лении, а не в цифровом, выбор интерфейса водителя сделан в пользу мнемонического с использованием сенсорного ЖК монитора.

Полагается удобным вывести на сенсорном мониторе "прозрачную" мнемосхему автомобиля. Отдельные контролируемые крупные узлы прорисованы линиями "спокойного" цвета. При обнаружении тенденции к выходу контролируемых параметров из допустимых пределов изображение данного узла окрашивается "предупреждающим" цветом, а при выходе параметра за допустимые пределы – "тревожным" цветом.

Детализация состояния крупного узла может выдаваться в виде фотоизображения, на котором "тревожным" цветом отмечен мелкий узел, состояние которого идентифицировано как предаварийное или аварийное. Детализирующее изображение может заменять на экране основную мнемосхему, а смена изображений (прямая и обратная) происходит при касании экрана в области выбранного узла.

*Диспетчер системы автоматизации* предназначен для координации функционирования всех ее программных подсистем в режиме РВ и в соответствии с заданной циклограммой.

В состав стандартного ПО описываемой системы входят соответствующая версия Windows Mobile и установленный изготовителем набор драйверов, в числе которых драйверы экрана и управления вводом. Для обеспечения переносимости ПО все прикладные подсистемы могут быть написаны на языке Omega Basic и эффективно компилированы в коды Mobile Java. Этим обеспечивается независимость программной компоненты данной системы автоматизации от типов бортовых компьютеров, т.е. обеспечивается возможность ее применения для автомобилей разных типов с минимальными переработками.

В заключение отметим, что предложенная автоматизированная система может быть применена не только практически к любому транспортному средству, но и в составе систем автоматизации различных технических систем. Например, предложенная форма представления прогноза позволяет использовать разработанную систему в качестве одной из подсистем управления автономным мобильным роботом или в системе автоматизации электростанции.

**Список литературы**


1. *Дунаев А.П.* Организация диагностирования при обслуживании автомобилей. М.: Транспорт. 1987.
2. *Дьяков И.Ф., Петров В.М.* Электронные системы для бортовой диагностики автомобиля // Сб. тез. докл. М.: НИИАЭ. 1999.
3. *Жовинский А.Н., Жовинский В.Н.* Инженерный экспресс-анализ случайных процессов. М.: Энергия. 1979.
4. *Ветлинский В.Н., Юрчевский А.А., Комлев К.Н.* Бортовые автономные системы управления автомобилем. М.: Транспорт. 1984.
5. *Черняк Б.Я., Васильев Г.В.* Управление двигателем с помощью микропроцессорных систем. М.: МАДИ. 1987.
6. *Ronald N.* Bracewell. The Fourier Transform and its Applications. McGraw-Hill Book Company, 1986.



***Герасимов Александр Владимирович*** *– аспирант,* ***Дмитренко Лариса Григорьевна*** *– канд. техн. наук, ст. научн. сотрудник Института проблем управления им. В.А. Трапезникова РАН*

*Контактный телефон (495) 334-88-70 E-mail: Ldmit@ipu.rssi.ru*


# Использование QNX Neutrino в системах автоматического управления для ответственных применений


**В.И. Клепиков, Д.С. Подхватилин, Г.В. Шарапов, Н.А. Захаров (ИТМиВТ им. С.А. Лебедева РАН)**



*Рассмотрено построение интегрированных САУ с применением жесткого временного разделения доступа к шине для обеспечения надежности (TTA – Time Triggered Architecture). Описано применение в TTA-архитектуре ОС РВ QNX. Приведен пример разработки с реализацией указанного подхода – процессорный модуль для САУ газотурбинного двигателя авиационного и наземного применения.*


Современный уровень развития аппаратных средств позволяет пересмотреть подходы к проектированию САУ для ответственных применений. В настоящее время в методах построения систем управления можно выделить, по крайней мере, три основных тенденции:

- отказ от представления САУ как набора специализированных программно-аппаратных блоков в пользу унифицированных аппаратно-программных средств;
- отказ от "федеративных" архитектур в пользу интегрированных решений;
- внедрение тактируемых архитектур (Time-Triggered Architecture).

Несмотря на то, что сегодня существует достаточно много примеров работоспособных САУ на базе специализированных модулей, все же такие системы обладают рядом существенных недостатков как на этапе разработки, так и в эксплуатации.

