Министерство Образования РФ
Пермский Государственный Технический Университет
Кафедра ИТАС

Методическое пособие по предмету

«Системы реального времени»

Составил:
ст. преподаватель
Артемов С.П.

Пермь, 2005

Основные задачи курса

В курсе «системы реального времени» рассматриваются следующие разделы:

1. Грамотное построение информационных систем.

1.1.Задачи проектирования:

1.1.1. Выбор средств для реализации системы.

1.1.2. Должны удовлетворяться требования, выдвигаемые заказчиком.

1.2. Особенности коммуникации узлов в системе.

1.3. Распределенный характер промышленной системы.

2. Обеспечение грамотного планирования систем реального времени (СРВ).

Анализ СРВ на возможность функционирования в надлежащем режиме. Конечная реализация строится по 3-4 вариантам.

3. Анализ классификации СРВ.

Классификация реализации систем реального времени

Реальное время – это время, которое удовлетворяет информационную систему в обслуживании внешних событий.

Основное применение СРВ находится в следующих областях:

1. Анализ протоколов передачи данных.

Анализатор протокола обеспечивает прием всех данных сети и анализ ошибок передачи данных. Существует программный и аппаратный анализатор протоколов.

2. Операционные СРВ. Система ориентирована на автоматизированные системы, в которых требуется своевременная адекватная реакция на события.

3. Применение в языках программирования. Языки СРВ предназначены для создания СРВ Assembler, C, Ada (спутниковые системы наблюдения).

4. Промышленные СРВ. Протоколы передачи данных АСУ: FieldBus, промышленные Ethernet.

5. SCADA-приложения. Предназначены для визуализации работы автоматизированной системы или автоматизированного объекта.

6. БД реального времени – базы данных, в которых предусмотрены функции с физическими данными, полученными в датчиках. Например, Industrial SQL.

7. SCADA-приложения и БД реального времени могут сформировать комплексные системы реального времени. Системы разрабатываются под ключ.

Классификация систем реального времени

1. По типу применения
2. По характеру работы

По типу применения различают:

- Универсальные.
- Специализированные.

СРВ называется специализированной, если она ориентирована на строго определенную задачу. Обычно применяются, где есть риск для человека. 70% - разработка, 30% - установка.

СРВ называется универсальной, если ее можно применять для различных задач автоматически. Человеческий фактор сведен к нулю. 90% - разработка, 10% - установка.

По характеру работы различают:

- Системы жесткого реального времени (hard) (HRT)

- Системы мягкого реального времени (soft) (SRT)

СЖРВ (система жесткого реального времени) – это СРВ, в которой невыполнение одной функции ведет к отказу всей системы. Система может функционировать в случае отказа «не важной» функции.

Система является СМРВ (система мягкого реального времени), если она не является СЖРВ (системой жесткого реального времени).

Среди СЖРВ и СМРВ выделяют следующие системы:

- истинные СРВ – это СЖРВ, в которых время ответа очень мало.
- Устойчивые СРВ – это СМРВ, в которых нет преимущества от смены операций между собой.

Средства разработки систем РВ

Средства разработки СРВ – это инструменты, позволяющие спроектировать СРВ на модельном объекте, отладить ее и перенести на реальный физический объект.

Проблемы проектирования систем

1. Распределенное управление. Сам объект систем автоматизации имеет несколько систем, требующих обеспечить взаимодействие между этими подсистемами. Подсистем может быть несколько, и одна подсистема может иметь несколько устройств управления. 1 устройство – 1 датчик.

2. Обеспечение реального масштаба времени.

Минимальный такт для СРВ – 20 мс. Минимальный цикл для СРВ – 20 мс.

3. Обеспечение заданного времени отклика на запрос.

Выделяют задачи:

1. Задача управления временем. Наличие таймеров программных и аппаратных, наличие функций контроля за выполнением операций. Время – это наивысший ресурс.

2. Планирование процесса выполнения задач. Построение очереди исполнения функций в автоматизированной системе. Данная последовательность реализуется в объекте, называемом планировщик.

3. Обеспечение коммуникации узлов в распределенных сетях. Сетевой планировщик задач.

4. Обеспечение логической корректности выполнения задач.

Построение алгоритма системы

Система называется системой реального времени (СРВ), если правильность ее функционирования зависит не только от логической корректности вычислений, но и от времени, за которое эти вычисления выполняются.

Основной задачей СРВ является получение надлежащих результатов за определенный кратчайший срок.

С точки зрения пользователя необходимо контролировать два параметра:

1. Логическая корректность вычислений.
2. Время возникновения событий в системе.

Организация систем реального времени

СРВ состоит из трех подсистем:

1. Контролируемая.
2. Контролирующая.
3. Операционная.

Между этими подсистемами существуют интерфейсы:

1-2 – интерфейс приложения,

2-3 – машинный интерфейс.

Контролируемая подсистема диктует требования в реальном масштабе времени и выдает основные характеристики объекта управления.

Контролирующая подсистема управляет вычислениями, управляет связью с внешним оборудованием.

Операционная подсистема обеспечивает связь с оператором. Контролирует полную деятельность системы.

Интерфейс приложения реализуется с помощью датчиков и исполнителя элементов.

Машинный интерфейс обеспечивает связь конечных устройств информационной системы с подсистемой визуализации оператора.

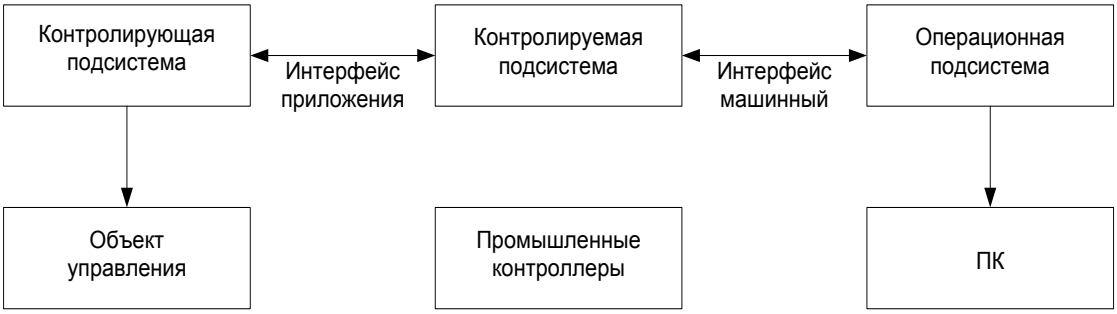

Рис.1.

Контролирующая подсистема должна обеспечивать распределение ресурсов таких как память, доступ к сети, устройство длительного хранения информации.

Любая контролирующая подсистема, кроме обеспечения интерфейса операционной системы (ОС), должна обеспечивать контроль записи на внешние устройства.

Место систем РВ в информационных системах

Сложность разработки связана с требованиями надежности и безопасности. Например, надежность – температура, осадки, ветер.

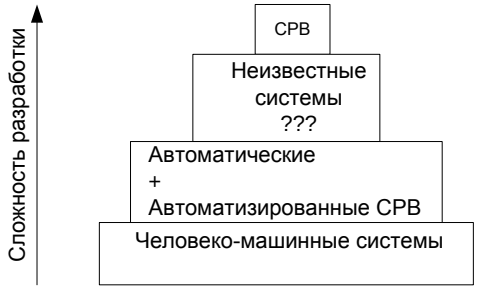

Рис.2.

Для СРВ требуется обеспечение вероятности отказа системы 10^{-10} в час. Данная вероятность обеспечивается на самолетах. На российских самолетах – 10^{-9}. в системах полета спутника – 10^{-7} часа.

Для СРВ выдвигается требование по работоспособности, позволяющее при отказе узла произвести восстановление его функций и обеспечить вероятность отказа на уровне 10^{-6}-10^{-4} часа.

Требования к системам реального времени

1. Требование по времени выполнения задач и функционированию систем предъявляется к следующим компонентам системы:

1.1. период функционирования;

1.2. крайний критический срок выполнения;

1.3. время выполнения;

1.4. суммарное время продолжительности выполнения задач (зависит от наличия ошибок).

2. Требование о возможности параллельного выполнения нескольких задач (возможность построения алгоритма псевдопараллельного исполнения задач).

3. Предсказуемость.

4. Учет максимального времени отклика на события (а не среднего, как в обычных системах).

5. Особые требования в вопросах безопасности систем (защита от постороннего вмешательства в алгоритм системы).

6. Возможность безотказной работы в течение длительного периода времени (включаются требования по обслуживанию в СРВ).

Для СРВ требуется проведение регламентных работ после 16 часов непрерывной работы.

Общие характеристики систем реального времени

Большие и сложные системы (разрабатываются больше года, поэтому алгоритмы функционирования могут быть сложными).

1. Распределенные системы.

2. Взаимодействие с окружающим миром. Жесткое взаимодействие с аппаратурой. Четкий контроль всех операций.

3. Выполнение задач зависит от времени (как и последовательность выполнения задач, так и время исполнения).

4. Сложность тестирования.

Так как в СРВ включается много различных функций и для тестирования СРВ требуется создание групп, не занимающихся разработкой этой системы, но понимающих ее функционирование.

Данные характеристики позволяют выявить множество систем РВ, функционирующих в реальном мире.

Задачи СРВ

СРВ представляют собой набор взаимодействующих между собой заданий или задач.

Задача является одиночным объектом, управление которым осуществляется оболочкой СРВ.

В зависимости от количества задач и от их вида определяется время функционирования СРВ.

Задачи классифицируют по двум категориям:

1. Требование по времени функционирования:

1.1. задачи в ЖРВ (жестком реальном времени);

1.2. задачи в МРВ (мягком реальном времени);

1.3. задачи в «нереальном времени».

2. Вид или тип функционирования:

2.1. периодические задачи;

2.2. апериодические задачи (асинхронные);

2.3. спорадические задачи;

2.4. фоновые задачи;

2.5. аппендикс.

Задача ЖРВ – это задача, чье логически правильное или своевременное исполнение считается критическим для действия всей системы.

Предельный срок исполнения называется жестким сроком исполнения. Неспособность удовлетворять этому требованию ведет к отказу всей системы.

Задача МРВ – это задача, в которой исполнение не критично по времени, но ее исполнение желательно для системы (предельный срок исполнения – мягкий крайний срок исполнения задается диапазоном).

Задача «нереального времени» - это задача, для которой нет требований по своевременному выполнению.

Периодические задачи

Периодические задачи – это задачи, которые переходят в состояние выполнения через строго заданный период и выполняются каждый цикл функционирования в системе. Например, обработка и контроль сигнала.

Для СРВ требуется четкое и своевременное выполнение каждой периодической задачи (см. рис.3).

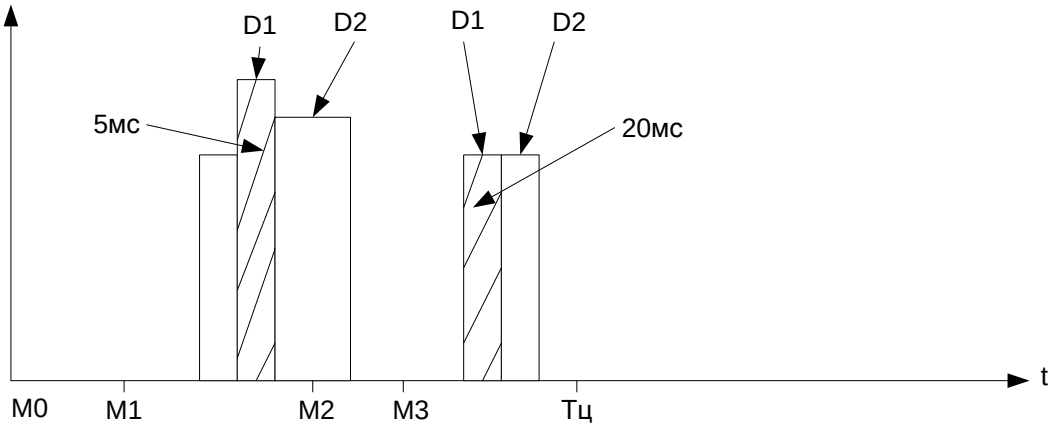

Рис.3.

$T_ц$ – время цикла (единица измерения в СРВ). Цикл делится на несколько групп (метки).

Периодическая задача выполняется в строго отведенное ей время, каждый цикл. Запуск периодической задачи может осуществляться несколько раз за цикл в зависимости от количества меток (сколько меток, столько раз можно запускать цикл). Характеризуется жестким крайним сроком исполнения.

Апериодические задачи

Апериодические задачи – это задачи, имеющие минимальный приоритет в системе и выполняющиеся по событию. Характеризуются наличием мягкого крайнего срока исполнения.

Функционирование осуществляется только в том случае, если периодические задачи не выполняются.

К функциям апериодических задач относятся функции диагностики, выдача справочной информации и сохранение информации на внешнем носителе.

Спорадические задачи

Спорадические задачи – это апериодические задачи с жестким крайним сроком исполнения.

Приоритет устанавливается на уровне периодических задач. Спорадические задачи имеют непредсказуемый характер.

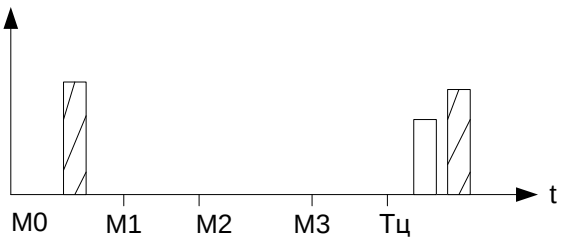

Рис.4.

Для обработки выделяется отдельная периодическая задача, которая будет контролировать выполнение.

Фоновые задачи

Фоновые задачи – это задачи, для которых предельный срок исполнения не задается, либо устанавливается мягкий крайний срок исполнения.

Функционируют в конце каждой метки и только при условии простоя вычислительного узла (при отсутствии других задач).

Может исполняться несколько циклов функционирования системы.

Задачи аппендиксы

Задачи аппендиксы – это задачи, которые исполняются до старта ОС и имеют приоритет выше, чем сама ОС.

Данные задачи связаны с доступом к аппаратуре, например, установка триггеров, регистров и временных меток.

Планирование задач

Планирование задач – алгоритм построения очереди задач на выполнение.

Алгоритмы:

- статические

- динамические

Статические алгоритмы основаны на применении основных характеристик задач и подразумевают построение примерного плана их исполнения.

Достоинства:

1. Если система предсказуема на первом шаге, то она будет предсказуема на всех других.

2. Система может пойти неверно, если последовательность построена неверно.

Недостатки:

1. Использование в каждом цикле исполнения задачи одной и той же последовательности задач.

2. Изменение очередности исполнения задач не допускается.

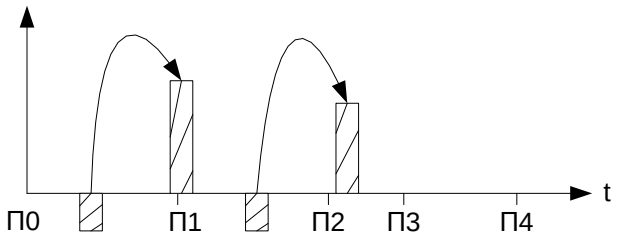

Рис.5.

В результате действия алгоритма существует вероятность накопления суммарной «нехватки времени». В системе могут накопиться остатки неиспользованного времени.

Динамические алгоритмы планирования предназначены для изменения последовательности задач во время функционирования системы. Изменение последовательности задач происходит перед новым тактом и требует от вычислительного узла дополнительных ресурсов для пересчета последовательности задач. В отличие от

статического планирования, динамическое позволяет адаптировать систему к текущему состоянию.

Достоинства:

1. Оптимальное распределение временных участков подзадач.

2. Возможность дополнения списка задач в процессе функционирования системы.

Недостатки:

1. Сложность реализации алгоритмов.

2. Повышенные требования к вычислительному узлу.

3. Предсказуемость системы зависит от алгоритма на каждом этапе функционирования.

Задачи динамического планирования реализуются в виде аппендикса. Далее запускается операционная система, потом периодические, апериодические, спорадические задачи, затем фоновые задачи.

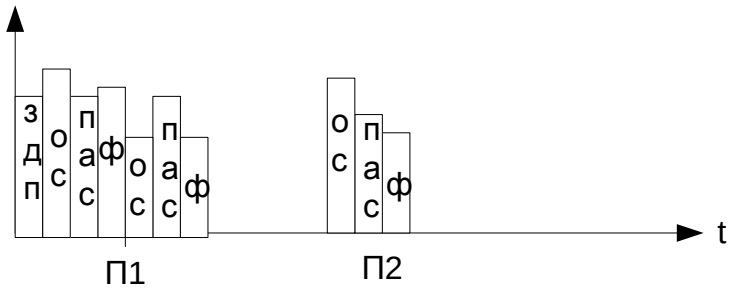

Рис.6.

Существует множество алгоритмов планирования задач. Рассмотрим 7 алгоритмов.

Планирование периодических задач

Планирование периодических задач связано с разработкой последовательности построения задач, выполняемых на одном вычислительном узле.

Есть два подхода к построению:

1. Фиксированный приоритет задач. Приоритет вычисляется один раз до запуска системы и остаётся неизменным в течение цикла функционирования задач.

2. Динамически назначаемый приоритет. Приоритет может быть установлен во время функционирования задач.

Назначение динамического приоритета производится крайним сроком исполнения задачи. В связи с этим были разработаны группы планирования:

1. Алгоритмы планирования задач с фиксированным приоритетом.

2. Вытесняющие алгоритмы планирования задач (подразумевает возможность вытеснения одной задачи другой, в зависимости от приоритета).

Существует три основных алгоритма планирования:

- RM
- EDF
- LSTF

RM (алгоритм с фиксированным приоритетом)

Приоритет задачи назначается согласно следующему принципу: чем меньше периодическая задача, тем больше приоритет. Данный алгоритм всегда формирует оптимальную последовательность задач, если это возможно.

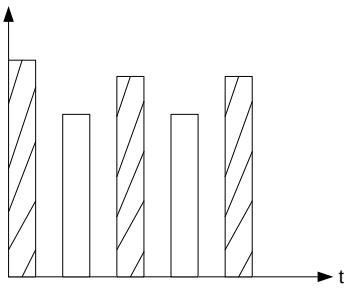

Рис.7.

Чем реже вызывается задача, тем выше у нее приоритет.

EDF (алгоритм с динамическим планированием задач)

Приоритет назначается согласно следующему принципу: чем меньше срок выполнения, тем выше приоритет.

В каждый цикл задачи последовательности выстраиваются заново в зависимости от критического срока выполнения. Реализованный алгоритм зависит от количества задач в определенный момент времени.

LSTF (алгоритм планирования)

Приоритет задачи назначается согласно следующему принципу: чем меньше время связывания задачи, тем выше приоритет.

$\Delta t \downarrow \quad p \uparrow$

$\Delta t = D - t$

t – физическая величина, время выполнения задачи, задается пользователем.

Свойства задач:

Задача – объект, который имеет метод, выполняемый в системе реального времени.

Задача – это единица измерения объектов системы исполнения реального времени.

1. Тип задачи (P, A, S, F, O).

P периодические

A апериодические

S спорадические

F фоновые

O аппендиксы

2. T – период.

3. t – время выполнения.

4. Критический крайний срок D исполнения.

5. tв – время начала функционирования задачи; определяет такт времени, в который задача начинает функционировать каждый цикл.

Приоритет зависит от алгоритма планирования и является произвольным.

Время связывания зависит от периода, типа задачи, крайнего критического срока исполнения.

Алгоритмы планирования спорадических и апериодических задач

Существует пять основных алгоритмов планирования спорадических задач:

1. а) Планирование спорадической задачи, как фоновой задачи.

Выделяется отдельная фоновая задача, которая отвечает за выполнение всех спорадических запросов. Поэтому все спорадические апериодические задачи исполняются тогда, когда не исполняются периодические задачи.

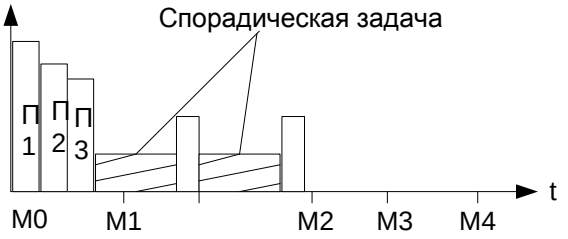

Рис.8.

Создание отдельной задачи не является обязательным.

б) Спорадическая задача как фоновая (без создания дополнительного процесса).

В а) – выделяют фиксированное время, а не любое свободное.

В б) – задача будет иметь приоритет общий с другими фоновыми задачами и ставится в очередь фоновых задач.

