае SCADA-пакета *Genie*, для формирования выходной переменной использовался оператор `output`. А в данном случае (после добавления знака «равно») получается, что выходная переменная `output` принимает результат формулы `5*sin (q0)`.

Остается только объявить данную выходную переменную, а точнее, выходной аргумент компонента «Программа#1». Для простоты под выходным аргументом мы будем понимать переменную специального вида (выходную переменную), значение которой может быть получено от данного компонента типа «Программа».

Для этого в столбце «Структура программы» нажмем на строку «Аргументы» (а не «Глобальные переменные», как в прошлый раз), что приведет к открытию таблицы аргументов, которая похожа на таблицу глобальных переменных.

В открывшейся таблице аргументов создадим аргумент «output». Для этого нажмем на значок  в левой верхней части этой таблицы.

Это приведет к созданию строки в таблице аргументов (рис. 87).

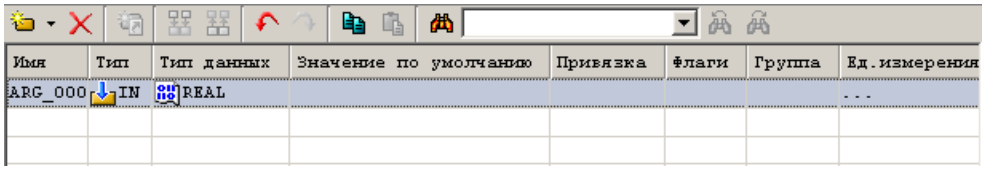


Рис. 87. Создание строки в таблице глобальных переменных

Таким образом, создается аргумент с именем «ARG_000» типом «IN» и типом данных «REAL».

В этой строке таблицы изменим имя «ARG_000» на «output» и изменим тип «IN» на «OUT» (рис. 88).

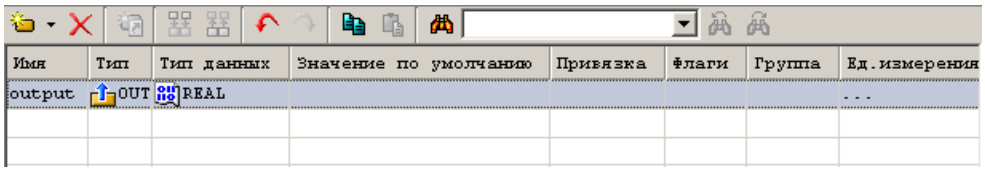


Рис. 88. Настройка выходного аргумента «output»

Теперь можно снова откомпилировать программу, нажав клавишу F7 или нажав соответствующую кнопку на панели инструментов  . При этом выводится окно с сообщением о результатах компиляции (рис. 89).

Это сообщение «compiled successfully» означает успешную компиляцию. Таким образом, получен вариант программы, не содержащий синтаксических ошибок.

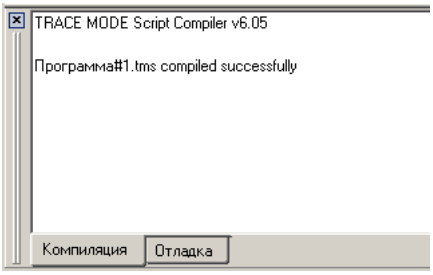


Рис. 89. Окно с результатами компиляции после добавления выходного аргумента «output»

Здесь надо добавить, что если окно вывода результатов компиляции было закрыто, то оно не будет появляться в случае успешной компиляции. Таким образом, отсутствие данного окна после компиляции само по себе является признаком успешности компиляция.

Теперь надо сделать так, чтобы выходной аргумент «output» можно было «использовать» вне компонента «Программа#1», в частности, чтобы можно было вывести значение этого аргумента (значение синусоидального сигнала) на график.

Для этого вернем отображение области «Навигатор проекта». Здесь надо напомнить, что эту область можно закрывать и открывать с помощью меню «Вид=>Навигатор проекта».

Теперь нажмем *правой* кнопкой мыши на компонент «Программа#1:2», который располагается в правой части навигатора проекта, и выберем в появившемся контекстном меню строку «Свойства». В нижней части откроется окно свойств компонента «Программа#1». В этом окне выберем вкладку «Аргументы», в которой отображается выходной аргумент «output», и *выделим* строку с этим аргументом (рис. 90).

В левой верхней части этого окна активизируется кнопка создания каналов с привязкой  . Нажмем на эту кнопку. В результате будет автоматически создан канал с именем аргумента (то есть с именем «output»), что можно наблюдать в правой части навигатора проекта (рис. 91), а также в строке выходного аргумента «output» появляется информация о привязке в соответствующем столбце (рис. 92).

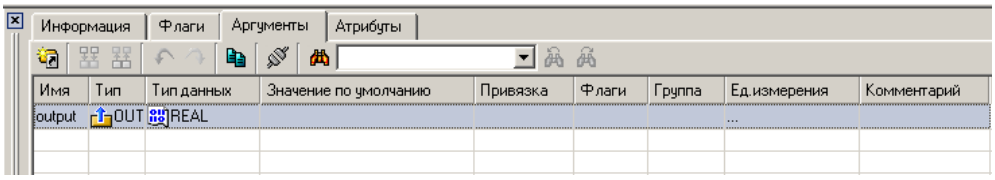


Рис. 90. Выделение аргумента «output» во вкладке «Аргументы» окна свойств компонента «Программа#1»

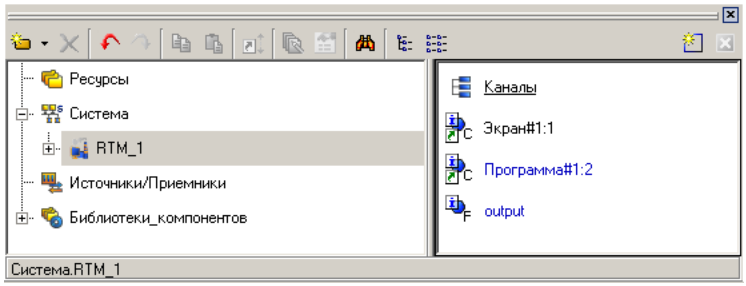


Рис. 91. Появившийся канал с именем «output» в навигаторе проекта

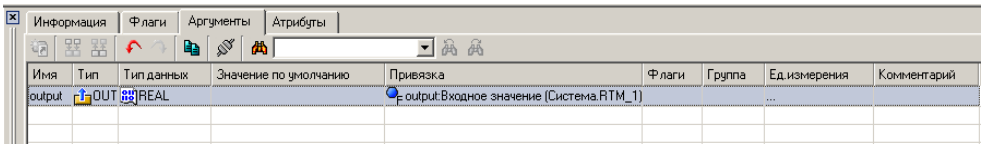


Рис. 92. Появившаяся информация о привязке выходного аргумента «output»

Однако чтобы был понятнее смысл вновь созданного канала, лучше переименовать его, задав ему новое имя «Синусоида». Для этого надо нажать *правой* кнопкой мыши на данный канал в навигаторе проекта и из контекстного меню выбрать пункт «Переименовать», а затем в появившейся строке ввода набрать новое имя. В итоге изменится имя канала (рис. 93), а также, естественно, изменится имя канала, к которому привязывается выходной аргумент «output» (рис. 96).

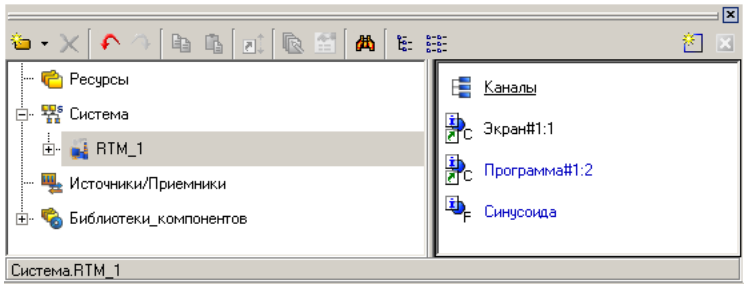


Рис. 93. Ранее созданный канал с новым именем «Синусоида»

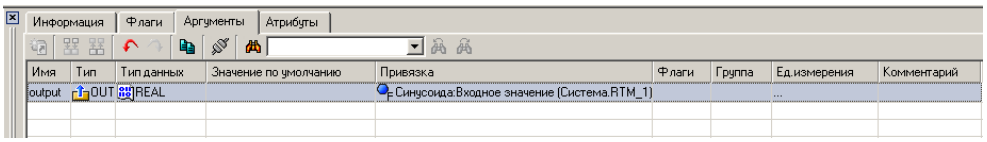


Рис. 94. Имя канала «Синусоида», появившееся в поле привязки выходного аргумента «output»

Теперь можно закрыть окно свойств компонента «Программа#1» (в этом окне находится вкладка «Аргументы», где назначалась привязка выходному

аргументу «output»). И можно приступить к созданию графика и выводу сигнала на график.

В навигаторе проекта дважды нажмем на элемент «Экран#1:1», и тогда правее откроется поле графического редактора, а также добавятся дополнительные панели инструментов в верхней части окна (см. рис. 74).

Теперь в пустое поле графического редактора, связанное с элементом «Экран#1:1», добавим графический элемент «Тренд». Для этого нажмем на соответствующую кнопку  на панели инструментов. После этого в поле графического редактора установим график, дважды нажав в произвольное место этого поля. При этом установится начальный вариант графического элемента «Тренд», а также откроется поле свойств этого графического элемента (рис. 95).

Теперь графический элемент «Тренд» можно перемещать по полю графического редактора, а также изменять размеры графического элемента «Тренд».

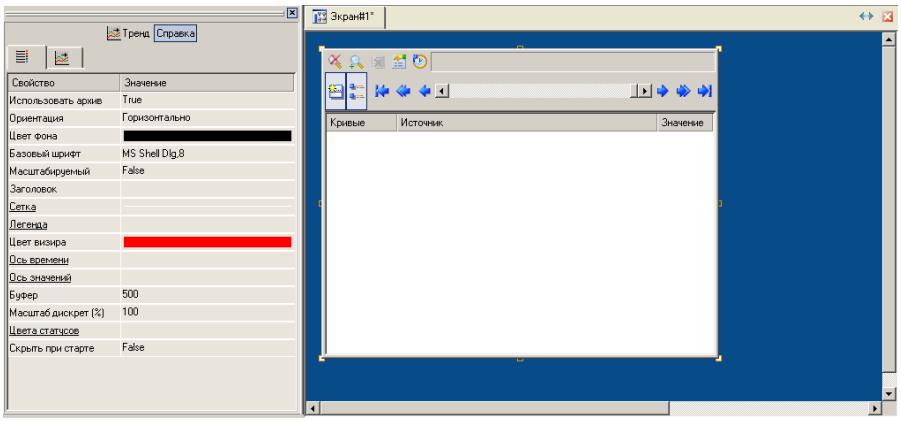


Рис. 95. Добавление графического элемента «Тренд» в поле редактирования

Слева от графического редактора открывается область свойств элемента «Тренд», которая подобна окну настроек аналогического графического элемента SCADA-пакета *Genie* (см. рис. 51). Только следует отметить, что в случае SCADA-пакета TRACE MODE этих настроек больше.

Для примера сделаем небольшое изменение этих настроек. Ранее при реализации программы на основе SCADA-пакета *Genie* в графиках ось времени не отображалась. По умолчанию в TRACE MODE она отображается.

Сделаем так, чтобы и в случае TRACE MODE ось времени не отображалась. Для этого надо в области свойств дважды нажать на поле «Ось времени» и в открывшемся поле «Показывать» изменить значение «True» на значение «False» (рис. 96).

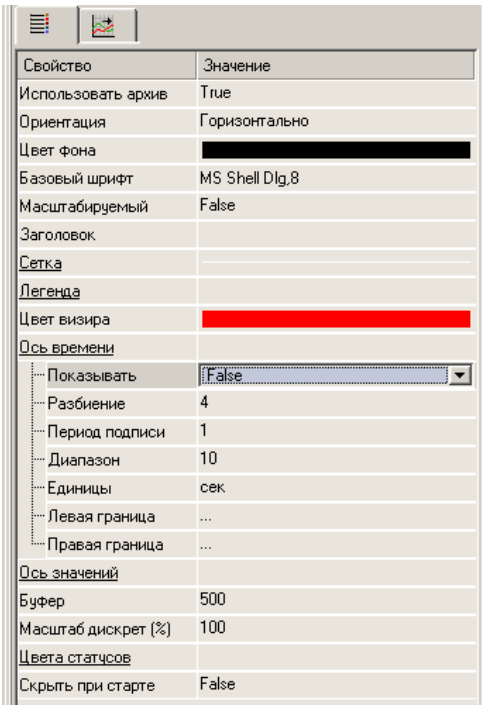


Рис. 96. Окно свойств элемента «Тренд» после изменения параметров оси времени

С помощью этого изменения ось времени для данного графика отключается. На практике это также отключает множество дополнительных возможностей этого графического элемента. Но в рамках рассматриваемого простого примера это как раз хорошо. Читателю в качестве отдельного упражнения оставляем возможность самостоятельно изучить дополнительные возможности этого графического элемента, реализуемые при включенной оси времени.

