Содержание

Введение

1. Понятие человеко-машинного интерфейса

1.1 Текстовый (текст ориентированный) интерфейс

1.2 Смешанный (псевдографический) интерфейс

1.3 Графический интерфейс

2. Особенности применения человеко-машинного интерфейса в промышленности

2.1 SCADA-системы: общие понятия и структура

3. Обзор программных средств для разработки человеко-машинного интерфейса

4. Структура SCADA - системы GENESIS 32

4.1 GraphWorX32

4.2 TrendWorX32

4.3 AlarmWorX32

4.4 ScriptWorX32

4.5 DataWorX32

4.6 WebHMI

4.7 AlarmWorX32 Multimedia

4.8 Менеджер экрана Sreen Manager

4.9 DataSpy

4.10 Dr. DCOM

4.11 Сервер администрирования Security Server

4.12 GEN-OPC сервер

4.13 ActiveX ToolWorX

4.14 OPC ToolWorX

5. Среда разработки мнемосхем Graphworx 32

6. Создание мнемосхемы модели дискретно-непрерывного технологического процесса

6.1 Создание статических объектов мнемосхемы

6.2 Создание анимационных объектов

6.3 Создание имитации наполнения/опустошения резервуара

6.4 Создание элемента отображения уровня в резервуаре.

6.5 Настройка анимации для изображения миксеров

Заключение

Список литературы

## Введение

В настоящее время всё больше сфер человеческой деятельности подвергаются автоматизации. Человек в силу своих физических способностей не может управлять технологическим процессом, например ядерный реактор. Так же автоматизируются те технологические процессы, в которых ошибка является катастрофой и может нанести огромный вред окружающей среде.

С ростом уровня автоматизации сфер производства выросло и число программных продуктов, управляющих технологическими процессами.

В данной дипломной работе пойдёт речь о разработке человеко-машинного интерфейса для технологического процесса в SCADA-системе Graphworx32.

## 1. Понятие человеко-машинного интерфейса

Человеко-машинный интерфейс довольно широкое понятие. Чтобы понять это понятие надо узнать, что представляет собой понятие интерфейс. Слово интерфейс (от англ. - поверхность раздела, перегородка) определяет место или способ соединения, соприкосновения, связи. Это слово стало популярным в эпоху компьютеризации, но его значение относится к любому сопряжению взаимодействующих систем. Например, вожжи - это главный элемент интерфейса между лошадью и кучером; руль, педали газа и тормоза, ручка КПП - интерфейс водителя для управления автомобилем.

Интерфейсы являются основой взаимодействия всех современных информационных систем. Если интерфейс какого-либо объекта (персонального компьютера, программы, функции) не изменяется (стабилен, стандартизирован), это даёт возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами. То есть, научившись работать с одной программой, например, под Windows, пользователь с лёгкостью освоит и другие, потому что они имеют одинаковый интерфейс.

В вычислительной системе взаимодействие может осуществляться на пользовательском, программном и аппаратном уровнях. В соответствии с этой классификацией можно выделить:

интерфейс командной строки: инструкции компьютеру даются путём ввода с клавиатуры текстовых строк (команд);

графический интерфейс пользователя: программные функции представляются графическими элементами экрана;

диалоговый интерфейс;

естественно - языковой интерфейс: пользователь "разговаривает" с программой на родном ему языке.

Из этого вытекает понятие пользовательского интерфейса. Пользовательский интерфейс - это совокупность средств, при помощи которых пользователь общается с различными устройствами. Разобрав понятие интерфейс можно перейти к человеко-машинному интерфейсу.

Так что же представляет собой человеко-машинный интерфейс. Человеко-машинный интерфейс - это широкое понятие, охватывающее инженерные решения, обеспечивающие взаимодействие оператора с управляемыми им машинами. Создание систем человеко-машинного интерфейса тесно связано с эргономикой (Эргономика - научная дисциплина, комплексно изучающая производственную деятельность человека и ставящая целью её оптимизацию), но не тождественно ей. Проектирование ЧМИ включает в себя создание рабочего места: кресла, стола, или пульта управления, размещение приборов и органов управления, освещение рабочего места, а, возможно, и микроклимат. Далее рассматриваются действия оператора с органами управления, их доступность и необходимые усилия, согласованность (непротиворечивость) управляющих воздействий и "защита от дурака", расположение дисплеев и размеры надписей на них.

Сложность создания человеко-машинного интерфейса состоит в том, что данные, которые нужно “донести” до пользователя, нужно “донести” так, чтобы пользователю было это “донесение” удобным и понятным.

Человеко-машинный интерфейс условно можно разделить на 3 подгруппы: текстовый (текст ориентированный) интерфейс; смешанный (псевдографический) интерфейс; графический интерфейс.

## 1.1 Текстовый (текст ориентированный) интерфейс

Как пример к такому виду человеко-машинного интерфейса можно привести интерфейс командной строки DOS или Shell - интерпретатор Linux. Пользователь взаимодействует с вычислительной системой с помощью клавиатуры, набирая специальные команды, для задания различных опций служат параметры. Система как ответ на действия пользователя тоже выдаёт или сообщения, или результат выполнения введенной команды, опять же в текстовом виде. Курсор может иметь вид мигающего прямоугольника или чёрточки, обозначающей место ввода. В таком режиме можно взаимодействовать лишь с одной программой, хотя потенциально могут выполняться несколько различных программ. Управлять взаимодействием этих программ можно лишь только опять с командной строки, причём проверить результат можно только по окончанию работы.

Рис. 1. Текстовый интерфейс режима MS DOS.

Для эффективного использования их пользователю необходимо знать синтаксис всех команд, плюс знать, какие нужно использовать ключи или опции для каждой из них. Это является главным недостатком этой подгруппы человеко-машинных интерфейсов. Кроме того, текстовая природа выводимых данных делает трудной, а под час и совершенно невозможной работу с определённым классом приложений, в первую очередь графических, или тех, где используются разнородные данные, например Web-браузеры.

В основном текстовый интерфейс применяется в таких сферах, как операционные системы, чаты и компьютерные игры. В операционных системах такой вид интерфейсов применяется как командная строка.

Единственной нишей, где подобный тип человеко-машинных интерфейсов сохранился и имеет более или менее прочные позиции, является удалённый доступ для администрирования или настройки сервера, когда требуется лишь shell-доступ, и в наличии есть только канал с ограниченной пропускной способностью.

Но с другой стороны преимуществом данного вида человеко-машинного интерфейса в том, что этот вид интерфейса требует определённой культуры поведения и чёткости мысли пользователя. Например, в Windows после нажатия Ctr-Alt-Del ошибка в позиционировании курсора на 20-30 точек влечет за собой или завершение выполнения одной программы, или завершения работы всей системы. В режиме командной строки такой ошибки не получится. Для выхода там как минимум нужно набрать совсем другую последовательность, "logout", "exit" или еще что-то столь же приметное, что с другой командой спутать не получится. Или, например, чтобы отформатировать дискету, нужно “пройти" через кучу окон и задать много опций, вместо того, чтобы набрать всем понятную команду format a: \. Получается, что текстовый вид интерфейсов останется как интерфейс там, где работают именно профессионалы, и где не требуется сомнительных удобств и красот в ущерб функциональности и гибкости.

## 1.2 Смешанный (псевдографический) интерфейс

В первую очередь следует различать понятия “оконный” и “графический” интерфейсы. “Оконный” базируется на принципе разделения реального окна монитора (или виртуального десктопа намного большего размера, чем физический дисплей) на прямоугольные области, внутри каждой из которых определенная программа направляет свой вывод и откуда получает команды. Никто и нечто не ставит никаких ограничений на природу этих окон - это могут быть как независимые текстовые терминалы, так и окна, куда выводиться графика (как результат работы, так и элементы интерфейса). А термин "графический" означает, что все выводиться в графическом режиме, так что может быть как оконный графический интерфейс, когда каждое окно отображает графический интерфейс, так и полноэкранный режим, когда выполняется только одна программа, которая осуществляет вывод в графическом режиме. То есть, оконный не обязательно графический, а графический не всегда оконный. Псевдографический интерфейс обособлен присутствием графических интерфейсных элементов, например, кнопки, индикаторы процесса выполнения, меню. Как пример можно привести известную программу FAR

Рис. 2. Псевдографический интерфейс оболочки FAR.

Для пользования этой системой уже не нужно наизусть помнить многочисленные команды и опции, сообщения имеют более удобный и привычный вид. Но интерфейс все равно остается текст ориентированным, а значит трудности с отображением различных данных остаются - о типе файла можно узнать только по расширению, а не как в Windows - еще и по иконке.

Псевдографический человеко-машинный интерфейс является промежутком между чисто текстовым интерфейсом и графическим. Такой вид человеко-машинного интерфейса в большинстве случаев обладает всеми преимуществами текстового интерфейса (использование мощных языков, расширяемость), и устраняет некоторые недостатки (позволяет легче управлять системой, нагляднее представить файловую систему, например). Но большинство недостатков практически те же - бедность вариантов представления данных, невыразительность интерфейса, нарастающая сложность при попытке перенести команду с множеством опций в режим, когда в окне нужно просто выбрать нужные пункты - на рис.2 видно, что в окне команды "Копировать" есть пункт "Дерево", выбор которого приведет к открытию еще одного окна, с деревом каталогов - так что уже есть где запутаться, тем более что переключаться произвольным образом между окнами нельзя.

## 1.3 Графический интерфейс

Графический человеко-машинный интерфейс - это система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т.п.). При этом, в отличие от текстового интерфейса, пользователь имеет произвольный доступ (с помощью клавиатуры или устройства координатного ввода типа "мышь") ко всем видимым экранным объектам.

Впервые Графический интерфейс был предложен учёными из исследовательской лаборатории Xerox PARC в 1970-х. В 1973 году в лаборатории Xerox PARC собирают молодых людей, недовольных политикой США (война во Вьетнаме) и дают свободу исследований. В результате на свет появляется концепция графического интерфейса WIMP (Windows, Icons, Menus, Point-n-Click). В рамках этой концепции создаётся компьютер Alto.

Графический интерфейс со своей многозадачностью был использован в 1985 году в операционной системе AmigaOS.

К этому виду интерфейсов относятся такие системы как, Mac OS, Solaris, GNU/Linux, Microsoft Windows, NeXTSTEP,

В графическом человеко-машинном интерфейсе все элементы пользовательского интерфейса, как и сами данные в окнах, отображаются в графическом режиме, с помощью 256, 16-битной или 32-битной глубины цветового буфера. Это позволяет сформировать привлекательные с точки зрения пользователя окна, кнопки, пиктограммы, ползунки, индикаторы. В таком режиме "объемность" интерфейсных элементов достигается с помощью искусственных приемов - например, за несколько пикселей до края рамки окна дают полоску белого толщиной в один пиксель - появляется иллюзия того, что рамка как бы выпукла.