2. «Политика выбора».

Заключается в создании периодического процесса, характеризующегося установленным приоритетом. Данный процесс отвечает за обслуживание всех запросов от спорадических и апериодических задач.

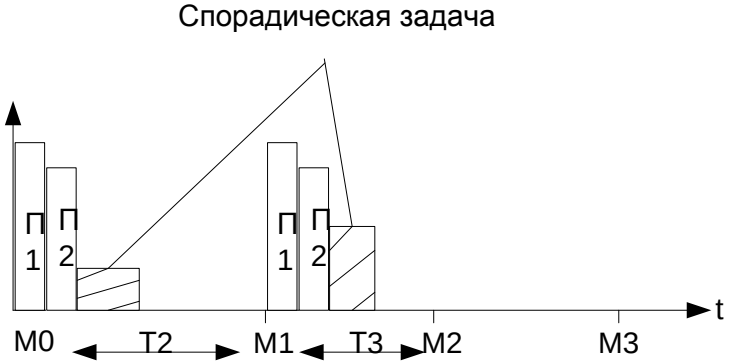

Рис.9.

Данная задача должна запускаться в строго установленное время запуска ($T_з$).

Недостатки: несовместимость циклического характера алгоритма и случайного характера спорадических задач.

Достоинства: позволяет четко спланировать время исполнения всех задач (самый определяющий алгоритм).

3. Обмен приоритетом.

Заключается в создании отдельного процесса обслуживания спорадической задачи с динамическим приоритетом.

Динамическое назначение приоритета позволяет изменять его в процессе функционирования системы.

В данном алгоритме для процесса, исполняющего спорадические задачи, устанавливается самый высокий приоритет (динамический). Система обменивает приоритеты между самым высокоприоритетным периодическим процессом и процессом, обслуживающим спорадические задачи. Обмен производится в начале цикла функционирования системы.

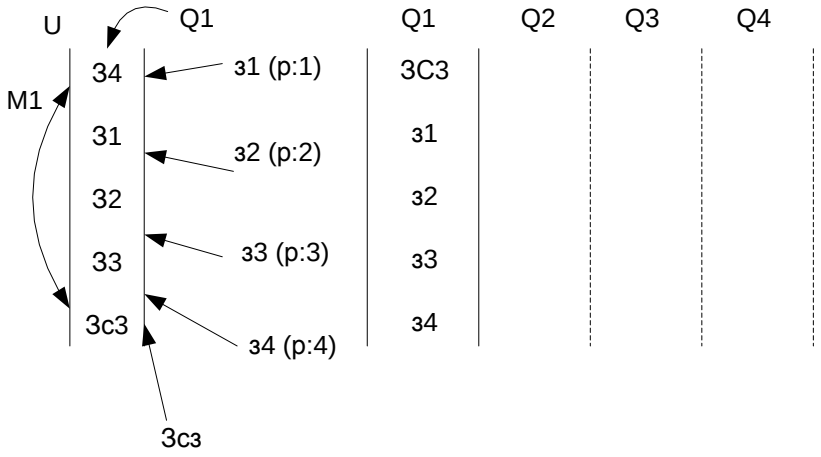

Рис.10.

Зсз – задача спорадических задач

з1…з4 – задачи

р – приоритет

М1 – метка (Q1)

Q_1^1 (1 цикл)

Q_1^2 (2 цикл)

Если в систему поступает периодическая задача с более высоким приоритетом во время между Q_1^1 и Q_1^2, то обмен приоритетом не производится, и данная задача может быть выполнена раньше, чем задача, обрабатывающая спорадические запросы.

4. Деферабельный сервер.

Основан на создании процесса обработки спорадических задач с четко установленным заданным приоритетом. Приоритет устанавливается на самом высоком уровне до запуска системы. Перед повторным запуском данный приоритет может изменить приоритет на самый высокий в текущей системе (см. рис.10.).

Зсз (p:max(p)+1)

В данном алгоритме процесс сохраняет ресурс, выделенный для обслуживания спорадических задач, и приостанавливает выполнение спорадических задач в случае, если этот ресурс исчерпан.

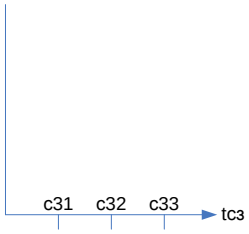

Рис.11.

tсз = $t_{сз}^1 + t_{сз}^2 + t_{сз}^3 + t_0$

Пока t_0>0 производится запуск спорадической задачи, иначе спорадическая задача не запускается. В процессе функционирования системы данная задача может прерывать периодические задачи несколько раз в зависимости от получения спорадических запросов.

5. Спорадический сервер.

Алгоритм заключается в создании периодического процесса исполнения спорадических задач, но приоритет этого процесса устанавливается на уровне приоритета спорадической задачи. При

поступлении спорадической задачи оценивается приоритет текущей задачи. Спорадическая задача устанавливается в очередь периодических задач, если ее приоритет меньше, чем у периодических задач.

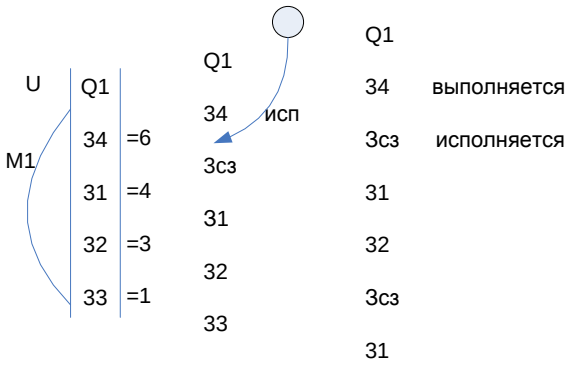

Рис.12.

Сама задача времени исполнения должна иметь наивысший приоритет. Данный сервер включает лишь одну задачу – изменение очереди задач.

Каждый алгоритм можно оценить с точки зрения производительности. Для оценки производительности используются 3 параметра:

1. Нагрузка системы на отказ (в обслуживании).

Breakdown Utilization (BU).

2. Нормализованное среднее время ответа (NMRT).
3. Гарантируемая скорость обработки задач (GR).

Нагрузка системы на отказ (в обслуживании) (BU) является степенью использования ресурсов, при которой система может гарантировать, что все задачи будут выполнены в заданные сроки. Чем больше значение BU, тем больше время процессора, в котором выполняются задачи.

Нормализованное среднее время ответа (NMRT) - это отношение между интервалами времени от готовности задачи к выполнению до ее

окончания к фактическому времени процессора. Используется для выполнения задач.

$$\frac{t_{\text{овз}} - t_{\text{нвз}}}{t_{\text{обр}}}$$

Чем больше NMRT, тем больше время простоя задач.

Гарантируемая скорость обработки задач (GR) является оценкой производительности системы для задач. Вычисляется как отношение определенного числа задач, выполнение которых можно гарантировать, к общему числу задач, ожидающих выполнения.

Если число больше 1, то система расписабельная, иначе – нерасписабельная. Чем больше это число, тем больше запас времени для выполнения задач.

Планировщик заданий

Планировщик заданий – это средство, которое предназначено для исполнения задач на вычислительном узле.

Планировщики бывают разных видов:

1. Глобальный (общий) планировщик – во-первых, распределяет задачи между несколькими вычислительными узлами в распределённой вычислительной системе. Обычно реализуется на узле типа server, либо на одном из узлов системы при децентрализованном управлении. Во-вторых, создаёт алгоритмы формирования образа узлов.

2. Местный планировщик – распределяет задачи на одном вычислительном узле за заданный цикл функционирования. Работа планировщиков на различных узлах является независимой.

Местный планировщик для глобального является обыкновенной задачей с повышенным приоритетом.

Планировщик заданий определяет:

1. Последовательность выполнения задач.

Каждый цикл функционирования узла планировщик может определять новую последовательность задач.

2. Распределение ресурсов между задачами.

Служит для борьбы с гонками.

Гонки – это ситуация по захвату доступа к ресурсу задачей с маленьким приоритетом.

3. Распределение времени между задачами.

Выделение заданного количества тиков для задачи, исполняющейся на узле.

Время – наиболее актуальный параметр.

Тик – минимальная измеряемая единица времени.

Количество тиков зависит от разных параметров:

1. частота процессора;

2. время одного цикла.

Алгоритм функционирования планировщика

Планировщик является частью операционной системы.

Из всех задач строится таблица запуска. Определяются списки задач по их виду (см. типы задач). В зависимости от типа задачи в списке задач устанавливаются параметры групп запуска.

Для периодических и фоновых задач должны быть установлены следующие параметры:

- стартовая метка запуска данной группы задач;
- период запуска данной группы задач;
- крайний критический срок исполнения.

Имя группы	Стартовая метка	Крайний критич. срок исполнения	Период
ptl0 ptl1	0 5	5 5	1 1

ptl2	10	5	1

ptl0 <список задач>

Для фоновых задач эта таблица расширится относительно стартовой метки, то есть расширится диапазон запуска. Также расширяется крайний критический срок исполнения (min и max).

Имя группы	Старт. метка min	Старт. метка max	Край. крит. срок исп. min	Край. крит. срок исп. max	Период

Для апериодических и спорадических задач также создаются таблицы, но они включают одну строку, если приоритет данного типа задач не учитывается.

Апериодические задачи: apt <список задач>

метка не описывается

Спорадические задачи: spt <список задач> запуск задач

Если задачи необходимо делить по приоритетам, то в таблице апериодических и спорадических задач необходимо указывать различные метки для различных приоритетов.

Задачи-аппендиксы описываются в виде таблицы, включающей 1 параметр – метку запуска.

Имя группы	Метка запуска
app0	0
app1	5

app1 <список задач>

Анализ таблиц

На основании этих таблиц строится список задач на каждом цикле исполнения.

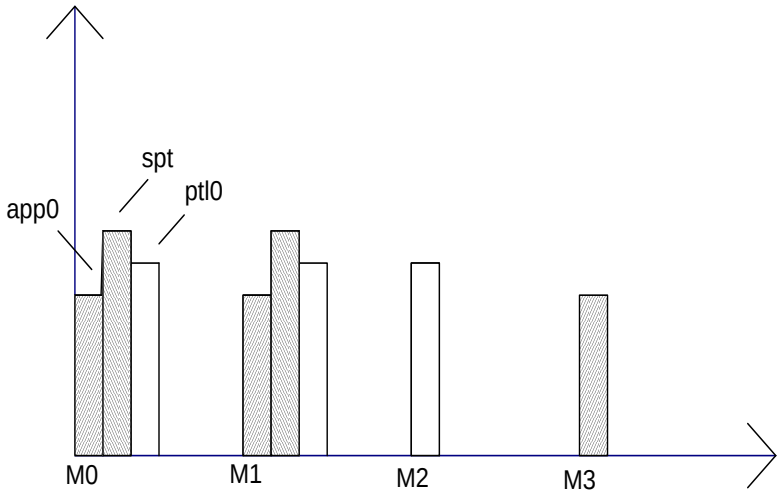

Рис.13.

Для аппендиксов, периодических и фоновых задач № метки должен совпадать с тем, который поставлен в таблице для построения списка.

Список задач может изменяться или не изменяться. Изменение зависит от типового алгоритма планирования.

Исполнение планировщика (запуск):

Завершающая стадия.

1. Создаётся бесконечный цикл – это планировщик заданий.

test_tabl // если планирование статическое

for (;;) // бесконечный цикл

{

2. Если планирование динамическое, то анализ таблицы в начале запуска test_tabl

3. После анализа таблицы построили списки задач.

// списки задач (4 метки)

4 период. + 4 апериод. +4 спорадич. + 1 фонов. + 1 аппенд.

<1…<14 (14 списков)

4. Следующей задачей будет вызов этих функций по данному списку.

Исполнение – вызов функции, которой является задача.

L1[1].exec.

}

Последовательность выполнения в пределах одной метки определяется структурой таблицы.

После вызова каждой функции проверяется количество элементов в списке, и в зависимости от количества элементов происходит переход на следующую метку либо исполнение функций из текущей метки.

Примечание 1: Планировщик во время запуска задач обязан контролировать занятость ресурса текущей задачи, а также должен осуществлять контроль временных характеристик задач, указанных в таблице.

Примечание 2: Если временные характеристики нарушены, должно сформироваться отказное состояние системы.

Классификация приложений систем реального времени

Основной параметр – предсказуемость.

Две парадигмы приложений для предсказуемости систем:

1. Архитектура приложения, работающего по событию. (ET – Event Type).

2. Архитектура приложения, функционирующего по времени. (TT – Time Type).

ET – подход

Любая деятельность системы начинается в ответ на возникающее специфическое событие. Вид события определяется самой системой. Предсказуемость достигается следующими способами:

1. Использование стратегии оценки для каждой прикладной задачи (оценивается потребность данной задачи в текущий момент времени). Проверяется несколькими способами:

а) Проверяется загруженность узла.

б) Проверяется время предыдущего запуска данной задачи и анализируется эффект от невыполнения задачи. При отрицательном эффекте задача обязательно должна быть запущена.

2. Оценка потребности ресурсов для данной задачи.

3. Оценка готовности ресурсов для удовлетворения потребностей и задач.

Достоинства: управляемость со стороны системы, независимость от времени (количества тактов).

Недостатки: сложность алгоритма оценки, отсутствие возможности синхронизации событий на разных узлах.

TT – подход

Деятельность системы начинается в определенный заданный момент глобально синхронизированного времени. Предсказуемость достигается путём приведения всех задач к периодическим. Для апериодических, спорадических и фоновых задач создаются мета-задачи, которые занимаются обработкой соответствующего типа задач.

Достоинства:

1. На всех узлах задачи могут исполняться по синхронизированному времени.

2. Таблица задач является фиксированной, для неё можно провести моделирование на возможность функционирования в режиме РВ.

Недостатки: слабая управляемость процесса исполнения задач. Не существует возможности управления последовательностью задач в процессе функционирования системы.

Моделирование систем реального времени

Моделирование систем реального времени необходимо для оценки разрабатываемой системы по временным характеристикам.

Основная временная характеристика – максимальное время задержки исполнения.

Рассмотрим пример: торможение машины

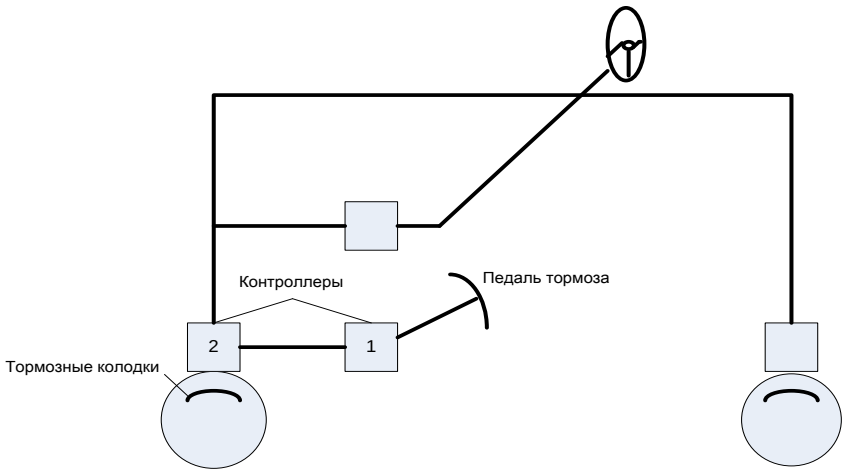

Рис.14.

I способ: обеспечение механической связи

II способ: обеспечение электрической связи. На каждый объект ставятся контроллеры, объединённые в сеть. Необходимо оценить время, за которое человек, нажавший на педаль тормоза, получит желаемый результат.

$\sum t = t_{\text{реакции человека}} + t_{\text{нажатия на педаль тормоза}} + t_{\text{обработки сигнала узлом 1}} + t_{\text{передачи информации}} + t_{\text{обработки сигнала узлом 2}} + t_{\text{выдачи управляющего воздействия}} + t_{\text{срабатывания}}$

$t_{\text{передачи информации}} = t_1 + t_2 + t_3 + t_4 + t_5$

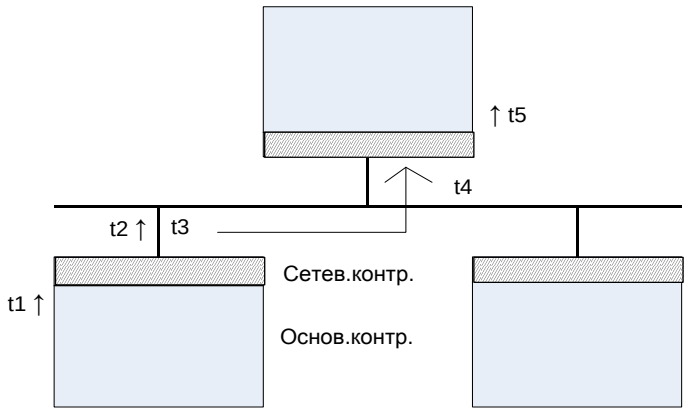

Рис.15.

Сеть состоит из основного и сетевого контроллеров.

t_1 – передаёт информацию от сетевого контроллера к основному.

t_2 – передаёт информацию от основного контроллера к шине.

t_3 – возникает разброс передачи сообщений в сеть из-за использования 1 канала.

t_4 – передача информации на сетевой узел.

t_5 – передача информации с сетевого контроллера узла 2 к основному контроллеру.

Задача моделирования – определение максимального времени передачи информации в распределенной системе управления от одного узла к другому.

Основным правилом при моделировании системы является гарантированное время передачи информации.

Гарантированное время передачи информации – это время, за которое система выполняет действие вне зависимости от внешних факторов.

При моделировании система делится на две модели:

1. Модель функционирования типового узла.
2. Модель распределенной системы управления.

Модель одиночного (типового) узла является детерминированной моделью (моделью с известным поведением). Следовательно, всегда можно получить время выполнения операции на данном узле. В системе реального времени это время является фиксированным и называется временем исполнения.

Существуют два способа передачи информации по сети:

1. Передача виртуального маркера (Token Ring)

2. Случайный доступ к среде с устранением коллизий (Ethernet, Cisma/CA)

Передача виртуального маркера

В данном случае система имеет детерминированное поведение, зависящее от скорости передачи информации и количества сообщений.

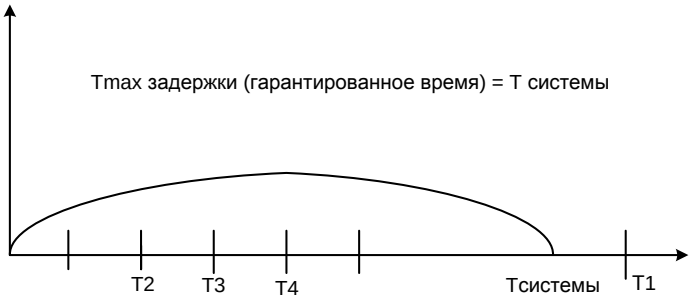

Рис.16.

Это время будет гарантированным при условии, что все процессы и все задачи являются периодическими. Задержка может возникнуть для апериодических запросов.

Проблема моделирования сетей при случайном доступе

При доступе узла к среде непосредственно при передаче информации проверяется занятость канала. Если канал занят, то сообщение ставится в очередь в соответствии с его приоритетом и отправляется только при освобождении канала (параметр t_3). Этот параметр называется **jitter** и является разбросом постановки сообщения в очередь. Параметр определяется пользователем, для каждого сообщения выбирается он отдельно.

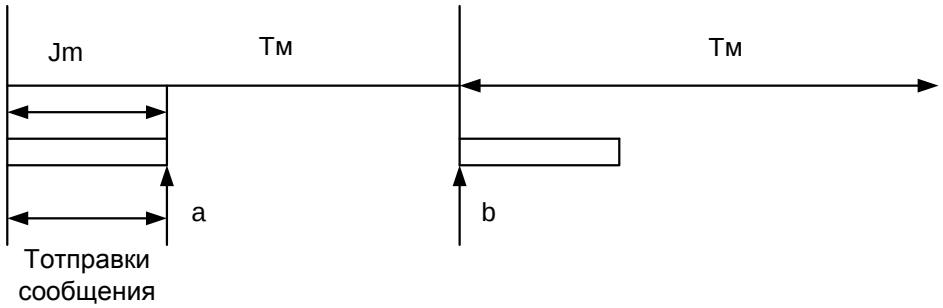

Рис.17.

$T_з = T_м - T_{отпр.сообщ}$, где $T_м$ – время передачи,

$J_m = T_{отпр.сообщ.}$ или $J_m \neq T_{отпр.сообщ.}$

J_m определяет возможность отправки сообщения в заданный цикл.