Таким образом, на основе отключения оси времени был рассмотрен пример изменения общих свойств данного элемента.

Но кроме общих свойств существуют также свойства отдельных *кривых*, выводимых на данном графике. Но для этого надо добавить хотя бы одну кривую.

Для этого в области свойств элемента «Тренд» (в левой части) перейдем на вкладку кривых, обозначаемую специальным значком  . В этой вкладке пока еще не добавлена ни одна кривая, и ее вид представлен на рис. 97.

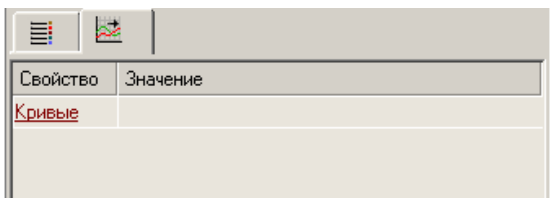


Рис. 97. Начальное содержимое вкладки кривых элемента «Тренд»

Для добавления кривой достаточно нажать правой кнопкой мыши в строке «Кривые» и выбрать единственный пункт контекстного меню «Кри-

вая». При этом в содержимое вкладки будут добавлены строки, соответствующие свойствам кривой, которая была добавлена (рис. 98).

Теперь надо сформировать привязку этой кривой к каналу «Синусоида». За счет этой привязки синусоидальный сигнал будет выводиться в виде кривой данного графика (элемента «Тренд»).

Для создании привязки в окне свойств кривой (см. рис. 98) нажмем на поле, содержащее «…», напротив поля со словом «Привязка». В результате этого откроется окно «Свойства привязки».

В левой верхней части этого окна нажмем кнопку создания аргумента  . В результате будет создан некоторый входной аргумент (рис. 99).

Изменим имя этого аргумента с «ARG_000» на «Синусоида», а также дважды нажмем на поле этого аргумента в столбце «Привязка». Это приведет к появлению окна конфигурирования связи, в котором раскроем (нажав на значок «+») компонент «Система», а затем компонент «RTM_1», и среди появившихся компонентов выберем канал «Синусоида» (рис. 100).

Теперь нажмем кнопку «Привязка» в этом окне. В результате окно закроется, и в окне «Свойства привязки» будет отображаться информация о сделанной привязке к каналу «Синусоида» (рис. 101).

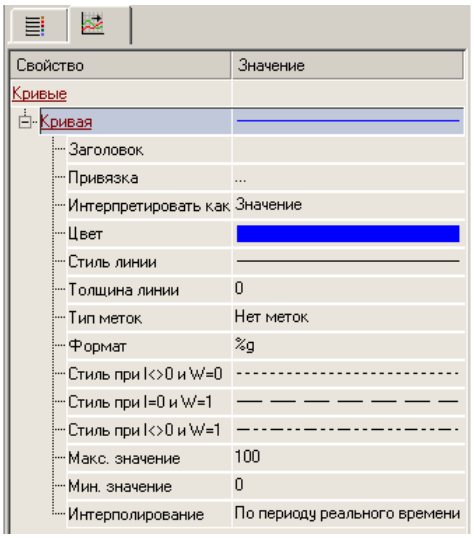


Рис. 98. Содержимое вкладки кривых элемента «Тренд» после добавления кривой

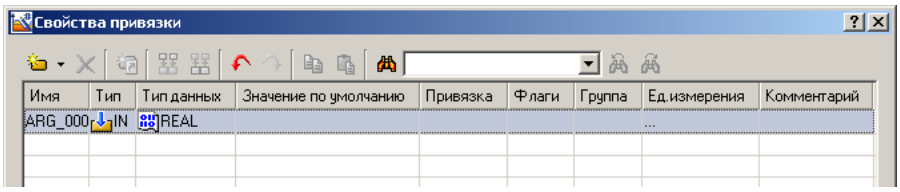


Рис. 99. Создание входного аргумента в свойствах привязки для кривой элемента «Тренд»

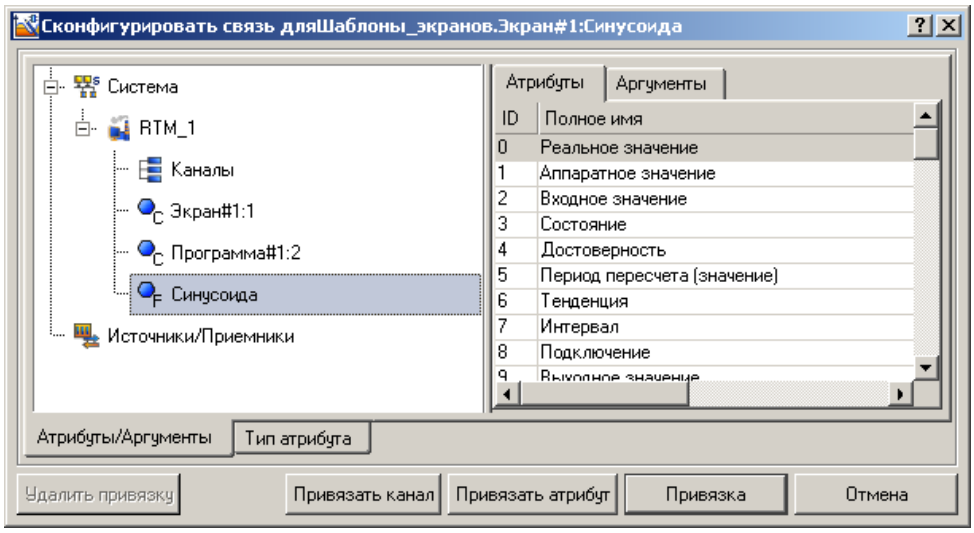


Рис. 100. Выбор канала «Синусоида» в окне конфигурирования связи

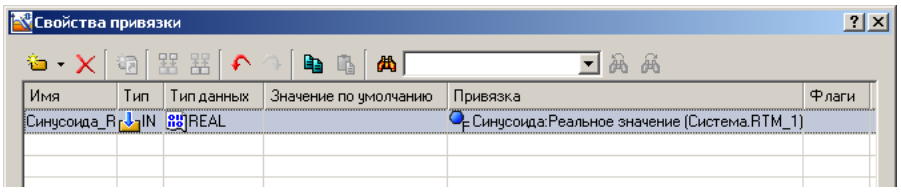


Рис. 101. Изменение входного аргумента в свойствах привязки для кривой элемента «Тренд»

При этом также автоматически поменяется имя аргумента «Синусоида» на «Синусоида_R».

Нажмем кнопку «Готово» в окне «Свойства привязки». В результате в окне свойств кривой в поле «Привязка» появится значение «Синусоида_R» (рис. 102).

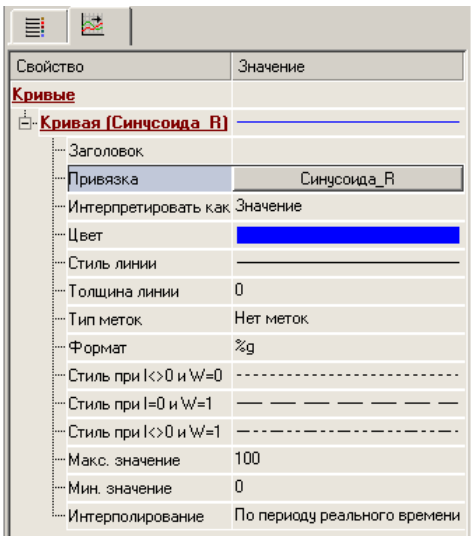


Рис. 102. Содержимое вкладки кривых элемента «Тренд» после привязки к каналу «Синусоида»

Сделаем еще некоторые изменения свойств кривой, а именно изменим:

– цвет кривой на зеленый (этот цвет использовался в примере для SCADA-пакета *Genie*);

– значения параметров «Макс. значение» и «Мин. значение» на 5 и –5 соответственно (диапазон, который соответствует диапазону синусоидального сигнала).

В результате окно свойств кривой примет вид, представленный на рис. 103.

Если обратиться к элементу «Тренд», расположенному в поле графического редактора экрана, то можно заметить, что этот элемент разделен на две части: собственно график (верхняя часть) и список кривых (нижняя часть) Можно не только изменять размеры и положение всего элемента, но также можно изменять соотношение занимаемой площади этими частями. Поскольку сейчас используется только одна кривая для графика, поэтому нижнюю часть можно уменьшить, «зацепив» мышью и сдвинув разделительную линию между этим частями. В итоге элемент «Тренд» примет вид, аналогичный тому, что представлен на рис. 104.

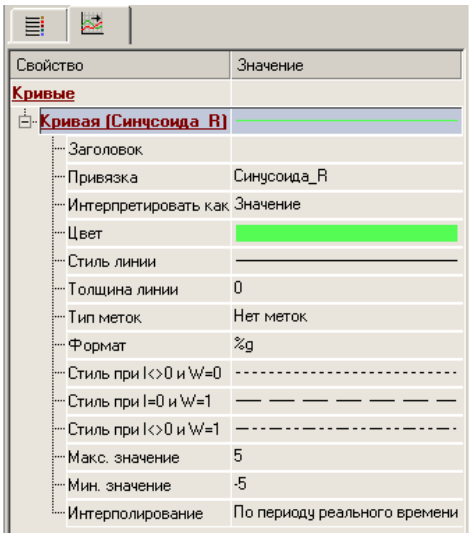


Рис. 103. Содержимое вкладки кривых элемента «Тренд» после изменения цвета и диапазона значения

Рис. 104. Элемент «Тренд» после изменения положения его частей

Надо отметить, что нижнюю часть элемента «Тренд» можно вообще скрыть, сдвинув разделительную линию до предела вниз. Но мы это делать не будем, чтобы было удобнее различать отдельные кривые данного графика.

Теперь можно попробовать запустить разработанную программу для проверки правильности ее создания.

Ранее при рассмотрении примера для SCADA-пакета *Genie* запуск программы осуществлялся с помощью программы-интерпретатора (*Advantech Genie Starter Runtime*).

В случае SCADA-пакета TRACE MODE программа-интерпретатор называется монитором реального времени. При этом имеется два типа таких мониторов:

– обычный монитор реального времени (используется для выполнения разработанной программы на целевом объекте, то есть на рабочем месте оператора);

– монитор реального времени, называемый профайлером (используется для отладки разрабатываемой программы).

Профайлер используется для отладки программы и входит в состав инструментальной системы. В эту инструментальную систему, помимо профайлера, еще входит интегрированная среда разработки, которую можно считать программой-редактором в ранее предложенной терминологии (см. рис. 35).

Попробуем запустить разработанную программу с помощью профайлера, который входит в состав инструментальной системы.

Для этого сначала сохраним проект с помощью меню «Файл=> Сохранить» или соответствующей кнопки  на инструментальной панели в верхней части окна. Надо отметить, что при первом сохранении следует задать имя проекта.

Теперь с помощью меню «Файл=>Сохранить для МРВ» или соответствующей кнопки  сохраним проект для монитора реального времени, что требуется для последующего запуска программы с помощью профайлера.

После этого в левой части навигатора проекта надо раскрыть компонент «Система» (если он свернут) и нажать на компонент RTM_1. Это делает активным пункт меню «Файл=>Отладка». Теперь можно запустить профайлер с помощью этого пункта меню или соответствующей кнопки  на инструментальной панели в верхней части окна.

При этом открывается окно профайлера, в котором отображается ранее сделанный график (элемент типа «Тренд») (рис. 105).

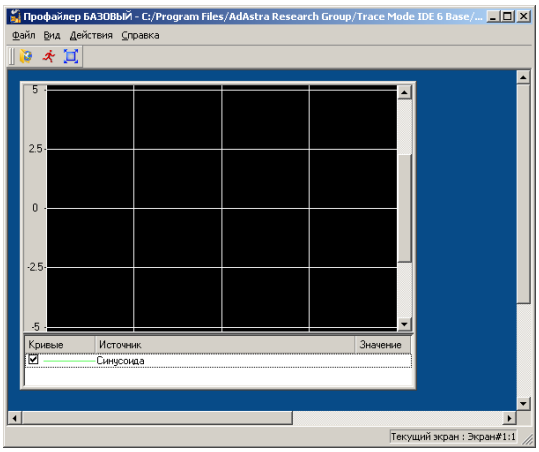


Рис. 105. Окно профайлера при его запуске для разрабатываемого примера программы

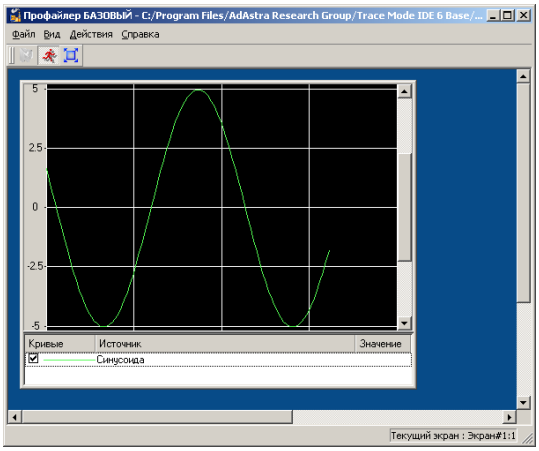


Рис. 106. Отображение синусоидального сигнала на графике в профайлере

Для запуска программы надо выбрать пункт меню «Файл=>Запуск/ Останов» или соответствующую кнопку  на инструментальной панели в верхней части окна. В результате на графике будет отображаться синусоидальный сигнал (рис. 106).