Общим для всех систем с графическим интерфейсом есть понятие окна. Окно - прямоугольная область экрана, куда программа выводит свои данные и откуда получает команды. Есть два различных подхода.

Первый - это десктоп. Десктоп - это окно, которое имеет максимальные размеры и занимает весь экран. Как пример можно рассмотреть 2 скриншота системы Microsoft Windows.

Рис.3. Оконная система ОС Microsoft Windows 98. Окно занимает максимальную область десктопа.

На рисунке 3 видно, что внизу, на панели задач, в данный момент открыто множество других окон.

Но сейчас они не активны, и что особенно, навигация между ними очень затруднена - панель задач имеет маленькие размеры, а окон много поэтому их заголовки отображаются сокращенно, и, как видно, по ним абсолютно нельзя сказать, что открыто в том или ином окне. Вот и главный недостаток подобной системы - при превышении некого лимита открытых окон практически невозможно при свернутом состоянии определить, что же там отображается. Значит, для поиска нужного окна сначала требуется по очереди открывать все окна и просматривать - как минимум, эта процедура занимает много времени. Следующий вариант - держать открытыми сразу много окон (в смысле - отображать все окна, просто с разной степенью "раскрытия" и перекрытия между соседними).

Рис.4. ОС Windows 98. Открыто и отображается сразу несколько окон.

Как видно, активным в каждый момент может быть только одно окно, и разница между активным и неактивным очень маленькая - всего лишь в цвете заголовка окна.

Для достижения отображения всех окон, пришлось так "урезать" их размеры, что работа с такими окнами очень утруднена, навигация между ними осталась практически такой же сложной.

Вообще графический интерфейс представляет собой рабочий стол - экран - поверхность стола, бумаги и документы - окна. Значит проблемы “традиционных столов” остаются, ведь обычно рабочий стол всегда завален кучей разных бумаг, и отыскать в этом всем нужный документ недельной давности ох как нелегко. То же твориться и на столе виртуальном (Рис 4). Пока на столе (экране) один документ (рис.3) то можно спокойно работать, но когда их количество переваливает за десяток... Перекрытие как метод отображения сразу нескольких документов не только не решает проблему, а наоборот, еще и усугубляет ее, так как нужно следить за тем, какой из документов/окон сейчас активен, а поиск в “мешанине" окон тоже не очень приятен. Вот это и есть еще один существенный недостаток оконных систем с перекрытием окон - трудность с расположением и навигацией между отдельными окнами.

Второй подход - система формирует виртуальный десктоп гораздо большего размера, чем дисплей. Окна могут размещаться на всей площади этого десктопа, а на экране отображается лишь те окна или их части, которые попадают в область отображения реального экрана. Такой подход обязывает постоянно и независимо от других окон держать в видимой области окно специальной программы - оконного менеджера, который показывает в масштабе расположение всех окон на виртуальном десктопе и ту область, которая отображается в данный момент.

Следует упомянуть ещё об одном скрытом минусе графических оконных систем. При их использовании возникает так званый эффект "когнитивной перегрузки" - когда на экран выводиться столько различной информации, за которой нужно следить, да еще и в разных частях дисплея, что пользователь просто теряется, "глаза разбегаются". Нужно следить за курсором, держать во внимании панель инструментов, где и какая кнопка нажата, следить за индикатором раскладки клавиатуры, за самой клавиатурой (Caps Lock), одновременно нужно знать, какое окно активно, и что делает система в данный момент. Это большой поток информации, за которым сложно следить. Кроме того, чтобы выполнить любую элементарную операцию, нужно найти курсором (значит, перевести внимание на него, но другие элементы, описанные выше тоже нужно держать в поле зрения) пункт меню или нужную кнопку, попасть на неё, удостовериться, что курсор находиться именно над той кнопкой, которую нужно нажать, а если это ползунок перемещения, то не останавливая нажатия, одновременно следить за тем, как исполняется команда. Часто бывает, что пользователь попадает по ненужной кнопке или пункту меню, вследствие чего приложение совершает не ту команду или вообще закрывается. Стандартизация элементов пользовательского интерфейса Windows по идее ее создателей должна было навести лад среди приложений и избавить пользователей от изучения нового интерфейса при переходе на новую программу. В какой то мере Windows сделал своё дело, и по крайней мере закрыть, минимизировать, раскрыть окно сможет каждый пользователь и с любой программой. Но все другие кнопки имеют стандартный цвет и размер. Примерно одинаковое размещение и функциональное предназначение приводит к тому, что пользователи очень часто даже не смотрят на кнопку, прежде чем ее нажать, срабатывает подсознательный рефлекс. И получается так, что разработчики некоторых программ определяют совсем другую функциональность для кнопок, с которыми пользователь знаком и не думает о том, что у неё совсем другая функциональность. Получается, что стандартизация элементов пользовательского интерфейса имеет и отрицательные стороны - пользователь зачастую даже не взглянет на надпись, которая находиться на той или иной кнопке. Получается что программа, в которой разработчики определи другие функциональность для кнопок, знакомых пользователю, переходит с категории утилит в разряд вредоносных программ. Если пользователь запустит описываемую программу в своей операционной системе и нажмёт не ту кнопку, то последствия могут быть не самыми лучшими, например, пользователь нажмёт кнопку с функцией форматирования жёсткого диска. Вот основные минусы графических интерфейсов:

Затруднена работа с несколькими окнами;

когнитивная перегрузка - на экране находится не только много не нужных элементов, но и главным образом тех, которые нужны, и на которых нужно в идеале одновременно концентрировать внимание;

применение сложных команд, например, при форматировании текста, требует знания что и где нажать, всего пути к нужному пункту меню, да еще и учета того, что сейчас нажато, или было нажато незадолго до этого;

большие трудности при смене общего стиля оформления системы;

двухмерность интерфейса накладывает определенные ограничения на размещение как документов, так и пиктограмм.

Но среди минусов есть большой плюс - это удобство. Ведь в графических интерфейсах можно реализовать то, что нельзя в остальных.

## 2. Особенности применения человеко-машинного интерфейса в промышленности

Раньше, когда ещё не было автоматизированных систем управления технологическим процессом, человеко-машинный интерфейс был довольно прост. Например, чтобы измерить уровень воды в резервуаре, человек который выполнял эту работу, брал линейку и измерял уровень воды. В данном случае интерфейсом служит сама линейка и шкала, расположенная на нём. Но с приходом прогресса в вычислительной технике, да и во многих сферах деятельности человека, человек стал стремиться к автоматизации. Это желание понятно, ведь человек уже не мог физически управлять такими процессами, как ядерная реакция, так как человек просто не успеет из-за своей реакции отреагировать на процесс. Если же выше описываемая ошибка с нажатием не той кнопки, могла привести лишь к форматированию жёсткого диска, то последствия такой ошибки в автоматизации промышленной сферы деятельности человека могут быть самыми печальными. Получается, что стабильная работа технологического процесса во многом зависит от разработчика человеко-машинного интерфейса к этому процессу, а не от диспетчера, который реагирует.

Для большего понимания особенностей применения человеко-машинного интерфейса в промышленности следует понять, что такое автоматизация. Автоматизация - одно из направлений научно-технического прогресса, применение саморегулирующих технических средств, экономико-математических методов и систем управления, освобождающих человека от участия в процессах получения, преобразования, передачи и использования энергии, материалов или информации, существенно уменьшающих степень этого участия или трудоемкость выполняемых операций. Требует дополнительного применения датчиков (сенсоров), устройств ввода, управляющих устройств (контроллеров), исполнительных устройств, устройств вывода, использующих электронную технику и методы вычислений, иногда копирующие нервные и мыслительные функции человека. Наряду с термином автоматический, используется понятие автоматизированный, подчеркивающий относительно большую степень участия человека в процессе. Целью автоматизации является повышение производительности труда, улучшение качества продукции, оптимизация управления, устранение человека от производств, опасных для здоровья. Автоматизация, за исключением простейших случаев, требует комплексного, системного подхода к решению задачи, поэтому решения стоящих перед автоматизацией задач обычно называются системами, например:

система автоматического управления (САУ);

система автоматизации проектных работ (САПР);

автоматизированная система управления технологическим процессом (АСУ ТП).

автоматизируются такие сферы деятельности, как:

производственные процессы;

проектирование;

организация, планирование и управление;

научные исследования.

Раз при автоматизации появляются новые сложные устройства, значит нужно создавать такой интерфейс для этих устройств, который бы был простым, понятным и интуитивным для диспетчера. Получается, что главное в системе управления - это организация взаимодействия между человеком и программно-аппаратным комплексом. Обеспечение такого взаимодействия и есть задача человеко-машинного интерфейса (HMI, human machine interface). В современных автоматизированных системах управления технологическим процессом существует, как минимум, 2 подхода по реализации человеко-машинного интерфейса:

на базе специализированных рабочих станций оператора, устанавливаемых в центральной диспетчерской;

на базе панелей локального управления, устанавливаемых непосредственно в цеху по близости к контролируемым технологическим объектам.

Иногда эти два варианта комбинируются, чтобы достичь наибольшей гибкости управления. Но дальше речь пойдёт о первом варианте организации операторского уровня.

Аппаратно рабочая станция оператора (OS, operator station) представляет собой ни что иное как персональный компьютер. Как правило, станция снабжается несколькими широкоэкранными мониторами, функциональной клавиатурой и необходимыми сетевыми адаптерами для подключения к сетям верхнего уровня (например, на базе Industrial Ethernet). Станция оператора несколько отличается от привычных для нас офисных компьютеров, прежде всего, своим исполнением и эксплуатационными характеристиками (а также ценой 4000 - 10000 долларов).

Рис.5. Промышленная рабочая станция оператора системы SIMATIC PCS7 производства Siemens

На станции оператора устанавливается программный пакет визуализации технологического процесса. Большинство пакетов визуализации работают под управлением операционных систем семейства Windows (Windows NT 4.0, Windows 2000/XP, Windows 2003 Server). Программное обеспечение визуализации должно выполнять следующие задачи:

отображение технологической информации в удобной для человека графической форме (как правило, в виде интерактивных мнемосхем) - Process Visualization;

отображение аварийных сигнализаций технологического процесса - Alarm Visualization;

архивирование технологических данных (сбор истории процесса) - Historical Archiving;

предоставление оператору возможности манипулировать (управлять) объектами управления - Operator Control;

контроль доступа и протоколирование действий оператора - Access Control and Operator’s Actions Archiving;

автоматизированное составление отчетов за произвольный интервал времени (посменные отчеты, еженедельные, ежемесячные и т.д.) - Automated Reporting.