$J_m + T_{отпр.сообщ.} < T_м$

Разброс передачи сообщения в системах со случайным доступом необходим для обеспечения гарантированного времени передачи.

D_m – крайний критический срок исполнения. Он определяет время, до которого текущая передача сообщения является актуальной. При превышении D_m сообщение должно быть отброшено.

Суммарно для систем со случайным доступом требуется получение неблагоприятного времени ответа R_m, то есть времени по истечении которого система не может гарантировать функционирование в режиме РВ.

$R_m = J_m + W_m + C_m$, где m – сообщение.

Если $R_m^{тек} > R_m^{расчет}$, то невозможно функционировать в системе РВ.

C_m – максимальное время физической посылки сообщения m по шине.

C_m включает:

1. $t_{перед.дан.}$ – время передачи информации по шине (полезная информация).
2. $t_{перед.наклад.расходов}$ – время передачи накладных расходов.

Накладные расходы – это биты информации, необходимые для передачи сообщения в защищённом виде, для указания источников и приёмников информации, CRC-кода и информации для аутентификации.

3. $t_{перед.дополн.данных}$ - время передачи дополнительных данных.

Дополнительные данные – это информация, определяемая протоколом передачи данных. Служит для передачи основной и накладной информации.

$$C_m = ([\frac{t_{н.р.} + 8S_m}{t_{д.д.}}] + t_{н.р.} + t_{д.д.} + 8S_m) * \tau_{bit}$$

S_m – представляет размер сообщения в байтах.

τ_{bit} – такт передачи шины.

Соответственно этот параметр зависит от протокола передачи информации, скорости передачи и длины сообщения.

W_m – задержка организации очереди сообщений.

,

где hp(m) – множество сообщений с приоритетом более высоким, чем у сообщения m;

lp(m) – множество сообщений с приоритетом более низким, чем у текущего сообщения.

$W_m = W_0 = 0$

$W_1=1$

$W_2=2 = W_1+E$

Wm является рекурсивным, так как при учёте задержек следующих сообщений необходимо учитывать задержки, полученные в результате формирования предыдущих сообщений.

$$W_m = B_m + \sum_{\forall j \in hp(m)} [\frac{W_m + J_j + \tau_{bit}}{T_j}] * C_j$$

Если у сообщения такой же приоритет, то помещается в hp(m).

$B_m = \max (C_k)$

$\forall$ k $\in$ lp(m), где C_k – время физической посылки сообщения.

B_m – максимальное время, на которое может задерживаться текущее сообщение более низкими по приоритету сообщениями.

В момент передачи более высокоприоритетного сообщения шина может быть занята сообщением с более низким приоритетом. Для передачи текущего сообщения требуется ожидание, равное времени передачи сообщения.

Если lp(m) $\rightarrow \infty$, то B_m = 130т – это наихудшее время, за которое должно отправиться текущее сообщение.

T_j – период передачи сообщения.

$$W_m^{n+1} = B_m + \sum_{\forall j \in hp(m)} [\frac{W^n_m + J_j + \tau_{bit}}{T_j}] * C_j$$

Для получения результата: W_m^0 = 0 нет никаких задержек.

Для высокоприоритетного сообщения: $W_1 = B_m$ = 130т

Приведённый алгоритм не учитывает возможность ошибок при передаче информации по сети.

Для эффективности использования данного алгоритма рекомендуется назначать приоритеты сообщений по принципу D - J, R_m.

Алгоритм оценки

Позволяет понять: работоспособна система или нет.

Если выполняется неравенство R<D-J, то система является устойчивой для функционирования в режиме РВ. Расчёт должен проводиться для разных скоростей передачи информации по шине. Реальная скорость зависит от типа принимаемого протокола:

Rm : 125, 250, 500, 1000 кб/сек

При наличии ошибок при передаче информации формула расчёта должна учитывать функцию ошибки:

$$W_m = B_m + \sum_{\forall j \in hp(m)} [\frac{W_m + J_j + \tau_{bit}}{T_j}] + E_m(W_m + C_m)$$

E_m – функция максимального времени восстановления системы после появления ошибки.

$$E_m(t) = (n_{er} + [\frac{t}{T_{er}}] - 1) * (M + \max_{\forall k \in hp(m) \cup m} (C_k)$$

n_{er} – число ошибок, которые могут произойти в заданном интервале времени (t).

Ter – остаточный период ошибки. Это время, показывающее интервал, до которого новые ошибки появиться не могут.

$\frac{t}{T_{er}}$ – общее количество ошибок за время t.

M – это количество данных, потерянных в результате ошибки.

max (c_k) – максимальная задержка.

Если n=4, T=10мсек при V=1Мбит/сек, то частота появления ошибок=1бит/10000 (1 ошибка за 10000 переданных битов)

Оценка разработанной модели производится по 3 параметрам:

1. Коэффициент использования сообщения показывает отношение числа полезной переданной информации к общему числу переданной информации.

$К_{исп.сооб.}=n_{полезн}/n_{общ.}$

5бит/15бит*100%=0,33%

2. Коэффициент использования шины показывает отношение числа переданной информации, включая накладные расходы, к числу всех возможных интервалов для передачи данных.

$К_{исп.шины}=n_{перед.интер.}/n_{возм.интер.}$

3. Коэффициент расписабельности – это число, показывающее во сколько раз может быть увеличен период передачи всех сообщений, при котором система продолжает функционировать в режиме РВ.

Измеряется в пределах: A= [1..∞)

При A=1 наращивание мощности шины невозможно.

При A<1 система является нерасписабельной, то есть не может функционировать в режиме РВ.

$A=1 \rightarrow К_{исп.шины} = 100\%$

$A<1 \rightarrow К_{исп.шины} > 100\%$

$$A = \frac{\sum_m (T_н + D_m + J_m)}{\sum_m R_m}$$

Если A<1, то для системы должны применяться методы оптимизации.

Оптимизация системы сообщений

1. Можно оптимизировать структуру системы, разделив глобальные функции на подсистемы и оставив для глобальных сообщений магистральный участок.

2. Для существующей структуры применяются методы стохастического программирования, позволяющие оптимизировать последовательность передачи сообщений.

3. Метод объединения сообщений.

Все пакеты, имеющие одинаковый период, объединяются в единое сообщение. Это делается для сокращения накладных расходов.

При объединении сообщений нарушается порядок передачи данных. Если с учётом крайнего критического срока получается превышение для общего сообщения, то объединение сообщений должно быть отменено.

Примечание: коэффициент расписабельности определяет степень использования времени функционирования системы. Определяется отношением времени, которое можно использовать для исполнения ко времени, которое используется.

Основная цель проведения моделирования - оценка распределенных систем.

Для нераспределенных систем данная модель не подходит (используется транспортная модель). Сам расчет зависит от первоначально построенной модели, а также от грамотно спроектированного проекта. На стадии проектирования системы реального времени необходимо определить ряд параметров:

Рис. 18.

1. Любая подсистема должна представлять отдельный вычислительный узел.

2. Необходимо определить коммуникацию между подсистемами; это означает построение таблицы всех сообщений системы.

№ сообщения	Источник	Приемник	Размер	Тип сообщения

Таблица дополняется типом сообщения, определяется его характер.

Определяется “возможный” период появления сообщений. Период всех спорадических сообщений является равным для всех сообщений и определяется как средний период всех сообщений: Тс.с. = среднее (Тп.с.)

3. Производится анализ системы передачи данных, использующихся в системе.

Необходимо знать базовые характеристики:

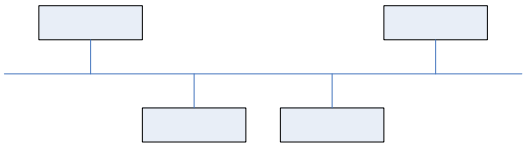

Рис. 19.

1. Скорость передачи данных.

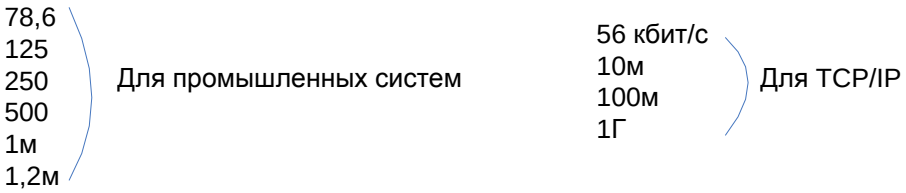

Рис. 20.

2. Накладные расходы на передачу данных. Дополнительные биты данных, используемые для кодирования сообщения, для

формирования источника и приемника, также анализируется количество данных, необходимых для восстановления системы.

3. IEEE определяет для всех протоколов свои характеристики. Например, для протокола CAN: 34 бит + 5 бит. Самые большие накладные расходы у протокола TCP/IP.

4. Расчет.

1. Определяются приоритеты каждого сообщения.
2. Строится таблица согласно заданных приоритетов.
3. Производится расчет параметра R для каждого сообщения.

5. Оценка результатов моделирования.

Применение модели реального времени

Роль модели:

1. Прогнозировать свойства и поведение объекта внутри области применения и за ее пределами.

2. Управление объектом путем выбора наилучших характеристик и воздействий, выявляющихся путем использования модели.

3. Изучение явлений или объекта на основе модели.

4. Получение навыков по управлению объектом.

5. Возможность улучшения свойств объекта путем использования модели.

Основное применение модели СРВ – разработка систем реального мира.

Построение моделей реального времени заключается в постановке и выработке требований поведения окружающей среды, а также поведение объектов в этой среде при воздействиях на ОС или объект.

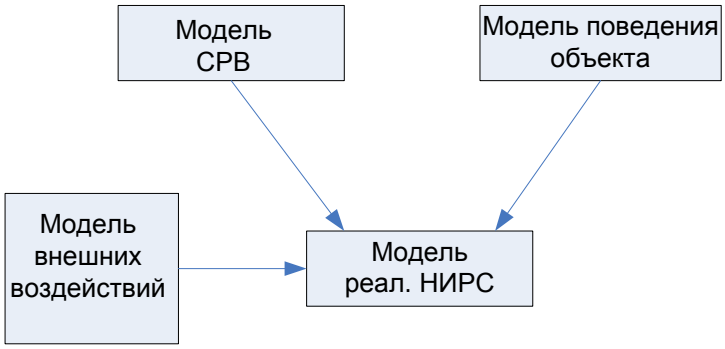

Рис. 21.

Первичная модель реального мира – модель интеллектуального здания.

Базовой моделью интеллектуального здания является модель СРВ.

Проблема построения модели – модель поведения человека (как сымитировать настроение человека?).

- Проблема управления человеком.
- Адекватная реакция модели на поведение человека.
- Создание благоприятных условий для функционирования реального мира и функционирования объектов внутри реального мира.
- Представленная модель удовлетворяет любой системе независимо от среды передачи данных.

Надежность в СРВ

Надежность в СРВ связана с реализацией систем с жестким РВ. К ним предъявляют следующие требования:

1. Функциональные – связаны с реализацией задач системы. Любая система должна удовлетворять предъявляемым требованиям.

2. Нефункциональные – это параметры определяются средой исполнения системы.

Выделяют:

2.1. Надежность:

а) собственно надежность, т.е. отказоустойчивость системы на протяжении определенного времени. Обеспечивается техническими средствами (дублирование функций, дублирование среды передачи, передача методом запрос/ответ).

б) Доступность и безопасность – к системе должен быть организован именованный доступ, позволяющий определить «своего» и «чужого», а также внешнее обеспечение безопасности.

в) Сохранность – включает в себя внешнюю сохранность объекта, сохранность информации (наличие средств архивирования информации, распределения, хранения).

2.2. Своевременность включает:

- «Отзывчивость» - подразумевает, что каждый узел должен реагировать на переданное ему событие по заданному алгоритму.

- Исполнительность – четкое выполнение функций в соответствии с временными диаграммами.

- Актуальность – любые данные, любые действия в текущий момент времени должны быть последними и соответствовать текущей ситуации.

- Временная предсказуемость – заключается в том, что в каждый последующий момент времени мы должны знать, в каком состоянии система будет находиться и как будет себя вести, а также, за какое время обрабатывается тот или иной параметр. Данные требования складываются в задачу моделирования.

- Контролируемость – любой объект системы должен быть управляемым. Это подразумевает наличие специальной функции в системе, позволяющей производить анализ функционирования узлов. Система должна адекватно реагировать на команды пользователя.

2.3. Динамическое управление изменениями.

Версионность (В1; В2)

Последующие версии должны функционировать без дополнительных технических изменений (т.е. это – гибкость системы, возможность расширения и модернизации).

Увеличение производительности достигается:

1) увеличением мощности вычислительного узла,

2) увеличением мощности системы в целом.

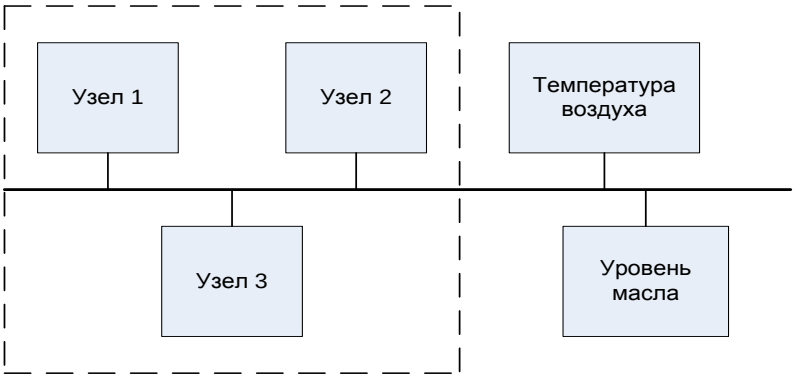

Рис. 22

Все группы требований являются требованиями среды исполнения. Реализация этих требования вытекает из процесса анализа будущей автоматизированной системы. Особенно актуален анализ для системы с жестким РВ.

На этапе проектирования систем надо учитывать параметры систем с жестким РВ.

Четкое разделение типов объектов на:

а) объекты,

б) действия.

При этом действия подразделяют на циклические и единичные.

Точное определение требований приложений по распределению времени для каждого объекта.

Определение относительной важности каждого объекта необходимо в том случае, если возникает ситуация, когда 2 объекта претендуют на 1 участок времени.

Точное определение и использование объектов и ресурсов.

Подбор наиболее подходящей для планировки распределения времени программной архитектуры.

Указанные параметры влияют на архитектуру систем жесткого РВ.

Архитектура делится на:

- Логическую – включает действия, которые выполняются независимо от требований, накладываемых средой исполнения. Эти требования направлены на удовлетворение функциональных требований.

- Физическую – включает расчет параметров и условий, которые обеспечивают функциональные и нефункциональные требования. (Нефункциональные требования учитываются на стадиях детального проекта и реализации).

Проектирование систем жесткого реального времени

Важнейшей стадией при разработке любой системы реального времени является создание проекта, который удовлетворяет ряду важных требований. Системы реального времени отличаются от обычных систем обработки данных тем, что к ним применяются некоторые нефункциональные требования (надежность и распределение времени). Как правило, стандартные методы проектирования не дают хороших результатов.

Обзор процесса проектирования

В настоящее время все больше заметно, что роль и важность нефункциональных требований в разработке комплексных приложений

оценивается неадекватно. Для разработчиков систем, для методов, которые они используют, характерна концентрация в первую очередь на функциональности, и лишь потом, сравнительно поздно, - на нефункциональных требованиях. Мы полагаем, что такой подход неверен при производстве безопасных ответственных систем. Например, часто требования по расчету времени рассматриваются в рамках производительности системы как целого. Отсутствие необходимой производительности часто выливается в какие-либо специальные ее изменения.

Нефункциональные требования включают в себя надежность (например, собственно надежность, доступность, сохранность и безопасность), своевременность (например, "отзывчивость", "исполнительность", актуальность, временная предсказуемость, контролируемость), и управление динамическими изменениями (т.е. занесение эволюционных изменений в работающую систему). Эти требования и условия, вносимые средой исполнения, должны приниматься во внимание во время разработки. Во время разработки необходима ранняя привязка программных функций к компонентам устройств с тем, чтобы можно было проводить анализ распределения времени и надежностных характеристик еще не отлаженной системы.

Учет особенностей жестких систем реального времени

Мы предполагаем, что если методы проектирования адекватно учитывают особенности жестких систем реального времени, то они должны поддерживать:

- четкое разделение типов действий/объектов, которые находятся в жестких системах реального времени (т.е. циклические и единичные действия).

- точное определение требований приложения по распределению времени для каждого объекта.
- определение относительной важности каждого объекта для успешного функционирования приложения.
- точное определение и использование объектов контроля ресурсов.
- переход к наиболее подходящей для планировки и распределения времени программной архитектуре.

Кроме того, методы проектирования должны допускать влияние планировки на проект как можно раньше.

Жизненный цикл жестких систем реального времени

Наш подход заключается в разделении архитектурного плана на две фазы:

- логическая архитектура;
- физическая архитектура.

Логическая архитектура включает действия, которые могут быть проделаны независимо от условий, накладываемых средой исполнения, и в первую очередь направлены на удовлетворение функциональных требований. Физическая архитектура принимает в расчет эти и другие условия и вдобавок охватывает нефункциональные требования. Физическая архитектура формирует основу для того, чтобы нефункциональные требования уже были удовлетворены, когда существуют детальный проект и реализация. Например, если все объекты построены с учетом худших условий по распределению времени и надежности, то сама система будет удовлетворять требованиям сохранности. Таким образом, физическая архитектура позволяет оценить параметры разработки для достижения компромиссного решения для всех требований задачи.

В этом документе мы в первую очередь касаемся жестких систем реального времени, поэтому физическая архитектура фокусируется на распределении времени и необходимой планировке, что будет гарантировать, что однажды построенная система будет работать корректно и по данным, и по времени. Чтобы провести этот анализ, необходимо оценить время исполнения имеющегося кода, получить зависимости по времени целевого процессора и другие параметры среды исполнения.

Когда архитектурные фазы закончены, можно начинать серьезное планирование деталей проекта. Когда это будет сделано, нужно измерить характеристики выполнения кода, чтобы гарантировать, что верхние оценки времени выполнения в самом деле корректны. Если же это не так (что обычно имеет место для новых приложений), фаза физической архитектуры пересматривается и обновляется. Если же все-таки система не реализуется, тогда либо должны быть пересмотрены детали проекта (при небольших отклонениях), или разработчик должен вернуться к фазе логической архитектуры (при серьезных проблемах). Когда измерения кода пройдены, выполняется тестирование приложения. Оно должно включать действительные распределения времени по коду.

Логическая архитектура

Существует два аспекта любого метода проектирования, которые облегчают создание логической архитектуры жестких систем реального времени. Во-первых, абстракции должна быть дана конкретная поддержка, что, как правило, и нужно проектировщикам систем. Во-вторых, логическая архитектура должна планироваться с тем условием, чтобы возможно было ее анализировать во время проектирования физической архитектуры.

Результатом иерархического разделения в ходе фазы логического планирования является коррекция конечных объектов с полным определением их взаимодействием. Предполагается, что некоторые формы процесса функционального разделения могут приводить к определению новых объектов.

Конечные объекты характеризуются как:

- циклические;
- единичные;
- защищенные;
- пассивные.

Циклические и единичные действия являются обычными в системах реального времени; каждое из них должен содержать поток, который создается во время выполнения. Приоритет потока устанавливается во время планировки в фазе физической архитектуры. Защищенные объекты управляют доступом к данным, которые доступны нескольким потокам (т.е. циклическим и единичным объектам); в частности, они предусматривают взаимное исключение. В однопроцессорных системах это достигается определением верхнего приоритета для каждого защищенного объекта, который равен максимальному количеству потоков, использующих его. Когда поток обращается к защищенному объекту, он начинает работать с верхним приоритетом и, следовательно, получает исключительный доступ к данным, хранимым в защищенном объекте. Защищенные объекты подобны перехватчикам, в которых вызывающий может быть заблокирован, если условия не позволяют ему продолжить. Это используется для задержки единичных объектов, пока не произойдет освобождающее событие. Последним важным типом объектов является пассивный. Он используется для объекта, который или

используется только одним объектом, или может использоваться совместно без ошибок.

Все четыре вышеуказанных типа объектов приемлемы как конечные объекты в жестких системах реального времени. Однако возможно, что система реального времени имеет подсистему, которая не является системой реального времени. Объекты в таких подсистемах являются пассивными или активными. Активные типы объектов могут участвовать в разделении главной системы, но должны быть трансформированы в один из вышеуказанных типов до достижения конечного уровня.