На рис. 106 отображен промежуточный результат реализации примера программы с помощью TRACE MODE, который ранее был выполнен на основе *Genie*. А именно этот промежуточный результат представляет собой вывод синусоидального сигнала.

Для выполнения оставшейся части примера необходимо:

– установить элемент графического интерфейса «Ползунок» и сделать управление верхним аварийным порогом с помощью этого элемента (результат реализации данного этапа с помощью *Genie* см. на рис. 58);

– сделать сравнение синусоидального сигнала с аварийным порогом и вывод сообщения о превышении этого порога с помощью индикатора (цветового и текстового) (результат реализации данного этапа с помощью *Genie* см. на рис. 59);

– реализовать два дополнительных источника сигнала и создать два дополнительных окна, то есть реализовать многооконный вариант программы (результат реализации данного этапа с помощью *Genie* см. на рис. 69, 70).

Читателю предлагается попробовать самостоятельно реализовать оставшиеся части примера с использованием SCADA-пакета TRACE MODE. Для того чтобы самостоятельно разобраться с особенностями работы в TRACE MODE, рекомендуется пользоваться встроенной справочной системой данного SCADA-пакета, которая предоставляет детальное описание основных компонентов SCADA-пакета, а также основ работы в его интегрированной среде разработки.

Справочная система вызывается с помощью меню «Справка=> Содержание».

Надо отметить, что справочная система является весьма подробной и содержит детально иллюстрированный материал (рис. 107). Поэтому удобно пользоваться этой справочной системой для самостоятельного изучения SCADA-пакета TRACE MODE.

Но даже рассмотрев только часть реализации примера с помощью TRACE MODE, уже можно сделать выводы об особенностях работы с этим SCADA-пакетом.

Во-первых, по сравнению с *Genie* в случае TRACE MODE мы имеем более обширный набор возможностей для реализации программы. Это, например, заметно при сравнении:

– набора готовых блоков графического интерфейса в *Genie* (см. рис. 49) и инструментальной панели TRACE MODE при активном графическом редакторе (см. рис. 74), учитывая, что каждая из кнопок инструментальной панели скрывает за собой набор различных вариантов готовых блоков (рис. 108);

– настроек отдельных блоков, например, настроек графика в *Genie* (см. рис. 57) и настроек графика в TRACE MODE, включающих в себя общие настройки (см. рис. 96) и настройки для отдельных кривых (см. рис. 103).

Во-вторых, SCADA-пакет TRACE MODE (по сравнению с *Genie*) рассчитан на гораздо более сложные программы, по крайней мере, в смысле количества компонентов, графических элементов и связей. В частности, это проявляется в том, что в случае TRACE MODE связи

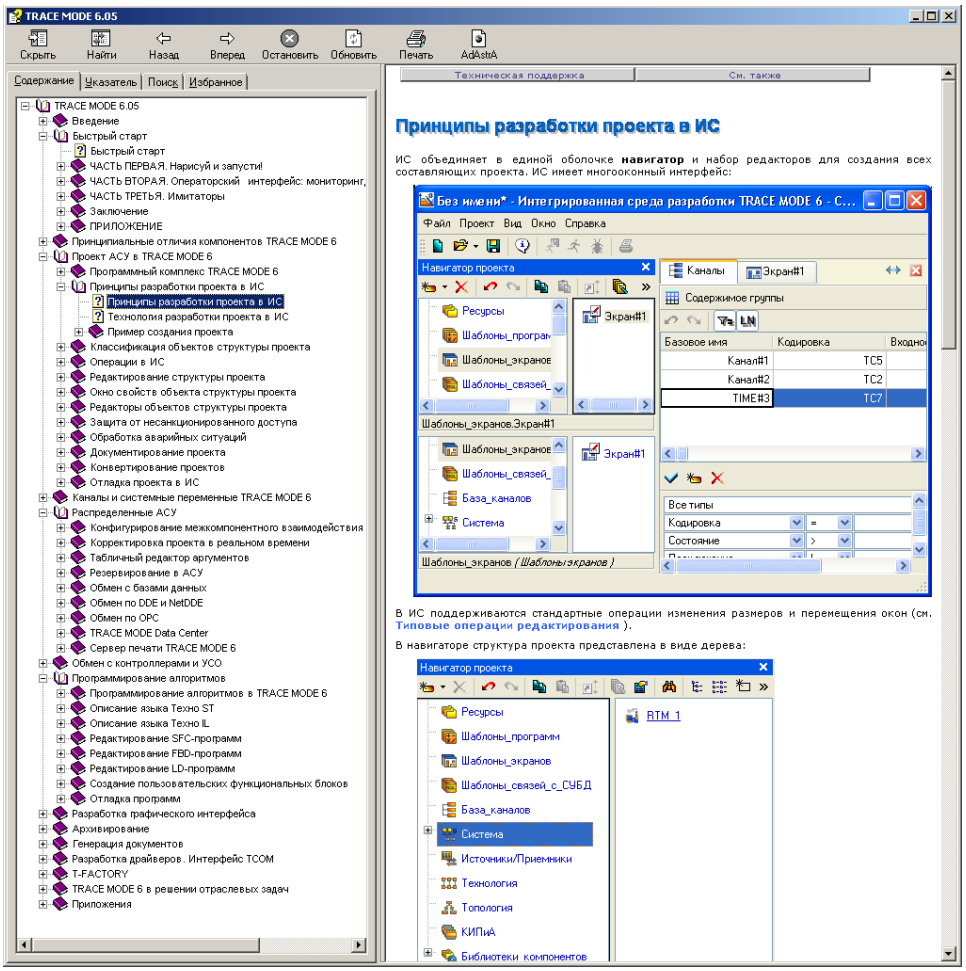


Рис. 107. Окно встроенной справочной системы SCADA-пакета TRACE MODE

Рис. 108. Инструментальная панель при активном графическом редакторе (показан набор различных видов трендов, появляющийся при нажатии на соответствующую кнопку)

отображаются не в графическом виде (как это делается в *Genie* с помощью стрелок), а скорее, в текстовом (табличном) виде и в виде так называемых привязок. Такой текстовый подход более удобен как раз для более крупных проектов. Ведь в большом проекте будет очень большое количество связей, и если попытаться отобразить эти связи в виде стрелок, то, скорее всего, получится очень запутанная картина, в которой будет довольно сложно разобраться. Но, с другой стороны, такой текстовый подход может оказаться сложным для начального изучения. Поэтому проще изучить принципы работы со SCADA-пакетом на основе такого пакета, как *Genie*, в котором используется графический подход для отображения связей. А уже затем переходить к более сложному SCADA-пакету. И при переходе к SCADA-пакету с текстовым подходом представления связей можно мысленно представлять графическое

отображение связей в виде стрелок, которое было ранее опробовано на таком простом SCADA-пакете, как *Genie*.

В качестве практического упражнения читателю рекомендуется реализовать этот же пример на основе других SCADA-пакетах, например, на основе CitectSCADA [11].

2.2. Разработка алгоритмов управления с помощью SCADA-пакетов

Под *алгоритмами управления* применительно к SCADA-системам мы будем понимать алгоритмы, которые формируют управляющие воздействия, передаваемые на нижние уровни САиУ, например, в качестве непосредственных команд для ИУ, передаваемых посредством УСО, или в качестве установочных значений для контуров управления, реализуемых на УВМ нижних уровней. Здесь важно подчеркнуть, что эти управляющие воздействия формируются автоматически (алгоритмом, реализованном в виде определенной части ПО), а не человеком-оператором.

SCADA-пакеты обычно реализуют ПО для верхних уровней САиУ, обеспечивая интерфейс с человеком-оператором, предоставляя ему необходимые возможности для реализации требуемых управляющих воздействий. Если рассматривать САиУ в целом, то большая часть *автоматически формируемых* управляющих воздействий реализуется на уровне УВМ, которые непосредственно связаны с целевым процессом, например, на основе УВМ в виде ПЛК. В каких же случаях может потребоваться *автоматически* формировать управляющие воздействия с помощью SCADA-пакетов? В целом, можно выделить три основных случая, когда это может потребоваться:

1) при реализации последовательностей управляющих воздействий (в виде так называемых *сценариев*), которые могли бы формироваться человеком-оператором, но осуществляются автоматически (в данном случае алгоритмы управления – это алгоритмы выполнения сценариев);

2) при использовании SCADA-пакета в качестве ПО для УВМ нижнего уровня;

3) при использовании SCADA-пакета в качестве ПО для УВМ в составе САиУ централизованной архитектуры (рис. 12).

Второй и третий случай по сути мало отличаются друг от друга за исключением того, что в третьем случае одно и то же ПО должно взаимодействовать как с процессом, так и с человеком (посредством УСО и УВО соответственно). Поэтому эти два случая могут рассматриваться в рамках одного и того же обобщенного случая, который предполагает использование SCADA-пакета на УВМ, взаимодействующей с целевым процессом.

В составе SCADA-систем могут быть уже готовые функциональные блоки для реализации типовых алгоритмов управления. Например, в составе TRACE MODE есть специальные функциональные блоки, которые позволяют реализовывать необходимые алгоритмы регулирования и управления (см. рис. 109, 110).

Рис. 109. Типовые функциональные блоки в составе SCADA-пакета TRACE MODE для реализации алгоритмов регулирования

Рис. 110. Типовые функциональные блоки в составе SCADA-пакета TRACE MODE для реализации алгоритмов управления

По каждому из этих блоков во встроенной справочной системе имеется подробная информация, например, на рис 111 представлена часть страницы справочной системы о блоке «Звено PID», который соответствует ПИД-регулятору. Этот рисунок демонстрирует пример реализации такого регулятора средствами TRACE MODE. Также это иллюстрирует ранее сделанное утверждение о том, что справочная система TRACE MODE является достаточно подробной. Поэтому для реализации нужного алгоритма управления на основе данного SCADA-пакета во многих случаях будет достаточно этой справочной системы, а также необходимых знаний, умений и навыков, связанных с областью теории автоматического управления.

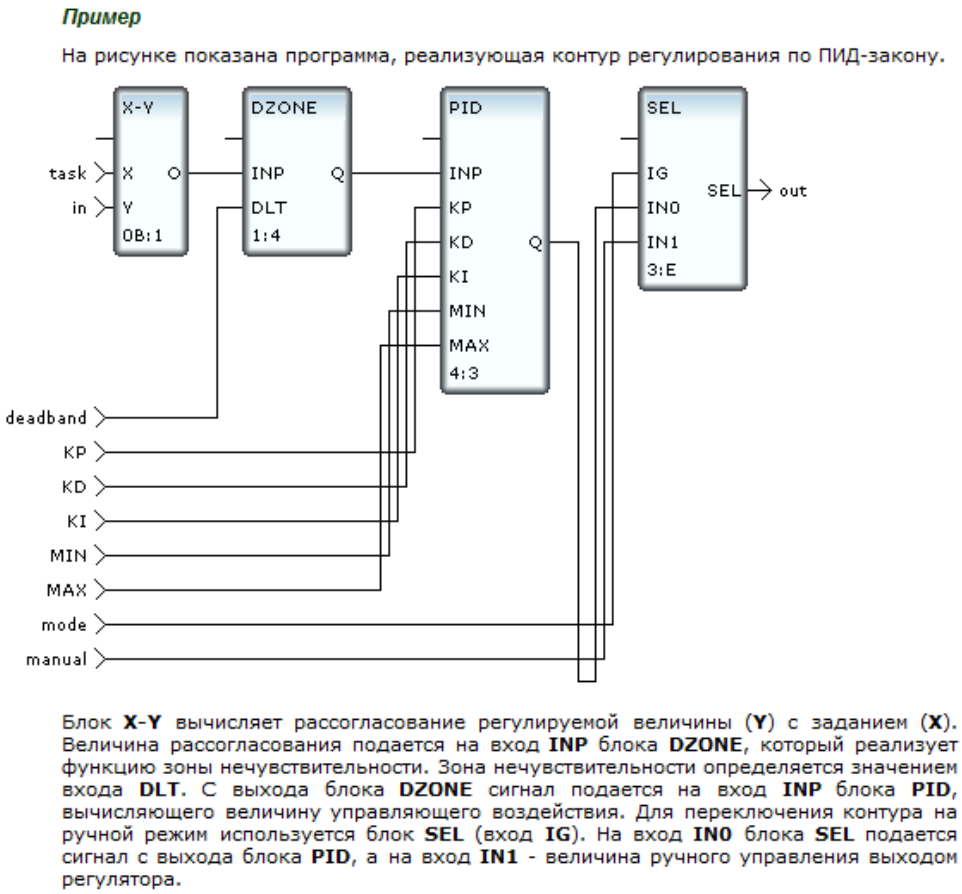


Рис. 111. Часть описания функционального блока «Звено PID» из справочной системы SCADA-системы TRACE MODE

Примеры использования алгоритмов управления для автоматического управления целевыми процессами совместно с *моделями* объекто управления будут продемонстрированы в п. 2.3, а также будут подробно рассматриваться на практических и лабораторных занятиях.

