WIND RIVER

Разработка критичного по безопасности ПО для Интегрированной Модульной Авионики

Paul Parkinson, Senior Systems Architect, Wind River
Larry Kinnan, Senior Engineering Specialist, Aerospace & Defense, Wind River

Эта техническая статья представляет современные тенденции в разработке критичных по безопасности систем. В статье обсуждается появление архитектур и стандартов Интегрированной Модульной Авионики (ИМА) и результат их влияния на разработку коммерческой ОСРВ, совместимой с этими стандартами.

Содержание:

Введение

Многие системы авионики были успешно разработаны с использованием специализированной аппаратуры и программного обеспечения. Но стоимость полного жизненного цикла специализированных систем заставила производителей оборудования обратить в последние годы свое внимание на использование коммерческих серийно-выпускаемых компонент (COTS). В то же самое время наметился переход от федерированных (federated) архитектур, в которых каждая отдельная подсистема выполняла отдельную функцию, к универсальным вычислительным платформам, которые могут быть использованы в приложениях различного рода и, в некоторых случаях, исполнять несколько приложений одновременно. Этот подход, известный как Интегрированная Модульная Авионика (ИМА) приводит к сокращению количества подсистем, снижению веса и потребляемой мощности и уменьшению избыточности. В ряде гражданских и военных исследовательских программ было проведено определение принципов архитектуры ИМА, и, несмотря на различия в подходах, у них были общие стратегические цели:

- *Унифицированные процессорные подсистемы*. Они должны позволить нескольким приложениям делить и повторно использовать одни и те же вычислительные ресурсы. Это приводит к сокращению количества развертываемых подсистем и более эффективному использованию системных ресурсов, что оставляет свободное пространство для будущих расширений.
- *Абстракция ПО*. Она должна сделать приложения независимыми не только от нижележащей шинной архитектуры, но и от нижележащей аппаратной архитектуры. Это повысит уровень переносимости приложений между различными платформами, а также позволит вводить новую аппаратуру для замены устаревших архитектур.
- *Максимизировать повторное использование*. Архитектура ИМА должна позволять повторное использование ранее разработанного кода. Это сокращает время разработки, предоставляя разработчику возможность использования уже существующих приложений без значительной модификации.
- *Стоимость изменений*. Архитектура ИМА должна понизить стоимость внесения изменений и повторного тестирования и упростить анализ последствий (impact analysis) путем изолирования отдельных частей платформы, которые исполняются на одном и том же процессоре.

ИМА также облегчает задачу поддержки приложений с постоянно растущей функциональностью, таких как индикаторы на лобовом стекле, системы отображения карт и дисплеи погодных радаров, включая взаимодействие этих сложных приложений между собой.

Хотя появилось несколько вариантов архитектур и стандартов ИМА, спецификации ACR (Avionics Computer Resource) [1] и ARINC 653 [2] оказались наиболее широко признанными сообществом разработчиков авионики. Спецификация ACR посвящена архитектурным вопросам, а спецификация ARINC 653 определяет высокоуровневую программную модель архитектуры ИМА. Эти и другие стандарты ИМА предъявляют новые требования к программным архитектурам, особенно к реализации ОСРВ коммерческими поставщиками. Компания Wind River придавала особое внимание этим задачам при разработке интегрированного пакета VxWorks 653 Platform [3], который эксплуатируется в C-130 AMP и 767 Tanker [4]. Компания Boeing выбрала Wind River VxWorks 653 Platform для разработки системы Common Core System (CCS) [5] Boeing 787. Другие заказчики Wind River, включая EADS [6], используют этот пакет для разработки систем авионики и критичных по безопасности приложений.

Следующие параграфы рассматривают технические требования к интегрированной программной платформе для поддержки ИМА-приложений и показывают, как пакет VxWorks