С помощью этих типов конечных объектов могут поддерживаться стандартные конструкции, используемые в жестких системах реального времени:

- Периодические действия - представляется циклическими объектами.
- Единичные действия - представляется единичными объектами.
- Действия, обусловленные приоритетом - влекут серии вычислений на конечных объектах. Вероятно появление в проектах, которые должны отражать транзакционные сроки.

Процесс планирования логической архитектуры может начаться с разработки активных и пассивных объектов. Процесс разделения приведет к разработке конечных объектов соответствующего характера. Например, требуемая циклическая транзакция от входа к выходу может быть сначала представлена как единичный активный объект, но потом может быть реализована как циклический объект с последующими сериями единичных объектов, связанных защищенными объектами.

Наложение на проект условий для анализа

Чтобы иметь возможность анализировать весь проект, необходимо поставить определенные условия. Они в основном связаны со связью или синхронизацией между объектами.

1. Циклические и единичные объекты не могут выполнять произвольные операции блокировки в других циклических или единичных объектах.

2. Циклические и случайные объекты могут выполнять асинхронную передачу операций управления в другие циклические или единичные объекты.

3. Защищенные объекты не могут выполнять операции блокировки в любых других объектах.

Пункты 1 и 2 требуют, чтобы циклическим и единичным объектам было разрешена только связь через посылку полностью асинхронных сообщений или защищенные объекты.

Любой метод проектирования систем реального времени должен учитывать эти условия в процессе планирования логической архитектуры.

Физическая архитектура

Основное внимание в физической архитектуре уделяется требованиям к распределению времени. Процесс проектирования должен поддерживать формирование физической архитектуры через:

1. Возможность ассоциирования атрибутов распределения времени с объектами.

2. Обеспечение такой внутренней структуры, в которой может быть предпринята планировка конечных объектов.

3. Создание абстракции, с помощью которой проектировщик может контролировать ошибки распределения времени.

Физический план должен быть осуществлен в контексте среды исполнения. Это гарантируется планировкой. Вопросы надежности также должны быть рассмотрены в этой фазе.

Атрибуты объектов

Все конечные объекты обладают ассоциированными с ними атрибутами реального времени. Многие атрибуты связаны с отображением на логический план требований распределения времени (например, срок, важность). Они должны быть установлены до того, как будет производиться планировка.

Каждый циклический или единичный объект имеет некоторое количество временных атрибутов. Например:

- Период исполнения для каждого циклического объекта.
- Минимальный интервал проявления для единичного объекта.
- Сроки для всех циклических и единичных действий.

Различаются две формы сроков. Одна применяется прямо к единичному или циклическому действию. Другая применяется к ранее обусловленному действию (транзакции). Сроки для других действий должны извлекаться таким образом, чтобы полная транзакция удовлетворяла ее требованиям распределения времени.

Для планировки нужно знать верхнюю оценку времени исполнения каждого потока и все операции (во всех объектах). После фазы логического планирования они могут быть оценены и присвоены соответствующие атрибуты. Чем лучше оценки, тем точнее планировка. Хорошие оценки могут быть получены при повторном использовании компонент или из аргументов сравнения (с существующими компонентами других проектов). В процессе детального проектирования и кодирования, а также при прямом

использовании измерений во время тестирования, могут быть получены лучшие оценки, которые потребуют перепланировки.

Планировка

Планировка является составной частью разработки физической архитектуры. Предложенный план должен быть осуществим. Это значит, что все сроки должны гарантироваться при всех предсказуемых обстоятельствах. Реализация этого требует знания скорости процессора, скорости памяти и ее емкости, временных характеристик ядра. Могут быть нужны также временные параметры других устройств. Если мы предполагаем, что среда исполнения поддерживает приоритетную опережающую диспетчеризацию потоков, то в этом случае планировка заключается в определении статических приоритетов потоков, заключенных в циклических и единичных действиях.

Контроль за временными ошибками

Описанная выше планировка может быть эффективна, если оценки/измерения верхней границы времени исполнения точны. В области временных характеристик определяются две стратегии для ослабления результатов ошибок в компонентах программ:

- Не давать объекту вычислительного времени больше, чем ему нужно.
- Не позволять объектам выполняться по истечении срока.

Операционные системы реального времени

Необходимы для обеспечения выполнения пользовательских задач и функций. Операционная СРВ базируется на планировщике задач (периодическом и апериодическом).

Имеет особенности по работе с объектами ввода/вывода. Любые ОС реального времени базируются на архитектуре. Существуют 3 основных архитектуры ОС:

(1) – монолитная,

(2) – на основе микроядра,

(3) – объектно-ориентированная.

Каждая из архитектур позволяет обеспечивать функционирование задач в режиме реального времени.

Монолитная архитектура ОС

Саму ОС можно разделить на 5 основных частей (см. рис.23): задачи, интерфейс прикладных программ (И.П.П.), собственно ОС или ее ядро, драйверы (Д.), аппаратные средства (А.С.).

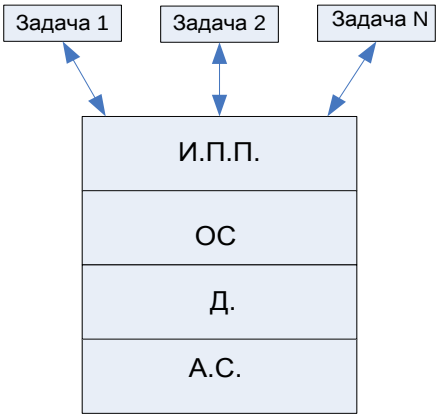

Рис. 23.

Самым простым примером является ОС DOS.

Достоинства: простота создания, простота управления задачами.

Недостатки: отсутствие гибкости в системе и возможности управления задачами в процессе функционирования систем; при зацикливании одного из блоков система блокируется и перестает функционировать.

Архитектура на базе микроядра

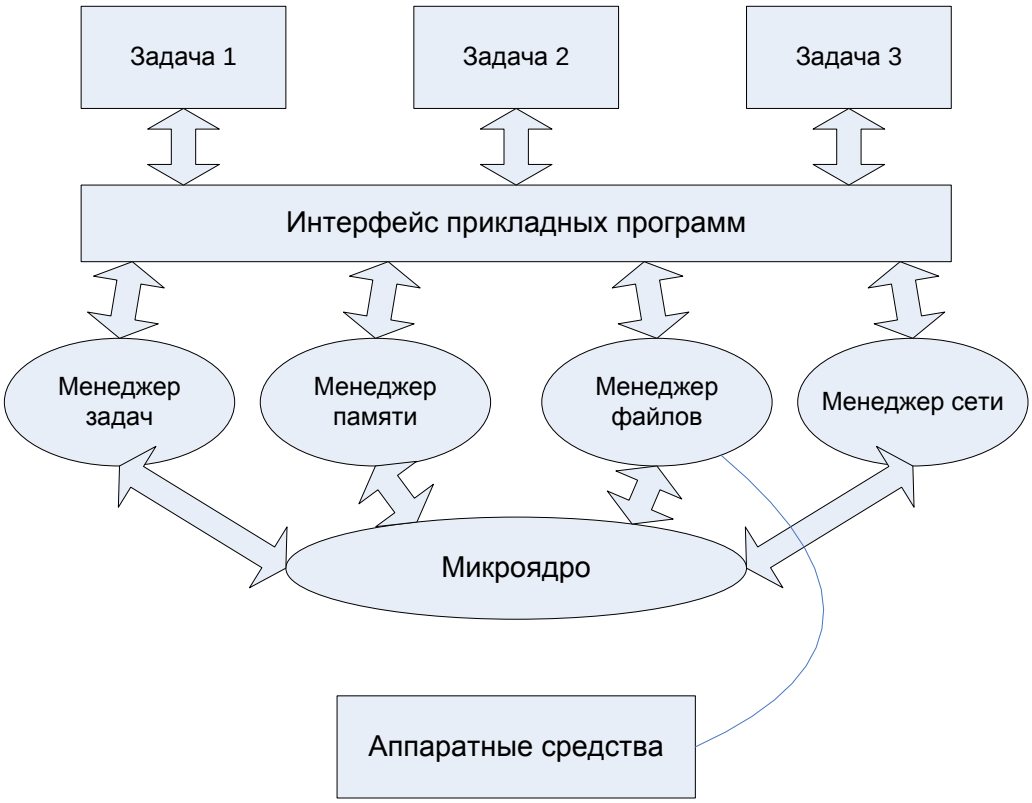

Рис. 24.

Принцип децентрализации функций ОС

Все базовые функции – задача микроядра, то есть запуск системы, управление доступом к аппаратным средствам и т.д. Все остальные функции выделены в отдельные менеджеры. Каждый менеджер выполняет строго определённые функции.

Достоинства: более гибкая система

Недостатки: каждая задача зависит от микроядра и от его реализации.

Большинство операционных систем РВ реализовано на базе микроядра (QNX и т.д.)

Объектно-ориентированная архитектура

Вводится понятие объекта. Объект для ОС – это задача. Для каждой задачи была предложена реализация ядра пользователя.

Каждый элемент архитектуры может взаимодействовать с любым другим. Виды и характер воздействия определяется конфигурацией ОС.

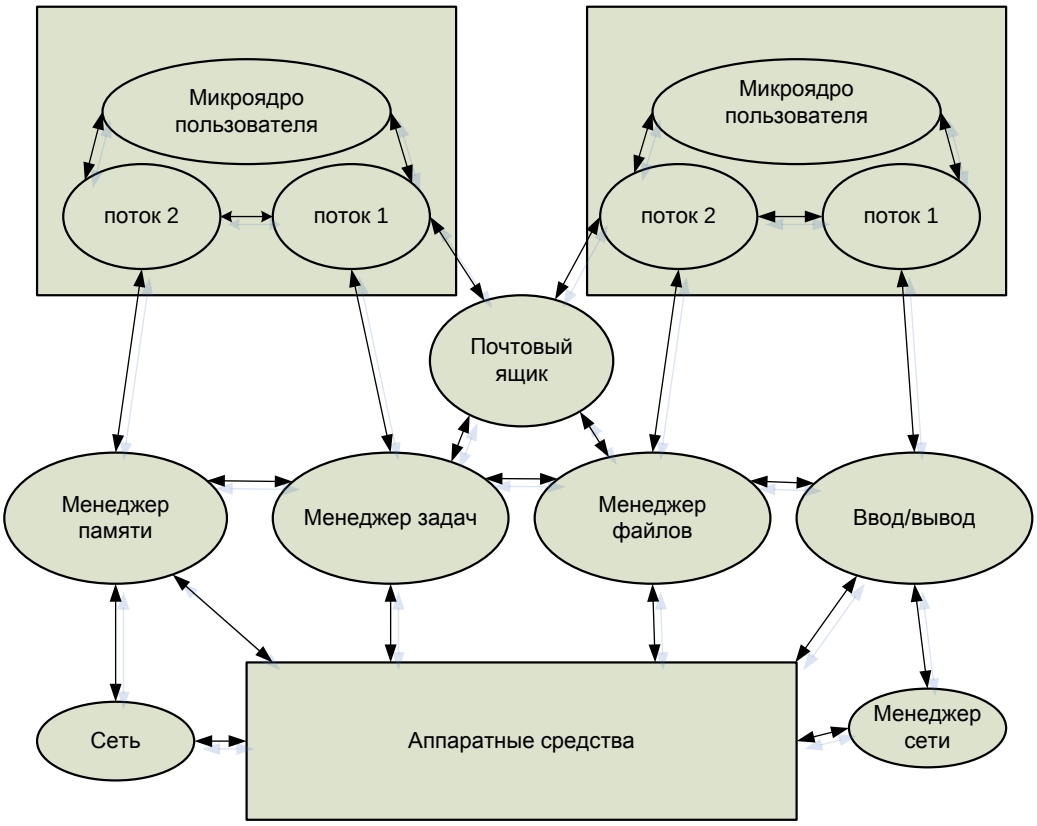

Рис. 25.

Каждая программа включает микроядро, которое обеспечивает выполнение основных функций, распределение приоритетов, загрузка задач и обеспечение взаимодействия с другими задачами. Взаимодействие с другими данными через потоки (обеспечивает передачу информации между программами и др.), то есть это буфер для обмена информацией. Для взаимодействия всех компонентов системы был реализован модуль «почтовый ящик». Это буфер системы (обеспечивает приём, обработку и передачу информации между компонентами). «Почтовый ящик» является сервером, он функционирует по запросу от других компонентов. Характер

взаимодействия компонентов не влияет на «почтовый ящик» и другие модули.

Особенности функционирования ОС РВ

Ядро ОС обеспечивает выполнение следующих функций:

1. Планирование задач.
2. Синхронизация задач.
3. Межзадачная коммуникация.
4. Управление памятью.

Вспомогательные элементы:

- Файловая система.
- Сетевая поддержка.
- Интерфейс с оператором.

Задача – готовая к выполнению программа, участвующая в вычислительном процессе.

Цель менеджера задач – определение последовательности выполнения задач пользователя. У менеджера задач есть ряд функций, которые называются функции ядра:

1. Планирование задач.

1.1. Циклическое планирование. Все задачи одна за другой. Нет возможности изменения функционирующей задачи по приоритетам.

1.2. Разделение времени.

1.2.1. Разделение времени на равные интервалы. Для каждой задачи выделяется интервал времени. Если задача не закончила своё исполнение в интервале, то выполняется в следующем доступном интервале (псевдопараллельное исполнение задач). Реальное параллельное выполнение задач возможно только при нескольких процессорах (вычислительных узлах).

1.2.2. Разделение времени с вытеснением. Разделение по времени, но при появлении более высокоприоритетной задачи более низкоприоритетная задача вытесняется.

2. Назначение приоритетов (см. алгоритм планирования).

3. Синхронизация задач – обеспечение согласованности действий программ на вычислительном узле.

Синхронизируются:

- Связанность задач (то есть логическая последовательность задач).
- Обеспечение доступа к общим ресурсам.
- Обеспечение синхронизации с внешними событиями.
- Обеспечение синхронизации по времени.

Связанность задач определяет последовательность функций реального объекта. Каждое действие в системе должно происходить последовательно в зависимости от предыдущих действий, должна сохраняться история. Для этих целей используется «почтовый ящик» - буфер и сортировщик сообщений. Принцип связности реализуется как алгоритм функционирования системы.

Обеспечение доступа к общим ресурсам необходимо для того, чтобы обеспечить доступ к разделяемым ресурсам для всех задач. Это физическое устройство или область памяти.

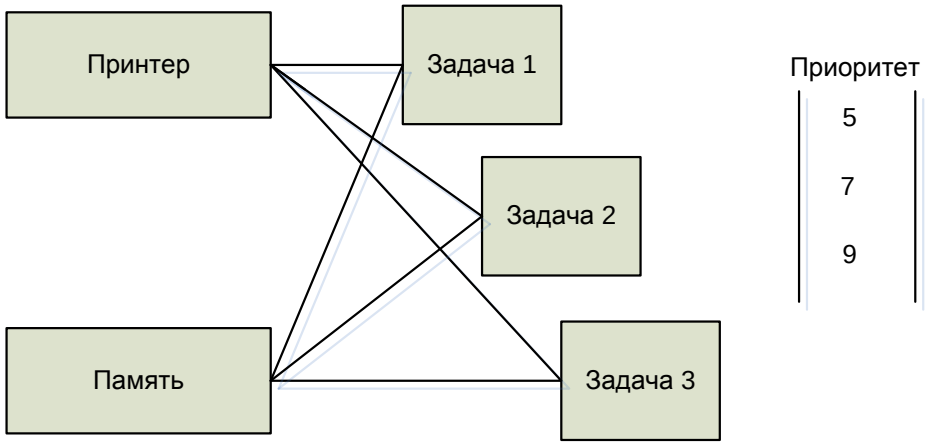

Рис. 26.

Проблемы возникают при использовании алгоритма «с вытеснением».

- Если одна из задач захватит ресурс и не обеспечит доступ для всех остальных или не успеет выполнить все действия на устройстве, то возникают ошибки функционирования, связанные с «гонками». Их можно обнаружить только в процессе эксплуатации системы.

Способы решения проблем гонок.

1. Создание сервера ресурса, то есть задачи, отвечающей за доступ к ресурсу. Она обеспечивает планирование последовательности выполнения.

2. Запрет прерывания доступа к ресурсу на время его использования.

3. Принцип «семафора».

Реализация проблем «гонок» осуществляется в критических секциях – участках кода программ, где происходит обращение к разделяемым ресурсам. Для решения можно реализовать несколько потоков и только 1 поток отвечает за доступ к ресурсу.

- Вторая проблема – «смертельный захват» (DeadLock). Возникает, если задачи не поделили между собой ресурс.

Пример:

Задача А → дисплей → клавиатура

Задача В → клавиатура → дисплей

захват требует

задачи зашли в тупик. Ситуация DeadLock.

Способы решения проблемы:

1. Принцип «либо всё, либо ничего». Только одна задача может претендовать на ресурсы. Вторая задача ожидает доступа только после освобождения обоих ресурсов первой задачей.

2. Организация сервера для доступа к ресурсу.

3. При требованиях доступа к занятому ресурсу задача, требующая ресурс отключается и ждёт выполнения задачи B.

- «Инверсия приоритетов» - это ситуация, при которой задача с более высоким приоритетом не может выполняться из-за задачи с более низким приоритетом.

Пример:

A – высокоприоритетная задача.

B – среднеприоритетная задача.

C – низкоприоритетная задача.

C → захватывает ресурс.

B → выполняется.

C → останавливается (ресурс захвачен).

A → пытается захватить ресурс и останавливает B, но C уже захватила ресурс. B не выполнится пока A не выполнится.

Решение:

1. Сервер ресурсов. Выполняется задача с самым высоким приоритетом, и доступ к ресурсам будет освобождаться.

2. Перераспределение ресурсов. Ресурс выделяется той задаче, которая начинает выполняться в данный момент.

Для оптимизации поведения системы (выполнение алгоритма и не возникновение опасных ситуаций) необходимо для каждого ресурса выделить свой сервер с заданным приоритетом и методом доступа, который обеспечит взаимодействие ресурсов и задач.

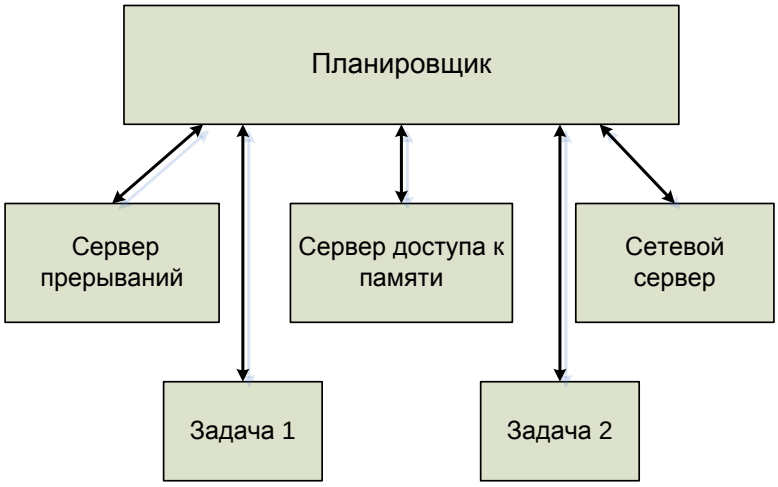

Рис. 27.

Наименее проблемная архитектура по обеспечению доступа к ресурсам – это объектно-ориентированная (из-за буфера – «почтового ящика» и микроядра).

Синхронизация с внешними событиями

Это обеспечение реакции системы на действия, передающиеся из внешних систем и не являющихся частью вычислительного узла.

Обеспечение синхронизации возможно:

- При организации распределённой системы и использовании сетевых компонентов.
- Синхронизация с узлами, подключенными к внутренней шине вычислительного узла.

Синхронизация с внешними событиями используется для обеспечения алгоритма функционирования при распределении системы по разным вычислительным узлам.

Синхронность функционирования обеспечивается наличием общих характеристик вычислительных узлов системы.

Для реализации синхронизации с узлами, подключенными к внутренней шине вычислительного узла, используются 2 метода:

1. метод прерывания

2. метод опроса.

Метод прерывания более скоростной, но в нём большой поток лишней информации. Применяется для быстрых систем и систем с большой градацией приоритетов.

В методе опроса задача, которой нужна информация, отправляет запрос на другой вычислительный узел именно в тот момент, когда ей это необходимо. Уменьшается поток информации, но обычно этот метод используется для более медленных задач, так как метод опроса требует подтверждение. Запрос → ответ → готовность к приёму информации → передача информации → подтверждение приёма (необязательно).