Осталось обратиться к вопросу реализации алгоритмов управления, обеспечивающих выполнение сценариев.

2.2.1. Разработка алгоритмов выполнения сценариев на основе SCADA-пакетов

Бывают ситуации, когда верхний уровень, на котором функционирует SCADA-пакет, должен формировать последовательности команд для нижних уровней САиУ. Причем эти команды могут формироваться человеком-оператором, то есть к ним не предъявляются такие требования по скорости реакции, которые были бы недоступны человеку. Но в силу того, что эти команды должны повторяться многократно в различных (определенных) последовательностях, может оказаться, что процесс их формирования человеком является монотонной, рутинной деятельностью, от которой этого человека можно освободить.

Например, рассмотрим САиУ, которая реализуется систему автоматических испытаний двигателя. Для того, чтобы всесторонне испытать этот двигатель на надежность при разных режимах функционирования, требуется имитировать различные режимы его работы. Для этого на двигатель со стороны САиУ с помощью ИУ передаются различные воздействия, имитирующие состояния, которые соответствуют тем или иным режимам работы этого двигателя в реальных условиях. То есть испытания стараются приблизить к реальным условиям эксплуатации двигателя. На верхнем уровень человек-оператор может вручную переключать режимы имитации для двигателя. Каждому такому режиму может соответствовать та или иная последовательность воздействий на двигатель. Но сами эти режимы необходимо время от времени изменять и определенным образом чередовать. Это может делать человек-оператор, но если эти действия требуется повторять длительное время в различных комбинациях, то целесообразно этот процесс автоматизировать. Его можно автоматизировать именно на верхнем уровне, то есть с помощью ПО, реализуемого на основе SCADA-пакета. Это вполне естественно, так как при этом обеспечивается единый уровень (уровень SCADA-пакета) для действий одного и того же типа, то для действий, который в данном примере переключают режимы испытаний. Тогда сохраняется возможность переключения режимов как в ручном, так и в автоматическом режиме, например, оператор может прервать автоматическое переключение режимов и начать переключать их вручную.

В таких ситуациях, могут формироваться последовательности команд на специальном языке, например, некотором языке сценариев, который потом исполняется ПО, реализованном с помощью SCADA-пакета. Подобные файлы сценариев могут даже формироваться человеком-оператором в ходе функционирования САиУ. Эти файлы сценариев могут считываться ПО на основе SCADA-пакетов и интерпретироваться (исполняться) согласно заданному языку сценариев. Другой вариант - последовательности команд (сценарии) могут программироваться стандартными средствами SCADA-пакета.

2.2.1.1.Пример разработки алгоритма выполнения сценариев на основе SCADA-пакета

В качестве примера SCADA-пакета используем пакет Genie, который ранее использовался для выполнения примера реализации пользовательского интерфейса (см. п. 2.1.3). Простота этого SCADA-пакета позволит показать, без отвлечения на лишние детали, простейший принцип реализации выполнения сценария.

Рассмотрим следующую задачу. Имеется три источника сигнала: «Синусоида», «Треугольник», «Случайный». Эти три сигнала ранее использовались

в примере п. 2.1.3. Теперь допустим, что требуется один из этих сигналов передавать на нижний уровень САиУ в течение определенного времени, а потом переключать на другой сигнал и т. д. То есть требуется реализовать некоторый сценарий переключения сигналов. Можно считать, что такая САиУ реализует систему автоматических испытаний некоторого изделия, проверяя поведение и характеристик этого изделия при подаче на него различных сигналов в разной последовательности. Здесь специально рассматривается очень упрощенный случай, но он позволит продемонстрировать некоторые базовые принципы реализации сценариев с помощью SCADA-пакетов.

Сценарии могут быть разными, для определенности выберем два конкретных сценария, чтобы в данном примере реализовать возможность их выполнения и выбора одного из них.

Сценарий 1 состоит в том, чтобы чередовать эти три сигнала в строгой последовательности с интервалом в 5 секунды.

Сценарий 2 состоит в том, что через случайный интервал времени (от 3 до 15 секунд) переключение осуществляется на один из сигналов, причем новый сигнал выбирается случайным образом среди трех имеющихся.

Оператор имеет возможность активировать один из этих сценариев.

Для реализации данного примера возьмем в качестве основы пример, ранее разработанный в п. 2.1.3, и дополним его следующим образом.

Для начала реализуем переключение трех сигналов вручную и вывод результата переключения на отдельный график в отдельное окно. Для этого добавим новое окно (блок «DISP4»), в этом окне разместим элементы графического интерфейса «График» и «Ползунок», подобно тому, как это показано на рис. 53. Отличие будет в том, что надо изменить свойства элемента «Ползунок» так, чтобы в нем было только три деления, причем дискретно переключаемых. Это позволит вручную переключать один из трех сигналов. Для этого надо открыть свойства этого элемента и установить свойства, как показано на рис. 112. Обратите внимание на свойство «Slider Action», которое сейчас имеет значение INCREMENTAL. Это позволяет дискретно переключать значения. Также изменяются свойства «Tics Number», «Tics Start», «Tics End».

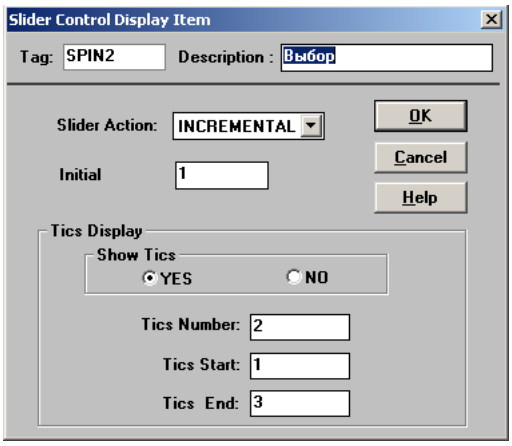


Рис. 112. Настройка свойств элемента «Ползунок» для ручного переключения сигналов

Теперь в конструкторе стратегии установим дополнительный блок «user prog», назовем его «Выбор» и подсоединим к нему все три сигнала, а также сигнала от только что установленного элемента ползунок (рис. 113). Этот блок будет использоваться для выбора (переключения) одного из трех сигналов, и его выход (Output 0) подсоединяется к «DISP4».

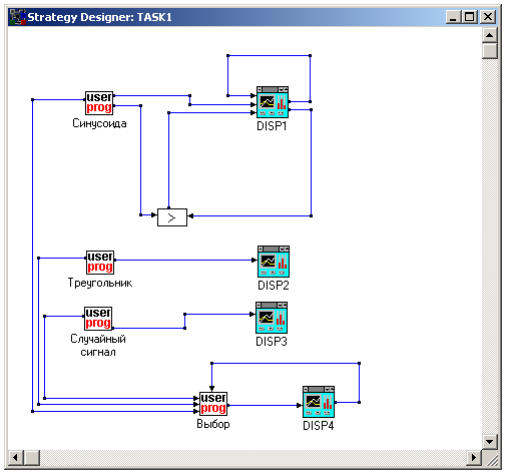


Рис. 113. Подключение блока «Выбор» для реализации переключения сигналов

В установленном блоке «Выбор» поместим программу следующего вида.

```
if (SPIN2==1) output(PRG1);
if (SPIN2==2) output(PRG2);
if (SPIN2==3) output(PRG3);
```

В этой программе SPIN2 соответствует элементу «Ползунок», который был только что добавлен. PRG1, PRG2, PRG3 соответствуют трех сигналам, который переключаются.

Теперь остается в новом окне, которое соответствует «DISP4», в свойствах элемента «График» подключить сигнал от блока «Выбор». Также в свойствах элемента «График» лучше поменять значение «Range of x axis» так,

чтобы в поле «to» было значение +500. Это изменит масштаб по оси времени, что позволит наблюдать больше переключений сигналов одновременно.

После этого можно запустить полученную программу и попробовать переключать сигналы. На графике можно будет наблюдать процесс ручного переключения сигнала (рис. 114).

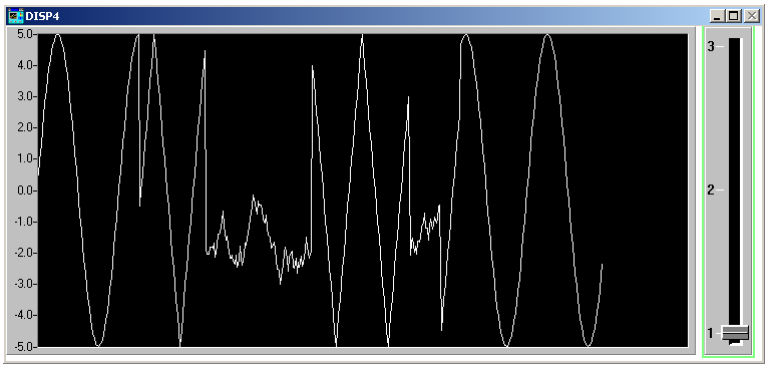


Рис. 114. Отображение на графике процесса ручного переключения трех сигналов

Теперь реализуем сценарии. Для этого разместим в окне для «DISP4» новый элемент «Ползунок», назначив ему те же свойства, что и предыдущему, но задав новое название: «Режим». Этот элемент будет служить для переключения режима. Значения 1 и 2 будут означать сценарий 1 и 2 соответственно, а значение 3 будет означать ручной режим, который уже реализован.

Теперь в конструкторе стратегии разместим два блока «user prog» для двух сценариев (рис. 115). При этом соединения от этих блоков поступают в блок «Выбор». Также в блок «Выбор» направляется соединение от элемента типа «Ползунок», который только что был добавлен.

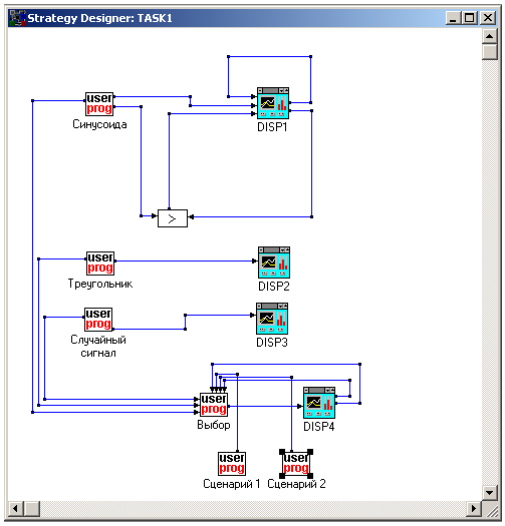


Рис. 115. Добавление блоков «user prog» для реализации сценариев

В блок «Сценарий 1» добавляем программу следующего вида.

```
if (t1 <= 0) {
```

```
 t1 = 50;
 s1 = s1 + 1;
 if (s1 > 3) s1 = 1;
}
t1 = t1 - 1;
output s1;
```

В этой программе реализуется задержка в 5 секунд между переключениями сигналов. Для этого используется переменная t1, которой присваивается значение 50, и значение этой переменной на каждом такте системе уменьшается на 1. Время одного такта системы ранее было установлено равным 100 мс (см. рис. 44), поэтому 50 тактов соответствует 5 секундам. Когда значение t1 в очередной раз становится равным 0, происходит новое присвоение значения 50, а также выполняется переключение сигнала за счет установки нового значения переменной s1, которая и подается на выход данного блока. И это реализует ранее сформулированный сценарий 1.

В блок «Сценарий 2» добавляем программу следующего вида.

```
if (t2 <= 0) {
 t2= rnd(0) % 121 + 30;
 r = rnd(0) % 2 + 1;
 s2 = s2 + r;
 if (s2 > 3) s2 = s2 - 3;
}
t2 = t2 - 1;
output s2;
```

Отличие этого блока от предыдущего (кроме других названий переменных) состоит в том, что:

1) Новое значение переменной t2 задается на основе генератора псевдослучайных чисел;

2) Новый сигнал выбирается случайным образом из двух оставшихся, для этого используется переменная r, которая реализует случайное смещение номера сигнала (на 1 или на 2).