Человеко-машинный интерфейс в промышленности реализуется с помощью мнемосхем. Мнемосхема - совокупность сигнальных устройств и сигнальных изображений оборудования и внутренних связей контролируемого объекта, размещаемых на диспетчерских пультах, операторских панелях или выполненных на персональном компьютере. Информация, которая выводится на мнемосхему, может быть представлена в виде аналогового, дискретного и релейного сигнала, а также графически. На мнемосхемах отражается основное оборудование, сигналы, состояние регулирующих органов. Вспомогательный и справочный материал должен быть расположен в дополнительных формах отображения, с возможностями максимально быстрого извлечения этих вспомогательных форм на экран.

Выше описанный программный пакет визуализации технологического процесса называется SCADA - системой. SCADA (сокр. от англ. Supervisory Control And Data Acquisition) - диспетчерское управление и сбор данных. Термин SCADA-система используют для обозначения программно-аппаратного комплекса сбора данных. SCADA-системы являются основным и в настоящее время остаются наиболее перспективным методом автоматизированного управления сложными динамическими системами (процессами) в жизненно важных и критичных с точки зрения безопасности и надежности областях. Именно на принципах диспетчерского управления строятся крупные автоматизированные системы в промышленности и энергетике, на транспорте, в космической и военной областях, в различных государственных структурах.

За последние 10-15 лет за рубежом резко возрос интерес к проблемам построения высокоэффективных и высоконадежных систем диспетчерского управления и сбора данных. С одной стороны, это связано со значительным прогрессом в области вычислительной техники, программного обеспечения и телекоммуникаций, что увеличивает возможности и расширяет сферу применения автоматизированных систем. С другой стороны, развитие информационных технологий, повышение степени автоматизации и перераспределение функций между человеком и аппаратурой обострило проблему взаимодействия человека-оператора с системой управления. Расследование и анализ большинства аварий и происшествий в авиации, наземном и водном транспорте, промышленности и энергетике, часть из которых привела к катастрофическим последствиям, показали, что, если в 60-х годах ошибка человека являлась первоначальной причиной лишь 20% инцидентов (80%, соответственно, за технологическими неисправностями и отказами), то в 90-х годах доля человеческого фактора возросла до 80%, причем, в связи с постоянным совершенствованием технологий и повышением надежности электронного оборудования и машин, доля эта может еще возрасти (Рис.6).

Рис.6. Тенденции причин аварий в сложных автоматизированных системах.

Основной причиной таких тенденций является старый традиционный подход к построению сложных автоматизированных систем управления, который применяется часто и в настоящее время: ориентация в первую очередь на применение новейших технических (технологических) достижений, стремление повысить степень автоматизации и функциональные возможности системы и, в то же время, недооценка необходимости построения эффективного человеко-машинного интерфейса, то есть период появления мощных, компактных и недорогих вычислительных средств, пришёлся пик исследований в США по проблемам человеческого фактора в системах управления, в том числе по оптимизации архитектуры и человеко-машинного интерфейса в SCADA - системах.

Изучение материалов по проблемам построения эффективных и надежных систем диспетчерского управления показало необходимость применения нового подхода при разработке таких систем: human-centered design (или top-down, сверху-вниз), т.е. ориентация в первую очередь на человека-оператора (диспетчера) и его задачи, вместо традиционного и повсеместно применявшегося hardware-centered (или bottom-up, снизу-вверх), в котором при построении системы основное внимание уделялось выбору и разработке технических средств (оборудования и программного обеспечения). Применение нового подхода в реальных космических и авиационных разработках и сравнительные испытания систем в Национальном управлении по аэронавтике и исследованию космического пространства (NASA), США, подтвердили его эффективность, позволив увеличить производительность операторов, на порядок уменьшить процедурные ошибки и свести к нулю критические (некорректируемые) ошибки операторов.

## 2.1 SCADA-системы: общие понятия и структура

SCADA - процесс сбора информации реального времени с удаленных точек (объектов) для обработки, анализа и возможного управления удаленными объектами. Требование обработки реального времени обусловлено необходимостью доставки (выдачи) всех необходимых событий (сообщений) и данных на центральный интерфейс оператора (диспетчера). В то же время понятие реального времени отличается для различных SCADA-систем.

Прообразом современных систем SCADA на ранних стадиях развития автоматизированных систем управления являлись системы телеметрии и сигнализации. Все современные SCADA-системы включают 3 основных структурных компонента (рис.7).

Рис.7. Основные структурные компоненты SCADA-системы.

Remote Terminal Unit (RTU) удаленный терминал, осуществляющий обработку задачи (управление) в режиме реального времени. Спектр его воплощений широк от примитивных датчиков, осуществляющих съем информации с объекта, до специализированных многопроцессорных отказоустойчивых вычислительных комплексов, осуществляющих обработку информации и управление в режиме жесткого реального времени. Конкретная его реализация определяется конкретным применением. Использование устройств низкоуровневой обработки информации позволяет снизить требования к пропускной способности каналов связи с центральным диспетчерским пунктом.

Master Terminal Unit (MTU), Master Station (MS) диспетчерский пункт управления (главный терминал); осуществляет обработку данных и управление высокого уровня, как правило, в режиме мягкого реального времени; одна из основных функций обеспечение интерфейса между человеком-оператором и системой (HMI, MMI). В зависимости от конкретной системы MTU может быть реализован в самом разнообразном виде: от одиночного компьютера с дополнительными устройствами подключения к каналам связи, до больших вычислительных систем (мэйнфреймов) или объединенных в локальную сеть рабочих станций и серверов. Как правило, и при построении MTU используются различные методы повышения надежности и безопасности работы системы.

Communication System (CS) коммуникационная система (каналы связи), необходима для передачи данных с удаленных точек (объектов, терминалов) на центральный интерфейс оператора-диспетчера и передачи сигналов управления на RTU (или удаленный объект в зависимости от конкретного исполнения системы).

В системе диспетчерского управления человек-оператор должен выполнять следующие функции:

планирование действий;

программирование компьютерной системы на следующие действия;

отслеживание результатов (полу) автоматической работы системы;

человек-оператор должен вмешиваться в процесс, в случае критических событий, когда автоматика не может справиться, либо при необходимости подстройки (регулировки) параметров процесса;

обучение в процессе работы (получение опыта).

Для выполнения этих функций, к SCADA системам предъявляются следующие требования:

надёжность системы (технологическая и функциональная);

безопасность управления;

точность обработки и представления данных;

простота расширения системы.

В целях безопасности SCADA система должна обладать следующими свойствами:

никакой единичный отказ оборудования не должен вызвать выдачу ложного выходного воздействия (команды) на объект управления;

никакая единичная ошибка оператора не должна вызвать выдачу ложного выходного воздействия (команды) на объект управления;

все операции по управлению должны быть интуитивно-понятными и удобными для оператора (диспетчера).

SCADA системы могут применяться в различных сферах: как в узких, так и в более широких. Они применяются в управлении передачи и распределения электроэнергии, в промышленных производствах, в производствах электроэнергии, в водозаборах и водоочистках, в добыче и транспортировки нефти и газа, в управлении космическими объектами, в телекоммуникациях, а так же в военной области.

В настоящее время в развитых зарубежных странах наблюдается настоящий подъем по внедрению новых и модернизации существующих автоматизированных систем управления в различных отраслях экономики; в подавляющем большинстве случаев эти системы строятся по принципу диспетчерского управления и сбора данных. Характерно, что в индустриальной сфере (в обрабатывающей и добывающей промышленности, энергетике и др.) наиболее часто упоминаются именно модернизация существующих производств SCADA-системами нового поколения. Эффект от внедрения новой системы управления исчисляется, в зависимости от типа предприятия, от сотен тысяч до миллионов долларов в год; например, для одной средней тепловой станции он составляет, по подсчетам специалистов, от 200000 до 400000 долларов. Большое внимание уделяется модернизации производств, представляющих собой экологическую опасность для окружающей среды (химические и ядерные предприятия), а также играющих ключевую роль в жизнеобеспечении населенных пунктов (водопровод, канализация и пр.). С начала 90-х годов в США начались интенсивные исследования и разработки в области создания автоматизированных систем управления наземным (автомобильным) транспортом ATMS (Advanced Traffic Management System).

## 3. Обзор программных средств для разработки человеко-машинного интерфейса

В настоящее время на рынке существует множество SCADA систем, но но речь пойдёт о более популярных. Первая такая система - это SIMATIC WinCC.

SCADA система SIMATIC WinCC (Windows Control Center) - это компьютерная система человеко-машинного интерфейса, работающая под управлением операционных систем Windows 2000/XP и предоставляющая широкие функциональные возможности для построения систем управления различного назначения:

простое построение конфигураций клиент-сервер;

поддержка резервированных структур систем автоматизации;

неограниченное расширение функциональных возможностей благодаря использованию ActiveX элементов;

открытый OPC-интерфейс (OLE for Process Control) интерфейс для реализации функций обмена данными;

простое и быстрое конфигурирование системы в сочетании с пакетом STEP 7.

Базовая конфигурация системы включает в свой состав набор функций, позволяющих выполнять событийно управляемую сигнализацию, архивирование результатов измерений, регистрировать технологические данные и параметры настройки конфигурации, функции управления и визуализации. Целый ряд функций может быть реализован с помощью встроенных ANSI-C компилятора и VisualBasic-script: от простейших операций до полного доступа к системным функциям SIMATIC WinCC. Кроме того, базовая система может дополняться опциональными пакетами WinCC и WinCC Add-ons.

На основе WinCC могут создаваться как простейшие системы человеко-машинного интерфейса с одной станцией оператора, так и мощные многопользовательские системы, включающие в свой состав десятки станций. Поддержка стандартных интерфейсов OLE, ODBC, OLE и SQL обеспечивает универсальность и открытость WinCC, позволяет использовать ее в сочетании с любым другим программным обеспечением. WinCC легко интегрируется во внутреннюю информационную сеть компании. Это не только снижает затраты на ее внедрение, но и повышает гибкость информационной системы.

Система SIMATIC WinCC разработана для решения задач визуализации и оперативного управления в различных областях промышленного производства. Система оснащена мощным интерфейсом для связи с процессом, пригодна для работы со всем спектром изделий SIMATIC, обеспечивает парольный доступ к управлению процессом, обладает высокой производительностью. Базовая конфигурация системы обладает высокой универсальностью и может быть использована для построения систем управления самого разнообразного назначения.

Открытая система визуализации фирмы SIEMENS WinCCTM позволяет легко и просто интегрировать компоненту визуализации и обслуживания в создаваемые или уже существующие системы технологического управления, избежав при этом непомерных затрат на проектирование и написание программного обеспечения, и конечно же она поддерживает русский язык. Ядро продукта WinCCTM образует нейтральная по отношению к отраслям промышленности и технологиям базовая система, которая оснащена всеми важнейшими функциями визуализации и обслуживания.