653 Platform (Рис. 1) выполняет эти требования, в частности, в контексте разработки приложений ARINC 653.

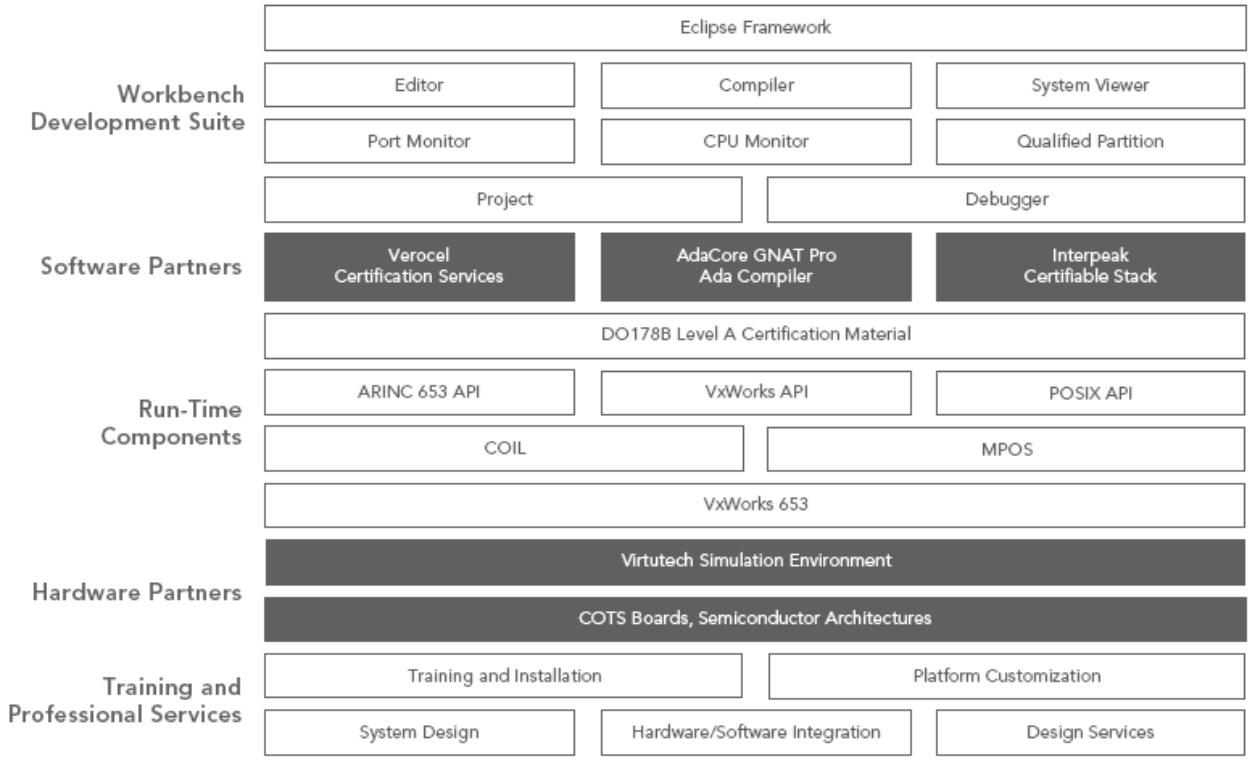

Figure 1: Wind River's VxWorks 653 Platform

Разработка приложений с Wind River VxWorks 653 Platform

Спецификация ACR определяет две важные концепции, широко используемые в ИМА: пространственное разделение и временное разделение.

Пространственное разделение

Пространственное разделение определяет требования по изоляции нескольких приложений, исполняемых одновременно на одной и той же вычислительной платформе, называемой «модулем». Согласно этим требованиям, приложения, исполняемые в разделе ИМА, не должны отбирать друг у друга разделяемые ресурсы, предоставляемые ядром ОСРВ. Чаще всего это реализуется через различные контексты виртуальной памяти, обеспечиваемые процессорным устройством управления памятью MMU (Memory Management Unit). В спецификации ARINC 653 эти контексты называются разделами (partition). Каждый раздел содержит приложения со своей динамически распределяемой областью памяти типа «куча» (heap) и стеком для процессов приложения (процесс – это исполняемая единица в ARINC 653). Эти требования влияют на конструкцию и реализацию ядра ОСРВ и исполнительной системы языка программирования. Например, VxWorks 5.5 использует разделяемое виртуальное адресное пространство для приложений и обеспечивает через MMU базовую защиту программного кода от доступа со стороны ошибочных приложений, не применяя полную модель процессов, влияющую на производительность. Операционные системы VxWorks 6.x и VxWorks 653 обеспечивают через MMU среду с полной изоляцией контекстов.

Однако одна только защита памяти в среде ИМА не может воспрепятствовать ошибочному приложению, исполняемому в разделе, занимать системные ресурсы, что может иметь вредоносное влияние на приложения, исполняемые в другом разделе. Это может иметь серьезные последствия, когда несколько приложения разных уровней критичности исполняются на одном и то же процессоре. Проблема не может быть решена одной только

реализацией полной модели процессов; требуется разработка специальной ОСРВ, предназначенной именно для ИМА. Операционная система VxWorks 653 было разработана специально для этой цели и поддерживает модель ARINC 653 на уровне реализации архитектуры ядра (Рис. 2).

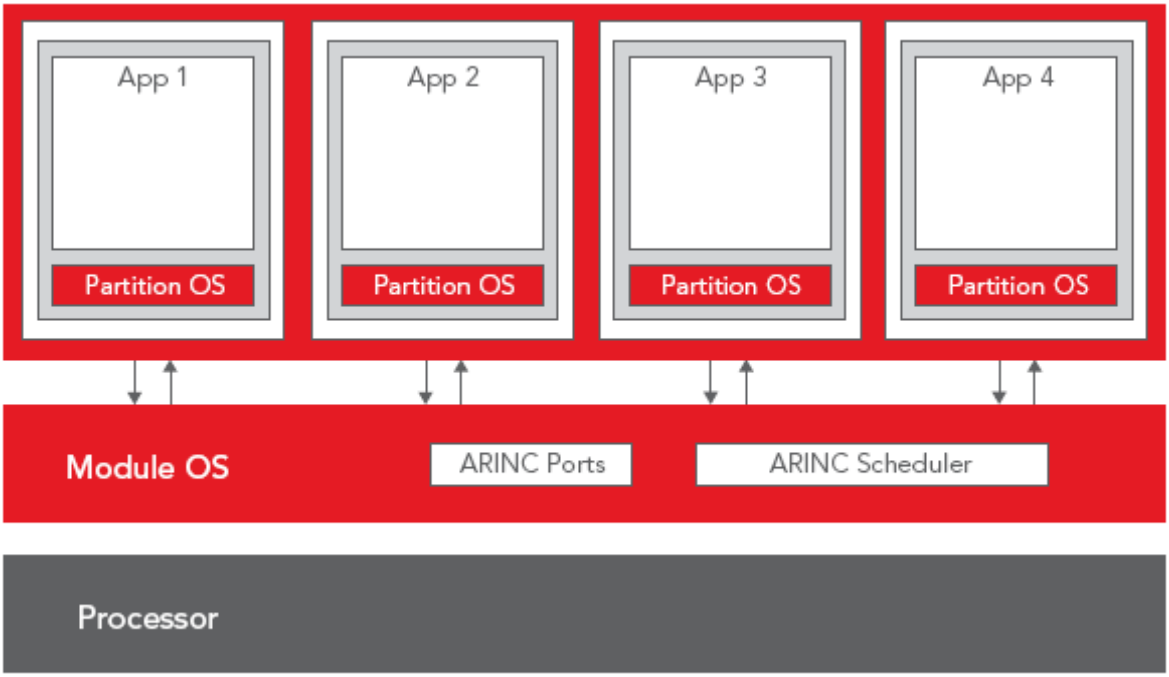

Figure 2: VxWorks 653 RTOS Architecture

- ОС модуля (Module OS) непосредственно взаимодействует с аппаратной платформой (core-модуль), обеспечивая управление общими ресурсами, планирование исполнения и мониторинг состояния каждого из разделов. Она также использует пакет поддержки модуля BSP (Board Support Package) – аппаратно-зависимые описания, необходимые для исполнения на различных процессорных и аппаратных конфигурациях.
- ОС раздела (Partition OS) реализована на микроядре VxWorks и обеспечивает планирование исполнения и управление ресурсами внутри раздела. Коммуникация с ОС модуля для обеспечения надежности происходит через внутренний закрытый интерфейс передачи сообщений. ОС раздела также обеспечивает программный интерфейс ARINC 653 APEX (Application/Executive), используемый приложениями.