Синхронизация по времени

Для этой цели задаётся квант времени. Он является эталонным измерителем и называется тиком. Фактически для операционной системы тик – это базовая единица измерения. Размер этого тика может быть различным для различных операционных систем и зависит от тех задач, которые в этой операционной системе исполняются. Размер тика определяется по таймеру, а также из механизма взаимодействия с таймером. Методы взаимодействия с таймером также могут быть различными в зависимости от операционной системы. Синхронизация задач по времени заключается в использовании этих таймеров. Таймеры бывают двух видов: программные и аппаратные. Количество аппаратных таймеров ограничено архитектурой вычислительного узла. Количество программных таймеров ограничено архитектурой операционной системы либо не ограничено. Одним из подвидов программных таймеров являются пользовательские таймеры. Пользовательский таймер – это некоторый счётчик и обработчик этого счётчика. Любой

программный таймер основывается на аппаратных таймерах. Обычно на одном вычислительном узле бывает один аппаратный таймер, который функционирует по прерыванию. Для таймера существует понятие истечения таймера.

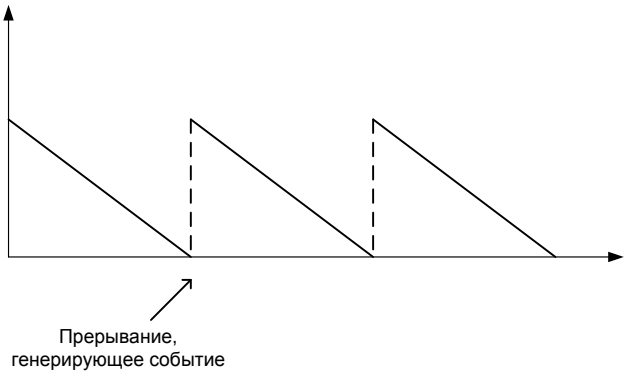

Рис. 28.

Синхронизация по времени обеспечивается по меткам прерывания. Выполнение каждой задачи реализуется по временному интервалу. Временной интервал определяется по таймеру. Любой программный таймер должен быть кратен аппаратному таймеру. Если это условие не выполняется, то синхронизация по времени для задач невозможна.

ВИДЫ ОПЕРАЦИОННЫХ СИСТЕМ

QNX

Данная операционная система (ОС) разработана в 1982 году компанией QNX Software Systems. Она сразу же стала ОС «двойного» назначения, то есть она применяется для хозяйственных работ и для военных целей. В 1990 году система QNX была распространена по всему миру. В России появилась в 1992 году. Основная особенность: QNX построена на базе FLEET-технологии.

F – Fault tolerance (отказоустойчивая)

L – Load balancing (регулирующая нагрузку)

E – Efficient (эффективная)

E – Extensible (расширяемая)

T – Transparent (прозрачная или открытая)

Особенности данной ОС:

1. Является гибридом 16-32-битной ОС.
2. Размер ядра от 8 до 20 кБ.
3. Процесс взаимодействия задач с помощью сообщений.
4. Поддержка распределённых сетевых вычислений.
5. Для поддержания связи с популярными ОС поддерживается SMB.
6. Поддержка файловых систем FAT, NTFS.
7. Данная ОС имеет свойство интероперабилить (interoperability – совместимость к использованию), совместима с программным обеспечением различных производителей.

Недостаток: данная ОС ориентирована на платформу Intel.

QNX распространяется бесплатно для некоммерческого использования и для платформ, ориентированных на персональный компьютер. Все остальные версии являются платными. Также весь софт под QNX является платным.

Под QNX разработаны собственные СУБД, являющиеся СУБД РВ:

1. Watcom SQL.
2. Faircom.
3. C-Tree.

В QNX есть большой набор графических средств для работы приложений. Для этой цели разработано специальное средство, позволяющее пользователю создавать собственные приложения.

ОС-9

Данная ОС относится к классу unix-подобных систем. Основное применение – мобильные телекоммуникационные устройства, а также интерактивные цифровые телевизионные приставки. Основные платформы: Motorola 68xxx, Intel 86-й серии и Hitachi. Данная ОС поддерживает функционирование 65535 задач одновременно, а также обеспечивает работу 255 пользователей.

Время переключения между процессами равно 14мс; для процесса Motorola 68040 на частоте 30 МГц. Данная операционная система обладает свойством переносимости приложений, поддерживает стандарт приложения на C++ и Java. Кроме того, ОС-9 имеет развитые сетевые средства: поддерживает протоколы TCP/IP, CAN, ArcNET, IPX. Поддерживается свойство совместимости. Существует большое количество программных продуктов для построения собственных приложений и для обеспечения сетевого взаимодействия. Производитель ОС-9 – Microware System Corporation. Система разработана в 1979 году по заказу компании Motorola.

ОС-9000 – переносимая версия ОС-9. функционирует однообразно на различных платформах, т.к. на 95% система написана на языке Си. 5% соответствует виду платформы. ОС-9 поставляется в исходных кодах для платформы, для которой она была заказана.

Кроме мобильных систем ОС-9 используется на данный момент в системах военного и аэрокосмического назначения, бытовой электронике, измерительных системах, промышленной автоматизации, связи. Основной упор делается на Power PC. С 1979-1997г было установлено 5млн.версий.

Основной метод – приоритет планирования с вытеснением.

VxWorks/Tornado

Данная система разработана компанией WindRiver Systems. Система ориентирована на следующие платформы: Intel 386, 86 и 960-й серий, Power PC, SPARC. Разработка ведется в два этапа:

Инструментнальная машина в среде Works/Tornado, выполнение осуществляется на Host-машине. Система реализуется по принципу микроядра; поддерживает следующие сетевые средства: Ethernet, RS232, Cross-шина. Поддерживается создание приложений на языке C, C++. Имеет средства разработки проекта, средства управления проектом и командный интерпретатор, поддерживает файловую систему NTFS и имеет специализированное CASE-средство Control Show.

Основное применение системы: сетевое и коммуникационное оборудование, промышленный контроль технологических процессов и бортовые вычислительные системы.

Применение:

1. Разработка тренажеров – пилотов, в системе обеспечиваются реальные условия для пилота с помощью алгоритмов VxWorks.
2. Регулирование движения на перекрестках в New York.
3. Система спутниковой связи «банкир», используется для обеспечения ЦБ РФ для связи с филиалами.

Система состоит из двух сегментов: космический (3 спутника), наземный.

Операционные системы реального времени для Windows

Любая ОС реального времени под Windows или Linux является расширением стандартной ОС.

Расширения

I. IA-SPOX

В расширении ОС Windows для реального времени. Разработано в 1994 году компанией Spectron microsystem. Данное расширение разработано для системы Windows95/98. Основное отличие от стандартной ОС заключается во включении виртуальных драйверов, взаимодействующих с ядром ОС.

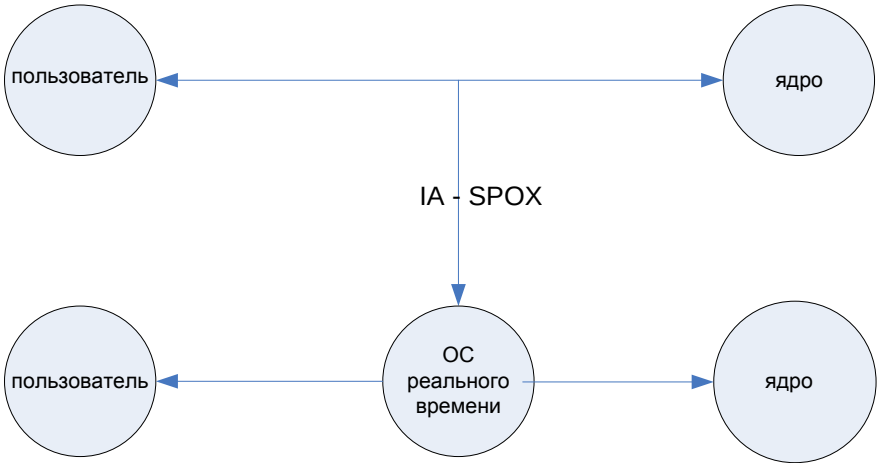

Рис. 29.

Набор виртуальных драйверов обеспечивает детерминированную реакцию ядра и не позволяет пользователю напрямую взаимодействовать с функциями ядра. Обслуживание запросов осуществляют виртуальные драйвера.

Основное применение: высокоскоростная связь в приложениях, а также в приложениях, требующих быстрые детерминированные реакции.

Система поддерживает необходимый набор прерываний. Система работает в нормальном состоянии, если приложений не больше 20. На данный момент такая система не поддерживается.

II. RTX

(Real Time Extension)

Производитель – VenturCom.

Данное расширение ориентированно на Windows NT и его линейку, а также на Linux. Ядро ОС является отдельной задачей и загружается как отдельная задача.

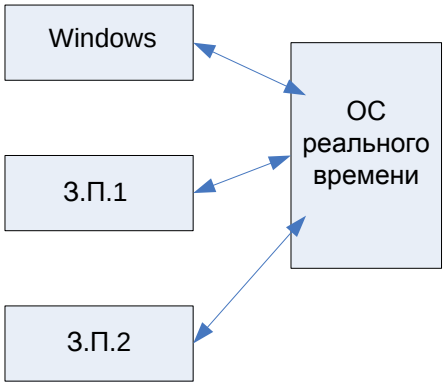

Рис. 30.

З.П.1 – задача пользователя 1, З.П.2 – задача пользователя 2.

Из вышеотмеченного следует уменьшение времени доступа к памяти (не приложений из-под Windows, а задач пользователя). Допускается взаимодействие с аппаратной частью напрямую из задач. Сократились задержки по работе с прерываниями. Если задача требует реального времени, то она запускается в отдельном планировщике; если она не требует реального времени, то запускается через Windows.

Основная особенность: добавлен планировщик задач.

Основное применение: мониторинг промышленных, мониторинг промышленных сетей, а также управление задачами в мягком режиме реального времени. При зависании задачи Windows система не функционирует.

Расширение RTX нашло поддержку компанией Microsoft, часть функций была введена в Windows XP.

III. Falcon

Разработчик: Radisys.

Применение в ОС класса Windows NT.

Расширение основано на объектно-ориентированной архитектуре, поэтому ядро Falcon взаимодействует с ядром Windows, как две равноправные задачи. Выполнение задач осуществляется по встроенному алгоритму:

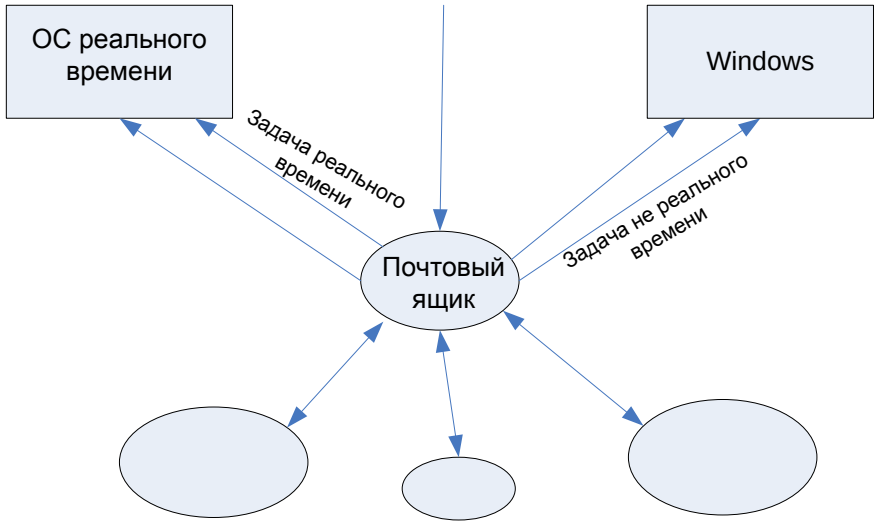

Рис. 31.

Поэтому Windows не влияет на работу всей остальной системы. При “подвисании” Windows задачи РВ будут продолжать выполняться, а задачи не РВ будут отправляться в ОС реального времени.

Данное расширение поддерживает создание системы «жесткого» реального времени.

Из всех расширений Falcon является более стабильным и более надежным.

IV. HyperKernel

Ориентировано для Windows NT. Все программное обеспечение делится на две части:

1. ПО операторного интерфейса.

2. Высокоскоростные задачи управления.

ОС реального времени и Windows – единое ядро, взаимодействия между ними нет. ОС реального времени – надстройка над Windows. При “подвисании” Windows система останавливается.

Применения не нашло, используется для задач «мягкого» реального времени.

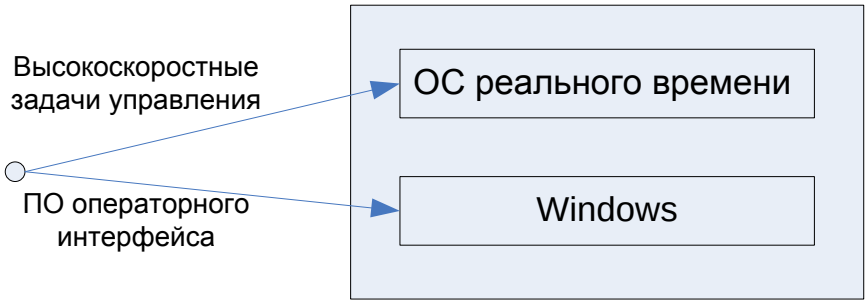

Рис. 32. Монолитная архитектура

Достоинства и недостатки операционных систем реального времени

Недостатки:

1. Узкоопределенная поддержка в промышленных контроллерах.
2. Отсутствие поддержки производителей контроллеров.
3. Высокая стоимость прикладного ПО ОС реального времени. Отсутствие связи с производителями ПО.

Достоинства:

1. Универсальность подхода. Разработчику не требуется создавать собственную ОС и специальное ПО, следовательно, уменьшаются затраты на проектирование.

Другие достоинства вытекают из реализации.

Вывод: наиболее популярными и широкоиспользуемыми ОС являются OS 9 и OS 9000 за счет наличия большого количества разработок под различные платформы и возможности создания ОС реального времени для новых платформ.

Технология разработки собственной ОС РВ

С позиции разработчика АСУ создание ОС реального времени делится на два этапа:

1. Разработка планировщика задач. Разработка на базе вытесняющей политики планирования, ориентированной на однопроцессорные системы. При создании планировщика для многопроцессорной среды производится его деление на глобальный и локальный. Глобальный планировщик также может использовать вытесняющее планирование, но наиболее эффективным является приоритетное планирование (перед запуском производится расчет).

2. Создание драйвера, обеспечивающего доступ к оборудованию и запуск планирования. Реализация драйвера зависит от типа используемого микроконтроллера. Выделяют менеджеры ресурсов, а также способы работы с таймером. Аппаратные и программные таймеры реализуются через драйвер.

Планировщик является платформонезависимым.

Средства создания операторского интерфейса автоматизированных систем (SCADA-приложения)

SCADA-Supervisory Control And Data Acquisition (централизованное управление и сбор данных).

Основное назначение: сбор информации в АСУ. Сбор информации является первичным для оценки ситуации автоматизированной системы. Управление может быть реализовано внутри системы. Под понятие «SCADA-приложения» попадают любые приложения, получающие данные с оборудования. Любое SCADA-приложение должно иметь набор инструментальных средств для подключения новых объектов. SCADA-приложения относятся к прикладному ПО. Разработка SCADA-приложения является

трудоемкой задачей. Поэтому большинство производителей используют стандарт SCADA-пакеты.

Возможности и средства, присущие SCADA-пакетам

1. Автоматизированная разработка, дающая возможность создать ПО системы автоматически без реального программирования.
2. Средство сбора первичной информации от устройств нижнего уровня. Наличие драйверов доступа к оборудованию либо серверов.
3. Средство управления и регистрации сигналов об аварийных ситуациях (alarm).
4. Средство хранения информации с возможностью ее пост-обработки. Существует 2 способа:

4.1. Включение собственной СУБД.

4.2. Реализация интерфейса через ODBC к популярной БД.

5. Средство обработки первичной информации. Преобразование информации из одного вида в другой.
6. Средство визуализации информации. Построение графиков, гистограмм. Осуществляется в режиме реального времени и в режиме пост-просмотра.

Основу SCADA-пакетов составляет несколько программных компонентов и несколько утилит администрирования.

Программные компоненты

1. База данных реального времени.
2. Ввод/вывод.
3. Предыстория.
4. Регистрация аварийных ситуаций.

Утилиты администрирования

1. Управление доступом.

2. Управление сообщениями.

3. Управление устройствами.

Технология создания операторного интерфейса состоит из основных пяти пунктов:

1. Разработка архитектуры системы автоматизации. Определяется функциональное назначение каждого узла.

2. Решение вопросов, связанных с поддержкой распределенной архитектуры операторного интерфейса, и необходимость наличия узлов с “горячим” резервированием (возможность замены оборудования во время функционирования системы).

3. Создание прикладной системы управления для каждого узла. Определяются все входы и выходы, используется данное SCADA-приложение.

4. Приведение в соответствие параметров прикладной системы с информацией, которой обмениваются устройства нижнего уровня.

5. Отладка созданной прикладной системы в режиме эмуляции и в режиме реального времени.

Классификация SCADA-приложений

Классификация SCADA-приложений возможна по нескольким категориям:

1. По используемой ОС:

1.1. Большинство SCADA-приложений выполнено под Windows 9x/NT.

1.2. SCADA-приложения для ОС реального времени

Специфические приложения: Real Flex, Sitex.

1.3. OS/2, Unix SCO, UMS на платформе VAX, AIX (RS6000), HP-UX (HP 9000).

2. По виду разработки:

2.1. Универсальные.

Предназначены для большого количества приложений, основной элемент OPC – сервер.

2.2. Специальные.

Реализуются под определенный вид контроллера заданной компании производителя (разработано для контроллеров Siemens).

3. По сетевой поддержке:

3.1. Поддержка стандартных сетевых сред.

3.2. Поддержка сетевых стандартов из класса FieldBus, ProfiBus, CAN, LON, ModBus, BitBus, IEB, BACNet.

3.3. Поддержка специальных протоколов.

Все протоколы, производные от RS485, являются разработкой компании - производителя сетевых контроллеров.

4. Поддержка встроенных SCADA-языков:

4.1. Поддержка Visual Basic.

4.2. Поддержка собственного языка.

4.3. Без языковой поддержки.

5. Поддерживаемые СУБД:

5.1. Поддержка ANSI SQL через ODBC.

5.2. Поддержка СУБД реального времени.

5.3. Без поддержки связей с СУБД.

6. Графические возможности:

6.1. Поддержка GUI (graphic user interface).

6.2. Без поддержки GUI, но наличие графического редактора.

6.3. Наличие поддержки векторной графики.

7. Эксплуатационные характеристики:

7.1. Скорость освоения продукта.

7.2. Скорость разработки типовых систем.

8. Удобство в применении или в использовании определяют сервисы, представленные на этапе разработки.

9. Наличие поддержки определяется наличием специалистов в районе по этим SCADA-приложениям. Зависит от количества копий.

Виды SCADA-приложений

1. ТРЕЙС МОУД

Самое распространенное приложение в России. Оно предназначено для разработки АСУТП широкого назначения. Разработано в 1992 году компанией AdAstrA Research Group, Ltd (Россия). В настоящее время около 40 тысяч инсталляций в следующих отраслях: энергетика, металлургия, нефтяная и газовая отрасли, химическая отрасль, коммунальное хозяйство. Данное SCADA-приложение является универсальным, применяется для автоматизации промышленных объектов и зданий.

Основные характеристики:

1. Поддержка модульной структуры (128 – 64000*16).
2. Количество тэгов не ограничено.
3. Минимальный цикл системы равен 1 мс.
4. Открытый формат драйвера для связи с любым УСО (устройством связи с объектом).
5. Программирование на Visual Basic.
6. Средство для «холодного» и «горячего» резервирования.
7. Встроенная поддержка контроллеров: Siemens, Adam, Tecon, Genius.
8. Возможность подключения к системе драйверов сторонних производителей.
9. Поддержка сетевого времени.
10. Возможность просмотра трендов в режиме реального времени и в режиме истории.
11. Поддержка обмена через DDE, OPC, DCOM, ODBC.
12. Полная русификация.
13. Web-управление.

14. Поддержка GSM, управление через WAP и sms.

15. Поддержка следующих сетевых протоколов: FieldBus, ModBus, LonWorks, ProfiBus, Industrial Ethernet, CAN и т.д.

16. Компоненты ТРЕЙС МОУД:

- редактор базы каналов;
- редактор представления системы;
- редактор шаблонов;
- профайлеры – средство создания стандартных профилей системы.

17. Защита от несанкционированного доступа осуществляется 2 методами:

- идентификация оператора с помощью login и password;
- закрытость протокола.