Разработка САУ на специализированных модулях требует наличия специалистов с очень широким спектром знаний в прикладной области, области аппаратно-программного обеспечения. Как правило, для каждого модуля используются свои средства отладки, зачастую недостаточно мощные и удобные. Модернизация таких систем осложнена из-за недостатка унификации подходов к разработке, скудной документации – на выпуск хорошей документации в рам-

ках одного проекта обычно не хватает ни времени, ни бюджета. Смена аппаратной части нередко приводит к глубокой модернизации всего ПО. Поддержку и развитие САУ на специализированных модулях, как правило, могут осуществлять только непосредственные разработчики этой системы.

Изменения в подходах к построению САУ для ответственных применений стали возможны, прежде всего, благодаря росту производительности и объема памяти современных процессорных модулей, что позволило использовать универсальные ОС РВ. Применение ОС РВ решает большинство из упомянутых проблем, что существенно снижает сроки разработки и увеличивает качество и удобство эксплуатации. ОС также представляют собой прослойку между аппаратным и прикладным программным обеспечением, благодаря которой аппаратура может модернизироваться без изменений в ПО и наоборот.

### Интегрированные САУ

Применение производительных вычислителей и ОС РВ привели к идее размещения большей части расчетов в едином процессорном модуле. Такие архитектуры САУ получили название "интегрированные" (в отличие от "старых", "федеративных" архитектур). САУ с интегрированной архитектурой обладает лучшими массогабаритными характеристиками при более низкой себестоимости.

При использовании интегрированной архитектуры остро встает вопрос надежности совместного выполнения различных задач на одном аппаратном модуле. Очевидным требованием является то, что сбои в одном из приложений не должны повлиять на выполнение других. В идеале должна обеспечиваться полная независимость выполнения функций, присущая федеративной архитектуре САУ.

В настоящее время видится только один подход, отвечающий данному требованию – применение специальной ОС, поддерживающей разделение приложений (partitioning). Идеи разделения приложений выражены стандартом ARINC 653. Каждое приложение (возможно, состоящее из нескольких процессов), которое должно выполняться обособленно относительно других приложений, помещается в раздел. Операционная система обеспечивает:

- невозможность доступа для приложения из одного раздела в память другого раздела (пространственное разделение);
- наличие у каждого раздела гарантированного бюджета времени, который будет ему предоставлен, даже если один или несколько других разделов имеют приложения с более высоким приоритетом (временное разделение).

Применение разделов, помимо повышения надежности САУ, дает следующие преимущества:

- независимость разработки. Поскольку разделы функционируют независимо друг от друга, они могут проектироваться, разрабатываться независимо (но с учетом ограничения бюджета времени и памяти). Большую часть верификации приложений (за исключением совместной доводки изделия) также можно провести независимо;
- гибкость в развертывании системы. Если в системе есть несколько вычислительных модулей, то раздел может быть при необходимости перенесен с одного вычислителя на другой без изменения кода приложений (при условии, что оба вычислителя работают под управлением одной ОС).

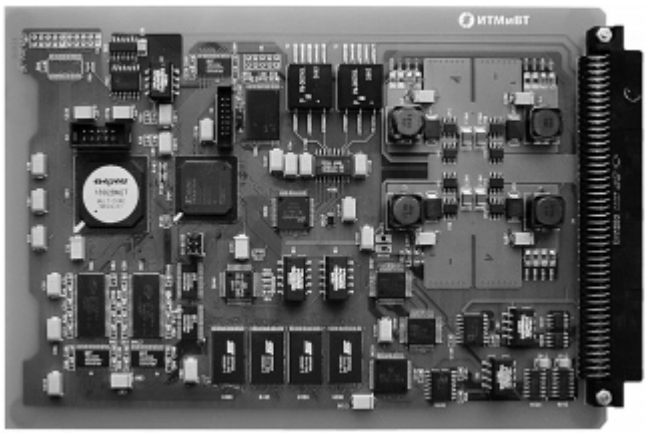


*Рис. 1. Бортовой процессорный модуль*

### Тактируемые архитектуры

Тактируемые архитектуры – одно из наиболее перспективных направлений в проектировании критичных к безопасности САУ. В таких системах модули связаны между собой с помощью единого интерфейса, как правило, с шинной топологией, и все передачи по этому интерфейсу производятся в соответствии с некоторым детерминированным расписанием.

Преимуществами тактируемой архитектуры являются:

- независимость передач по интерфейсу для каждого из приложений, соответственно сбои в работе одного из приложений не повлияют на работу других приложений;
- предопределенность задержки передачи информации между модулями;
- возможность компенсации одного или нескольких одновременных сбоев при передаче.