Эта архитектура, соответствующая виртуальной машине, описанной в литературе [7] «Разделение в архитектуре авионики: требования, механизмы и обеспечение», удовлетворяет требованиям ARINC 653 и, в то же время, предоставляет гибкий и расширяемый каркас (framework), что трудно обеспечить с помощью монолитного или UNIX-похожего ядра. Отдельные разделы внутри каркаса реализованы на базе контейнеров с защищенной памятью, в которых размещаются процессы, объекты и ресурсы, и это разделение обеспечивается устройством MMU (виртуальной машиной). Каждый раздел имеет собственный стек и локальную «кучу», которые не могут быть использованы приложениями других разделов.

На рисунке 2 показана концептуальная реализация архитектуры VxWorks 653. В ОСРВ допускается в разделе общего доступа иметь одну системную библиотеку (разделяемый код с защитой только-чтение), чтобы упростить конфигурирование, тестирование и сертификацию. Также допускается иметь отдельную определенную конфигурацию ОС раздела для одного или более разделов, называемую MPOS (multiple partition operation systems – операционная система нескольких разделов).

Временное разделение

Временное разделение определяет требования по изоляции нескольких приложений, исполняющихся одновременно на одной и той же вычислительной платформе. Одно приложение не может занимать процессор дольше, чем ему предназначено, чтобы не навредить другим приложениям. В ARINC 653 уделено внимание этой проблеме и определена реализация, в которой используется планирование исполнения разделов. Каждому разделу выделяется для исполнения временной интервал определенной длительности, и интервалы различных разделов могут быть одинаковой или различной длительности. Внутри своего интервала раздел может применять собственную политику планирования, по окончании временного интервала ARINC-планировщик переключит контекст на следующий по расписанию раздел. Это модель достаточно универсальна, чтобы разместить на core-модуле существующие федерированные приложения или IMA-приложения, разработанные независимо друг от друга. Но планирование исполнения разделов и проверка целостности расписания и границ исполнения, а также необходимые корректирующие действия, неизбежно вносят дополнительную сложность.

В VxWorks 653 ОС модуля выполняет планирование исполнения отдельных разделов в соответствии с ARINC 653. Внутри каждого временного интервала ОС раздела использует планировщик VxWorks для выполнения планирования на основе вытеснения по приоритету (preemptive priority-based) (Рис. 3). Это значит, что все планирование на уровне процессов происходит в пространстве раздела, что обеспечивает более высокую масшабируемость и стабильность (минимальный джиттер) в системе даже при высокой частоте системных тактов (чаще 1 миллисекунды). Также полностью реализован протокол силинга приоритета (priority ceiling) для предотвращения неограниченной инверсии приоритетов (см. след. параграф «Инверсия приоритета, наследование приоритета, силинг приоритета»).

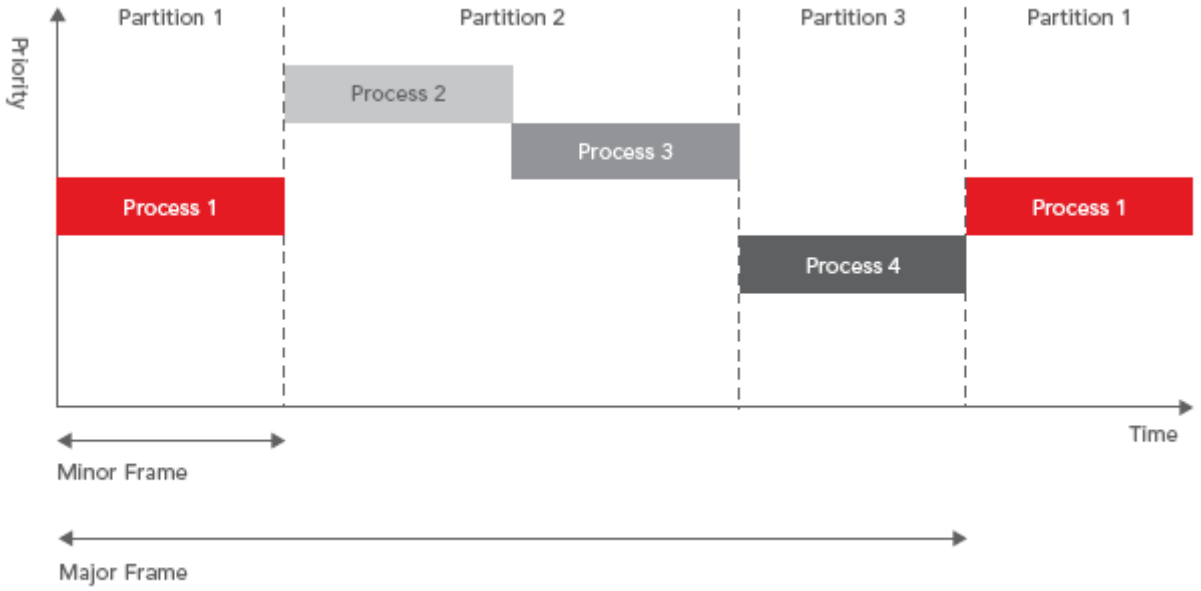

Figure 3: VxWorks 653 Temporal Partitioning

Реализация ARINC 653 в VxWorks 653 полностью удовлетворяет спецификации Supplement 2 (653-2) [8]. Она также поддерживает дополнительное планирование по режиму (mode-based scheduling), в котором могут быть определены до 16 расписаний, и они могут быть применены для различных режимов полета или для многоступенчатой инициализации. Переход между режимами выполняется через API-вызов ограниченного доступа *arincSchedSet()*, который может быть выполнен на границе между основными кадрами (major frame), между интервалами разделов или между системными временными тактами. Система мониторинга состояний HMS (Health Monitoring System) проводит валидацию нового

расписания до его принятия к исполнению (это будет обсуждаться далее в параграфе «Система мониторинга состояний и рестарты»).