18. Поддержка предоставляется бесплатно при условии лицензионной покупки пакета. Стоимость зависит от количества точек ввода/вывода и находится в следующих пределах: 590 – 9950$.

2. Wizcon

Приложение разработано компанией eMation. Входит в состав комплекта, представляющего комплексное решение задач автоматизации, Wiz Factory.

Основные характеристики:

1. Архитектура реализует многозадачный режим функционирования.

2. Поддержка вывода трендов в режиме реального времени и в режиме истории.

3. Сетевое деление объектов на станции и сервера. Все сетевые компоненты объявляются на сервере, а станция лишь использует методы доступа к этому устройству.

4. Поддержка «горячего» резервирования.

5. Встроенная технология на базе Web. Позволяет управлять через сеть Internet; поддержка создания мобильных систем.

6. Для доступа к базам данных используется специальный модуль Wiz SQL. Он поддерживает интерфейс с наиболее популярными СУБД.

7. Поддержка программирования на Visual Basic.

8. Одновременная регистрация 10 тысяч сообщений.

9. Векторная графика.

10. Поддержка OPC для связи с приложениями.

11. Организация распределенных систем мониторинга.

12. Существует 3 вида Wizcon:

- Wizcon Runtime
- Wizcon Development Runtime
- Wizcon с модулем Wiz SQL

13. Защита от несанкционированного доступа осуществляется с помощью 32х-битного идентификатора пользователя.

14. Имеется документация на русском языке.

15. Стоимость: 1140-7080$, зависит от количества точек ввода/вывода (80-65000).

3. iFix

Разработано в 1996 году компанией Intellution. По данным на 2000 год, было произведено 130 тысяч инсталляций. Основная особенность – объектно-ориентированная структура.

Основные характеристики:

1. Поддержка «горячего» резервирования.

2. Обеспечение клиент-серверной архитектуры.

3. Конфигурирование в режиме on-line.

4. Программирование на Visual Basic.

5. Объектно-ориентированная графика.

6. Web-поддержка.

7. Возможность резервирования серверов в сети.

8. Доступ к базам данных осуществляется через OPC, исключая следующие СУБД: Oracle, CBase, Informix (через встроенные средства).

9. На русском языке какой-либо информации по данному приложению нет.

10. Представитель – компания Indosoft.

4. CI Tech

1. Данное приложение разработано в Австралии CI-Technologies. По данным на 2000 год, было произведено больше 35 тысяч инсталляций. Самое большое внедрение – это алмазный рудник в ЮАР, где 50 тысяч контролируемых регулируемых параметров. Приложение работает под любую версию ОС Windows. За 1 секунду приложение производит опрос 5 тысяч точек в сетевом режиме. Клиент-серверная технология

Основные характеристики:

1. Автоматическое обновление конфигурации по сети.

2. Одновременный доступ к данным из любой точки сети.

3. Возможность наращивания системы без изменения конфигурации.

4. Автоматический переход на резерв и восстановление.

5. «Горячее» резервирование.

6. Программирование на CI COM.

7. Поддержка OPC, DDE.

8. Наличие средств для разработки драйверов.

9. Поддержка ODBC и наличие прямого SQL-интерфейса.

10. Защита от несанкционированного доступа. Обеспечивается 8 уровней защиты по паролям для заданного участка. Защита осуществляется как для данных, так и для операций.

11. Наличие готовых шаблонов.

12. Стоимость исчисляется в зависимости от тегов, минимум 1200$. Среда разработки поставляется бесплатно.

5. Geni DAQ

Является SCADA-приложением для построения локальных систем сбора, анализа и визуализации информации. Основная особенность – открытая архитектура, позволяющая интегрировать с другими приложениями через механизмы OLE, DDE, ODBC. Разработано компанией Advantech. Именно эта особенность определила недостаток: данное SCADA-приложение поддерживает всю номенклатуру Advantech и небольшое количество других контроллеров. Geni DAQ относится к классу систем начального уровня, поэтому в нём в отличие от других SCADA-приложений отсутствуют следующие возможности:

- масштабирование системы (количество точек ограничено, и система не охватывает большие производственные участки);

- защита от несанкционированного доступа.

Следовательно, Geni DAQ используется для решения задач некритических ко времени опросов достаточно простой архитектуры.

Основные характеристики:

1. Лёгкий для освоения человеко-машинный интерфейс (HMI).
2. Объектно-ориентированная графика.
3. Программирование на Visual Basic.
4. Многозадачный режим работы.

5. Возможность генерации отчётов, использование истории.

6. Стоимость.

Geni DAQ выпускается в 3 различных номинациях:

Development Edition – 919,49 $

Runtime Edition – 257,99 $

Upgrade Edition – 522,59 $

От количества каналов стоимость не зависит, так как максимальное количество каналов достигает порядка 200.

7. Документация на английском языке, компания Prosoft осуществляет поддержку с русской стороны.

6. InTouch

Данное SCADA-приложение разработано компанией Wonderware и входит в состав программного комплекса Factory Suite, который предназначен для разработки систем автоматизации промышленных предприятий. Область применения: крупные промышленные предприятия металлургической, пищевой, машиностроительной индустрии. Количество инсталляций по всему миру составляет порядка 150 тысяч.

Основные характеристики:

1. Ориентировано на внешние приложения. То есть фактически не поддерживает встроенные системы драйверов. Поддерживает DDE, OLE, OPC.

2. Взаимодействует с СУБД PB, входящей в состав Factory Suit.

3. Удобный человеко-машинный интерфейс.

4. Наличие инструментов для графического отображения состояния процесса.

5. Отображение трендов в режиме PB и в режиме «истории».

6. Поддержка российскими компаниями-интеграторами.

7. Часть большой системы управления (Factory Suit).

8. Стоимость зависит от количества точек ввода/вывода.

Система бывает 2 видов:

InTouch Development – от 650 $

InTouch Runtime – от 1500 $

При приобретении осуществляется бесплатная поддержка в течение 1 года.

Другие SCADA-приложения

1. **Genesis**. Является широкоприменимым SCADA-приложением со встроенным микроядром. Разработано компанией Iconix. Часть контроллеров на уровне ядра поддерживает связь с Genesis.

2. **WinCC**. Разработано компанией Siemens. Поддерживает только контроллеры Siemens.

3. **Simplicity**. Разработано российской компанией GE Fanuc Automation.

Технология разработки SCADA-приложений

Основное назначение – построение человеко-машинного интерфейса.

Построение человеко-машинного интерфейса включает 2 шага:

1. Создание поддержки аппаратного обеспечения сетевых систем.

2. Поддержка связи с пользователем и СУБД.

При построении первого шага необходимо выполнить следующее условие: включить сеть контроллеров в информационную базу SCADA-приложений. Если это собственный контроллер, включить поддержку собственных контроллеров.

При построении второго шага необходимо реализовать:

1. Средства графического отображения состояния контроллера.

2. Возможность программирования данных контроллеров либо на собственном языке, либо на Visual Basic, Visual C.

3. Интерфейс СУБД (через ODBC или собственную БД).

Стоимость разработки SCADA-приложения состоит из следующих составляющих:

1. Фонд заработной платы сотрудников, занимающихся этим SCADA-приложением. При этом время разработки SCADA-приложения составляет примерно 1,5-2 года.

2. При реализации интерфейсов ODBC, OLE, DDE требуется наличие документов, подтверждающих право на использование этих технологий.

3. Необходимость сертификации SCADA-приложения, как средства, управляющего технологическим процессом или распределённой системой.

То есть дешевле купить SCADA-приложение, чем создавать собственное.

Базы данных РВ

Существует возможность в любой системе РВ сохранять информацию в структурированном виде. Данная особенность реализуется с помощью средства, называемого реляционной базой данных. Реляционная БД позволяет обеспечить доступ к информации с помощью языка SQL и позволяет хранить информацию в заданном пользователем виде. Для большинства систем, не являющихся системами «жёсткого» реального времени реляционная БД является оптимальной. Для систем «жёсткого» реального времени требуется обеспечить:

1. Высокоскоростной сбор информации.

2. Возможность сохранения больших объёмов информации.

3. Обеспечение доступа к информации с различных рабочих станций.

Для решения этих проблем были разработаны базы данных реального времени. Основным представителем БД РВ является Industrial SQL Server.

Особенности Industrial SQL Server:

1. Внутризаводский хранитель архивной информации включает данные о событиях и соответствующих реакциях. В Industrial SQL Server реализованы внутренние механизмы быстрого сбора информации на небольшом дисковом пространстве. Industrial SQL Server работает в 100 раз быстрее реляционной БД на аналогичной платформе.

2. Включение Industrial SQL Server в программный комплекс Factory Suit. Соответственно он напрямую имеет доступ к драйверу, что позволяет сохранить необходимую информацию.

3. Базовым ядром служит Microsoft SQL Server.

Схема реализации:

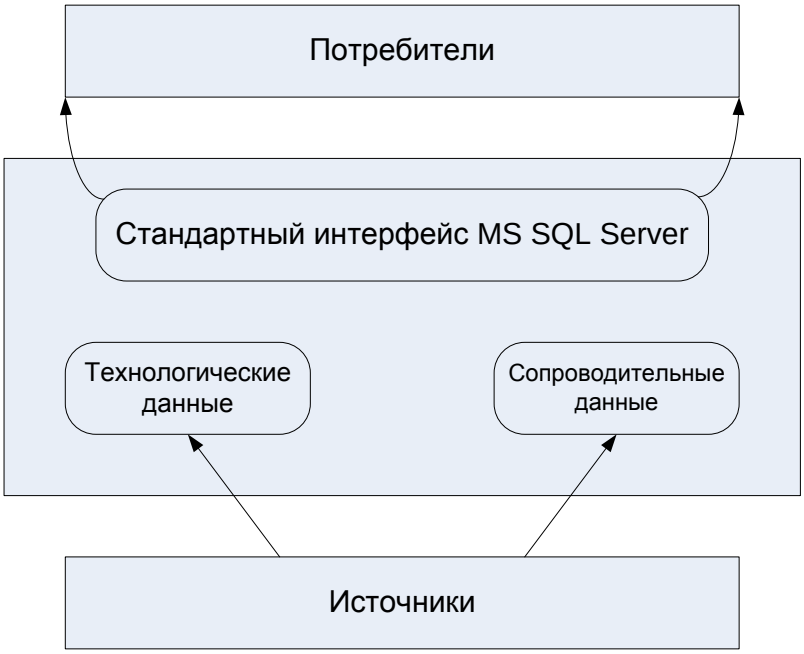
Потребители
Стандартный интерфейс MS SQL Server
Технологические данные
Сопроводительные данные
Источники

Рис. 31.

Принцип функционирования Industrial SQL Server:

Не критичная к РВ информация сохраняется в стандартных таблицах БД Microsoft SQL Server. Вся технологическая информация, получаемая с датчиков, сохраняется в специальных таблицах расширения. С помощью этого механизма поддерживается большая пропускная способность. Запросы пользователя осуществляются через стандартный интерфейс не в реальном времени. За счёт таблиц расширения поддерживается сохранение больших потоков информации, разделение пользовательских данных и данных с датчиков для исполнительных элементов. Внутренняя поддержка целостности информации. Управление пользовательскими таблицами осуществляется через стандартный интерфейс, семантическая и ссылочная целостность информации находится на контроле пользователя. Создание технологических таблиц осуществляется через SCADA-приложение InTouch. Соответственно целостность данных контролируется самой системой, и пользователь на эту целостность никак не влияет. Industrial SQL Server построен на архитектуре «клиент-сервер».

Функциональные возможности сервера базы данных

1. Высокопроизводительный сервер.

Достигается путём возможности масштабирования системы, а также многоуровневой клиент-серверной архитектурой. Фактически источником данных является SCADA-приложение с датчиками и исполнительными механизмами. А приёмником служит АСУП.

2. Уменьшение объёма хранения информации.

Выполняется за счёт механизмов упаковки информации, приём сжатия.

3. Отсутствие всевозможных алгоритмов защиты данных.

4. Защита данных осуществляется за счёт средств SCADA и АСУП. Например, 2-месячный архив с 4000 параметров, которые регистрируются каждые 10 сек., занимает 2 мБ.

5. Достоверная информация.

Достигается путём того, что Industrial SQL Server входит в состав Factory Suit. Позволяет одновременно сохранить порядка 800 значений. Для проверки достоверности полученная информация с датчиков сравнивается со следующей информацией:

- конфигурация;
- операционная информация;
- сведения о событиях;
- аварийная информация;
- информация системы контроля;
- технологическая информация.

Соответственно объединение этой информации представляет пользователю структуру всего технологического процесса. При доступе и изменении информации сохраняется предыстория.

6. Сервер реального времени.

Реализован через протокол обмена (Factory Suit), называемый SuiteLink. Протокол характеризуется наличием меток времени, по которым осуществляется передача информации, а также обеспечение качества передаваемой информации.

7. Система регистрации событий.

Касается сохранения данных, поступающих непрерывно. Сохранение данной информации осуществляется по методу событий. Событие служит переключателем между массивами данных.

8. Гибкий открытый доступ.

Доступ к открытой информации со стороны пользователя осуществляется через стандартный интерфейс Microsoft SQL Server, который также определяет группу пользователей и права пользователей, следовательно, обеспечивается защита технологической информации и доступ к любой информации.

9. SQL с поддержкой временных параметров.

Эта поддержка осуществляется добавлением в SQL-запрос временных характеристик (по принципу задачи).

10. Открытая и гибкая БД.

Industrial SQL Server поддерживает доступ к информации PB, а также к архивным и конфигурационным данным. Пользователю доступны следующие типы данных:

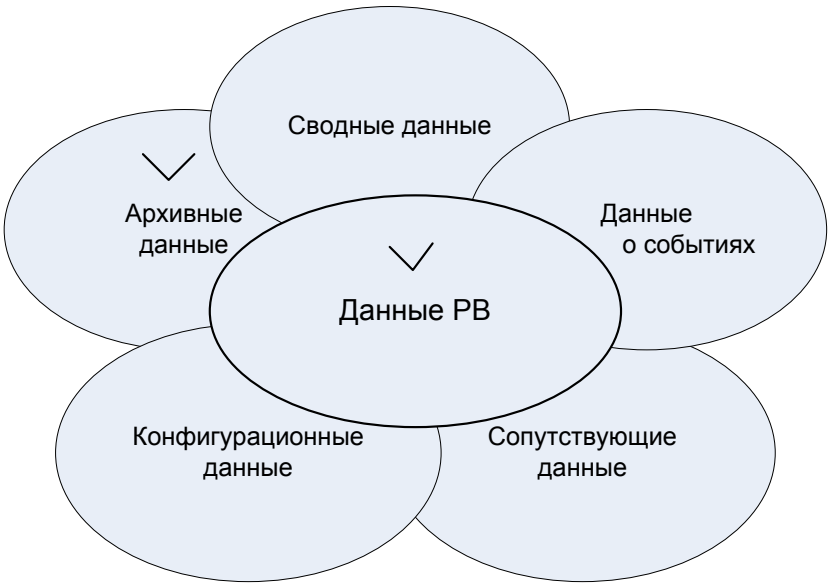

Рис. 34.

⩒- эти данные относятся к Industrial SQL Server. Остальные – к Microsoft SQL Server.

Выделяют следующие области применения:

1. Анализ протекания процесса. Диагностика, оптимизация.
2. Управление запасами (управление потреблением сырья).

3. Техническое обслуживание систем. Анализ системы для предварительного и превентивного ремонта.

4. Контроль качества выпускаемой продукции.

5. Функционирование в качестве системы управления производственным процессом.

10. Простота использования.

Заключается в том, что Industrial SQL Server не требует от пользователя знания SQL-языка. Он ориентирован на готовые наборы функций. Управление резервированием и управление базой данных осуществляется средствами Microsoft SQL.

11. Интеграция с другими компонентами комплекса.

Выполняется с помощью стандартных средств операционной системы: RTFX, DDE, SuitLink. Интеграция производится со следующими компонентами:

- SCADA-приложения InTouch;
- система контроля ввода данных;
- система доступа к архивной информации.

12. Возможность организации клиент-серверной системы.

Вычислительный узел, на котором установлено Industrial SQL Server, может являться сервером для других приложений (приложений сторонних производителей). Сервер может быть организован как для технологического процесса, так и для уровня пользовательских приложений. Доступ может обеспечиваться через Internet.

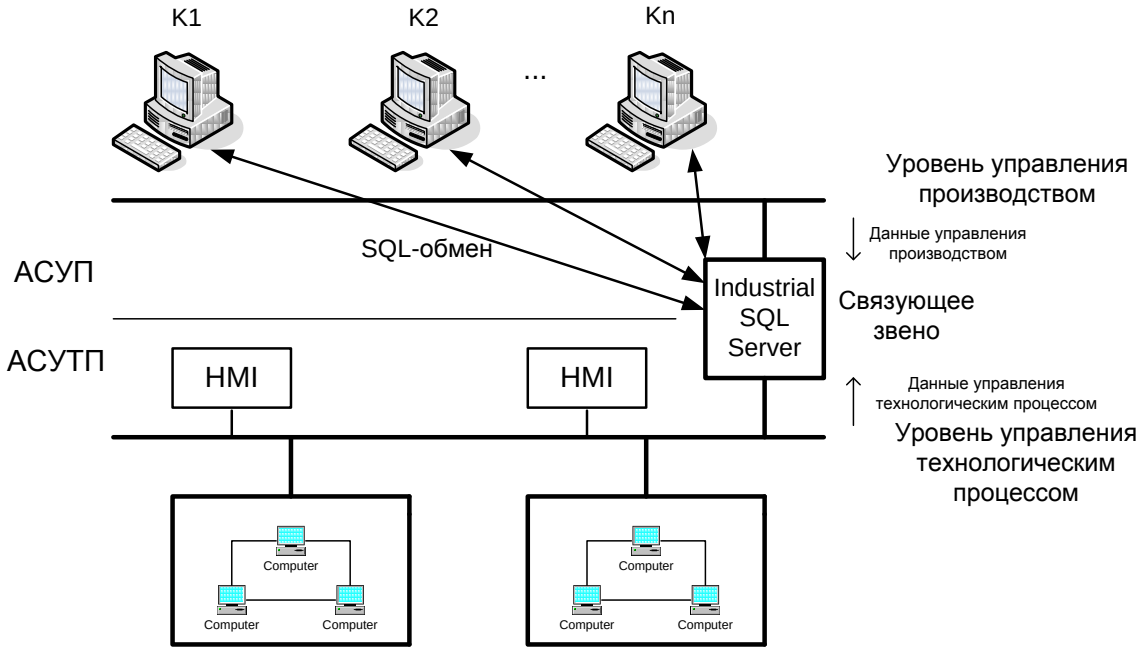

Рис. 35.

Существует возможность отправки сообщений по электронной почте, а также возможность коммуникаций с другими СУБД.

13. Возможность расширения.

Конечные пользователи для данной СУБД могут разрабатывать собственные специализированные программы, необходимые для анализа и представления данных. Для этого используется интерфейс ODBC.

Данные функциональные возможности позволяет использовать Industrial SQL Server в режиме РВ.

Комплексные программные средства разработки приложений РВ.

Комплексный программный продукт – это система, позволяющая автоматизировать уровень производства и уровень технологического процесса. Соответственно, данная система должна представлять информацию для руководителя предприятия и для отделов разработки и эксплуатации.

2 наиболее популярных продукта в этой области:

Wiz Factory (Wizcon)

Factory Suite (InTouch)

(В скобках указаны SCADA-приложения).

Программный комплекс – это комплекс для эффективного управления промышленным процессом, для неразрывной связи его с бизнес - менеджментом предприятия. Фактически, это представление информации о технологическом процессе руководству предприятия в удобном виде и необходимом формате.

Назначение продукта – анализ производства в целом и моделирование отдельных этапов.

Моделирование – это создание прогноза по внедрению основных средств предприятия на базе существующей и накопленной информации.

Продукт позволяет создать стратегию развития предприятия.

Состав комплексных продуктов:

1. SCADA-приложение, которое выводит информацию о технологических процессах, протекающих на предприятии.

2. Реляционная база данных.

3. Система для управления контроллерным оборудованием и процессами – средство для формирования управляющих воздействий на основе решения руководства. Позволяет строить диаграммы релейной логики и диаграммы функциональных схем.

4. Средство просмотра данных (локально и через Internet). Должно иметь возможность подключения к нескольким узлам одновременно.

5. Система управления производством. Должна позволять создавать прикладные программы управления производством, моделируя и отслеживая каждую стадию производственного процесса. Система должна управлять последовательностью работ. Связь с другими приложениями через OLE.

Основное применение.