Так же на рынке популярна SCADA-система TRACE MODE.

Рис.8 Графическая мнемосхема процесса, созданная в SCADA-системе TRACE MODE.

TRACE MODE предназначена для разработки крупных распределенных АСУТП (автоматизированных систем управления технологическим процессом) широкого назначения. TRACE MODE создана в 1992 году фирмой AdAstra Research Group Ltd (Россия), и к настоящему времени имеет свыше 7000 инсталляций. Системы, разработанные на базе TRACE MODE, работают в энергетике, металлургии, нефтяной, газовой, химической и других отраслях промышленности и в коммунальном хозяйстве России. По числу внедрений в России TRACE MODE значительно опережает зарубежные пакеты подобного класса. TRACE MODE - основана на инновационных, не имеющих аналогов технологиях. Среди них: разработка распределенной АСУТП как единого проекта, автопостроение, оригинальные алгоритмы обработки сигналов и управления, объемная векторная графика мнемосхем, единое сетевое время, уникальная технология playback - графического просмотра архивов на рабочих местах руководителей. TRACE MODE - это первая интегрированная SCADA - и softlogic-система, поддерживающая сквозное программирование операторских станций и контроллеров при помощи единого инструмента. Разработка графического интерфейса операторских станций проекта осуществляется в объектно-ориентированном редакторе представления данных.

Графические изображения создаются в векторном формате DBG, однако возможно использовать и растровые изображения в формате BMP. Размер графического поля и число экранов не ограничены. Редактор содержит библиотеки объемных изображений мнемосхем технологических объектов, включающих баки, емкости, трубы, задвижки, а также их различные сечения и сопряжения. Формы динамизации содержат все необходимые элементы, в т. ч. гистограммы, графические, цветовые и звуковые сигнализаторы, тренды, бегущие дорожки, мультипликацию и т.д. Обширный набор библиотек технологических объектов, включающий емкости, теплообменники, электротехнические символы и др., а также панели управления, ввода заданий, регуляторов, приборов и т.д. Любая часть изображения может быть включена в объекты и анимирована произвольным образом. Для импорта изображений из других приложений Windows (например из AutoCAD) редактор поддерживает форматы WMF и EMF. Графические мнемосхемы возможно редактировать в реальном времени.

Исторические тренды TRACE MODE позволяют вводить неограниченное число переменных с неограниченной глубиной. В реальном времени пользователь может добавлять и удалять выводимые на тренд параметры, производить масштабирование и смещение трендов по осям X и Y. Тренды имеют визир и вывод значений в точке курсора. Основные функции:

модульная структура - от 128 до 64000х16 I/O. Количество тегов неограниченно;

0,001 с - минимальный цикл системы;

открытый формат драйвера для связи с любым УСО.

открытость для программирования (Visual Basic, Visual C++ и т.д.);

разработка распределенной АСУТП как единого проекта;

средства сквозного программирования АСУТП верхнего (АРМ) и нижнего (ПЛК) уровня;

встроенные библиотека из более чем 150 алгоритмов обработки данных и управления в т. ч. фильтрация, PID, PDD, нечеткое, адаптивное, позиционное регулирование, ШИМ, управление устройствами (клапан, задвижка, привод и т.д.), статистические функции и произвольные алгоритмы;

автоматическое горячее резервирование;

поддержка единого сетевого времени;

средства программирования контроллеров и АРМ на основе международного стандарта IEC 1131-3;

более 200 типов форм графического отображения информации в т. ч. тренды, мультипликация на основе растровых и векторных изображений, ActiveX;

просмотр архивной информации в реальном времени в т. ч. в виде трендов и таблиц;

сеть на основе Netbios, NetBEUI, IPX/SPX, TCP/IP;

обмен с независимыми приложениями с использованием OPC client/server, DDE/NetDDE client/server, SQL/ODBC, DCOM;

автоматическое резервирование архивов и автовосстановление после сбоя;

мониторинг и управление через Internet;

полностью русифицирована;

техническая поддержка на русском языке.

Вышеописанные SCADA-системы являются популярными на рынке SCADA-систем. Но следующая SCADA-система GENESIS32, которая будет описана подробней в данной дипломной работе, на мой взгляд, является лидером.

GENESIS32 является комплексом клиентских и серверных приложений, основанных на технологии OPC (OLE for Process control - технология связывания и внедрения объектов для промышленной автоматизации), которые предназначены для разработки прикладного программного обеспечения визуализации контролируемых параметров, сбора данных и оперативного диспетчерского управления в автоматизированных системах управления технологическими процессами. GENESIS32 является комплексом 32-разрядных приложений для Windows 98, Windows NT,Windows 2000, Windows XP и Windows Vista, построенных в соответствии со спецификацией OPC. Комплекс предназначен для создания программного обеспечения сбора данных и оперативного диспетчерского управления верхнего уровня систем промышленной автоматизации. В состав GENESIS-32 также входит среда разработки и исполнения сценарных процедур VBA, обеспечивающая возможность разработки части программного обеспечения средствами Microsoft Visual Basic for Applications 6.0 (Visual Basic для приложений), входящего в популярный пакет MS Office 2000. Все программные компоненты реализованы на базе многопоточной модели и поддерживают технологию ActiveX.

Первый пакет, выпущенный фирмой для Windows в 1992 году - WinWorX - реализовал не все функции, необходимые для пакета программ класса SCADA. Фактически в него были включены все средства операторского интерфейса - динамическа графика, тренды, аварийные сообщения, рапорты, но не было собственно средств сбора и первичной обработки параметров процесса, а также автоматического управления. Эти задачи возлагались на пакет Genesis в версии для DOS, от которого по сети получал данные WinWorX. Обмен информацией шел в стандартном для Genesis'а протоколе, а специальная программа под Windows - DDE-сервер, приняв данные из сети, отдавала их WinWorX'у через стандартный уже для Windows протокол динамического обмена данными - DDE. Впрочем в качестве источника и приемника данных могли служить и любые другие Windows-программы, поддерживающие этот протокол. Реализованный подход дал возможность предложить средства, позволяющие организовать дополнительные сетевые рабочие места операторов, которые хотя и не связаны непосредственно с объектом, но функционально решают все задачи, необходимые для верхнего уровня системы. Для обеспечения успеха нового продукта на рынке, фирма установила на него невысокую цену, которая еще снижается при приобретении групповой лицензии (в 1,5 раза при приобретении лицензии на 6 рабочих мест и в два раза при приобретении лицензии на 12 рабочих мест, в последнем случае) (стоимость одного рабочего места оказывается ниже $1000, что существенно меньше, чем цена других известных на рынке продуктов аналогичного класса). Еще один маркетинговый шаг также повышал привлекательность WinWorX'а и уменьшал его стоимость - любой из модулей пакета, в случае, если на каком-либо из рабочих мест системы не нужны все его функции, можно приобрести отдельно (WinWorX Open Seriesa).

Пакет GENESIS for Windows полностью решает вопросы автоматизации производства на всех уровнях АСУТП, он также позволяет просто и эффективно решить проблему взаимодействия АСУТП с АСУП. Graphworx является модульной системой, состоящей из полностью независимых приложений, взаимодействующих между собой в самой современной на сегодняшний день программной архитектуре клиент-сервер. При создании системы можно комбинировать модули GENESIS for Windows и любые другие Windows приложения. Клиенты получают информацию от серверов по программной магистрали передачи данных в реальном времени Talx Data Bus. Несмотря на то, что пакет программ Genesis for Windows состоит из многих модулей, он является интегрированной системой, которую пользователь воспринимает как единое целое благодаря наличию общей оболочки - менеджера проекта, а также благодаря возможности использования общих средств автоматического формирования сценария работы в каждом модуле, где они могут понадобиться.

Прямые связи Graphworx с оборудованием управляются высокопроизводительным сервером реального времени (RTS), включающем в себя интерфейс драйверов ввода-вывода и функции контроля и управления. RTS сканирует ввод-вывод, вычисляет алгоритмы обработки сигналов, формирует сигналы тревоги и обслуживает все запросы на данные от приложений-клиентов. Благодаря использованию механизма приоритетной вытесняющей многозадачности обеспечивается гарантированное выполнение с заданным периодом наиболее критических функций, например обновление ввода-вывода. RTS работает с выбранным пользователем интервалом времени (50, 100, 250 миллисекунд, 1 или 2 секунды), прерывая на несколько миллисекунд работу Windows.

Конфигуратор, с помощью которого для RTS задается программа работы в режиме реального времени, это уже зарекомендовавший себя в DOSовской версии мощный графический редактор языка функциональных блоков. Создание конфигурации сводится к рисованию потока обработки данных с помощью связанных между собой алгоблоков. Библиотечные алгоритмы размещаются на схеме в виде прямоугольников, выходы одних блоков связаны линиями со стрелкой в точке входа с другими блоками. Созданную схему можно экспортировать в формат AutoCAD'а и использовать затем, например, в качестве мнемосхемы. Кроме того имеется возможность экспорта в формат dBase и последующего импорта из этого формата, что позволяет использовать широко распространенные программы баз данных и электронных таблиц для редактирования конфигурации. Такая технологи создания конфигурации бывает особенно удобна при необходимости тиражирования множества однотипных блоков или при выполнении операций поиска и замены. Кроме входов и выходов блоки имеют имя и настроечные параметры. Библиотека содержит свыше 60 алгоритмов нескольких типов (ввод-вывод, первичная обработка сигналов, арифметика, математические функции, логика, регулирование, управление и др.). Именно обширная библиотека алгоритмов позволят использовать Graphworx как средство прямого программного управления.

Приложения-клиенты обращаются к данным RTS, используя составное имя, включающее имя сетевого узла, имя конфигурации, имя блока и обозначение конкретного параметра в данном блоке. Сервер реального времени использует существующие, хорошо проверенные драйверы DOSовского GENESIS'а для интерфейса с полевым оборудованием.

Рабочая станция GENESIS for Windows может быть связана с другими GENESIS с помощью модуля GEN-NET, позволяющего передавать по сети данные, файлы и аварийные сообщения. Удаленные узлы GENESIS действуют как независимые серверы данных, связанные с устройствами ввода-вывода. Все данные, содержащиеся в удаленном узле GENESIS, доступны в реальном времени дл приложений-клиентов. Такая распределенная система позволяет обрабатывать огромное количество данных, не перегружая отдельные станции. В некоторых системах бывает целесообразно для сбора и обработки данных выделить отдельные машины, которые могут не иметь операторского интерфейса - дисплея и клавиатуры ("слепой узел"). Тем самым одни узлы распределенной системы оказываются специализированы на функциях сбора и обработки информации (GEN-NODE - специальная, без графики, сетевая версия Genesis'а дл DOS), другие используются исключительно для функций операторского интерфейса. Как правило, такое построение системы бывает целесообразным, когда несколько операторов должны иметь доступ к одним и тем же данным с объекта. При этом изменение архитектуры системы практически не требует переделок конфигурации, поскольку драйверы устройств и конфигурация RTS полностью совместимы между GENESIS for Windows и GENESIS'ом для DOS, а ссылки на переменные связаны только с именами сетевого узла и его конфигурации

Еще один метод доступа к данным с объекта используют серверы ввода-вывода. Каждый сервер ввода-вывода обеспечивает интерфейс с каким-то одним типом оборудования, например с определенным типом логического или регулирующего контроллера какой-либо фирмы. Сервер ввода-вывода состоит из двух частей - конфигуратора и исполняемого модуля.