Для того, чтобы блок «Выбор» мог выбирать между сценариями и ручным режимом изменим его программу на следующую.

```
if (SPIN3==1) {
if (PRG5==1) output(PRG1);
if (PRG5==2) output(PRG2);
if (PRG5==3) output(PRG3);
}
if (SPIN3==2) {
if (PRG6==1) output(PRG1);
```

```
if (PRG6==2) output(PRG2);
if (PRG6==3) output(PRG3);
}
if (SPIN3==3) {
if (SPIN2==1) output(PRG1);
if (SPIN2==2) output(PRG2);
if (SPIN2==3) output(PRG3);
}
```

В этой программе проверяется значение переключателя режимов (SPIN3) и на основе этого определяется, значение с какого блока (PRG5, или PRG6, или SPIN2) является номером сигнала. За счет этого значения (с блока «Сценарий 1», или «Сценарий 2», или «Выбор») происходит переключение сигналов.

В итоге при запуске программы можно переключаться между сценариями и ручным режимом. Можно наблюдать, что график результирующего сигнала соответствует сценарию (рис. 116, 117).

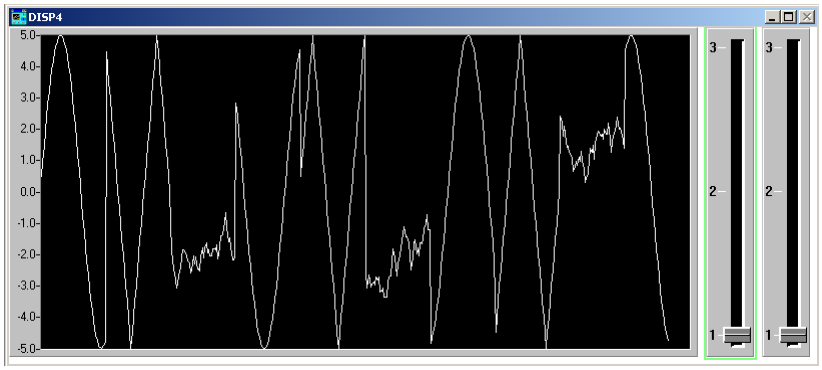


Рис. 116. Отображение на графике процесса переключения трех сигналов при выборе сценария 1

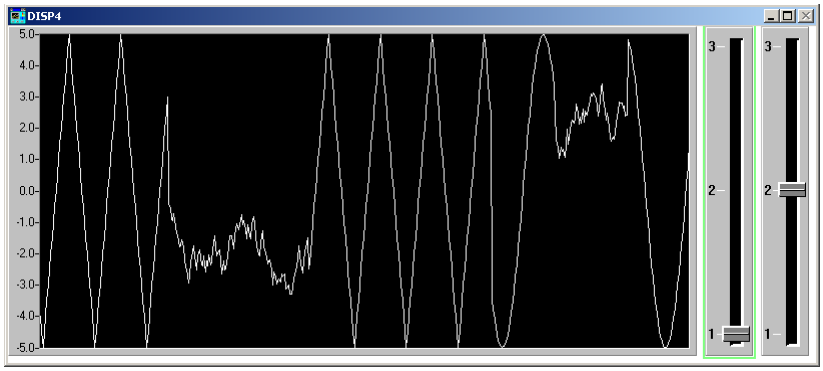


Рис. 117. Отображение на графике процесса переключения трех сигналов при выборе сценария 2

Таким образом, с помощью SCADA-пакета *Genie* на базе предыдущего примера реализован алгоритмы управления согласно заданным сценариям. В

качестве практического упражнения читателю рекомендуется реализовать этот пример на основе других SCADA-пакетах, например, TRACE MODE, CitectSCADA [11].

2.3. Компьютерное моделирование при разработке и отладке программного обеспечения систем автоматизации и управления

При разработке и отладке ПО САиУ часто возникает необходимость в имитации объекта управления, так как начальная разработка ПО обычно осуществляется вне связи с реальным объектом управления. Потребность в имитации объекта управления возникает из-за того, что алгоритмы управления надо протестировать, чтобы на ранних стадиях выявить какие-либо серьезные недочеты. Также важно иметь хотя бы грубую модель реального объекта, чтобы проще было реализовывать и тестировать пользовательский интерфейс и другие функции ПО верхних уровней САиУ.

Проблема построения математических моделей объектов управления относится к отдельной области исследований и более подробно разбирается в рамках соответствующей учебной дисциплины. Здесь мы рассматриваем вопрос, как из готовых математических описаний моделей можно построить их программную реализацию.

Один из типовых подходов отладки алгоритмов управления на основе имеющихся математических моделей – это использование специализированных пакетов математического моделирования, таких как, например, MATLAB.

Даже в случае, если ПО для промышленного компьютера или контроллера разрабатывается с использованием специализированных систем программирования, которые несовместимы с языком пакета для моделирования, все равно существуют возможности сборки гибридной системы для тестирования. Например, пакет моделирования функционирования на обычном компьютере, которые через те или иные интерфейсы взаимодействует с контроллером, на котором тестируется и отлаживается ПО, реализующее алгоритмы управления.

Как уже отмечалось и обосновывалось, в рамках данной учебной дисциплины мы концентрируемся на системах программирования для верхних уровней САиУ, а именно на SCADA-пакетах. Поэтому обратимся к тому, как используются и разрабатываются модели применительно к SCADA-пакетам.

Конечно, можно создавать гибридные системы на взаимодействующих компьютерах, как было указано выше. Можно также создавать гибридные системы в рамках одного компьютера, когда на этом компьютере совместно функционируют SCADA-пакет и пакет для моделирования. В этом случае

следует отдельно рассматривать две задачи: реализация модели объекта управления в пакете моделирования и реализация алгоритма управления в SCADA-пакете. Что касается реализации алгоритмов управления средствами SCADA-пакета, то это был рассмотрено ранее (см. п. 2.2). Реализация модели объекта управления в том или ином пакете моделирования – это задача, которая специфична для конкретного пакета моделирования и решается обычно путем преобразования математического описания модели объекта управления в программную реализацию согласно имеющимся средствам пакета моделирования. Также отдельный и сложный вопрос – это совмещение SCADA-пакета и пакета моделирования в рамках единой системы.

Но есть еще один интересный подход, который основан на том, что модели объекта управления могут создаваться средствами самого SCADA-пакета. В этом случае отпадает необходимость в построении гибридной системы и решении сложной задачи совмещения SCADA-пакета и пакета моделирования в единую систему.

SCADA-пакеты как правило обладают достаточно универсальными средствами программирования, которые позволяют помимо алгоритмов управления также реализовывать и модели объектов управления. Этот вопрос подробно рассматривается в ходе практических и лабораторных занятий. Здесь же в качестве начального знакомства с этой темой разберем конкретный пример использования и разработки моделей объектов управления при использовании SCADA-пакетов.

2.3.1. Использование и разработка компьютерных моделей объектов управления при применении SCADA-пакетов

2.3.1.1.Пример на основе SCADA-пакета TRACE MODE

Рассмотрим использование модели объекта на примере SCADA-пакета TRACE MODE.

В рамках данного пакета имеется специальный функциональный блок «OBJ», который позволяет реализовать модель объекта управления (см. рис. 118).

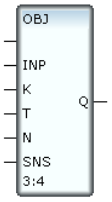


Рис. 118. Функциональный блок «OBJ» SCADA-пакета TRACE MODE

Этот блок реализует апериодическое звено 1-го порядка с передаточной функцией

$$W(p)=\frac{K}{Tp+1},$$

а также позволяет добавлять чистое запаздывание и случайную помеху. При этом коэффициенты *K*, *T*, которые являются одними из входов данного блока (см. рис. 118) задают параметры данной передаточной функции. Кроме того, параметр *N* задает время чистого запаздывания, которое можно считать результатом добавления звена чистого запаздывания к указанно передаточной функции. Еще один коэффициент *SNS* задает параметры случайных помех. Вход *INP* – это вход объект, на него можно подавать управляющее воздействие, сформированное алгоритмом управления. Данный блок имеет единственный выход, который является выходом данной модели объекта.

Таким образом, с помощью данного блока, а также комбинаций этих блоков можно задавать различные варианты сложных объектов управления для задач моделирования, например, с целью проверки качества алгоритмов управления.

Рассмотрим следующий простой пример. Пусть имеется модель объекта управления, которая задается одним блоком «OBJ» (см. рис 118) с параметрами *K*=1, *T*=100, *N*=0, *SNS*=0. Кроме того, модель включает в себя возможность потери управляющего сигнала, подаваемого на объект управления. Такая потеря сигнала, например, может быть связана с помехами или другими причинами, вызывающими потерю связи с УСО или ИУ, которые должны передавать управляющее воздействие на объект. Тем самым, моделируется аварийная ситуация, в которой сигнал с выхода регулятора в течение некоторого промежутка времени не поступает на вход объекта.

Допустим, что было принято решение использовать ПИ-регулятор для управления этим объектом, который является частным случаем ПИД-регулятора. Коэффициенты регулятора были рассчитаны и проверены на модели объекта без учета потери связи. При этом ПИД-регулятор может быть реализован на основе схемы, подобной той, что представлена на рис. 111.

Пусть теперь требуется проверить с помощью компьютерного моделирования, каким образом поведет себя регулятор, когда управляющее воздействие в течение некоторого периода времени перестает подаваться на объект управления.

В ходе экспериментов оказалось, что использование обычного ПИ-регулятора приводит к тому, что в течение этого периода (когда воздействие не поступает на объект) будет происходить неограниченное накопление интегральной суммы (суммы, используемой для расчета интегральной составляющей ПИ-регулятора). Когда же сигнал снова начнет поступать на вход объ-

екта, то тогда накопленная сумма приведет либо к резкому превышению допустимых значений выхода объекта, либо к резкому ухудшению качества регулирования. Во многих случаях не помогает и ограничение на выход регулятора, так как слишком жесткое ограничение, хотя и предотвращает резкие превышения, но при этом ухудшает качество регулирования. Решением в этой ситуации может быть ограничение на значение интегральной суммы. Но такое ограничение нельзя реализовать на основе стандартного функционального блока «Звено PID» (см. рис. 111).

Однако можно реализовать подобный алгоритм управления на основе совокупности других блоков. И это хороший пример того, что хотя в некоторых случаев стандартные блоки данного SCADA-пакета могут не удовлетворять заданным требования, часто остается возможность реализовать нужную функцию на основе комбинации других блоков.

Так, в случае данного примера, можно реализовать ПИ-регулятор из простых блоков, а также дополнить его возможностью установки допустимых граничных значений для интегральной суммы: min_sum, max_sum. Кроме того, как и в случае стандартного блока «Звено PID», добавим ограничения на выход регулятора (min_out, max_out). В итоге получается схема, представленная на рис. 119.

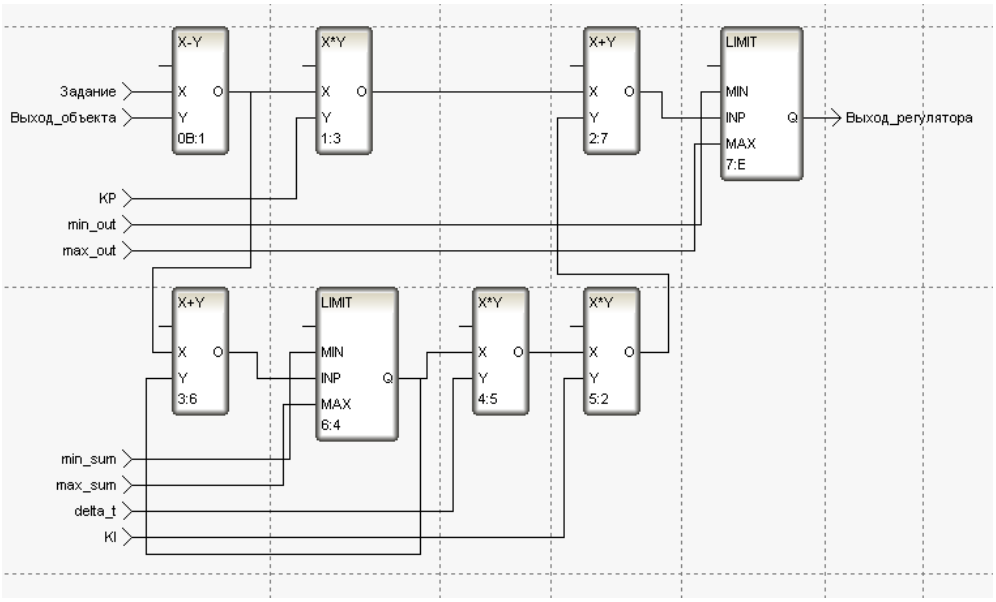


Рис. 119. Реализация ПИ-регулятора с возможностью ограничения интегральной суммы

В верхнем ряду этой схемы (будем перечислять блоки слева направо) сначала идет блок расчета ошибки рассогласования, которая поступает на вход расчета пропорциональной составляющей с коэффициентом *KP*, а также в нижнюю часть схемы для расчета интегральной составляющей. Далее в верхнем ряду имеется блок для суммирования пропорциональной и инте-

гральной составляющих, выход которого поступает на блок, формирующий выход регулятора с учетом ограничений согласно границам min_out, max_out. В нижнем ряду схему формируется интегральная составляющая. Крайний левый блок этого ряду реализует суммирование ошибки рассогласования с уже имеющейся суммой. Следующий блок ограничивает величину этой суммы согласно заданным границам min_sum, max_sum. Полученная сумма поступает на цепочку из двух блоков-умножителей, которые реализуют вычисление интегральной составляющей. Тем самым, реализуется ПИ-регулятор, модифицированный за счет ограничения интегральной составляющей.