Стандарт ARINC 653 предоставляет методологию высокодетерминированного планирования, которое может не подходить для некоторых приложений, поскольку может привести к излишнему свободному времени в разделе или слишком высокой частоте системного такта для высокочастотной выборки данных. Чтобы удовлетворить потребности таких приложений, VxWorks 653 обеспечивает дополнительную возможность планирования разделов с вытеснением по приоритету. Этот метод позволяет назначенным разделам захватывать простои (slack stealing) и использовать незанятое при обычном ARINC 653 планировании время.

Инверсия приоритета, наследование приоритета, силинг приоритета

Инверсия приоритета является проблемой для разработчиков ПО устройств. Она возникает тогда, когда высокоприоритетная задача не может начать исполняться потому, что она хочет получить доступ к мьютексу (двоичному семафору взаимоисключающего доступа), которым владеет низкоприоритетная задача, выполнению которой, в свою очередь, препятствует среднеприоритетная задача (Рис. 4)

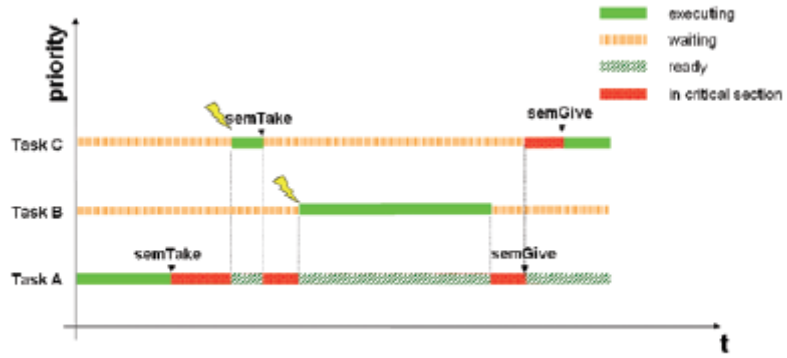

Figure 4: Priority Inversion Example

Эта проблема преодолена во многих ОСРВ путем реализации протокола наследования приоритета (priority inheritance). Эта схема привязывает к каждому мьютексу приоритет, и приоритет задачи, владеющей мьютексом, повышается до уровня приоритета задачи, запрашивающей мьютекс, как показано на рис. 5. Обычно при использовании протоколов наследования приоритета или силинга приоритета применяется анализ RMA (Rate Monotonic Analysis), но иногда для определения времени исполнения для наихудшего случая может потребоваться дополнительный анализ, поскольку при использовании наследования приоритета возможно возникновение каскада блокировок.

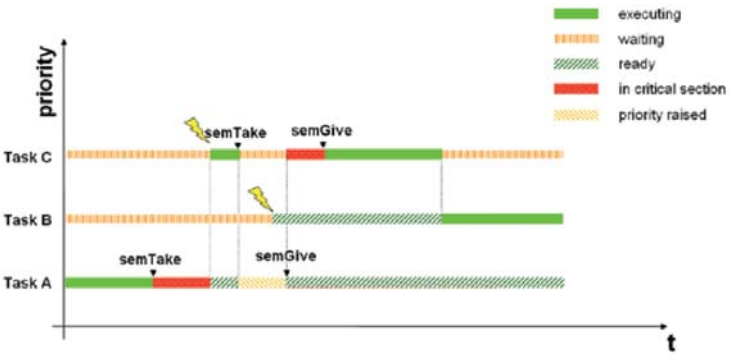

Figure 5: Priority Inheritance Example

Протокол силинга приоритета (ceiling = потолок) является альтернативным методом, предотвращающим каскадные блокировки. В этом случае мьютекс инициализируется с приоритетом выше, чем приоритет любой из задач, способных его захватить. Когда задача блокирует мьютекс, ее приоритет поднимается до «потолка» (Рис. 6). Некоторые производители ОСРВ применяют собственные реализации протокола силинга приоритета, а Wind River использует POSIX-реализацию [9] потому, что это международный общепризнанный стандарт, позволяющий портировать существующие приложения на платформы ИМА на базе VxWorks 653 (см. след. параграф «Поддержка разнородных приложений».

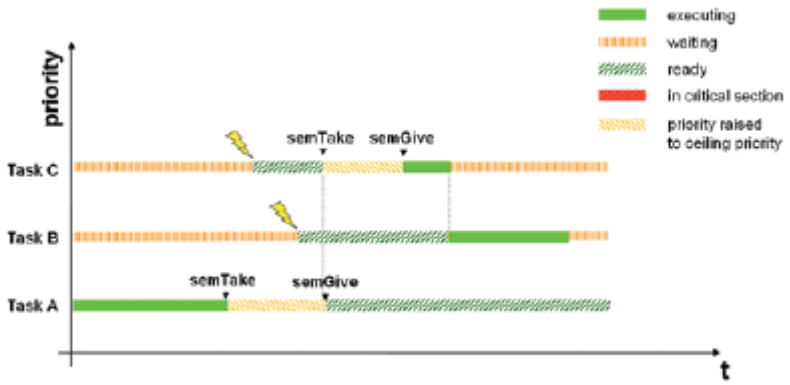

Figure 6: Priority Ceiling Example

Необходимо также обеспечить отсутствие взаимовлияния приложений во время выполнения действия внутри ядра, инициированных приложениями. В ARINC 653 это достигается путем применения сложной модели. В ней обслуживание APEX-вызовов выполняется на уровне раздела, и только требующие обслуживания на уровне модуля вызовы передаются дальше. Время, затрачиваемое на обслуживание вызовов ядра, включается в расписание исполнения раздела, что предотвращает занятие приложениями времени других разделов.

В VxWorks 653 эта концепция расширена возможностью для системного интегратора ограничить на этапе компиляции количество одновременных блокирующих APEX-вызовов. Это сделано путем статической конфигурации количества worker-потоков ядра (worker-поток выполняет операции в ядре от имени ARINC-процесса, выполнившего APEX-вызов). Если все worker-потоки заняты обслуживанием вызовов ядра, следующий APEX-вызов, требующий взаимодействия с ядром, будет блокирован до тех пор, пока не освободится worker-поток. Это управляется конфигурационными данными, которые определяют разрешенные операции и ресурсы в разделах. Конфигурационные данные отделены от разделов и их приложений, так что изменения могут быть изолированы и реконструированы без изменения самого приложения раздела. Это значительно снижает стоимость внесения изменений.