Конфигуратор создает базу данных, совместимую со стандартом ODBC, для определения параметров ввода-вывода (имя, тип точки - аналоговый или дискретный, расположение данных в памяти, масштаб и многое другое). Совместимость с ODBC делает возможным экспорт информации в различные внешние базы данных, а также импорт данных из таких систем программирования контроллеров, которые поддерживают этот стандарт.

Исполняемый модуль сервера ввода-вывода обеспечивает связь с драйверами, обновляет в реальном времени базу данных текущими значениями переменных и формирует сигналы тревоги при отклонениях параметров за заданные пределы. Данные могут обновляться как непрерывно (например, для контроля аварийного отклонения), так и по запросу от приложения-клиента. Такой гибкий подход позволяет серверу ввода-вывода одновременно обрабатывать тысячи точек.

Сервер ввода-вывода работает как DDE-сервер, что позволяет использовать его не только в рамках GFW, но и как источник данных дл любых Windows-приложений, поддерживающих DDE.

GENESIS for Windows поддерживает стандартный протокол динамического обмена данными - Windows Dinamic Data Exchange (DDE). Модуль DDEWorX в качестве DDE-клиента может получать данные от внешних программ и передавать их приложениям-клиентам GFW по внутренней программной магистрали передачи данных Talx Data Bus. Для связи с другими рабочими станциями Windows в сети используется NetDDE. GFW поддерживает определяемые пользователем списки DDE-имен, благодаря чему соединение осуществляется простым нажатием кнопки мыши. Не менее просто использование таких инструментов, как Cut and Paste (Вырезать и Вклеить); для установления связи достаточно этими стандартными функциями редактирования вставить имя переменной из внешнего приложения в GFW. При отладке системы возможен просмотр и фильтрация DDE вызовов, ошибок, сообщений и т.п.

Модуль GraphWorX позволяет пользователю легко и быстро создавать динамические цветные экраны-мнемосхемы. Объектно-ориентированная графика дает возможность связать любой графический объект с технологическими параметрами процесса. Например, объект может одновременно менять цвет, размер и положение на экране в соответствии со значениями различных сигналов. Среди множества средств анимации изображения есть и мультипликация. Весь процесс создания мнемосхемы сводится к рисованию и заданию в диалоге динамических свойств изображения. При создании мнемосхемы используются ставшие уже привычными в Windows'овских программах последнего поколения плавающие инструментальные линейки, которые можно расположить в любом месте экрана.

Изображение всей мнемосхемы или ее отдельных частей может быть скопировано из других Windows приложений, импортировано из AutoCAD или GENESIS for DOS. Также имеется библиотека таких часто используемых символов как задвижки, насосы, панели регуляторов и т.п. Любое изображение на экране можно сохранить в собственной библиотеке. Особенно удобно то, что наряду со статическими в библиотеках могут храниться и динамические символы. Как библиотеки символов, так и целые мнемосхемы могут использоваться без переделок в различных проектах, позволяя накапливать не только опыт работы с пакетом, но и готовые решения, тиражируемые во всех последующих системах.

GraphWorX обеспечивает возможности конфигурирования непосредственно в режиме on-line, что обеспечивает отладку динамики мнемосхемы на реальных данных с объекта.

Окно GraphWorX, как и любое окно системы Windows, имеет свойства изменения размера и перемещения по экрану, при этом, естественно, вся графика и тексты растягиваются пропорционально изменению масштаба без потери графической информации. Эта возможность позволяет Вам следить за процессом во время работы других программ или расположить на экране множество мнемосхем одновременно.

В качестве элементов мнемосхемы могут быть использованы такие GFW-приложения, как TrendWorX и AlarmWorX. Во врем конфигурирования мнемосхемы доступны все конфигурационные возможности этих приложений, а в режиме реального времени меню управления просмотром тренда или буфера сообщений вызывается простым щелчком клавиши мыши на изображении тренда или списка сообщений.

Данные из GFW могут передаваться другим Windows приложениям, таким как электронные таблицы, базы данных, VisualBASIC, С++ и др. Модуль DataSpy является DDE-сервером и с точностью до направления передачи информации функционально аналогичен описанному выше модулю DDEWorX. DataSpy совместим со стандартом встраиваемых документов OLE 2.0. Пользователь может создавать встраиваемые "документы", содержащие сотни опрашиваемых точек от DDE, RTS, Gen-Net, а также переменные сценариев. Все они могут быть не только прочитаны, но и изменены.

GENESIS for Windows обладает возможностями написания сценариев дл выполнения вычислений, последовательностей команд и других интеллектуальных функций. Visual BASIC-совместимые команды позволяют Вам задать реакцию системы на события и действия оператора. По сценарию можно менять мнемосхемы, выдавать рапорты, считывать и записывать данные, просматривать тренды, квитировать аварийные сообщения и даже запускать мультимедийные клипы. Более 300 различных команд содержится в библиотеке. Выполнение сценария может начаться после нажатия оператором какой-либо кнопки на клавиатуре или экране, периодически и по событиям. Непрограммисты без труда создадут сценарий с помощью мощного Мастера Сценариев - модуля Script Wizard. Script Wizard сам генерирует сценарии на основе ответов пользователя на вопросы, предлагаемые ему в диалогах и меню. Доступ к созданию сценария возможен из приложений-клиентов GraphWorX, AlarmWorX, TrendWorX, а также из программы управления проектом. В сценариях поддерживаются глобальные переменные, которые могут быть определены в одном сценарии, а использоваться не только в нем, но и в любом другом, а также в мнемосхемах и трендах.

Модуль разработки сценариев содержит полнофункциональные редактор и отладчик, работающие не только в режиме конфигурации, но и непосредственно в ON-LINE. В тексте сценариев можно использовать фрагменты программ, заимствованные из приложений, разработанных на языке Visual BASIC.

GFW - модульный пакет, который может в конкретных применениях содержать различный набор модулей. Project Manager (Менеджер Проекта) объединяет их в единую интегрированную систему, в которой есть прямой доступ к любому приложению. Менеджер Проекта позволяет задать дл использования в режиме Runtime внешний вид экрана после старта пакета. Например, половину экрана может занимать мнемосхема, четверть - электронная таблица, а остальные запущенные приложения представлены в виде, свернутом до символизирующего их значка (иконки).

Project Manager дает возможность задать для каждого из зарегистрированных в проекте операторов один из 256 уровней доступа к системе, а также определить объем прав для каждого из этих уровней. С помощью Менеджера Проекта можно определить доступные серверы и приложения, конфигурировать действия серверов данных, например, используемые ими базы данных, контролировать в режиме реального времени их состояние и т.д.

GENESIS32 является комплексом клиентских и серверных приложений, основанных на технологии OPC (OLE for Process control - технология связывания и внедрения объектов для промышленной автоматизации), которые предназначены для разработки прикладного программного обеспечения визуализации контролируемых параметров, сбора данных и оперативного диспетчерского управления в автоматизированных системах управления технологическими процессами. GENESIS32 является комплексом 32-разрядных приложений для Windows 98, Windows NT,Windows 2000, Windows XP и Windows Vista, построенных в соответствии со спецификацией OPC. Комплекс предназначен для создания программного обеспечения сбора данных и оперативного диспетчерского управления верхнего уровня систем промышленной автоматизации. В состав GENESIS-32 также входит среда разработки и исполнения сценарных процедур VBA, обеспечивающая возможность разработки части программного обеспечения средствами

Microsoft Visual Basic for Applications 6.0 (Visual Basic для приложений), входящего в популярный пакет MS Office 2000. Все программные компоненты реализованы на базе многопоточной модели и поддерживают технологию ActiveX. Основные возможности GENESIS32:

соответствие стандартам OPC: ОРС Data Access 3.0, ОРС A&E 1.1, OPC HDA 2.0, OPC DX и OPC XML DA;

поддержка OPC UA (Unified Archtechture);

двусторонний обмен информацией с базами данных и электронными таблицами (MS Access, MS Excel, ORACLE, SQL Server, SAP и т.д.);

единый навигатор данных, позволяющий одновременно получать доступ к текущим и историческим данным, тревогам, псевдонимам, базам данных.

мощные средства визуализации;

разработка сценариев на языках VBA 6.3, VBScript, JScript;

поддержка технологий OLE и ActiveX;

архивирование информации в открытых базах данных;

средства анализа архивов: генерация отчётов в различных форматах и визуальное воспроизведение исторических данных: "обратная прокрутка" изображений, трендов и тревог с заданной скоростью, пошаговый просмотр, сравнение данных с "идеальной моделью", поиск критических событий и т.д.

горячее резервирование узлов, серверов текущих и исторических данных, тревог, архивных баз данных, тегов ОРС;

оперативное переключение источников данных;

широкий набор графиков: зависимость от времени и от другой переменной, гистограмма, самописец, логарифмическая зависимость, круговая диаграмма. Отображениe на трендах текущих и исторических данных, значений из баз данных и электронных таблиц;

оповещение персонала о событиях и тревогах, в том числе с помощью средств мультимедиа;

широкие возможности построения распределённых систем, в том числе организация соединений Internet/Intranet. Передача данных ОРС с помощью транспортных протоколов TCP/IP и SOAP/XML. Кэширование данных, механизмы обработки нарушения связи и переключения на резервные серверы;

поддержка интерфейса SNMP (Simple Network Management Protocol);

средства управления проектами: ведение истории проекта, поиск и замена, Web-публикация, упаковка и развертывание, совместимость с пакетом Microsoft Source Safe, настройка параметров запуска проекта;

переключение "на лету" языка пользовательского интерфейса проекта;

система безопасности;

установка прав пользователей на выполнение операций в приложениях проекта, доступ к файлам и элементам управления, к сигналам и тревогам, контроль исполнения сценариев, а также места и времени регистрации пользователей;

контроль и регистрация действий оператора;

предусмотрена возможность интегрирования с системными настройками безопасности, а также регистрация с помощью биометрических устройств;

встроенные средства диагностики.