Автоматизация сложных и больших производств (нефтедобыча, газодобыча, производство потребительской продукции, система управления транспортным движением, измерительно-информационные системы).

Инструменты разработчиков системы автоматизации.

Основная тенденция развития данных средств заключается в применении CASE-подхода для создания автоматизированных систем. CASE-подход должен включать в себя возможность применения стандартных средств как на конечных узлах, так и на узлах оператора. Его применение заключается в исключении разработчика из процесса создания микропрограммного обеспечения и представление пользователю стандартных общепринятых средств управления технологическим оборудованием.

Достоинства:

1. Гибкость и универсальность подхода создания любой системы автоматизации и любой сложности.

2. Простота обучения.

3. Независимость от аппаратной платформы.

4. Возможность распределённой разработки.

Недостатки:

1. Невозможность выявления ошибок микропрограммного обеспечения на стадии тестирования системы.

2. Невозможность дополнения системы функциями пользователя.

3. Высокая стоимость узлов средств разработки

Isagraph Pro - компонент, послуживший средством появления таких систем. Это средство включает 3 уровня обеспечения функционирования системы:

- Базовый уровень.
- Уровень переносимости функций.
- Уровень переносимости приложений.

Эти 3 уровня позволяет применять одни и те же решения для нескольких различных задач.

Базовый уровень предполагает, что системы совместимы на некотором подмножестве базовых компонентов, определяемых стандартом. К этим базовым компонентам относятся типы переменных, языковые конструкции, исходные тексты. Совместимость обеспечивается производителями контроллеров. Они должны стремиться к стандарту «1131».

«1131» - это система, реализующая возможность программирования на 5 стандартных языках. Разработана в 1993 году независимой международной организацией «PLC Open».

Уровень переносимости функций - уровень совместимости функций и функциональных блоков между различными системами. Для этой цели создаётся специальный формат файлов обмена.

Уровень переносимости приложений определяет степень совместимости различных реализаций в различных системах.

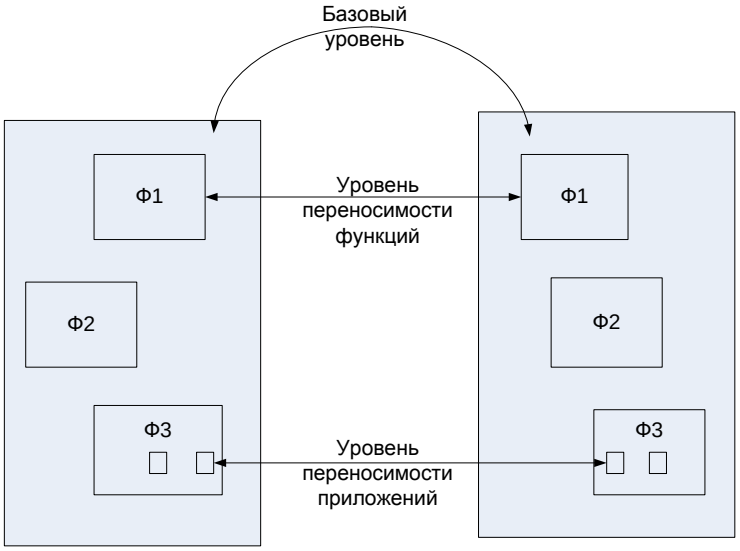

Рис. 36.

Возможности IsaGraph Pro.

Позволяет, используя базовые функции, являющиеся частью контроллера, передавать в различные человеко-машинные интерфейсы через уровень приложений. Для этой цели встроено приложение Genesis 32.

Состав системы IsaGraph Pro.

Управление осуществляется с помощью ядра IsaGraph Pro визуальной машины IsaVM.

Рис. 37.

Системный компонент позволяет создать связь с ОС с обеспечением функций.

Компонент связывания ресурсов позволяет объединять все промышленные элементы разрабатываемой системы.

Сервер обслуживания запросов позволяет обеспечить взаимодействие пользователя с разработанной системой автоматизации.

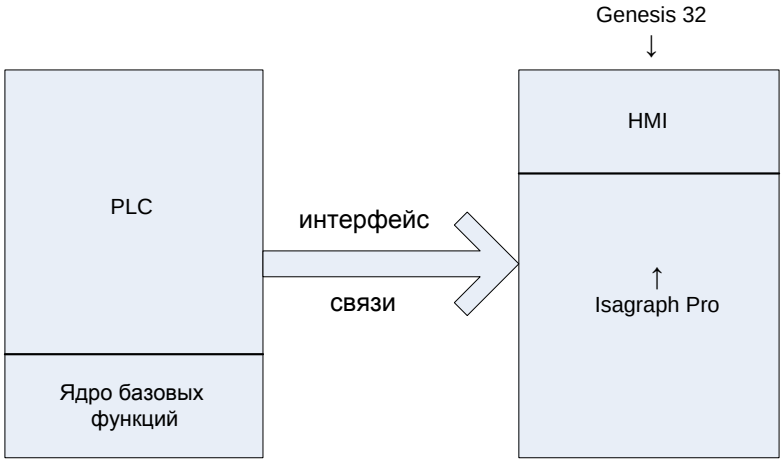

Рис. 38.

ПРИЛОЖЕНИЕ

Лабораторная работа №1

«Разработка приложений для систем автоматизации на базе промышленного Ethernet»

Необходимо знать:

- Принципы программирования для WEB
- Организацию интерфейса узлов в сети Ethernet
- Состав лабораторного оборудования
- Интерфейс между узлами и исполнительными элементами/датчиками
- Программирование на GGI
- Принципы конфигурирования узлов
- Основы программирования для сетевых приложений

Задание на лабораторную работу «Разработка приложений для систем автоматизации на базе промышленного Ethernet»

1. Спроектировать приложение для контроллера
2. Написать программу для IPC Чипа, согласно заданию
3. Загрузить программу на контроллер – IPC Чип
4. Подготовить отчет для сдачи лабораторной работы
5. Содержание отчета: Описание задания, Предполагаемое решение, Программа с комментариями, Описание основных функций. Описание стандартных функций из библиотек IPC Чип и GGI, применяемых в программе.

Варианты заданий:

1. Разработать утилиту для создания текстовых файлов (telnet)
2. Разработать GGI-программу для создания текстовых файлов

3. Создать программу для чтения и исполнения текстовых файлов с описаниями ввода-вывода в порты

4. Разработать программу для чтения портов. Результаты – в файл. (telnet)

5. Написать GGI-программу для чтения текстовых файлов из браузера

6. Разработать многозадачную программу для работы с портами: задача 1 – запись в порт данных с консоли (telnet); задача 2 – ежесекундный вывод текущего состояния порта (в файл или на консоль)

7. Создать приложение для чтения данных с термометра (шина I2C) и запись в файл

8. Разработать программу управления WEB-камерой из браузера (GGI)

9. Разработать программу управления WEB-камерой по программе из файла

10. Создать приложение - WEB-чат на языке GGI

11. Разработать программу управления сигнализацией в помещении

12. Написать программу – счетчик помещений сайта (GGI), запись в файл статистики

13. Создать программу контроля температуры с применением соединения по telnet

14. Создать программу контроля температуры по программе из файла

15. Создать программу контроля температуры из браузера на языке GGI

16. Построить приложение о принятии решения о вкл/выкл внешнего устройства по голосованию из Интернета с применением языка GGI

17. Разработать программу мониторинга состояния внешних устройств на языке GGI

18. Показать в браузере названия всех запущенных GGI с применением функций языка GGI

19. Разработать WEB-browsing по каталогам чипа (WEB-оболочка для RTOS)

20. Создать программу вывода информации встроенными световыми индикаторами (разработать протокол или использовать азбуку Морзе)

21. Создать программу ввода информации внешними переключателями (разработать протокол или использовать азбуку Морзе)

22. Разработать протокол RTOS для пользователя на базе варианта 20

23. Разработать оболочку RTOS для пользователя на базе варианта 21

24. Разработать программу передачи информации о состоянии контактов IPC@Chip по команде от пользователя (telnet)

25. Разработать программу передачи информации о состоянии контактов IPC@Chip по команде от пользователя (web-интерфейс)

26. Разработать программу синхронизации часов ЧИПА с указанным – ПК (по IP адресу) (из консоли ЧИПА)

27. Разработать комплексное приложение для 2 чипов, обменивающихся информацией о состоянии контактов. (Замыкание

контакта на 1 чипе должно сгенерировать включение инверсного контакта на втором чипе) (1-8, 2-7 3-6 5-4)

28. Разработать приложение вывода на экран пользователя состояния контактов в режиме реального времени. (web)

29. Создать программу управления памятью IPC@Chip – (вывод информации о состоянии памяти, сохранение данных, очистка памяти по запросу пользователя) (web)

30. Разработать приложение вывода информации обо всех IPC@Chip найденных в сети (web) (по IP таблице)

31. Разработать программу обмена информацией между IPC@Chip по RS232 интерфейсу

32. Разработать приложение отображающее состояние элементов стенда охраны (web приложение)

33. Разработать программу передачи на IPC@Chip с ПК управляющих команд на включение/отключение контактов. (web)

34. Разработать программу передачи на IPC@Chip с ПК управляющих команд на включение/отключение контактов. (telnet)

35. Разработать программу передачи на IPC@Chip с ПК управляющих команд на включение/отключение контактов из внешней программы ПК (OPC, ORB)

36. Разработать программу управления контактами IPC@Chip из web-интерфейса (создаем файл, грузим в чип + исполняем)

37. Управление шаговым электродвигателем с двух кнопок

38. Утилита передачи файла между двумя чипами через последовательный интерфейс

39. «Переговорное устройство». Снятие сэмплов с контактов чипа с заданной частотой, передача потока данных по сети, выдача их на контакты другого чипа с такой же частотой

40. Разработка клиента для конфигурации чипа. Клиент должен слушать заданный TCP порт, принимать команды на изменение параметра в chip.ini, выдачу значения параметра по имени

41. Калькулятор с cgi-интерфейсом

42. Управляемый преобразователь частоты до 10Кгц

43. «Кодовый замок» 5 входов под кнопки,1 выход – открытие двери, 1 выход – тревога. Хранение базы пользователей в памяти, изменение базы и передача сигнала «тревога» через telnet

44. «Считыватель телефонных карт» См. Описание стандарта ISO 7816

45. Служба «chargen»

46. Служба «echo»

47. Служба «discard»

48. Служба «time». См. RFC 868

49. Редирект. Получение данных с одного TCP порта и перенаправление их на другой

50. Простой прокси сервер

51. Отправка e-mail сообщения при изменении состояния контактов

52. Изменение состояния контактов при получении e-mail сообщения

53. Поиск чипов в сегменте сети и составление таблицы их параметров с записью в файл

54. «Электронный шпион». Составление графика работы компьютера с указанным IP адресом.

Лабораторная работа №2

Наименование «Разработка планировщика периодических и спорадических задач»

Описание задания:

1. Существует структура задачи в виде объекта, обладающего следующими свойствами:
   a. Имя,
   b. Тип,
   c. Период,
   d. Крайний критический срок,
   e. Время запуска,
   f. Время исполнения,
   g. Приоритет.
2. Необходимо разработать планировщик задач, который на основании таблиц задач строит список исполнения задач и запускает их в соответствии с полученным списком.
3. В приложении должно существовать от 10 до 100 задач разного типа.
4. Типы задач определяются пользователем в зависимости от системы управления, определенной заданием. (3-4 типа задач).
5. Экземпляры задач создаются пользователем и должны сохраняться в конфигурационном файле. Минимальное количество задач должно быть определено по умолчанию.
6. Каждый тип задач должен в интерфейсе пользователя выводить графическую или текстовую информацию.
7. В интерфейсе пользователя должны быть реализованы следующие функции управления:
   a. Запустить планировщик

b. Остановить планировщик

c. Добавить задачу в таблицу задач

d. Удалить задачу из списка задач

e. Изменить свойства задачи

f. Запустить задачу

g. Остановить задачу

8. Планировщик реализуется отдельно для периодических и спорадических задач в соответствии с зданием.

9. Реализация программы должна осуществляться в любой среде визуального программирования.

10. Необходимо определить такт функционирования планировщика. Такт его функционирования необходимо разделить на метки. Для каждой метки устанавливается относительное время запуска (от начала такта планировщика) и список исполняемых задач

a. Установка списка исполняемых задач осуществляется в зависимости от свойств задач (крайний критический срок исполнения, время запуска, время исполнения)

b. Для каждой метки должен быть построен собственный список задач, по которому строится общий список и осуществляется исполнение задач.

Постановка задачи

1. Необходимо разработать приложение для планирования периодических и спорадических задач

2. Необходимо разработать планировщик периодических задач по алгоритму из задания

3. Необходимо разработать планировщик спорадических задач по алгоритму из задания

4. Реализовать интерфейс пользователя для управления задачами

5. Подготовить отчет для сдачи лабораторной работы

Структура отчета

1. Титульный лист с указанием номера варианта
2. Описание задания.
3. Описание типа «Задача»
4. Описание алгоритма планирования периодических задач
5. Описание алгоритма планирования спорадических задач
6. Описание планировщика заданий.
7. Описание основных функций приложения.

Варианты заданий (в скобках примерные варианты типов задач):

1. Система измерения температуры (измерение температуры, Перевод из мВольт в градусы Цельсия, Контроль температуры)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Деферабельный сервер

2. Система измерения освещенности и выдачи управляющих сигналов (Измерение уровня освещенности, выдача сигнала на включение, выдача сигнализации)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

3. Система измерения давления (Измерение, Контроль, Выдача в 3 цикла 2- предварительное, 3 - настоящее)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Обмен приоритетом

4. Система работы с данными (заполнение массива данных, выполнение однотипной операции над данными, выдача значений)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Спорадический сервер

5. Система перекрестного контроля параметра (Измерение параметра группы 1, Измерение параметра группы 2, сравнение параметров и выдача результата)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Выбор

6. Система изменения состояния объекта (Измерение температуры, построение объекта «квадрат», заполнение объекта в зависимости от температуры соответствующим цветом от синего до красного)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

7. Система контроля климата (включение вентилятора, включение обогревателя, в зависимости от температуры - измеренной)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

8. Система управления светофорным объектом (изменение цвета, переключение режима)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Спорадический сервер

9. Система управления освещением (Включение/Выключение, регулирование яркости, переключение режимов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Обмен приоритетом

10. Система управления АЗС (вкл/выкл насосов, выдать режим обслуживания, изменить режим обслуживания)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Выбор

11. Система измерения вибраций на объекте (контроль по модели, анализ опасных и повышенных вибраций, сигнализация вибраций)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач — Планирование как задачи фона (динамической)

12. Система управления гирляндой (смена режимов, переключение ламп, смена режимов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Деферабельный сервер

13. Система управления рекламным щитом (управление подсветкой, изменение рекламы, выдача сообщений)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Обмен приоритетом

14. Система управления питанием на объекте (переключение по 2 источникам и 2 фазам, анализ загруженности, анализ качества)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Выбор

15. Система управления охранной сигнализацией (Анализ состояния датчиков - Движения и Открытия, выдача сигнализации на сирены и пульт)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

16. Система управления камерами наблюдения (задать угол поворота, задать время поворота, вывести состояние)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

17. Система управления автомобилем (управление расходом топлива, анализ скорости движения, анализ степени нажатия на педаль газа, анализ оборотов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Спорадический сервер

18. Система управления климатом (анализ температуры, анализ влажности, включение/выключение вентилятора и нагревателя)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Деферабельный сервер

19. Система управление выводом и вводом данных (чтение данных с клавиатуры, выдача сигналов на экран, чтение по группам (цифры, буквы), формирование слов)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Спорадический сервер

20. Система управления входом в помещение (выдача состояния дверей, открытие, закрытие, световая сигнализация)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Обмен приоритетом

21. Система измерения освещением (через диммер установка значения 1..12, выдача информации на другие объекты, поддержка освещенности на заданном уровне)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

22. Система управления переключения между камерами наблюдения (переключение режима, включение/выключение, сохранить данные)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

23. Система управления конвейером (Движение, Смена операций, Запуск)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Выбор

24. Система изменения освещенности объекта (Изменение степени освещенности, скорость изменения цвета, вывод состояния)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

25. Система управления движение ж/д транспорта на станции (переключение семафора, перевод стрелок, вывод состояния пути)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Обмен приоритетом

26. Система пожарной сигнализации (анализ датчиков дыма, анализ датчиков огня, выдача световой индикации, вывод состояния на пульт)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Деферабельный сервер

27. Система контроля прохода судов через шлюз (поиск свободного пути, выбор направления суда, вывод информации о судах)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

28. Система формирования «зеленой полосы» на автомобильной дороге по светофорным объектам (Вывод сигнала светофора, оценка потока, вывод информации)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Выбор

29. Система учета электрической энергии (измерение, выявление пиковой нагрузки, выдача сигнализации о пиках)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Обмен приоритетом

30. Система контроля трубопровода нефти (измерение потока в сечениях, включение насосов, выдача информации о рассогласовании потоков)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Спорадический сервер

31. Система управления освещением коттеджа (включение, выключение, задание свойств, изменение яркости)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

32. Система контроля движения транспортного средства (контроль закрытия/открытия дверей, исправности частей и контроль давления на подножки, выдача сигнализации)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Выбор

33. Система управления документооборотом (поиск документа, добавление документа, архивирование документа)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

34. Система управления водоснабжением (измерение потока жидкости, формирование квитанций, контроль утечек)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Деферабельный сервер

35. Система управления БД (добавление данных, удаление данных, создание мета данных)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Спорадический сервер

36. Система управления кадрами на предприятии (расчет з/п, учет рабочего времени, поиск, изменение)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Деферабельный сервер

37. Система проверки очистных сооружений (измерение качества, измерение содержания вредных компонентов, выдача сигнализации, формирование отчета)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Спорадический сервер

38. Система проверки работоспособности оборудования (контроль цепей, проведение проверки, выдача результата)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Обмен приоритетом

39. Система управление аудиториями (проверка свободных, проверка потребностей в аудиториях, формирование запросов, выдача результата)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

40. Система управления филиалами (подготовка отчета, проверка кадров, выдача результата деятельности)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Спорадический сервер

41. Система управления звуковым потоком (выдача сигнала, смешивание сигнала, анализ поступающего сигнала, проверка соответствия сигналов)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Деферабельный сервер

42. Система учета тепла (измерения, расчет, выдача квитанции)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

43. Система управления соединением на телефонной станции (включение/выключение связи, выдача задолженности, оценка времени разговора)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Обмен приоритетом

44. Система управление трафиком сети (контроль, отключение, сигнализация)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Спорадический сервер

45. Система управления складом (Прием потребностей, контроль остатков, резервация, выдача сообщений)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Выбор

46. Система управления производством автомобилей (Передача с участка на участок, контроль соединения деталей, выдача сигнализации)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

47. Система управления объектом (задать информацию через экран, выдать информации на экран, анализ клавиатуры промышленного узла)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

48. Система выдачи графиков угла (sin, cos, tg, ctg)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Деферабельный сервер

49. Система управления геометрическими фигурами (изменение типа (квадрат, треугольник, круг), изменение типа линии, изменение цвета линии)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Обмен приоритетом

50. Система управления охранной сигнализации административного здания (контроль датчиков, контроль периметра,

контроль камер наблюдения, выдача световой индикации на пульте оператора)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Выбор

51. Система управления перекрестком (Анализ загруженности линий, переключение сигналов, вывод информации на световые табло)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Выбор

52. Система управление лесопилкой (Измерение объема полученного материала, измерение объема поданного материала, контроль качества полученного материала)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Деферабельный сервер

53. Система управления теплицей (измерение температуры по участкам, измерение освещенности по участкам. Выдача сигнализации)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

54. Система контроля загруженности аудиторий (проверка расписания групп, сортировка аудиторий по типам, выдача информации о свободных аудиториях по каждой паре каждого дня)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

55. Система контроля функционирования шахты (объем добываемого угля с участка, отгрузка угла по направлениям, контроль аварийности участок по признакам наличия опасных газов, выдача статистики)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Спорадический сервер

56. Система управления потоком автомобилей на дороге (выдача рекомендуемых направлений движения, оценка движения на участках дорого по загруженности, оценка аварийности, выдача статистики)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

57. Система контроля работоспособности системы производства газа (проверка давления, проверка исходящего потока, проверка входящего потока, выдача сигнализации)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Спорадический сервер

58. Система контроля мишенного поля (контроль попадания мишени, подъем мишени, опускание мишени, выдача результата о попадании)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Деферабельный сервер

59. Система выдачи графиков по времени (sin, cos, tg, ctg)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Спорадический сервер

60. Система допускового и перекрестного контроля параметров (проверка одиночного параметра ft.min и max значение, проверка рассогласование параметров с двух измерений, выдача сигнализации)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Выбор

61. Система контроля записи информации на носители (архивирование данных, передача на сетевые узлы, выдача статистики функционирования, запуск системы по времени)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Деферабельный сервер

62. Система управление производством (оценка потребности в материалах, оценка эффективности, формирование отчетов о производстве, передача документации по отделам)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Деферабельный сервер

63. Система передачи телеграфных сообщений (передача телеграмм, формирование ответа о получение, выдача уведомления о получении телеграммы, формирование отчета по переданным телеграммам)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Обмен приоритетом

64. Система контроля проведения экспертиз автомобилей (учет стоимости работ, учет стоимости материалов, оценка степени износа объекта, формирование отчетов)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

65. Система управления контроля успеваемости студентов (выдача экзаменационных ведомостей, занесение результатов сдачи экзамена, формирование статистики по сданным предметам, формирование списка задолжников, передача информации преподавателям по e-mail)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Выбор

66. Система управление вывозом мусора (выбор объектов по наименованию отходов, оценка пути движения, выбор пути следования, формирование отчетов, оценка стоимости перевозки)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

67. Система синхронизации времени на ПК (запрос серверов времени, оценка среднего времени на системе, выдача времени для установки его на различных ПК, выдача статистики запросов)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

68. Система контроля продукции предприятий (выбор предприятий для контроля, формирование статистики о качестве товара (по замерам), формирование степени качества товаров предприятия)

a. Планировщик периодических задач - LSTF

b. Планировщик спорадических задач - Обмен приоритетом

69. Система контроля печати на сетевых принтерах (контроль узлов, осуществляющих печать, контроль пользователей, оценка количества напечатанных страниц, выдача статистики

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Обмен приоритетом

70. Система управления положением фигур (изменение места (квадрат, треугольник, круг), изменение скорости движения линии, изменение цвета линии,)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Планирование как задачи фона (статический)

71. Система управления бильярдной (контроль времени игры на столе, контроль тарифа в зависимости от времени, выдача стоимости игры, формирование статистики)

a. Планировщик периодических задач - RM

b. Планировщик спорадических задач - Выбор

72. Система учета водоснабжения на объектах (контроль подачи воды, оценка параметров горячей воды, формирование отчетов по стоимости по объектам, выдача сигнализации об аварийных ситуациях)

a. Планировщик периодических задач - EDF

b. Планировщик спорадических задач - Планирование как задачи фона (динамической)

Лабораторная работа №3

Наименование «Расчет автоматизированной системы управления на базе современного протокола передачи данных на возможность функционирования в режиме реального времени»

Рассчитать систему автоматизации на возможность работы в режиме «реального времени». Расчет производится по методическому пособию для студентов 5 курса специальности АСУ по предмету «Системы реального времени».