Теперь полученный регулятор можно проверить на имеющейся модели и оценить его работу. Для этого на вход регулятора будем подавать задание в качестве периодически изменяющегося прямоугольного сигнала. Это позволит наблюдать переходные процессы на каждом из ступенчатых воздействий, формируемых этим сигналом. В графическом интерфейсе (рис. 120, 121) реализуем компоненты, которые позволяют включать имитацию потери управляющего сигнала, чтобы проследить, как это скажется на качестве переходного процесса. Также с помощью элементов графического интерфейса можно задавать параметры регулятора.

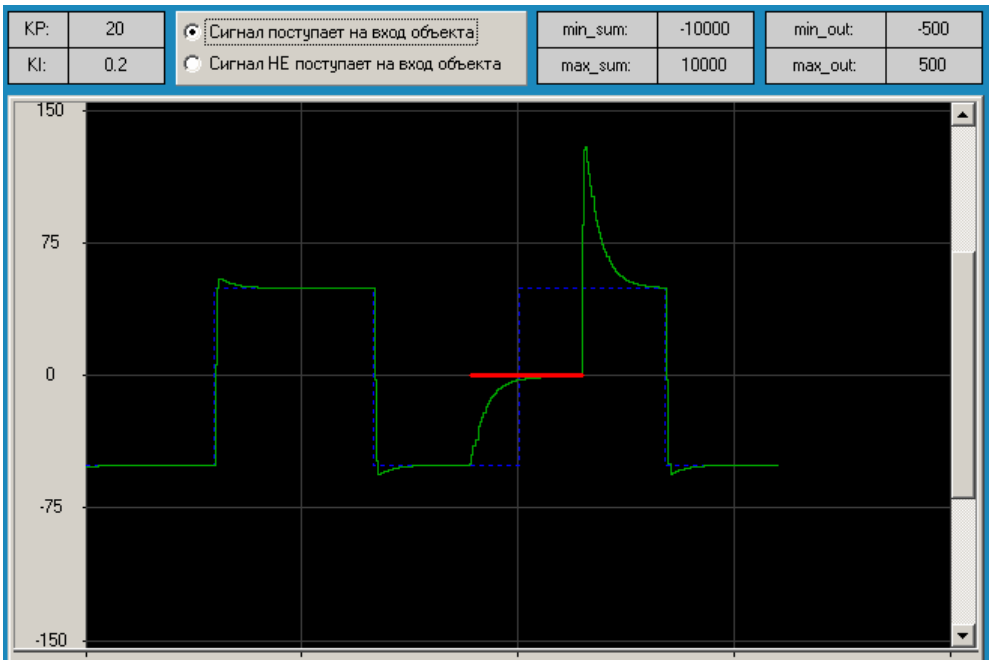


Рис. 120. Переходные процессы в модели, когда ослаблено ограничение на интегральную сумму регулятора

Для начала значительно ослабим ограничение на интегральную сумму, задав min_sum=-10000, max_sum=10000. На рис. 120 представлен график переходных процессов в этом случае. На данном графике зеленым цветом показан выход объекта управления, синим цветом - задание регулятора. Красным

цветом показан интервал времени, в течение которого управляющий сигнал не поступал на объект управления. Как можно видеть из рисунка, после восстановления связи оказалось, что выход объекта резко увеличился, а потом постепенно вернулся в норму. Это как раз связано с тем, что на время указанного интервала времени накопилась достаточно большая интегральная сумма, которая и привела к такому резкому скачку выходного значения объекта управления.

Теперь установим более жесткое ограничение на интегральную сумму, а именно min_sum=-100, max_sum=100. И промоделируем аналогичную ситуацию. На рис. 121 представлен график переходных процессов в этом случае. Оказывается, что качество переходного процесса улучшилось, в частности, теперь отсутствует скачок выходного значения объекта управления после восстановления связи.

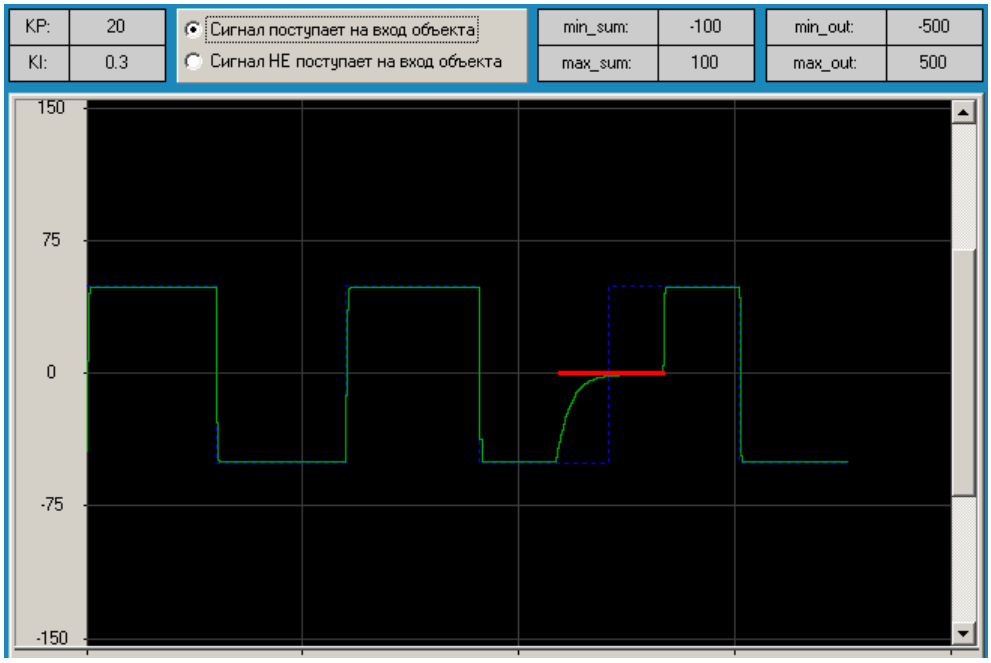


Рис. 121. Переходные процессы в модели, когда установлено более жесткое ограничение на интегральную сумму регулятора

Таким образом, использование модели объекта управления, реализованной на основе блока OBJ (см. рис. 118), позволяет оценить влияние ограничения интегральной суммы ПИ-регулятора на качество управления.

2.3.1.2.Пример на основе SCADA-пакета Genie

Даже если в составе SCADA-пакета нет специальных функциональных блоков для реализации моделей объектов управления, то их можно разработать программно. И потом эти модели можно состыковать с разработанными алгоритмами управления для проверки качества управления и других харак-

теристик, связанных с процессами управления объектом, которому соответствует данная модель.

Для примера возьмем все тот же простейший SCADA-пакет *Genie* и посмотрим, как на его основе может быть построена среда для проверки качества регулирования для различных моделей объектов управления.

В данном случае мы не будем подробно разбирать разработку ПО в рамках данного примера, а посмотрим сразу же на готовый пользовательский интерфейс и опишем особенности данной системы.

Для простоты будем рассматривать обычный ПИ-регулятор, который управляет некоторым объектом (точнее, его моделью)

ПИ-регулятор реализован на основе соотношения $y_i = \text{Кп} \cdot (u_i - x_i) + \text{Ки} \cdot S_i$, где Кп – коэффициент пропорциональной составляющей; Ки – коэффициент интегральной составляющей; y_i – выход регулятора на i-м такте; u_i – уставка (желаемое значение параметра объекта управления) на i-м такте; x_i – измеренное значение параметра объекта управления на i-м такте; S_i – значение интегральной суммы на i-м такте, определяемое на основе $S_i = S_{i-1} + (u_{i-1} - x_{i-1})$. Так как период пересчета система постоянный, то для простоты этот период учтен в коэффициенте Ки. То есть можно считать, что «истинный» коэффициент интегральной составляющей – это Ки, деленный на период пересчета системы, который для удобства устанавливается равным 100 мс.

Предполагается, что управление объектом управления выполняется в условиях случайных помех. Для устранения этих помех также добавляется алгоритм фильтрации, который реализован по следующему принципу. Если на i-м такте выполняется условие $|x_i - x_{i-1}| > \Delta x$ и $e = 0$ (где Δx – пороговое значение разности соседних значений; e – счетчик, определяющий необходимость $x_i = x_{i-1}$), то тогда значению x_i присваивается значение x_{i-1} и значение счетчика e устанавливается равным 1. Если на i-м такте выполняется условие $|x_i - x_{i-1}| > \Delta x$ и $e < 3$, то тогда также значению x_i присваивается значение x_{i-1}, а значение счетчика e увеличивается на 1. Если на i-м такте выполняется условие $|x_i - x_{i-1}| > \Delta x$ и $e = 3$, то тогда значение x_i остается без изменений. Если на i-м такте не выполняется ни одно из указанных условий, то тогда значение x_i остается без изменений, а значение счетчика e устанавливается равным нулю.

Также для устранения погрешностей измерения (которые отдельно моделируются) используется алгоритм усреднения, который работает по принципу «плавающего среднего», а именно реализуется соотношение $x_i = (x_i + x_{i-1} + \ldots + x_{i-v+1})/v$, где v – параметр усреднения (количество значений, по которым выполняется усреднение).

Пользовательский интерфейс системы представлен на рис. 122.

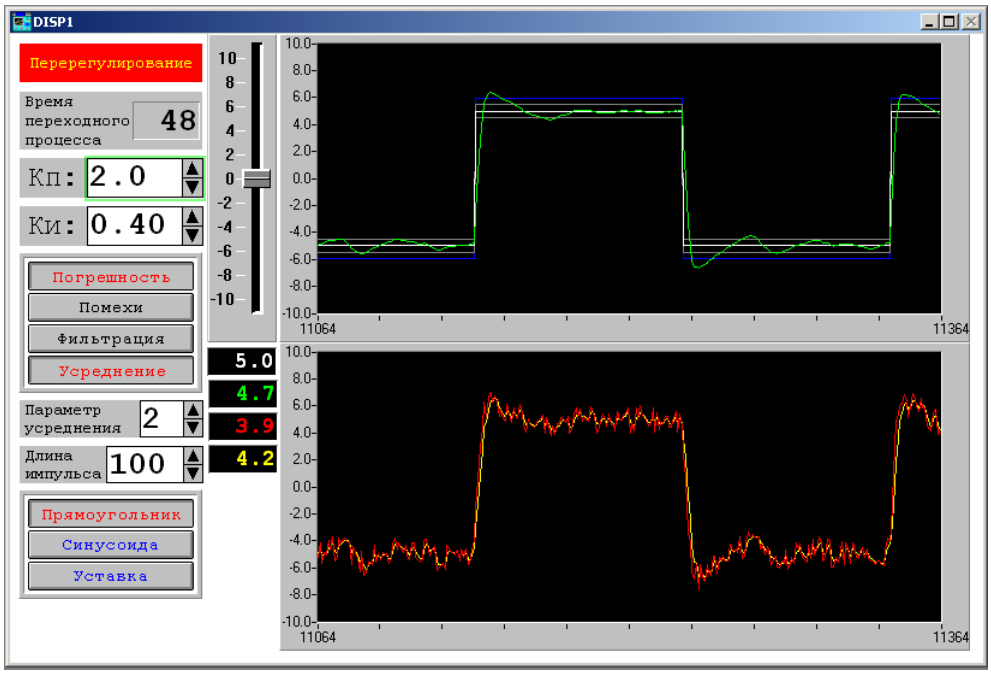


Рис. 122. Регулирование при наличии погрешностей измерения и усреднения по двум отчетам

Значение уставки u_i можно задавать вручную (кнопка «Уставка»), а также изменять автоматически (кнопки «Синусоида» и «Прямоугольник»).

Для анализа переходного процесса удобно использовать прямоугольные импульсы (кнопка «Прямоугольник») с настраиваемой длиной импульса (поле ввода «Длина импульса»). По умолчанию значение длины импульса установлено равным 100.

В дальнейшем считается, что время переходного процесса – это интервал времени (в тактах) до того момента, когда реальное значение параметра объекта (зеленая линия на верхнем графике) больше не будет отличаться более чем на 5% от значения уставки (белая линия на верхнем графике). При использовании прямоугольных импульсов границы 5%-й области отображаются серыми линиями на верхнем графике, кроме того время переходного процесса для каждого предшествующего импульса отображается на цифровом индикаторе (если переходный процесс не успел завершиться до окончания периода импульсов, то в качестве времени переходного процесса отображается значение периода импульсов).

Переходный процесс должен удовлетворять следующему условию. Перерегулирование не должно превышать 10%. При использовании прямоугольных импульсов предельное значение перерегулирование отображается линией синего цвета на верхнем графике, кроме того при перерегулировании, превышающем 10%, активизируется текстовый индикатор.