Надо сказать, что концепция разделов хорошо сочетается с концепцией тактируемых архитектур. Обе концепции призваны обеспечивать независимость приложений друг от друга и останавливать распространение ошибок. Приложения в тактируемой архитектуре должны разделять не только процессорное время, но и время на шине. Детерминированное расписание передач должно также поддерживаться ОС.

Одной из стратегических задач ИТМиВТ им. С.А. Лебедева является создание современной аппаратно-программной платформы для разработки САУ в ответственных приложениях. Ключевой особеннос-

тью нашей платформы является использование отечественной элементной базы и ОС QNX Neutrino.

В рамках создания указанной выше платформы специалистами ИТМиВТ разработан интегрированный аппаратно-программный комплекс для построения систем управления. В его состав входят процессорный модуль (рис. 1) на двухядерном микропроцессоре, функционирующий под управлением ОС QNX, набор драйверов различных интерфейсов и флэш-памяти, средства программирования и отладки.

Ключевой особенностью данного комплекта является разгрузка разработчика встраиваемой системы на его основе от трудоемкого рутинного программирования обслуживания системных устройств, высвобождение ресурсов разработчика для написания собственно управляющего приложения по заданным алгоритмам, сокращение сроков работы над проектом, повышение качества продукта в целом.

### Использование QNX Neutrino для ответственных применений

За многие годы применения в области САУ и промышленной автоматизации ОС QNX отлично зарекомендовала себя как очень надежная и эффективная ОС для построения САУ. Вот перечень свойств, выводящих QNX Neutrino на передовые позиции среди ОС для встраиваемых применений и во многом упрощающих решение упомянутых выше проблем со сложностью разработки и модернизации, переносимостью ПО:

• QNX Neutrino – это ОС жесткого РВ, она может применяться в системах для ответственных применений;

• высокая надежность за счет микроядерной архитектуры. Ядро QNX, имеющее небольшой размер, хорошо отлажено за десятки лет разработки;

• наличие развитых средств разработки и отладки QNX Momentics, включающих интегрированную среду разработки, необходимые библиотеки, пакет качественной документации. QNX Momentics упрощает процесс разработки и поддержки ПО;

• удобство разработки системного ПО (драйверов). Данное свойство также следует из микроядерной архитектуры: драйвера в понимании QNX – это обычные приложения;

• совместимость с POSIX дает хорошую переносимость ПО на платформу QNX из широко распространенных ОС Unix, Linux.

К указанным преимуществам в последнее время добавились открытость исходных кодов ядра и доступность специальных пакетов инструментальных средств (Technology Development Kits – TDK), упрощающих выполнение типичных проектов.

### Разделение приложений

Концепция разделения приложений также поддерживается QNX Neutrino. В версии 6.3.2 в ядро включен пакет адаптивного разделения (Adaptive Partitioning), обеспечивающий защиту разделов по памяти и гарантирующий наличие временного бюджета у каждого раздела, но его реализация имеет отличия от традиционного понимания (в духе ARINC 653) управления разделами.

Обычная реализация временного разделения подразумевает использование статической схемы разделения времени. То есть вводится понятие цикла системы, и составляется расписание работы разделов. Простой пример: цикл системы 50 мс; раздел №1 работает в период 0...15 мс, раздел №2 – в период 15...20 мс, раздел №3 – в период 20...50 мс. Планировщик ОС гарантирует следование расписанию.

QNX Neutrino предлагает адаптивное временное разделение. Каждому разделу разработчиком назначается бюджет времени в процентах от общего времени "цикла". На самом деле, вместо понятия "цикла" – жестко определенного периода времени с детерминированным расписанием – используется понятие "усредняющее окно", в рамках которого планировщик обеспечивает заданный процент времени каждому разделу. Приведенный выше пример в терминах QNX Neutrino означал бы: размер усредняющего окна 50 мс; раздел №1 имеет 30% времени, раздел №2 – 10%, раздел №3 – 60%. Никакого жесткого расписания переключений между разделами в QNX нет.

Надо отметить, что QNX гибко подходит к распределению времени между разделами. Если какой-либо из разделов не использует полностью свой бюджет, неизрасходованное время может быть отдано другим разделам. Механизм гарантии указанного разработчиком процента начинает работать только при больших нагрузках.

Адаптивное разделение имеет следующие преимущества:

• уменьшение простоев процессора за счет передачи неиспользованного бюджета другим разделам, что снижает требования к производительности аппаратных средств;

• сохранение быстрой реакции на события, характерной для QNX (в том числе уменьшение задержки обработки прерываний).

Но применение адаптивной стратегии управления разделами несколько усложняет дизайн упомянутых выше тактируемых архитектур. Очевидно, что статическое расписание передач по интерфейсу лучше сочетается со статическим расписанием работы разделов, нежели с адаптивным планированием, при котором фактически никакого расписания и нет.