Разработка приложений ARINC 653

Программный интерфейс ARINC 653 APEX, иногда упоминаемый как AINC 653 API, обеспечивает интерфейс общего назначения между операционной системой и прикладным ПО. Интерфейс ARINC 653 API обеспечивает уровень абстракции, который скрывает от приложения детали реализации конкретной ARINC 653 совместимой ОСРВ и нижележащей архитектуры core-модуля. Это упрощает перенос приложений на другие платформы ARINC 653, что является важным фактором для критичных по безопасности ИМА-систем, таких, как компьютер системы управления полетом, который требует двойного или даже тройного, как в случае Boeing 777 [10], резервирования.

Интерфейс ARINC 653 APEX также обеспечивает модель статической системной конфигурации и инициализации. В ней количество ARINC-процессов известно заранее, и они

создаются в разделе через вызов CREATE_PROCESS(). Все другие объекты раздела создаются из «кучи» раздела, и после того, как это сделано, раздел активируется вызовом SET_PARTITION_NODE(NORMAL). В этот момент активируется планировщик ОС раздела и принимает управление исполнением процессов внутри раздела. После запуска приложения никакие другие процессы или объекты динамически созданы быть не могут. Это обеспечивает контролируемую детерминированную инициализацию и детерминированное, фиксированное использование ресурсов.

Интерфейс ARINC 653 обеспечивает также отличные конструкции для коммуникации как между разделами, так и внутри раздела. Блэкборды (blackboards) и буферы помогают обмену внутри раздела. Блэкборды обеспечивают удобный механизм однократная-запись/многократное-чтение, а буферы обеспечивают возможность посылки и приема сообщений, которые записаны в порядке FIFO, но допускают чтение как в порядке FIFO, так и в порядке приоритета. Кроме этого, для синхронизации могут быть использованы семафоры и события.

Порты ARINC 653 предназначены для коммуникации между разделами. Одна и та же схема именования портов может использоваться для портов, находящихся на одном процессоре или на разных core-модулях, в одном ИМА-отсеке или в разных. Это делает приложения архитектурно- и конфигурационно-независимыми, способствует переносимости и упрощает реконфигурацию системным интегратором. Механизм портов ARINC 653, реализованный в VxWorks 653, также позволяет определение и использование псевдопорта, посредством которого сканирующий (sampling) или (буферизующий (queuing) порт ARINC 653 соединяется с драйвером устройства в ОС модуля для обмена между модулями, представленного приложению в виде стандартного ARINC 653 API.

Поддержка разнородных приложений

Хотя многие ИМА-приложения разрабатываются «с нуля», есть множество существующих приложений на федерированных системах. Эти приложения могли быть разработаны на различных языках программирования и могли использовать различные модели планирования исполнения, но в среде ИМА между ними может потребоваться обмен.

Wind River обратил внимание на эту задачу и обеспечил поддержку разнородных приложений внутри ARINC-разделов. Это позволяет приложениям Ada 95, использующим профиль Ravenscar с ограниченной мультизадачностью, исполняться поверх ОС раздела VxWorks 653 (это детально обсуждено в литературе [11]). Аналогично, POSIX-совместимые приложения могут исполняться в разделе, и обмен между приложениями может быть выполнен с использованием ARINC-портов. (Рис. 7)

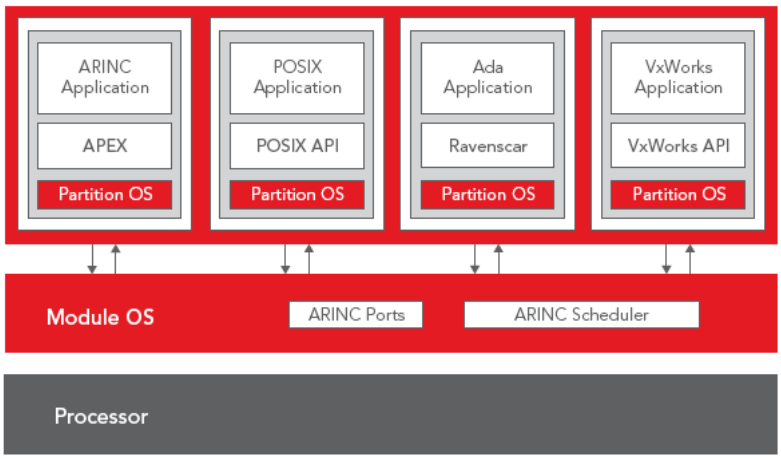

Figure 7: VxWorks 653 Heterogeneous Application Support

Системное конфигурирование

Архитектура ARINC 653 гарантирует доступность ресурсов путем использования конфигурационных записей для системы и разделов (также известные в области ИМА как «синие копии» - blueprints). Они предназначены для того, чтобы позволить системному интегратору ИМА-платформы конфигурировать ИМА-приложения, разработанные несколькими отдельными производителями. Конфигурационные записи раздела определяют характеристики приложения в терминах потребностей в памяти, в процессоре и в использовании ARINC-портов. Конфигурационная запись системы определяет функциональные возможности и допустимую нагрузку ИМА-платформы и связана ссылками с конфигурационными записями отдельных разделов и ратифицирует (validate) их. Такая схема позволяет системному интегратору обеспечить соответствие запросов отдельных приложений производительности платформы и то, что отдельные приложения не выйдут за пределы выделенных им ресурсов.

Современное издание спецификации ARINC 653 включает в себя только высокоуровневое описание структуры и содержания конфигурационных записей и оставляет детали реализации на усмотрение разработчика ОСРВ, хотя образцы конфигураций на языке XML-приведены. Операционная система VxWorks 653 использует следующий подход:
Шаг 1.При инициализации системы сначала загружается ОС модуля и конфигурационные записи системы и разделов.
Шаг 2. ОС модуля самоинициализируется и запускает свои подсистемы.
Шаг 3. ОС модуля загружает прикладные разделы и их приложения.
Такой процесс отделяет двоичное представление конфигурации ОС модуля и двоичное представление конфигурационных записей системы и разделов от приложений раздела. Вследствие этого отдельные приложения и подсистемы могут быть разработаны отдельно друг от друга, а затем беспрепятственно интегрированы на целевой системе. Приложения отдельных разделов можно обновлять обычным путем, не требующим изменений в конфигурации ОС модуля. Это значительно снижает затраты на повторную сертификацию и дает больше свободы производителям и системным интеграторам.