В дальнейшем требуется добиваться минимального значения времени переходного процесса при максимальном перерегулировании меньше, чем 10%.

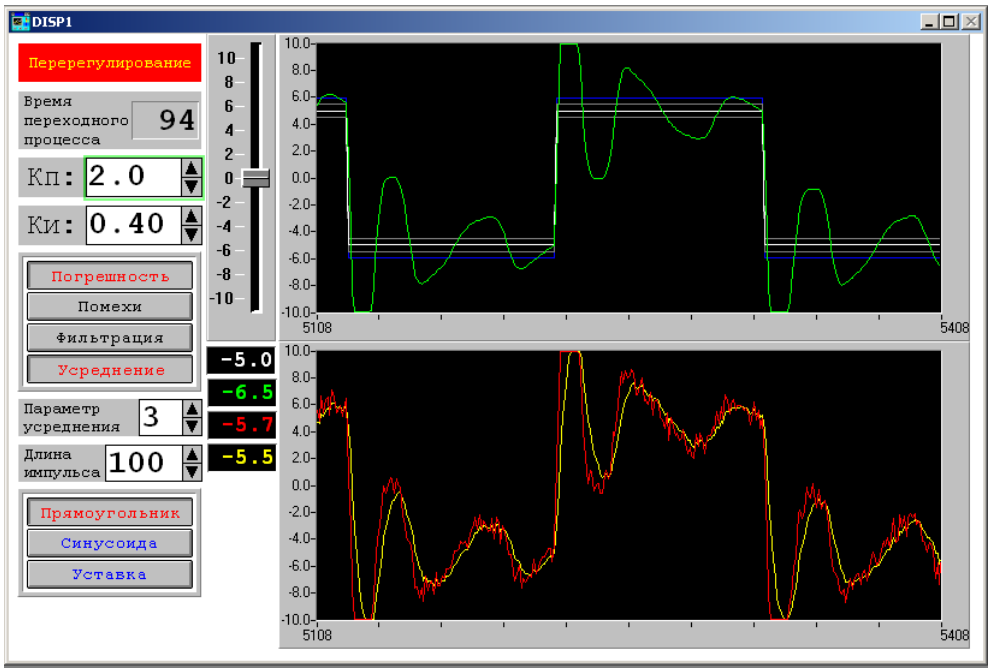


Рис. 123. Регулирование при наличии погрешностей измерения и усреднения по трем отчетам

На рис. 122 представлен случай регулирования при наличии существенных погрешностей измерения (см. красный линию на графике в нижней части), а также при усреднении по двум соседним отчетам (см. желтую линию в нижней части). При изменении параметра усреднения v переходные процессы могут существенным образом меняться. Так например, при установке $v=3$ при тех коэффициентах регулятора получаем существенно другую картину переходных процессов (см. рис. 123).

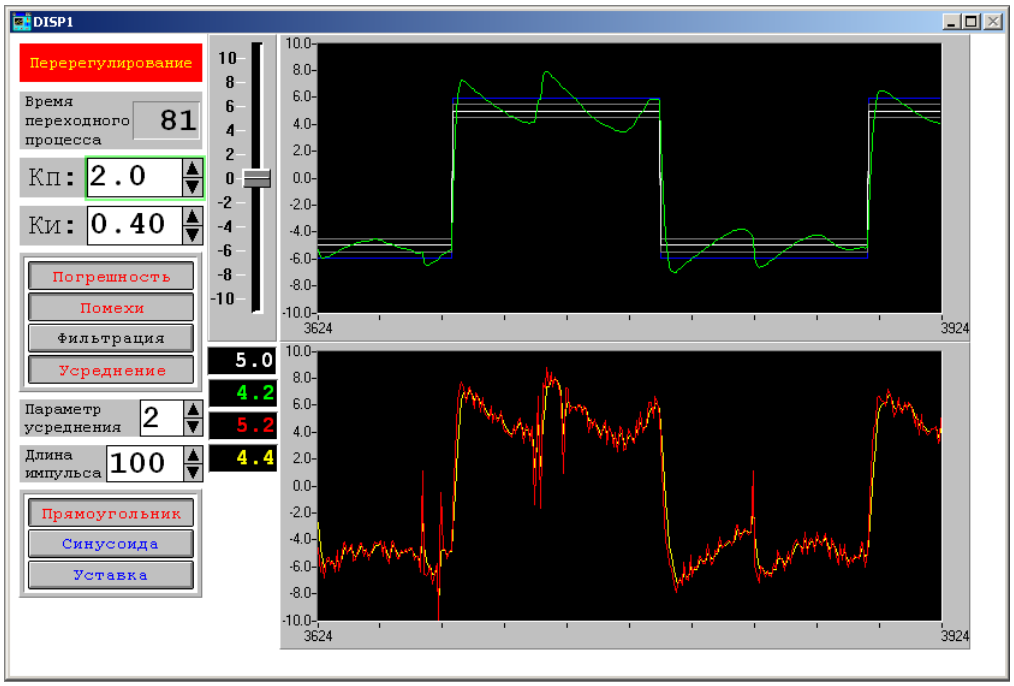


Рис. 124. Регулирование при наличии погрешностей измерения и усреднения по двум отчетам, а также при наличии помех

Также можно рассмотреть случай, когда кроме погрешностей измерения также присутствуют помехи, которые тоже влияют на измеряемую величину параметра объекта управления (рис. 124). В данном случае можно видеть, что существенные помехи частично сглаживаются за счет усреднения, а также инерционности регулятора. Однако влияние помех (резких выбросов значений на нижнем графике) сказывается на регулируемом параметре (обратите внимание на форму линии зеленого цвета).

За счет анализа подобных ситуаций можно оценивать влияние тех или иных параметров на качество регулирования. Это становится возможным на основе реализации модели объекта управления, а также модели погрешностей измерения и модели помех. Схема соединения блоков в конструкторе стратегии для данной системы представлена на рис. 125.

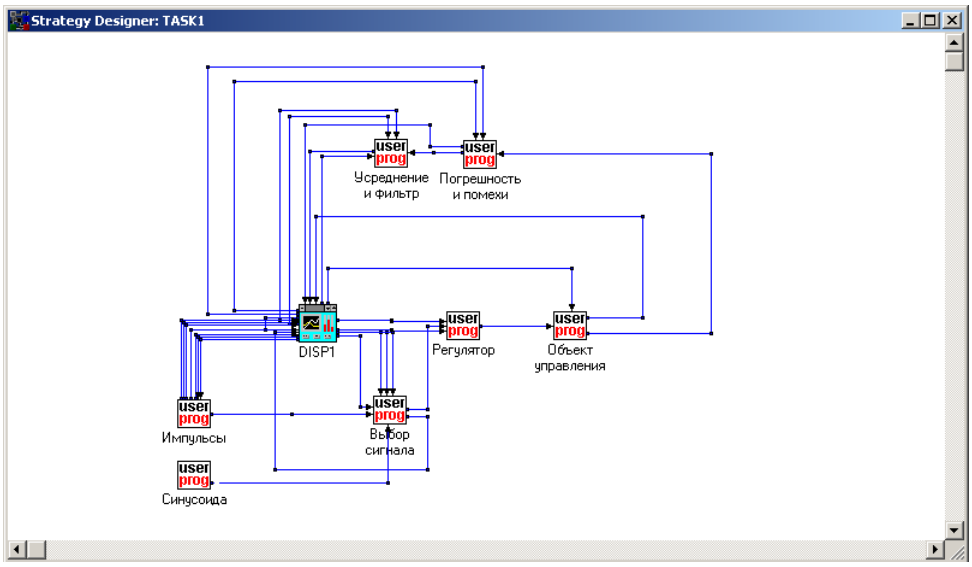


Рис. 125. Схема соединения блоков в конструкторе стратегии

Рассмотрим основные блоки, которые связаны с моделями и регулятором.

Объект управления (и соответствующий блок «user prog») представлен следующей программой

```
i=PRG2;
i=i/3;
i=i*i*i;
o=o+(i-o)/50;
if(o>20) o=20; if(o<-20) o=-20;
output(#0,o);
```

Блок «Регулятор» содержит довольно простую и очевидную программу, реализующую ПИ-регулятор:

```
v=PRG3-b;
y=NCTL1*v+NCTL2*s;
s=s+v;
output(#0,y);
```

Блок «Погрешность и помехи» реализует модели погрешности измерения и помех на основе следующей программы:

```
a=PRG1;
if(BBTN1==1)
{
 a=a+1*2*(rnd(0)-16384.0)/32767.0;
}
if(BBTN2==1)
{
 if(rnd(0)>(1-0.03)*32767.0){
  a1=(rnd(0)-16384.0)/32767.0; // -0.5 .. 0.5
  if(a1>0){
   a=a+5+5*a1;
  }else{
   a=a-5+5*a1;
  }
 }
}
output(#0,a);
```

Как можно видеть, модели для данной системы представлены довольно простыми программами, которые, тем не менее, позволяют получать довольно интересные результаты при моделировании регулирования объекта управления в условиях погрешностей и помех. Более подробно этот пример рассматривает в ходе лабораторных занятий.

Данный простой пример иллюстрирует утверждение о том, что использование и разработка моделей на основе SCADA-пакетов является удобным инструментом, который может быть очень полезен при разработке ПО верхних уровней САиУ.

Контрольные вопросы

1. Каковы основные принципы разработки пользовательского интерфейса с помощью SCADA-пакетов?
2. Какие можно заметить наиболее существенные различия между SCADA-пакетами Genie и TRACE MODE?
3. Как можно реализовать программный компонент при разработке программного обеспечения с помощью SCADA-пакета *Genie*?
4. Как можно реализовать программный компонент при разработке программного обеспечения с помощью SCADA-пакета TRACE MODE?
5. В каких случаях может понадобится разработка алгоритмов управления с помощью SCADA-пакетов?
6. Как могут быть реализованы алгоритмы выполнения сценариев с помощью SCADA-пакетов?

7. С какой целью может применяться компьютерное моделирование при разработке и отладке программного обеспечения САиУ?

8. Каким образом могут создаваться модели объектов управления с помощью SCADA-пакетов?

ЗАКЛЮЧЕНИЕ

Представленное учебное пособие соответствует учебной дисциплине «Компьютерные технологии управления в технических системах», входящей в состав магистерской программы 22040051.68 «Распределенные компьютерные информационно-управляющие системы».

В учебном пособии изложены базовые понятия, относящиеся к: системам автоматизации и управления, компьютерным технологиям управления, SCADA-пакетам. Также рассмотрены основы решения задач управления в технических системах и базовые принципы работы со SCADA-пакетами.

В качестве примеров SCADA-пакетов используются пакеты Genie и TRACE MODE. На примере этих пакетов продемонстрированы базовые принципы работы со SCADA-пакетами, а также продемонстрированы их возможности по разработке алгоритмов управления и моделей объектов управления.

В целом данное пособие является составной частью необходимых учебных материалов, используемых для преподавания дисциплины «Компьютерные технологии управления в технических системах».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Андреев Е.Б., Куцевич Н.А., Синенко О.В. SCADA-системы: взгляд изнутри. – М.: РТСофт, 2004. – 176 с.

2. Арменский Е.В. Фалк Г.Б Электромеханические устройства автоматики [Электронный ресурс]. Учебное пособие // Единое окно доступа к образовательным ресурсам. 2002. URL: http://window.edu.ru/window_catalog/redir?id=24734&file=1.pdf (дата обращения 28.04.2014).

3. Бодяжин А., Трофанчук В. Автоматизированный эколого-аналитический мониторинг источников загрязнения поверхностных вод // Современные технологии автоматизации. – 2002. – №2. – С. 68–73.

4. Брандина Е.П. Электрические машины [Электронный ресурс]: Письменные лекции. Примеры решения задач // Единое окно доступа к образовательным ресурсам. 2004. URL: http://window.edu.ru/window_catalog/redir?id=40524&file=1700.pdf (дата обращения 28.04.2014).

5. Войтович И.Д., Корсунский В.М. Интеллектуальные сенсоры [Электронный ресурс]: Учебное пособие. Глава 1 // Единое окно доступа к образовательным ресурсам. 2009. URL: http://window.edu.ru/window_catalog/redir?id=65340&file=Voytovich_978-5-9963-0124-9/Glava1_cC0124-9.pdf (дата обращения: 28.04.2014).

6. Втюрин В.А. Автоматизированные системы управления технологическими процессами. Основы АСУТП [Электронный ресурс]: учебное пособие // Единое окно доступа к образовательным ресурсам. – 2006. – URL: http://window.edu.ru/window_catalog/redir?id=66030&file=asu2.pdf (дата обращения: 28.04.2014).

7. Втюрин В.А. Автоматизированные системы управления технологическими процессами. Программно-технические комплексы [Электронный ресурс]: учебное пособие // Единое окно доступа к образовательным ресурсам. – 2006. – URL: http://window.edu.ru/window_catalog/redir?id=66031&file=asu3.pdf (дата обращения: 28.04.2014).