## 4. Структура SCADA - системы GENESIS 32

GENESIS 32 является комплексом 32-разрядных приложений для Windows 98, Windows NT, Windows 2000, Windows XP и Windows Vista построенных в соответствии со спецификацией OPC. Комплекс предназначен для создания программного обеспечения сбора данных и оперативного диспетчерского управления верхнего уровня систем промышленной автоматизации. В состав GENESIS 32 также входит среда разработки и исполнения сценарных процедур VBA, обеспечивающая возможность разработки части программного обеспечения средствами Microsoft Visual Basic for Applications 6.0 (Visual Basic для приложений), входящего в популярный пакет MS Office 2000. Все программные компоненты реализованы на базе многопоточной модели и поддерживают технологию ActiveX. В состав GENESIS 32 входят приложения соответствующие спецификации OPC, представленные в табл.1:

Табл.1 Приложения, входящие в состав Genesis 32.

|  |  |
| --- | --- |
| Название приложения | Описание |
| GraphWorX32 | GraphWorX32 объединяет средства разработки и просмотра графических мнемосхем, автоматизированных рабочих мест оператора АСУТП.  |
| TrendWorX32 | Даёт возможность высокопроизводительного построения графических зависимостей контролируемых параметров.  |
| AlarmWorX32 | Мощная подсистема обнаруженияфильтрации и сортировки аварийных и других событий, связанных с контролируемым технологическим процессом и состоянием технических средств АСУТП.  |

Указанные приложения могут заказываться и применяться как в составе комплекса, так и автономно. Кроме того, фирма ICONICS поставляет дополнительные приложения и инструментальные средства разработки приведённые в табл.2:

Табл.2 Дополнительные приложения и инструментальные средства разработки, поставляемые фирмой ICONICS

|  |  |
| --- | --- |
| Название доп. приложения | Описание |
| ScriptWorX32 | Позволяет выполнять одновременно вычислительные операции и любые другие действия, доступных в языке программирования VBA.  |
| AlarmWorX32 Multimedia | Мультимедийное приложение, которое предназначено для локального и удаленного оповещения оперативного персонала об аварийных и других событиях, связанных с контролируемым технологическим процессом и состоянием технических средств АСУТП.  |
| WebHMI | Осуществляет просмотр графических мнемосхем контролируемого технологического процесса в глобальной сети Internet или в корпоративной Intranet.  |
| DataWorX32 | DataWorX32 является OPC-сервером, который предназначен для организации единого моста между множеством клиентских и серверных компонентов системы.  |
| Библиотека символов Symbols32 Library | Набор символов, с помощью которых строится мнемосхема.  |
| ICONICS GEN-OPC Сервер | Обеспечивает интерфейс OPC между приложениями.  |
| OPC ToolWorX | Является инструментальным средством быстрой разработки серверов и клиентов OPC.  |
| ActiveX ToolWorX | Предназначен для быстрой разработки управляющих элементов Activex.  |

## 4.1 GraphWorX32

GraphWorX32 объединяет средства разработки и просмотра графических мнемосхем, автоматизированных рабочих мест оператора АСУТП. Мнемосхемы (экранные формы) могут создаваться как на основе встроенных средств рисования, так и управляющих элементов ActiveX других производителей. Алгоритмы вторичной обработки данных и процедуры управления экранными формами могут разрабатываться в интегрированной среде разработки и исполнения сценариев Visual Basic для приложений (VBA). GraphWorX32 является инструментальным средством, предназначенным для визуализации контролируемых технологических параметров и оперативного диспетчерского управления на верхнем уровне АСУТП, который полностью соответствует требованиям к клиенту OPC и поддерживает технологии ActiveX и OLE. Основные характеристики GraphWorX32:

многопоточное 32-разрядное приложение;

возможность обмена данными с любыми серверами OPC;

мощные инструменты для создания экранных форм и динамических элементов отображения;

возможность встраивания элементов управления ActiveX и объектов OLE;

встроенная среда редактирования сценарных процедур Microsoft Visual Basic for Applications;

динамизация элементов отображения со временем обновления графической информации 50 мс;

поддержка шаблонов экранных форм, содержащих наиболее часто используемые слои графических объектов;

возможность встраивания в HTML страницы и другие контейнеры OLE (MS Word, MS Excel, MS Access и др.);

возможность просмотра браузерами Интернет, такими как MS Internet Explorer;

обширная библиотека элементов отображения, ориентированных на построение мнемосхем промышленных объектов;

возможность встраивания графиков TrendWorX32 и экранов AlarmWorX32;

средства импорта графических метафайлов (WMF) и растровых изображений (BMP);

публикация экранных форм в глобальной сети Интернет;

встроенный редактор выражений для выполнения математических, функциональных, логических и других операций над данными.

## 4.2 TrendWorX32

TrendWorX32 представляет отрытое решение по высокопроизводительному построению графических зависимостей контролируемых параметров. Поддерживает спецификацию OPC доступа к историческим данным (OPC HDA), устанавливающую требования к подсистеме извлечения и представления исторических данных из баз данных архива. Пакет TrendWorX32 обеспечивает накопление и представление текущих данных в виде графических зависимостей от времени. Кроме того, TrendWorX32 является мощным средством архивации накапливаемой информации в базах данных с возможностью последующего извлечения и просмотра на графиках. Полностью соответствует спецификациям OPC доступа к текущим и историческим данным. Основные функциональные возможности TrendWorX32: представление значений контролируемых параметров, получаемых от серверов OPC, на графиках различных типов в реальном масштабе времени; архивирование значений контролируемых параметров в базах данных MS Access, MS SQL Server, Oracle, Microsoft Data Engine; генерация отчетов на основе данных архива и публикация отчетов в Интернет; вычисление статистических характеристик выборок значений контролируемых параметров; извлечение значений контролируемых параметров из архивов и представление в виде графиков различных типов; вывод графиков на печатающее устройство; разработка и исполнение сценарных процедур на встроенном Visual Basic для приложений; возможность вставки элементов просмотра графиков TrendWorX32 ActiveX в различные контейнеры ActiveX; встроенное средство генерации отчетов в базах данных и MS Excel.

## 4.3 AlarmWorX32

AlarmWorX32 мощная подсистема обнаружения, идентификации, фильтрации и сортировки аварийных и других событий, связанных с контролируемым технологическим процессом и состоянием технических средств АСУТП. AlarmWorX32 является набором программных компонентов, предназначенных для обнаружения аварийных событий, оповещения оперативного персонала, приема подтверждений восприятия информации об аварийных событиях и регистрации информации об авариях в базе данных. Основные функциональные возможности AlarmWorX32:

обнаружение аварийных событий по множеству признаков и критериев, настраиваемых пользователем;

передача информации об обнаруженных авариях клиентским приложениям, расположенным на разных узлах локальной или глобальной сети;

анализ аварийных событий и действий ответственного персонала;

объединение всех аварийных событий и подтверждений восприятия системных сообщений ответственным персоналом в сводки аварийных событий;

отображение вспомогательной информации для аварийных событий, позволяющей локализовать и устранить причины аварии;

связь с аппаратными средствами системы через интерфейсы OPC;

возможность запуска сервера обнаружения аварий в качестве сервисного процесса (службы) Windows NT;

мощное средство конфигурирования условий аварийных событий;

встроенная среда редактирования сценарных процедур Microsoft Visual Basic for Applications.

## 4.4 ScriptWorX32

ScriptWorX32 одновременное выполнение вычислительных операций и любых других действий, доступных в языке программирования VBA, управление базами данных, формирование отчетов и заданий. ScriptWorX32 является мощных средством разработки и исполнения сценарных процедур Microsoft Visual Basic for Applications (VBA) версии 6.0. ScriptWorX32 содержит мультизадачную среду параллельного исполнения сценариев с поддержкой симметричных многопроцессорных архитектур. VBA сценарии, разрабатываемые пользователем, могут выполнять операции обмена данными с серверами OPC. Основные функциональные возможности ScriptWorX32: многопоточное 32-разрядное приложение; возможность работы в операционных системах Windows NT,98/2000/XP; контейнер сценариев VBA 6.0; visual Basic for Applications 6.0 входит в установочный комплект; одновременное исполнение сценариев VBA 6.0; ускорение разработки сценариев при помощи Мастера сценариев; исполнение сценариев по расписанию или периодически; исполнение сценариев при выполнении условий, вычисляемых на основе значений тегов OPC-серверов; исполнение сценариев по событиям от серверов OPC Alarms and Events (серверов системных и аварийных событий OPC); диагностика текущих состояний сценариев; возможность компиляции сценариев в многопоточные библиотеки динамической компоновки (DLL); наличие глобальных сценариев для интеграции с другими приложениями; открытый интерфейс OLE Automation.

## 4.5 DataWorX32

DataWorX32 сервер организации единого централизованного списка контролируемых параметров с возможностью создания глобальных переменных, доступных всех клиентским приложениям программного комплекса, а также обладающий механизмом резервирования серверов OPC. DataWorX32 является OPC-сервером, который предназначен для организации единого моста между множеством клиентских и серверных компонентов системы. Основные функциональные возможности DataWorX32:

централизация параметров контролируемого процесса, обслуживаемых множеством серверов OPC, в едином списке;

оптимизация запросов множества клиентов OPC к одним и тем же параметрам в разных серверах OPC;

организация списка глобальных переменных с возможностью непосредственного обмена данными между клиентскими приложениями GENESIS32;

возможность выполнения арифметических, функциональных, логических и других операций над глобальными переменными;

возможность оперативного изменения привязки глобальных переменных к источникам данных в серверах OPC;

резервирование серверов OPC на узлах локальных и глобальных сетей с автоматическим перенаправлением запросов клиентских приложений в случае выхода из строя основных узлов.

## 4.6 WebHMI

WebHMI средство просмотра графических мнемосхем контролируемого технологического процесса в глобальной сети Internet и/или в корпоративной Intranet. WebHMI предназначен для предоставления данных и графической информации о контролируемом технологическом процессе любого клиентского приложения комплекса GENESIS32 любому компьютеру, на котором установлен браузер Интернет Microsoft Internet Explorer. WebHMI основывается на архитектуре Microsoft DNA и использует технологии ActiveX и DCOM. Помимо просмотра информации о контролируемом процесса WebHMI обеспечивает возможность оперативного диспетчерского управления, что позволяет строить недорогие распределенные системы верхнего уровня. Основные функциональные возможности WebHMI:

возможность работы в операционных системах Windows NT,98/2000/XP;

тонкий Web-клиент позволяет использовать на операторских станциях бесплатный Microsoft Internet Explorer для просмотра графических мнемосхем контролируемого процесса;

работа с графической информацией и данными GENESIS32 с "НУЛЕВОЙ инсталляцией" на операторских станциях;

публикация управляющих элементов ActiveX и HTML-страниц;

требует наличия Microsoft Internet Explorer 4 или 5;

передача данных OPC через Интернет;

администрирование действий пользователей и приложений на уровне NT.