Операционная система VxWorks 653 дополняет образцы XML-конфигураций и обеспечивает разработчика приложений, производителя платформы и системного интегратора полным и квалифицированным набором средств управления данными конфигурации и инициализации ИМА-платформы. Этот процесс, называемый IBLL (independent build, link and load = независимое построение, связывание и загрузка), снижает стоимость внесения изменений и, в тоже время, обеспечивает полностью конфигурируемую среду исполнения. Это процесс полностью реализует цели, поставленные в документе DO-297 «IMA Development Guidelines and Certification Considerations» [12].

Конфигурация системы и разделов может быть изменена без реконструкции всего приложения или платформы, что значительно снижает нагрузку на системного интегратора по анализу последствий (impact analysis) при обновлении и модификации существующей системы. Поскольку инструментарии для создания конфигурационных записей в VxWorks 653 генерируют двоичные данные непосредственно из конфигурационных XML-данных, то такой инструментарий гораздо проще использовать и квалифицировать, чем другие реализации. Они обычно используют инструменты более общего назначения, например С-компилятор, для генерации двоичных конфигурационных данных, используемых в системе.

Система мониторинга состояний и рестарты

Стандарт ARINC 653 определяет понятие монитора состояний HM (Health Monitor) внутри ИМА-системы. Монитор состояний HM отвечает за «мониторинг неисправностей и отказов в аппаратуре, приложениях и операционной системе», и функцией HM является «изоляция

ошибок и предотвращение распространения отказов». Хотя эта функция достаточно очевидна, в действительности она требует сложного общесистемного монитора состояний для того, чтобы отслеживать ошибки и выполнять реконфигурацию и восстановление. Реакция на конкретную неисправность зависит от природы неисправности, ее серьезности и политики обработки ошибок, определенной системным интегратором.

Система мониторинга состояний HMS (HM Systems) в VxWorks 653 – это сложный каркас, который функционирует как неотъемлемая часть архитектуры VxWorks 653. Он удовлетворяет всем требованиям ARINC 653 и обеспечивает расширения, относящиеся к динамической реконфигурации (в частности, планирование по режиму), которыми могут воспользоваться нуждающиеся в них системные интеграторы. Конструкция и реализация VxWorks 653 HMS достаточно развитая, здесь можно провести только ее краткий обзор.

Архитектура HMS состоит из общесистемного HMS-сервера и HMS-агентов (называемые в ARINC 653 process-level handlers = обработчики уровня процесса), которые располагаются в отдельных разделах, и в эту архитектуру также включена поддержка ОС модуля. Система HMS обрабатывает события, нуждающиеся в рассмотрении; они называются неисправностями (fault), хотя могут иметь как негативный, так и позитивный характер, например, аппаратная исключительная ситуация или достижение порогового значения (неисправность, представляемая в ПО как «тревога» = alarm). Каркас системы мониторинга также поддерживает другой тип событий - «сообщения», которые используются для регистрационных или других действия, заданных системным интегратором.

Каркас HMS обеспечивает возможность проводить мониторинг состояний на трех уровнях - HM процесса, HM раздела и HM core-модуля - и через три типа сервиса: детектирование тревоги, регистрация тревоги и реакция на тревогу. Мониторы HM раздела и HM модуля управляются табличными описаниями и устанавливают соответствие между кодом ошибки и ее обработчиком. Для лучшей переносимости, каркас HMS использует ARINC 653 определения кодов ошибок, таких как истечение критического срока (missed deadline), ошибки с конечным числом вариантов (numeric errors), недопустимый запрос (illegal request) и падение питания (power fail). Для конфигурирования каркаса HMS используются XML-данные, которые устанавливают таблично-управляемое соответствие для использования системой во время исполнения. Реакция на тревогу зависит от уровня ошибки: например, реакцией уровня модуля может быть сброс или отключение, реакцией уровня раздела – рестарт раздела.

В момент создания раздела используется холодный старт, включающий размещение и инициализацию объектов раздела, при рестарте раздела применяется теплый старт, который только реинициализирует уже размещенные объекты. Для ошибок в процессах реакция управляется приложением; действие зависит от типа ошибки и ее контекста. Чтобы способствовать теплым рестартам, в VxWorks 653 имеются персистентные (persistent) типы данных , которые используются для обеспечения сохранности критичных данных во время операции теплого рестарта.

Инструменты разработки критичных по безопасности систем

Хотя исполнительная среда ОСРВ является основной рассматриваемой темой, обсуждение проблем разработки ИМА-приложений было бы неполным, если не рассмотреть средства разработки и отладки. Качество средств разработки и отладки может иметь значительное влияние на временные рамки этапа разработки. Средства разработки, созданные для федерированных систем, могут не подойти для разработки ИМА, поскольку они должны поддерживать модули и режимы планирования ИМА.

Пакет Wind River VxWorks 653 Platform представляет собой интегрированную систему разработки, включающую инструментальную среду Wind River Workbench, основанную на Eclipse [13]. Эта современная среда включает в себя средства конфигурации проекта, просмотра и построения кода, симулятор VxWorks 653, отладчик целевой системы и визуализатор системных событий Wind River System Viewer. На Рис. 8 показана среда Workbench в сеансе отладки приложения ARINC-раздела. К возможностям среды Workbench, поддерживаемым Wind River, можно добавлять Eclipse-плагины из других источников, открытых или коммерческих, для расширения функциональности среды и создания специализированного набора средств.