8. Гибридный SCADA-пакет на примере pvbrowser / А.В. Антинескул, М.В. Кавалеров, А.А. Сулейманов, Н.А. Фарафонова // Системы мониторинга и управления: сб. науч. тр. – Пермь: Изд-во Перм. гос. техн. ун-та, 2010.

9. Гордеев А.С. Основы автоматики [Электронный ресурс]: Учебное пособие для вузов // Единое окно доступа к образовательным ресурсам. 2006.

URL: http://window.edu.ru/window_catalog/redir?id=64503&file=0284.pdf (дата обращения: 28.04.2014).

10. Гринь С. Система управления линией по производству гофрокартона // Современные технологии автоматизации. – 2005. – №4. – С. 66–67.

11. Давыдов В.Г. Система супервизорного управления Vijeo Citect 7.30 SP1. Базовый курс [Электронный ресурс]: учебное пособие // Санкт-Петербургский государственный политехнический университет. URL: http://elib.spbstu.ru/dl/2/3412.pdf/download (дата обращения 28.04.2014).

12. Деменков Н.П. SCADA-системы как инструмент проектирования АСУ ТП. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 326 с.

13. Емельянов А.В., Шилин А.Н. Шаговые двигатели [Электронный ресурс]: Учебное пособие // Единое окно доступа к образовательным ресурсам. 2005. URL: http://window.edu.ru/window_catalog/redir?id=61155&file=Step_motors.pdf (дата обращения 28.04.2014).

14. Засов В.А., Савченков Н.Н. Устройства ввода-вывода аналоговых и дискретных сигналов для компьютерных систем [Электронный ресурс]: Учебное пособие // Единое окно доступа к образовательным ресурсам. 2002. URL: http://window.edu.ru/window_catalog/redir?id=29213&file=samiit230.pdf (дата обращения 28.04.2014).

15. Зюзев А.М., Нестеров К.Е., Головин И.С. SCADA-системы [Электронный ресурс]. – Екатеринбург: Изд-во УГТУ-УПИ, 2009. – 24 с.

16. Карпенко Е. IEC 61131-3: языки и средства программирования [Электронный ресурс] // Компьютерное обозрение. 2007. URL: http://ko.com.ua/iec_61131-3_yazyki_i_sredstva_programmirovaniya_34561 (дата обращения 28.04.2014).

17. Ключев А.О., Ковязина Д.Р., Петров Е.В., Платунов А.Е. Интерфейсы периферийных устройств [Электронный ресурс]: Учебное пособие // Единое окно доступа к образовательным ресурсам. 2010. URL: http://window.edu.ru/window_catalog/redir?id=72751&file=itmo500.pdf (дата обращения 28.04.2014).

18. Ключев А.О., Кустарев П.В., Ковязина Д.Р., Петров Е.В. Программное обеспечение встроенных вычислительные систем. – СПб.: СПбГУ ИТМО, 2009. – 212 с. URL: http://window.edu.ru/resource/411/63411/files/itmo368.pdf (дата обращения 28.04.2014).

19. Лиференко В. Программирование ПЛК и стандарты IEC 61131-3 [Электронный ресурс] // Компоненты и технологии. 2006 URL: http://www.kit-e.ru/assets/files/pdf/2006_04_82.pdf (дата обращения 28.04.2014).

20. Низовой А.Н., Бойчук В.С. Разработка SCADA-системы управления энергосетью предприятия // Электротехнические комплексы и системы управления. – 2006. – № 1. – С. 58–61.

21. Новиков Г.А. Основы метрологии [Электронный ресурс]: Учебное пособие // Единое окно доступа к образовательным ресурсам. 2010. URL: http://window.edu.ru/window_catalog/redir?id=71794&file=ulstu2010-60.pdf (дата обращения: 28.04.2014).

22. Олссон Г., Пиани Д. Цифровые системы автоматизации и управления. – СПб.: Невский Диалект, 2001. – 557 с.

23. Пользовательский интерфейс, SCADA-пакеты // Энциклопедия АСУ ТП. – URL: http://bookasutp.ru/Chapter9_4.aspx (дата обращения: 28.04.2014).

24. Пономарев О.П. Наладка и эксплуатация средств автоматизации. SCADA-системы. Промышленные шины и интерфейсы. Общие сведения о программируемых логических контроллерах и одноплатных компьютерах [Электронный ресурс]: Учебное пособие // Единое окно доступа к образовательным ресурсам. 2006. URL: http://window.edu.ru/window_catalog/redir?id=37097&file=kvshu04.pdf (дата обращения: 28.04.2014).

25. Практическое введение в компьютерное зрение [Электронный ресурс] // 2013, URL:http://my-it-notes.com/2013/05/практическое-введение-в-компьютерно/ (дата обращения 28.04.2014).

26. Пьявченко Т.А., Финаев В.И. Автоматизированные информационно-управляющие системы [Электронный ресурс] // Единое окно доступа к образовательным ресурсам. – 2007. – URL: http://window.edu.ru/resource/206/61206/files/pos_ASU_TP_2.pdf (дата обращения: 28.04.2014).

27. Сергеев С.Ф. Введение в инженерную психологию и эргономику иммерсивных сред [Электронный ресурс]: учебное пособие // Единое окно доступа к образовательным ресурсам. – 2011. – URL: http://window.edu.ru/resource/819/72819/files/itmo518.pdf (дата обращения: 28.04.2014)

28. Сергеев С.Ф., Падерно П.И., Назаренко Н.А. Введение в проектирование интеллектуальных интерфейсов [Электронный ресурс]: учебное пособие // Единое окно доступа к образовательным ресурсам. – 2011. – URL: http://window.edu.ru/window_catalog/redir?id=72820&file=itmo519.pdf (дата обращения: 28.04.2014).

29. Системы диспетчерского управления и сбора данных (SCADA-системы) [Электронный ресурс] // Сайт журнала «Мир компьютерной автоматизации». – URL: http://www.mka.ru/?p=41524 (дата обращения: 28.04.2014).

30. Туманов М.П. Технические средства автоматизации и управления: цифровые средства обработки информации и программное обеспечение

[Электронный ресурс]: Учебное пособие // Единое окно доступа к образовательным ресурсам. 2005. URL: http://window.edu.ru/window_catalog/redir?id=24739&file=6.pdf (дата обращения 28.04.2014).

31. Яшнов Л., Кухаренко В., Ткачук М., Чуйков С. Автоматизированная система управления стендовыми огневыми испытаниями ракетных двигателей малой тяги // Современные технологии автоматизации. – 2005. – №3. – С. 38--44.

32. Bailey D., Wright E. Practical SCADA for industry. – Oxford (GB): Elsevier, 2003. – 304 p.

33. Boyer S.A. SCADA: supervisory control and data acquisition. – Research Triangle Park (USA): ISA Publishing, 2004. – 219 p.

34. Fernandez E.B., Wu J., Larrondo-Petrie M.M., Shao Y. On building secure SCADA systems using security patterns / // Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research, 2009.

35. Germanus D., Khelil A., Suri N. Increasing the Resilience of Critical SCADA Systems Using Peer-to-Peer Overlays // Architecting Critical Systems. – 2010. – Vol. 6150. – P. 161–178.

36. Giani A., Karsai G., Roosta T., Shah A., Sinopoli B., Wiley J. A testbed for secure and robust SCADA systems // ACM SIGBED Review. – 2008. – Vol. 5. – P. 1–4.

37. Shaw W.T. Cybersecurity for SCADA systems. – Tulsa (USA): PennWell, 2006. – 299 p.

38. Williams R. Real-Time Systems Development. – Elsevier. 2006. – 455 p.

СПРАВОЧНЫЕ ДАННЫЕ ПО РАБОТЕ С ПАКЕТОМ GENIE v2.0

Общие положения

Активизация блока осуществляется двойным нажатием левой кнопки или одним нажатием правой кнопки мыши.

Соединение блоков в редакторе стратегии производится в режиме соединения. Этот режим включается при нажатии кнопки  библиотеки функциональных блоков (курсор мыши изменяется). Соединение осуществляется от выхода к входу. Сначала выбирается выход (с помощью мыши выбирается блок). Аналогично выбирается вход. Для выхода из режима соединения нажать правую кнопку мыши.

Для связи формы отображения в редакторе экрана с блоком в редакторе стратегии необходимо открыть свойства формы отображения и в поле *Input from* выбрать имя блока.

Редактор стратегии. Типы блоков

 **Display.** Установка этого блока приводит к формированию окна отображения. Для формирования нескольких окон необходимо установить соответствующее количество этих блоков в редакторе стратегии. При активизации этого блока происходит переход в редактор экрана для данного окна. Для отображения значений, формируемых другими блоками, в данном окне необходимо соединить выходы этих блоков с входами данного блока *Display*. Тогда в редакторе экрана для данного окна будут доступны значения присоединенных блоков.

User Prog. Имеет 8 входов и 8 выходов. Используется для составления небольших программ на Си-подобном языке.

Особенности написания программ

Могут использоваться только глобальные переменные, состоящие из одной буквы (от *a* до *z*) или из буквы и цифры от 1 до 9 (то есть от *a*1 – *z*1 до *a*9 – *z*9). Эти переменные являются глобальными, то есть они доступны из любого блока USER PROG. На первом цикле (после запуска *Runtime*) все эти переменные равны 0. Программа каждого блока USER PROG выполняется на каждом цикле *Runtime*.

Для преобразования к целому числу часто требуется использовать функцию «int ()».

Основные операторы, используемые при написании программ

Оператор	Описание оператора
a=b;	Присвоение

if (логическое условие) {выражения}
else {выражения}
Пример:
if (a>b) {a=b; c=a; }
else {a=b; c=b; }
	Условие
	(перед закрывающей фигурной скобкой следует ставить точку с запятой)

= =! = > < >= <= && ||! Пример: if (((a>b) || (a==c)) && (b! =c)) {a=b; b=0;}	Логическое равенство, неравенство, больше, меньше, больше или равно, меньше или равно, И, ИЛИ, НЕТ
+ – * / %	Сложение, вычитание, умножение, деление, остаток от деления
& | ~ ^	Битовое И, ИЛИ, НЕТ, XOR
output (#n,v);	Вывести на *n*-й выход значение *v*
display (n);	Активизация *n*-го окна

Полный список операторов расположен в поле *Operators* (в свойствах блока USER PROG).

Конструктор стратегии. Основные блоки

Single Operator Calculation Block (SOC). Используется для выполнения одного оператора. Имеет два входа (два операнда) и один выход. При активизации данного блока в поле *Result Data Type* определяется тип результата: *Integer* (целый) или *Floating Point* (*Real*) (вещественный с плавающей точкой).

FAI, AI, AO, DI, DO. Блоки аналогового ввода, вывода (AI, AO), ускоренного аналогового ввода (FAI), дискретного ввода, вывода (DI, DO) используются для связи с аппаратным обеспечением (задается в поле *Device*).

Timer. Программируемый таймер. Устанавливаются период и величина цикла таймера, а также сигнал сброса.

Time Stamp Block. Используется для вывода текущего времени в различных форматах.

Triggering Block. Пропускает данные в соединении между блоками по заданному условию.

Event Counter Block. Счетчик изменений из 0 в 1.

Hardware Event Counter/Frequency Measurement/Pulse Output Block. Используется для связи с аппаратным обеспечением для измерения частоты.

PID Control Block. Реализует ПИД-регулятор.

 **On/Off Control Block.** Включение/выключение по превышению/занижению уровня.

 **Ramp Block.** Формирует пилообразный сигнал.

 **Running/Moving Average Block.** Усреднение сигнала.

 **Data File Block, Log File Block.** Используются для записи, чтения файлов данных.

 **Beep Block.** Формирует звуковой сигнал на внутреннем динамике.

 **RS-232 Block.** Обеспечивает взаимодействие по RS-232.

 **Hardware Alarm Block.** Обеспечивает получение аварийной информации от специального аппаратного обеспечения.

 **DDE Blocks (Client and Server).** Обеспечивают DDE-взаимодействие.

 **Network Input, Network Input.** Сетевое взаимодействие между рабочими станциями.

 **Alarm Log Block.** Формирование отчетов событий.

Конструктор экрана. Основные блоки

 **Slider Control Display Item.** Линейный регулятор. Start Tics, End Tics определяют разметку шкалы.

 **Knob Control Display Item.** Поворотный регулятор. Start Tics, End Tics определяют разметку шкалы.

 **Numeric Control Display Item.** Цифровой регулятор. High Limit, Low Limit определяют границы изменения.

 **Numeric/String Display Item.** Цифровой индикатор.

 **Trend Graph Display Item.** Форма отображения, определяющая график.

 **Indicator Display Item.** Двухцветный индикатор. *Style* определяет форму (прямоугольник или эллипс).

 **Conditional Text Display Item.** Вывод строки из сформированного пользователем списка.

Учебное издание

Кавалеров Максим Владимирович,

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Учебное пособие