## 4.7 AlarmWorX32 Multimedia

AlarmWorX32 Multimedia является мультимедийным приложением, поддерживающим технологию OPC, которое предназначено для локального и удаленного оповещения оперативного персонала об аварийных и других событиях, связанных с контролируемым технологическим процессом и состоянием технических средств АСУТП.

В состав AlarmWorX32 Multimedia входит набор агентов оповещения, позволяющих выполнять следующие оповещающие действия:

отправка сообщений об аварийных событиях на пейджеры (имеется поддержка технологии SMS);

взаимодействие с пейджерами в дуплексном режиме;

отправка сообщений по электронной почте;

отправка сообщений по факсу;

речевое оповещение посредством технологии преобразования текста в речь;

речевое оповещение при помощи воспроизведения звуковых файлов в формате wav;

видеооповещение путем воспроизведения видеороликов и отображения фотоснимков, оцифровка видеоинформации, поступающей от видеокамер;

оповещение во всплывающих окнах;

оповещение путем вывода сообщений в окне бегущей строки (Marquee);

оповещение бегущей строкой, выводимой во внешние устройства, подключенные к последовательному порту компьютера или к локальной вычислительной сети;

оповещение посредством отправки мгновенных сообщений через агента MS Instant Messaging;

работа с Конфигуратором сервера оповещения AlarmWorX32 существенно облегчается наличием в составе системы Мастера конфигураций, а также интерактивного учебного пособия Icon-Nick Tutorial.

## 4.8 Менеджер экрана Sreen Manager

Менеджер экрана является сервисной программой, которая позволяет управлять расположением окон приложений на экране компьютера. Менеджер экрана обеспечивает возможность задания конфигурации экранов одного и более мониторов для систем с несколькими мониторами.

## 4.9 DataSpy

DataSpy является сервисной программой, которая предназначена для проверки работоспособности серверов OPC, расположенных на локальном и удаленных компьютерах. DataSpy позволяет проверять совместимость со спецификацией серверов OPC Data Access версий 1.0 и 2.0, а также OPC Alarms and Events 1.0.

## 4.10 Dr. DCOM

Dr. DCOM является сервисной программой, которая облегчает настройку взаимодействия удаленных клиентов и серверов OPC через DCOM. Dr. DCOM обеспечивает поиск удаленных серверов, подключение к ним и тестирование вызовов интерфейсов OPC.

## 4.11 Сервер администрирования Security Server

Сервер администрирования предназначен для управления доступом пользователей и групп пользователей к функциям приложений, входящих в состав GENESIS32. Конфигуратор сервера администрирования позволяет создавать учетные записи для пользователей и групп пользователей с последующим ограничением или разрешением доступа к множеству операций в приложениях GENESIS32. Имеется возможность задания следующих ограничений:

ограничение доступа к файлам;

запрещение записи значений в теги OPC;

вход пользователя в систему на ограниченный промежуток времени;

вход пользователя в систему только на определенных станциях сети;

управление политикой учетной записи по аналогии с принятой в Windows NT/2000;

ограничение доступа к использованию конфигураторов GENESIS32;

ограничение доступа к выполнению множества операций в приложениях GENESIS32.

## 4.12 GEN-OPC сервер

GEN-OPC сервер обеспечивает интерфейс OPC между приложениями комплекса GENESIS for Windows 3.51 в существующих системах и клиентскими приложениями GENESIS32, что позволяет сохранить вложения пользователей и безболезненно перейти на новую версию продукта.

## 4.13 ActiveX ToolWorX

ActiveX ToolWorX предназначен для быстрой разработки управляющих элементов ActiveX, являющихся клиентами OPC, с возможностью последующего использования в приложениях-контейнерах, подобных GraphWorX32.

## 4.14 OPC ToolWorX

OPC ToolWorX имеет в своем составе мастер для автоматической генерации кода клиентов и серверов OPC в среде MS Visual C++ на базе примера полнофункционального OPC-сервера для протокола Modbus, а также тестовое клиентское приложение. OPC ToolWorX является инструментальным средством быстрой разработки серверов и клиентов OPC, который позволяет производителям серийного оборудования для промышленной автоматизации в кратчайшие сроки перейти к использованию наиболее передовой технологии обмена данными и обслуживания устройств в среде Windows. OPC ToolWorX содержит комплекты разработки серверов и клиентов OPC. Каждый комплект имеет в своем составе примеры исходных текстов двух серверов OPC, документацию, тестовое клиентское приложение, а также средство генерации интерфейсов диспетчеризации OLE Automation с тестовым примером на Visual Basic. Основные функциональные возможности OPC ToolWorX:

модель свободных потоков;

DLL автоматизации OLE;

мастера для генерации приложений Visual C++;

навигатор тегов OPC;

одновременная поддержка спецификаций OPC Data Access и OPC Alarms and Events;

возможность создания внутри задачных серверов для Windows CE.

## 5. Среда разработки мнемосхем Graphworx 32

Графические мнемосхемы автоматизированных рабочих мест оператора, называемые экранными формами, разрабатываются при помощи GraphWorX32 и сохраняются в файлах экранных форм, имеющих расширение \*. GDF.

Экранные формы могут включать в себя элементы просмотра графиков текущих и исторических данных TrendWorx32 Viewer ActiveX, элементы просмотра событий и тревог AlarmWorx32 Viewer ActiveX и элементы просмотра архива событий AlarmWorx32 Reporter ActiveX. Конфигурирование указанных элементов просмотра может выполняться как в самом GraphworX32, так и в многооконных контейнерах TrendWorX32 и AlarmWorX32 соответственно. Подсистема обнаружения и обработки событий строится на базе Конфигуратора сервера аварийных событий AlarmWorX32. Единый список переменных проекта может быть создан средствами DataWorX32, который не входит в комплект поставки GENESIS32 и должен заказываться отдельно. При этом обеспечивается возможность структуризации глобальных переменных путем перемещения их в отдельные многоуровневые группы, а также выполнения простых вычислительных операций над переменными.

Вторичная обработка данных и другие пользовательские алгоритмы могут выполняться в многопоточных сценариях VBA 6.0, разработка и исполнение которых осуществляется при помощи ScriptWorX32. Отдельные сценарии группируются в проект ScriptWorX32, который компилируется в многопоточную библиотеку динамической компоновки (DLL). Подсистема архивации данных реализуется на основе стандартных СУБД (MS Access, MS SQL Server, Oracle или Microsoft Data Engine), а также сервера архивации данных TrendWorx32 SQL Data Logger и сервера архивации событий AlarmWorX32 Logger. Извлечение информации из архива TrendWorx32 SQL Data Loggerможет выполняться при помощи элемента просмотра графиковTrendWorx32 Viewer ActiveX, генератора отчётов TrendworX32 Reporting, сценариев ScriptWorX32 или приложений, написанных на Visual Basic, Visual C++, C++ Builder с использованием интерфейсов OLE DB. Все компоненты комплекса открыты через интерфейс OLE Automation. Это значит, что каждое из приложений GENESIS32 может быть встроено в любую имеющуюся или создаваемую программную систему, разработка которой ведётся с использованием инструментальных средств общго назначения, подобных Visual C++, Visual Basic, Delphi, C-builder и т.п.

Внешний вид главного окна Graphworx32 с его основными компонентами, включая панель инструментов, набор инструментов рисования, упорядочивания, шрифтов и динамики, показан на рис.11.

Рис.9. Внешний вид главного окна GraphWorX32

## 6. Создание мнемосхемы модели дискретно-непрерывного технологического процесса

Для создания данной мнемосхемы необходимо чётко поставить задачу. Предполагается, что имеется два компонента: *CompА* и *CompB*, которые могут быть смешаны друг с другом в определённой пропорции. После смешивания будет происходить отгрузка готового продукта. Предположим, что компоненты *CompA* и *CompB* являются жидкостями. Подача жидкости *CompA* в смесительный резервуар будет управляться задвижкой *ValveA*. Подача жидкости *CompB* будет управляться задвижкой *ValveB*. Задвижки *ValveA* и *ValveB* имеют два положения: Открыта и Закрыта. Запуск технологического процесса производится нажатием кнопки “**Пуск".** В смесительном резервуаре имеются 3 лопастных смесителя, расположенных на разных уровнях. Включение смесителя *Mixer* 1 происходит при достижении 30% -ного наполнения резервуара. Смеситель *Mixer* 2 включается при 50% -ном наполнении резервуара. Смеситель *Mixer* 3 включается при уровне наполнения, равном 70%. Остановка смесителей происходит при понижении уровня наполнения до соответствующих отметок включения. Цвет изображений смеси будет изменяться по мере смешивания. Насос *Pump*1 будет использоваться для отгрузки приготовленной смеси. Производительность насоса равна производительности. эквивалентной подаче жидкости в резервуар через одну задвижку.

## 6.1 Создание статических объектов мнемосхемы

Для создания статических объектов мнемосхемы будет использована встроенная библиотека символов.

Рис.10 Главное окно библиотеки символов

Для данной мнемосхемы необходима категория символов под названием *TrainingClass. sdf.* Из этой категории выбираем нужные символы и переносим на мнемосхему, образуя нужную модель, показанную на рис.11.

Рис.11 Расположение статических объектов из Библиотеки символов на мнемосхеме

## 6.2 Создание анимационных объектов

В данном пункте будет описано представление изменяющегося уровня в резервуаре. Для этого следует копировать изображение разреза в стенке резервуара. Одно из этих изображений следует перекрасить в чёрный цвет для эффекта пустоты в резервуаре. Изображение чёрного цвета нужно поместить на задний план, а поверх его наложить его копию синего цвета для эффекта заполнения резервуара. Затем к этому изображению применяется динамика изменения размера. В поле источника данных заведётся переменная ***~~tank~~***, значение которой будет определять степень заполнения резервуара.

## 6.3 Создание имитации наполнения/опустошения резервуара

Ниже будет описана настройка динамического действия для графического объекта *Switch On* создаваемой мнемосхемы. Как следует из названия, данный графический объект представляет переключатель с двумя фиксируемыми положениями и служит для запуска дискретно-непрерывного процесса. После выполнения однократного щелчка левой клавишей мыши над объектом *Switch on* (переключатель зелёного цвета), переключатель перейдёт во включённое положение и будет происходить загрузка значений в локальную переменную ***~~tank~~***.