Тем не менее, гибкие средства ОС позволяют решить и эту задачу даже с дополнительными выгодами.

### Использование QNX в тактируемых архитектурах

В QNX введен механизм критических потоков управления, которые выполняются немедленно, как только становятся активными. Критический поток может быть запущен, даже если раздел исчерпал свой бюджет, за счет чего достигается маленькая латентность потока, характерная для систем жесткого РВ.

*Рис. 2. Рабочее место для разработки и отладки ПО*

Поток считается критическим в пределах критического бюджета раздела, к которому он принадлежит. Критический бюджет назначается разработчиком в миллисекундах (то есть выражается в абсолютном, а не относительном времени). По исчерпанию критического бюджета критический поток рассматривается как обычный поток управления и не имеет никаких привилегий относительно других потоков. Ограничением критического бюджета обеспечивается защита от монопольного захвата процесса в случае сбоя в критическом потоке.

Возвращаясь к построению тактируемых архитектур, можно предложить достаточно простой способ организации разделов QNX. Каждый раздел должен иметь, по крайней мере, один критический поток, который:

• активируется по таймеру в соответствии с расписанием передач;

• выполняет только те операции, которые критично выполнить в соответствии с расписанием (как правило, это операции ввода/вывода).

Операции, которые можно выполнить не синхронно с расписанием передач, должны выполняться в обычных потоках. QNX будет управлять ими по адаптивной стратегии, обеспечивая низкую латентность для действительно критичных ко времени операций.

**Пример САУ ответственного применения**

В ИТМиВТ был разработан процессорный модуль для САУ газотурбинными двигателями (ГТД) для авиационного применения. В качестве элементной базы для этого модуля были использованы только компоненты отечественного производства или разрешенные к использованию Министерством обороны РФ.

После тщательного анализа имеющихся на рынке ОС РВ для построения платформы на базе модуля была выбрана ОС QNX. В сжатые, нехарактерные для встроенных систем сроки на вышеупомянутый модуль была установлена (адаптирована) ОС РВ QNX Neutrino 6.3.

В соответствии со стандартом POSIX разработаны драйверы:

• CAN (последовательная магистраль, обеспечивающая увязку в сеть "интеллектуальных" устройств ввода/вывода, датчиков и др. исполнительных устройств);

• интерфейса ARINC-429;

• интерфейса QSPI (Quick Serial Peripherial Interface) – быстрый (буферизованный) последовательный периферийный интерфейс;

• TPU (Time Processing Unit) – устройства обработки временных интервалов;

• MIL-STD1553B – магистрального последовательного интерфейса;

• универсального асинхронного порта RS-232;

• флэш-памяти, а также некоторые другие, которые вместе с ОС РВ QNX Neutrino 6.3 составляют на данном процессорном модуле полнофункциональную среду исполнения для прикладного ПО жесткого РВ.

Несмотря на то, что основное применение процессорного модуля бортовое, в составе САУ ГТД функциональные возможности "среды разработки QNX Momentics" позволили организовать на его основе рабочее место для разработки и отладки ПО жесткого РВ (рис. 2).

Рабочее место состоит из процессорного модуля для САУ ГТД с загруженной на него полнофункциональной средой исполнения, и ПК с установленной на нем средой разработки QNX Momentics. Созданное рабочее место позволяет не только разрабатывать программы, загружать их на модуль и управлять их исполнением. Используя символьный отладчик можно непосредственно с ПК (без предварительной загрузки на процессорный модуль), работая через интерфейс универсального асинхронного порта, исполнять откомпилированные программы в режиме пошаговой отладки, что сокращает время разработки и отладки ПО.

Рабочее место для разработки и отладки ПО может легко трансформироваться, например, в стенд для проверки и наладки САУ. Для этого достаточно подсоединить к модулю через один из внешних UART (а их на плате модуля четыре) еще один ПК с загруженным в него необходимым прикладным ПО.

К настоящему времени подобные рабочие места развернуты как в стенах ИТМиВТ, так и у компании-заказчика.

***Клепиков Владимир Иванович*** *– канд. техн. наук, руководитель лаборатории,*
***Захаров Николай Анатольевич*** *– канд. техн. наук, руководитель направления,*
***Подхватилин Дмитрий Станиславович*** *– руководитель проектов,*
***Шарапов Геннадий Викторович*** *– начальник отдела Института точной механики и вычислительной техники им. С.А. Лебедева РАН.*
*Контактный телефон (495)649-12-70.*