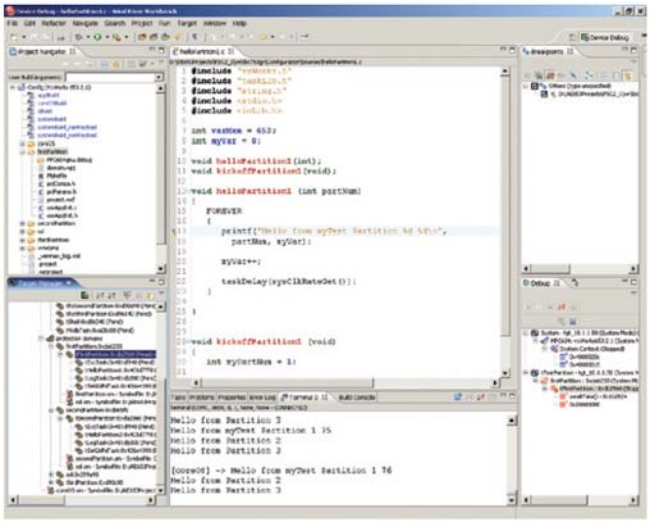

Figure 8: Wind River Workbench Showing VxWorks 653 Partition Debugging

Возможности динамической визуализации очень полезны разработчику приложений, так как дают возможность наблюдать в графическом виде поведение приложений ARINC, POSIX и VxWorks, взаимодействие между разделами и операции системы мониторинга состояний HMS. Среда Workbench используется для просмотра, навигации и понимания содержания приложений Ada, ARINC, POSIX и VxWorks. Визуализатор Wind River System Viewer может отображать поведение процессов ARINC, потоков POSIX и задач VxWorks и их взаимодействие в разделах. Он может оказать неоценимую помощь при отображении внутренних деталей поведения приложений, таких как обмен между ARINC-процессами через буферизующий порт, как показано на Рис. 9.

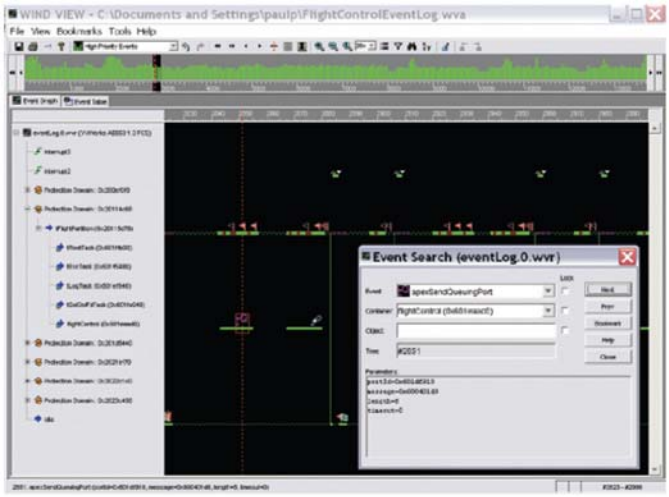

Figure 9: Wind River System Viewer Showing ARINC Partition Behavior

Для разработчика приложения ARINC важна не только визуализация поведения внутри раздела ARINC, но и наблюдение работы приложения в сертифицируемом окружении, включая коммуникацию между разделами через порты и каналы ARINC. Эта возможность

обеспечена средствами мониторинга времени CPU, использования памяти и использования портов, которые встроены в ОСРВ и сертифицированы как часть исполнительной системы VxWorks 653. Эти мониторы ОСРВ работают совместно с инструментальными средствами, квалифицированными по DO-178B, которые отображают и регистрируют данные при тестировании. Мониторы и инструментальные средства обеспечивают детальное и глубокое понимание внутреннего поведения системы во всех ее конфигурациях – от варианта для разработки до варианта для полетной сертификации.

Вопросы защищенности в сетевых системах ИМА

Защищенность (security) в системах авионики все чаще становится предметом обсуждения [15]. Защита во внутренней сети самолета может быть обеспечена сетевыми экранами, ограничивающими взаимодействие между различными типами подсистем и отделяющими полетные системы от служебных и сервисных [16]. Но с приходом ИМА выросла потребность в сетевом обмене – в сертифицированном его виде - внутри этих доменов. Эта задача породила некоторые интересные проблемы.

Протокол TCP/IP и связанные с ним протоколы требуют высоких затрат при сертификации, и системные проектировщики вынуждены искать компромисс между функциональностью и сертифицируемостью. Полный стек TCP/IP сертифицировать по Уровню A DO-178B крайне затруднительно. Некоторые разработчики выступают за специализированные реализации, использующие дополнительный (slave) процессор, исполняющий сетевой стек для основного (master) процессора. Но специализированные конфигурации используют дополнительную аппаратуру и ограничивают переносимость ПО. Это противоречит двум основным целям ИМА, подчеркнутые ранее, и может рассматриваться только как шаг назад. Решение Wind River – начиная с версии 2.2 VxWorks 653 выпустить в качестве дополнительного продукта сетевой стек UDP/IPv4, сертифицируемый по Уровню A DO-178B. Это обеспечит достаточный на сегодняшний день для многих заказчиком уровень функциональности, и в то же время, предоставит возможность дальнейшего расширения путем добавления других протоколов, например IPv6.

На уровне core-модуля также существует проблема защищенности, требующая внимания. Например, приложение раздела, исполняющееся в пользовательском режиме процессора, не может выполнять привилегированные процессорные инструкции. Кроме этого, если ОС раздела не может обслужить APEX-вызов приложения, то она передает его в ОС модуля для валидации перед исполнением. Типы валидации включают в себя проверку адреса на нахождение внутри области памяти раздела, проверку границ, проверку прав доступа к объекту ОС модуля и проверку целостности/совместимости структуры данных. Операционная система VxWorks 653 включает также масшабируемый механизм привилегий системных вызовов, по которому один раздел может иметь больше полномочий, чем другие, что обеспечивает фундамент для выполнения требований стандарта МЭК-15408 [17] (см. также соответствующую публикацию Wind River [18]). Имеются также ограничения по соображениям защищенности, касающиеся системы мониторинга состояний HMS, например, только привилегированный раздел, функционирующий как менеджер режимов, может запросить изменений политики планирования. В совокупности, все эти методы создают повышенную защищенность.