Данный алгоритм будет иметь следующие свойства:

уровень в резервуаре будет установлен в 0, если значение переменной ***~~tank~~*** станет меньше 0;

если уровень в резервуаре больше либо равен 0, при одновременном равенстве 1 сигнала запуска процесса то уровень в резервуаре будет представляться следующей формулой:

***Уровень = Текущий уровень + ICONICS. Simulator.1\diplom. dip. bool. out1 + ICONICS. Simulator.1\diplom. dip. bool. out2 - отгрузка***, где: ***ICONICS. Simulator.1\diplom. dip. bool. out1 = 1***, если задвижка *ValveA* (расположена слева от резервуара) открыта (*CompA* поступает в резервуар)

***ICONICS. Simulator.1\diplom. dip. bool. out1 = 0***, если задвижка *ValveA* (расположена слева от резервуара) закрыта (*CompA* не поступает в резервуар)

***ICONICS. Simulator.1\diplom. dip. bool. out2 = 1***, если задвижка *ValveB* (расположена справа от резервуара) открыта (*CompB* поступает в резервуар)

***ICONICS. Simulator.1\diplom. dip. bool. out2 = 0***, если задвижка *ValveB* (расположена справа от резервуара) закрыта (*CompB* не поступает в резервуар)

***Отгрузка = 1***, если отгрузочный насос включён; ***Отгрузка = 0***, если отгрузочный насос выключен.

Для реализации построенного алгоритма необходимо завести нужные переменные, приведённые ниже в табл.3:

Табл.3 Описание переменных, необходимых для создания дискретно-непрерывного процесса

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип | Описание |
| ***ICONICS. Simulator.1\diplom. dip. start*** | Boolean | Сигнал запуска процесса |
| ***~~tank~~*** | Double | Уровень в резервуаре (в диапазоне от 0 до 1000см)  |
| ***ICONICS. Simulator.1\diplom. dip. bool. out1*** | Boolean | Положение задвижки *ValveA* |
| ***ICONICS. Simulator.1\diplom. dip. bool. out2*** | Boolean | Положение задвижки *ValveB* |
| ***~~disch~~*** | Boolean | Положение отгрузочного насоса |

Для создания этих переменных будет использоваться OPC-сервер имитации сигналов *OPC Simulator* из программной группы ICONICS Genesis-32. Для данной мнемосхемы будет создана отдельная группа тегов, названная *diplom. dip*. В нужной группе создаются нужные теги и конфигурируются на усмотрение пользователю.

Рис.12 Добавление набора тегов в OPC-сервере имитации сигналов

Теперь, когда есть нужные переменные можно перейти непосредственно к созданию имитации заполнения/опустошения резервуара. Для этого нужно выделить изображение *Swtich On* и в панели **“Динамика"** выбрать свойство **“Указание и щелчок мышью".** После чего на экран монитора будет выведена диалоговая панель **“Инспектор свойств"** с выбранной закладкой **“Указание и щелчок".** В выпадающем списке **“Операция**" следует выбрать действие **“Передать значение".**

Рис.13 Внешний вид страницы **“Указание и щелчок”** диалоговой панели **“Инспектор свойств"** после выбора операции **“Передать значение”**

В выпадающем списке **“Тип”** следует выбрать “**С фиксацией”**. Если выбрать тип без фиксации, тогда элемент *Switch On* будет подавать 1 (или *True*) на зависимые от него переменные (под словом зависимые следует понимать то, что элемент *Switch on* управляет всей моделью, то есть если он включен, то процесс может работать, а если выключен то процесс остановится) только при нажатии левой кнопкой мыши. Получается, что весь процесс будет работать лишь в момент нажатия клавишей мышью *Switch On*. Поэтому следует выбрать тип **“С фиксацией"** (фиксация позволяет зафиксировать положение/значение элемента до тех пор, пока пользователь не изменит положение/значение этого элемента). Теперь если нажать левой клавишей мыши на элемент *Switch On*, то он станет 1 (или *True*) до тех пор, пока пользователь не остановит процесс, тем самым изменив *True* на *False*. В группе параметров **“Порядок выполнения”** отмечается флажок **“Пока нажато”**, после чего в поле **“Интервал"** устанавливается значение 50 мс (это значит, что в переменную будет записываться значение раз в 50 мс). Источником данных будет являться локальная переменная ***~~tank~~***. После настройки параметров остаётся ввести выражение, которое будет отображать описанный ваше алгоритм. Для этого в инспекторе свойств есть меню **“Редактор выражений”**.

При вызове редактора выражений будет выведена диалоговая панель, в которую и будет введёно выражение. Выражение выглядит так: ***X= (if (~~tank~~>1000,1000,~~tank~~ +***

***{{ICONICS. Simulator.1\diplom. dip. start}}\*{{ICONICS. Simulator.1\diplom. dip. bool. out1}}+{{ ICONICS. Simulator.1\diplom. dip. start}}\****

***{{ICONICS. Simulator.1\diplom. dip. bool. out2}} - {{ICONICS. Simulator.1\diplom. dip. start}} \* ~~disch~~)) \* (if (~~tank~~<0, 0,1))***

Далее следует установить соединение между положением переключателя *Switch On* и переменной ***ICONICS. Simulator.1\diplom. dip. start***. Для этого необходимо выбрать инструмент **“Указание и щелчок мышью".** В появившейся диалоговой панели, в списке **“Операция"** нужно выбрать **“Передать значение”** и в поле **“Значение"** поставить 1. Далее переключатель *Switch On* будет настроен таким образом, что в режиме исполнения он будет видимым только в случае, если дискретно-непрерывный процесс остановлен. Для этого нужно выделить элемент *Switch On* и выбрать инструмент **“Скрыть/блокировать"** в панели **“Динамика”**. На экран будет выведена диалоговая панель, которой в качестве источника данных будет переменная ***ICONICS. Simulator.1\diplom. dip. start*** и указываем параметр **“Скрыть/блокировать, если true”**. Получается, что если переменная ***ICONICS. Simulator.1\diplom. dip. start*** будет равна 1, то объект *Switch On* не будет видимым в режиме исполнения. Теперь, когда настроен элемент *Switch On*, нужно настроить другой элемент - *Switch Off*, который предназначен для остановки процесса. Переключатель *Switch Off* должен быть видимым только в случае, если процесс запущен. В то же время щелчок левой клавишей мыши на данном переключателе должен приводить к сбросу в 0 значение переменной ***~~tank~~***.

Выделяем объект *Switch Off* и используем инструмент **“Указание и щелчок мышью"** в панели **“Динамика”**. В появившемся окне нужно выбрать операцию Передать значение и в поле Порядок выполнения необходимо указать Если отпущено. Источником данных будет переменная ***ICONICS. Simulator.1\diplom. dip. start*** и в поле значение указываем 0. Снова выделяем объект *Switch Off* и выбираем инструмент **“Скрыть/блокировать"** в инструментальной панели **“Динамика”**. В инспекторе свойств с выбранной закладкой **“Скрыть"** нужно выполнить нужную настройку. Источником данных выбираем ***ICONICS. Simulator.1\diplom. dip. start*** и выбираем опцию **“Скрыть/блокировать, если False".** Теперь когда все настройки с элементами *Switch On* и Switch Off, следует наложить их друг на друга для динамического эффекта.

## 6.4 Создание элемента отображения уровня в резервуаре

Выбираем инструмент **“Численное отображение параметра"** в инструментальной панели **“Динамика”**. В появившейся диалоговой панели в поле Источник данных следует указать локальную переменную ***~~tank~~***. Теперь уровень в резервуаре будет отображаться не только графически, но и в текстовом виде.

## 6.5 Настройка анимации для изображения миксеров

Каждое изображение миксера в зависимости от значения уровня в резервуаре должно представлять одно из двух состояний: включён или выключен. Во включённом состоянии лопасть миксера должна вращаться. Данный визуальный эффект создаётся при помощи динамического действия **“Анимация".** Изображение лопасти каждого миксера, вставленного из библиотеки символов, состоит из графических объектов.

Рис.14 Составные части изображения лопасти миксера

Все эти элементы данного объекта в итоге будут представлять анимацию. Для начала следует выделить лопасть миксера и разгруппировать его при помощи инструмента **“Разгруппировать”** в инструментальной панели **“Расположение".** После данной операции необходимо выделить торец миксера и выдвинуть его на передний план. После этого нужно выделить всё изображение лопасти и выбрать инструмент **“Анимация"** в инструментальной панели **“Динамика”**. Во всплывшем окне инспектора свойств с выбранной закладкой **“Анимация"** нужно в редакторе выражений ввести следующее выражение: ***X= (~~tank~~>300) && {{ICONICS. Simulator.1\diplom. dip. start}}***. Благодаря данному выражению нижняя лопасть миксера будет вращаться, когда переменная ***~~tank~~*** (уровень в резервуаре) будет больше 300, но при условии, что переменная ***ICONICS. Simulator.1\diplom. dip. start*** будет равна 1 (то есть если процесс включён). Так же в инспекторе свойств следует отметить опцию **“Анимация, если true".** Порядок смены кадров будет равным 50 мс. Так же следует установить опцию **“Видимый, если выкл”** и **“Первый кадр, если выкл”**.

Рис.15 Настройка свойств динамического действия Анимация для изображения лопасти

Затем для упрощения процесса делаем ещё 2 копии настроенных лопастей и размещаем выше оригинальной лопасти. В итоге получится миксер, состоящий из 3 лопастей.

Рис.16 Размещение изображений лопастей на оси

Далее следует вызвать инспектор свойств средней лопасти с вкладкой **“Анимация"** и в выражении, находящемся в поле Источник данных изменить 300 на 500, в результате чего формула будет иметь следующий вид: ***X= (~~tank~~>500) && {{ICONICS. Simulator.1\diplom. dip. start}}***.

После этого нужно проделать аналогичную операцию с верхней лопастью, но 300 заменить на 700: ***X= (~~tank~~>700) && {{ICONICS. Simulator.1\diplom. dip. start}}***.

После всей проделанной работы в итоге получилась мнемосхема, показанная на рис.17.

Рис.17. Мнемосхема дискретно-непрерывного процесса

## Заключение

В данной дипломной работе речь шла не только о распространении автоматизации во многие сферы человеческой деятельности, но и о том, какую важную роль играет человеко-машинный интерфейс в этой автоматизации. Мною были представлены самые популярные SCADA-системы на современном рынке. Но детально описана была SCADA-система Genesis-32, так как именно в этой системе была сделана данная дипломная работа. Так же в ходе выполнения дипломной работы была создана модель дискретно-непрерывного процесса.

## Список литературы

1. www.prosoft.ru
2. www.asutp.ru
3. www.iconics.com
4. Джеф Раскин “Интерфейс: новые направления в проектировании компьютерных систем” (Символ-плюс, 2005).
5. Fastwel SD Dept “Руководство пользователя ICONICS GraphWorX32" (19.05.2000).
6. Fastwel SD Dept “Начальные сведения и ускоренное ознакомление с основными подсистемами ICONICS GraphWorX32" (10.06.1997)