Задание на лабораторную работу.

Необходимо произвести расчет для спроектированной системы управления. В процессе расчета необходимо

1. Выделить подсистемы (от 5 до 7). Каждая подсистема представляет собой отдельный модуль управления. Если в системе присутствуют «типовые» узлы, они выделяются в отдельную подсистему.
2. Для каждой подсистемы определить набор входных и выходных сообщений, а также источники и преемники данной информации.
3. На основании выделенных входных и выходных данных необходимо построить таблицу сообщений, включающих следующие данные:
   a. Номер сообщения
   b. Источник сообщения (наименование)
   c. Приемник сообщения (наименование)
   d. Размер сообщения (в битах)
   e. Тип сообщения (Периодический, Апериодический, Спорадический)
   f. Периодичность сообщения (в единицах времени, T)

g. Крайний критический срок исполнения (в единицах времени, D)

h. Задержка отправки сообщения в очередь (jitter в единицах времени, J)

i. Размер пакета при передачи данных, включая «накладные расходы»

j. Приоритет сообщения (вычисляется, в зависимости от используемого метода назначения приоритета)

4. На основании таблицы исходных данных, производится расчет максимальной задержки передачи сообщения, от источника до приемника для каждого сообщения. (R). Расчет производится для четырех скоростей передачи данных. (По указанном в варианте - протоколу передачи данных)

5. По полученным результатам строится таблица 2, включающая информацию о задержках передачи сообщения

a. Номер сообщения

b. Источник сообщения (наименование)

c. Приемник сообщения (наименование)

d. Размер сообщения (в битах)

e. Задержка для скорости 1 (в единицах времени, R1)

f. Задержка для скорости 2 (в единицах времени, R2)

g. Задержка для скорости 3 (в единицах времени, R3)

h. Задержка для скорости 4 (в единицах времени, R4)

6. По результатам таблицы 2 формируется вывод о системе. Для оценки о системе требуется вычисление трех параметров:

a. Message Utilization – Коэффициент передачи полезных данных в сообщении (%)

b. Bus Utilization – Коэффициент передачи всех данных в сообщении (%)

c. Breakdown Utilization – Коэффициент возможности увеличения размеров сообщения. (коэффициент расписабельности)

7. В случае, если коэффициент Breakdown Utilization<1, необходимо произвести оптимизацию системы методом объединения коротких сообщений.

Этапы выполнения лабораторной работы следующие:

1. Проектирование системы. При проектировании необходимо выделить подсистемы и сообщения в системе, а также произвести словесное описание функционирования системы при заданных условиях.

2. Расчет автоматизированной системы на критерий функционирования в режиме реального времени. Расчет производится по параметру R(J,D,T), а также на основании информации по протоколу передачи данных.

3. Оптимизация автоматизированной системы для увеличения коэффициента Breakdown Utilization. (Оптимизация выполняется методом объединения коротких сообщений)

4. Подготовка отчета. Состав отчета:

a. Проектирование системы с обоснованием разделения системы (почему именно так).

b. Каждое сообщение системы должно содержать информацию. (Между такими подсистемами и зачем оно было нужно).

c. Словесное описание функционирования системы

d. Характеристики протокола передачи данных, с включением общего описания технологии, скоростей передачи данных, накладных расходов, применение протокола передачи данных.

e. Расчет автоматизированной системы, с указанием таблиц 1, таблиц 2, формул расчета.

f. Оптимизация расчета (при необходимости).

g. Выводы по расчетам (возможность функционирования, плюсы автоматизации)

Данные по протоколам передачи данных

Наименование протокола	Скорость передачи данных	Накладные расходы + восстановление при сбоях	Разработчик
EIB	9,6 кБит/сек 100 кБит/сек 500 кБит/сек 1200 кБит/сек	70 бит + 6 бит	Siemens
LonWorks	78,6 кБит/сек 125 кБит/сек 250 кБит/сек 1200 кБит/сек	84 бит + 10 бит	Echelon
CAN	125 кБит/сек 250 кБит/сек 500 кБит/сек 1000 кБит/сек	35 бит + 5 бит	Bosh
P-NET	56,6 кБит/сек 78,6 кБит/сек 120 кБит/сек 240 кБит/сек	24 бит + 5 бит	Process DATA
TCP-IP	56,6 кБит/сек 1000 кБит/сек 2000 кБит/сек 10000 кБит/сек	48 бит + 5 бит	
Profibus	9,6 кБит/сек 12,4 кБит/сек 56,6 кБит/сек 78,6 кБит/сек	32 бит + 5 бит	

Задание состоит из двух частей (наименование системы автоматизации, в скобках указана базовая технология, на основе которой должна быть реализована система автоматизации.)

Варианты заданий:

1. АЗС (P-NET)
2. Биологическая лаборатория (LonWorks)
3. Тепловая машина (P-NET)

4. Автопилот (CAN)
5. АТС (LonWorks)
6. Здание (EIB)
7. Прогноз погоды (TCP-IP)
8. Сточные воды (P-NET)
9. Управление освещением в помещении (EIB)
10. Управление движением автобуса (LonWorks)
11. Охранная система квартиры (EIB)
12. Система управление движением на перекрестке с разной интенсивностью движения, связь с другими перекрестками (TCP-IP)
13. Безопасность движения на автомобиле (CAN)
14. Климат в помещении (LonWorks)
15. Охрана гаражного кооператива (TCP-IP)
16. Пропускная система предприятия (EIB)
17. Управление светофорным объектом (P-NET)
18. Управление ж/д станцией (P-NET)
19. Управление бильярдом (TCP-IP)
20. Управление мишенями на стрельбище (LonWorks)
21. Автоматизированное кафе (CAN)
22. АЗС (CAN)
23. Биологическая лаборатория (EIB)
24. Тепловая машина (TCP-IP)
25. Автопилот (LonWorks)
26. АТС (EIB)
27. Здание (TCP-IP)
28. Прогноз погоды (CAN)
29. Сточные воды (TCP-IP)
30. Управление освещением в помещении (LonWorks)

31. Управление движением автобуса (P-NET)
32. Охранная система квартиры (TCP-IP)
33. Безопасность движения на автомобиле (P-NET)
34. Климат в помещении (EIB)
35. Охрана гаражного кооператива (P-NET)
36. Пропускная система предприятия (CAN)
37. Управление светофорным объектом (TCP-IP)
38. Управление ж/д станцией (TCP-IP)
39. Управление бильярдом (CAN)
40. Управление мишенями на стрельбище (EIB)
41. Автоматизированное кафе (LonWorks)
42. АЗС (TCP-IP)
43. Биологическая лаборатория (CAN)
44. Тепловая машина (LonWorks)
45. Автопилот (TCP-IP)
46. АТС (CAN)
47. Здание (CAN)
48. Прогноз погоды (LonWorks)
49. Сточные воды (CAN)
50. Управление освещением в помещении (CAN)
51. Управление движением автобуса (TCP-IP)
52. Охранная система квартиры (LonWorks)
53. Безопасность движения на автомобиле (TCP-IP)
54. Климат в помещении (CAN)
55. Охрана гаражного кооператива (EIB)
56. Пропускная система предприятия (LonWorks)
57. Управление светофорным объектом (CAN)
58. Управление ж/д станцией (EIB)

59. Управление бильярдом (LonWorks)
60. Управление мишенями на стрельбище (TCP-IP)
61. Автоматизированное кафе (EIB)
62. АЗС (LonWorks)
63. Биологическая лаборатория (TCP-IP)
64. Тепловая машина (CAN)
65. Автопилот (EIB)
66. АТС (P-NET)
67. Здание (LonWorks)
68. Прогноз погоды (EIB)
69. Сточные воды (LonWorks)
70. Управление освещением в помещении (TCP-IP)
71. Управление движением автобуса (CAN)
72. Охранная система квартиры (P-NET)
73. Безопасность движения на автомобиле (LonWorks)
74. Климат в помещении (TCP-IP)
75. Охрана гаражного кооператива (LonWorks)
76. Пропускная система предприятия (TCP-IP)
77. Управление светофорным объектом (LonWorks)
78. Управлении ж/д станцией (CAN)
79. Управление бильярдом (EIB)
80. Управление мишенями на стрельбище (P-NET)
81. АТС (TCP-IP)
82. Прогноз погоды (P-NET)
83. Управление движением автобуса (EIB)
84. Охранная система квартиры (CAN)
85. Охрана гаражного кооператива (CAN)
86. Пропускная система предприятия (P-NET)

87. Управление светофорным объектом (EIB)

88. Управлении ж/д станцией (LonWorks)

89. Управление мишенями на стрельбище (CAN)

90. Управление системой сборки объекта на конвейере (Profibus DP)

Лабораторная работа №4

Наименование «Разработка интерфейса оператора автоматизированной системы управления в SCADA приложении»

Исходные данные для выполнения лабораторной работы:

Лабораторная работа выполняется в SCADA приложении Citect 5.0., либо Citect Facilities 5.1 (передается студентам на CD-ROM (R, RW), DVD-R (+R, -RAM))

Для создания интерфейса оператора необходимо использовать виртуальное внешнее устройство (Generic, либо OPC).

Для создания проекта в Citect используются три программных компоненты:

Citect Explorer – создание страниц проекта, выбор компонент системы (устройства, переменные, (теги), сервера, платы ввода вывода). Основное средство управления проектом.

Citect Builder – для просмотра и создания всех элементов системы, а также ошибок компиляции.

Citect Runtime – система запуска приложения, разработанного в SCADA и ее проверки функционирования в режиме реального времени и режиме эмуляции.

При использовании не тривиальных функций управления, осуществляется программирование проекта. Программирование функций выполняется на встроенном языке – Cicode, вызов редактора осуществляется из Citect Explorer.

Алгоритм создания проекта:

1. Создание проекта, либо создания ссылки на проект. При создании в окне Citect Explorer для заданного проекта формируется шаблон, включающий средство создания страниц, а также всех

компонент системы. Компоненты системы – теги переменных, устройства, сервера ввода вывода, системные компоненты.

2. Создается сервер ввода вывода. При выборе данной функции необходимо указать имя.

3. Создается устройство ввода вывода, с которым будет взаимодействовать ПК. Для устройства необходимо указать уникальный идентификатор, адрес (от 1 и выше), тип устройства (Generic), тип памяти, из которой будут приниматься данные (MEMORY).

4. Для всех датчиков (устройства передающие сигналы и управляемые пользователем) создаются теги (переменные). При создании Variable Tag важно указать его имя и тип формируемых данных. Имя – уникальный в системе идентификатор, тип – INT, DECEMAL и пр. Также необходимо для всех элементов указать адрес устройства. При указании адреса следует именовать их начиная с буквы, именующий тип. При типе DECEMAL адрес должен быть D1…D100, при INT – I1…I100 и так далее. Адрес состоит из двух частей – буквенное обозначение (первая буква типа данных), число обозначающее адрес.

5. Для исполнительных устройств также могут быть созданы теги, если вывод информации осуществляется на внешние устройства, либо производится опрос. Если используется виртуальный объект теги указать необязательно.

6. В проекте создаются страницы интерфейса оператора. В проекте может быть одна стартовая страница и несколько вызываемых страниц в процессе функционирования. (Требуется создать 1 – 3 страницы). Создание осуществляется через Citect Explorer.

7. Переключение между страницами реализуется через кнопки управления, выбираемых в панели управления. Необходимо реализовать кнопку «Выход».

8. Для интерфейса оператора на страницах необходимо разместить все датчики и исполнительные элементы. Добавление осуществляется через панель инструментов.

9. Для всех датчиков (управляются пользователем) в окне свойств необходимо перейти к вкладке Access или Input и указать в окне ввода текста реакцию на действие датчика (например установить значении тега в заданное значение).

10. Для исполнительных элементов реакция указывается на одном из окон (Appearance, Movement, Scaling, Fill, Slider). Реакцией может быть изменение масштаба объекта, изменение цвета заливки, изменение кадра, движение по заданным величинам, задается степень видимости объекта. В окне ввода текста указывается условие, при котором заданное действие наступает.

11. Каждый элемент также может иметь текстовое описание при выводе на экран.

Примечание:

1. При необходимости в проект могут быть включены элементы из внешних графических редакторов.

2. Для каждого тега необходимо указывать устройство ввода вывода.

Задание:

Необходимо разработать интерфейс оператора автоматизированной системы в соответствии с вариантом задания.

1. Для автоматизированной системы предусмотреть 4-8 управляющих воздействий на алгоритм функционирования.

2. Разработать несколько «графических» исполнительных элементов, которые иллюстрируют действия на систему.

3. Создать автономный алгоритм функционирования системы.

4. Подготовить отчет.

Содержание отчета:

1. Титульный лист с указанием номера варианта.

2. Описание варианта задания.

3. Описание тегов проекта.

4. Описание алгоритмов функционирования.

5. Графическое представление системы.

Варианты заданий:

1. Модель железнодорожной стрелки

2. Модель 2 перекрестков: равнозначного и с приоритетом

3. Модель химического производства

4. Модель подачи воды в жилой дом, измерение потока, дублирование при аварии

5. Модель системы управления освещением при входе в помещения (2-3 помещения)

6. Модель системы охранной сигнализации на объектах, вывод информации при срабатывании трех типов датчиков

7. Модель системы управления шлюзованием

8. Модель системы отопления помещения в зданиях (2 здания по 5 помещений в каждом)

9. Модель системы подачи электрической энергии на объекты (2-3 объекта, 2-3 станции)

10. Модель системы производства руды

11. Модель системы управления движения скорой помощи по автодорогам (выбор пути, 2-3 пути)

12. Модель системы управления отправкой железнодорожных составов на станции

13. Модель системы управления вылетом и посадкой самолетов

14. Модель конвейера

15. Модель системы пожарной сигнализации в помещениях здания (5-7 помещения)

16. Модель системы коммутации объектов через общую сеть (сетевая активность узлов – 4-5 узлов)

17. Модель системы учета посещаемости

18. Модель АЗС

19. Модель зернохранилища

20. Модель контроля климата в помещениях

21. Модель телефонной станции

22. Модель системы прогноза погоды

23. Модель системы биллинга электрической энергии

24. Модель системы управления зданием

25. Модель системы контроля газовой станции

26. Модель системы управления самолетом

27. Модель системы управления движением автобусом

28. Модель системы управления складом готовой продукции

29. Модель системы управления автомобильным краном

30. Модель системы управления процессов выкачивания нефти

31. Модель системы управления погрузкой груза в вагоны

32. Модель системы управления морским портом

33. Модель системы управления пайкой плат и выпуском готовой продукции

34. Модель системы управления пожарной сигнализации

35. Модель системы управления движением поезда

36. Модель системы записи архивных данных на оптические носители

37. Модель системы контроля освещенности в помещении (жалюзи, уровень освещенности и т.п.)

38. Модель системы контроля прохождения транспортом точек управления

39. Модель системы управления производством на молочной фабрике

40. Модель системы управления движением пассажирского поезда

41. Модель системы управления аэропортом

42. Модель системы контроля погодных условий и принятие решения о включении световой сигнализации

43. Модель системы управления питанием на объекте (с резервированием)

44. Модель системы управления отоплением в квартире

45. Модель системы контроля потока транспорта на трассе и вывод информации на дисплей по массе, количеству и видам транспорта

46. Модель системы управления гидропонной системы (тепличное хозяйство)

47. Модель системы управления электростанцией

48. Модель системы управления сигнализацией люков канализации

49. Модель системы управления бильярдной

50. Модель системы управления влажностью в помещении

51. Модель системы контроля автобаном (движение по полосам)

52. Модель системы управления химическим процессом (миксер и нагрев)

53. Модель системы вывода информации на ж/к дисплей о температуре, коде нажатой клавиши

54. Модель системы игрового автомата (выигрыш/проигрыш)

55. Модель системы контроля пассажиропотока в общественном транспорте

56. Модель системы видео-наблюдения

57. Модель системы контроля летательного аппарата

58. Модель системы газопровода

59. Модель управления елочной гирляндой (переключение режимов, смена цветов)

60. Модель системы управления расписанием перемещения транспорта

61. Модель системы управления рекламными щитами (изменение реклам, смена подсветки и т.п.)

62. Модель системы управления трамвайным движением

63. Модель системы управления входом в помещения (по расписанию) дома (видеодомофон)

64. Модель системы управления сборкой автомобиля и отгрузкой продукции

65. Модель системы управления складом готовой продукции

66. Модель контроля проходной на предприятии

67. Модель системы управления лифтами (грузовой и пассажирский)

68. Модель системы контроля безопасности в супермаркете

69. Модель системы мониторинга транспорта на таможенном пункте

70. Модель системы кондиционирования в помещениях

71. Модель системы контроля периметра вокруг здания

72. Модель системы управления подачей воды в бассейн и сменой воды по расписанию

73. Модель системы управления воспроизведением музыки с различных дисков в зависимости от указанной последовательности песен, поиск записей

74. Модель системы мониторинга багаж в зале прилета и вылета аэропортов

75. Модель системы управления выдачей дискретной информации с промышленного контролера. Выдача информации на запросы с различных каналов и отображение полученных значений.

76. Модель тренажера обработки детали на станке

77. Модель системы управления движением троллейбуса

78. Модель системы управления рекламной службой

79. Модель системы контроля парковой по времени

80. Модель системы управления электропоездами

81. Модель системы управления управление зданием (потребление электрической энергии бытовыми приборами). Принцип договоренности приборов, увеличение потребления – увеличение тарифа. Вывод информации о потреблении.

82. Модель системы управления сливным бачком.

83. Модель системы управления сточными водами предприятия

84. Модель системы управления очистки воды на водозаборе

85. Модель системы управления моечной станцией

86. Модель системы управления прачечной

87. Модель системы управления зернохранилищем (перемещение по контейнерам)

88. Модель системы управления перемещением контейнеров
89. Модель системы управления пилорамой
90. Модель системы управления сауной
91. Модель системы управления фермой
92. Модель системы управления сбором мусора
93. Модель системы управления свинокомплексом