Вопросы безопасности в системах ИМА

Задача сертификации ИМА-систем относительно новая, но многие ее аспекты основаны на методах, использованных при сертификации существующих федерированных систем по различным сертификационным стандартам [19][20]. Например, концепция повторно используемых компонент при DO-178B сертификации критичных по безопасности

приложений хорошо документирована [21] и успешно использована при сертификации федерированного приложения на базе VxWorks в программе FAA [22]

Изоляция ОС модуля и разделов в VxWorks 653 путем пространственного разделения позволяет расширить эту концепцию. Теперь приложения VxWorks 5.x, ранее сертифицированные по Уровню C DO-178B, могут быть использованы в отдельном ИМА-разделе как новое приложение Уровня A DO-178B без ресертификации его по Уровню A. Этот метод может быть использован для драйверов ввода/вывода и сетевых стеков (таких, как TCP/IP). Они размещаются в отдельном VxWorks 653 разделе ввода/вывода, который изолирован от ОС модуля и прикладных разделов. Обмен с прикладными разделами происходит посредством ARINC-портов, и взаимодействие с ОС модуля ограничено подпрограммами драйвера, исполняемыми в режиме супервизора. Это препятствует возможности воздействия на корректную работу ОС модуля и прикладных разделов со стороны несертифицированного кода (Рис. 10).

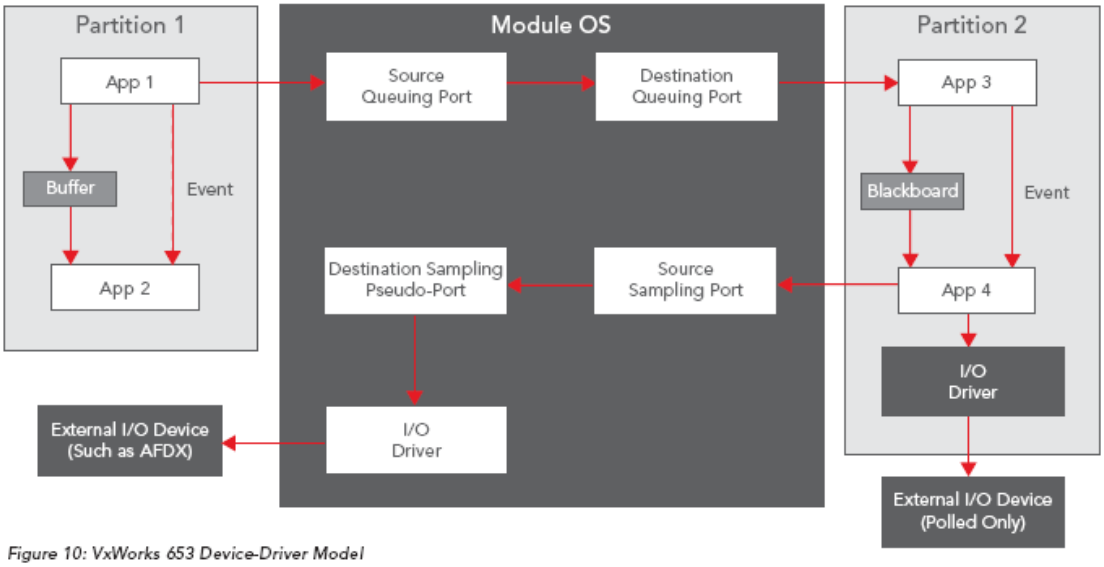

Figure 10: VxWorks 653 Device-Driver Model

Заключение

Индустрия авионики находится в середине пути к ИМА несмотря на то, что архитектуры и стандарты ИМА продолжают эволюционировать, и это создает проблемы для стандартизирующих организаций, производителей оборудования и поставщиков приложений.

Wind River предоставляет интегрированную программную платформу. Пакет Wind River VxWorks 653 Platform объединяет удовлетворяющую стандартам коммерческую ОСРВ и все инструментальные средства, необходимые для успешной разработки критичных по безопасности ИМА-приложений. Это пакет не только повышает продуктивность разработчика, но и дает ему уверенность при столкновении со сложной и трудозатратной задачей сертификации.

Поддержка разнородных приложений ARINC 653, Ada, POSIX и VxWorks в среде ИМА способствует максимальному повторному использованию и переносу существующих федерированных приложений в VxWorks 653.

Список литературы:

[1] DO-255, “Requirements Specification for Avionics Computer Resource (ACR).” www.rtca.org
[2] ARINC Specification 653, “Avionics Application Software Standard Interface,” January 1, 1997. www.arinc.com
[3] Wind River VxWorks 653 Platform product page. www.windriver.com/products/platforms/safety_critical/
[4] Wind River, ACT, and Smiths Aerospace C-130AMP press release. www.windriver.com/news/press/pr.html?ID=296
[5] Smiths Aerospace Boeing 7E7 Dreamliner Common Core System. www.windriver.com/customers/customer-success/aerospace-defense/smiths787.html
[6] EADS/CASA. www.windriver.com/news/press/pr.html?ID=201
[7] John Rushby, DOT/FAA/AR-99/58, “Partitioning in Avionics Architectures: Requirements, Mechanisms and Assurance,” March 2000
[8] ARINC Specification 653-2, “Avionics Application Software Standard Interface,” December 1, 2005. www.arinc.com
[9] POSIX Specification, ANSI/IEEE POSIX 1003.1-1995; ISO/IEC standard 9945–1:1996
[10] Y. C. (Bob) Yeh, “Design Considerations in Boeing 777 Fly-by-Wire Computers.” Third IEEE International High-Assurance Systems Engineering Symposium, 1998. http://doi.ieeecomputersociety.org/10.1109/HASE.1998.731596
[11] P. Parkinson & F. Gasperoni, “High Integrity Systems Development for Integrated Modular Avionics Using VxWorks and GNAT.” 7th International Conference on Reliable Software Technologies, Ada Europe, 2002. http://link.springer.de/link/service/series/0558/bibs/2361/23610163.htm
[12] DO-297, “Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations.” www.rtca.org
[13] Eclipse Consortium. www.eclipse.org
[14] Wind River Workbench product page. www.windriver.com/products/development_suite
[15] P. Tingey & P. Parkinson, “Avionics Security.” Defense Procurement Analysis, Summer 2003
[16] Jean Paul Moreaux, EADS-Airbus, “Evolution of Future Aircraft Data Communications.” NASA Workshop on Integrated CNS Technologies, May 2001. http://spacecom.grc.nasa.gov/icnsconf/docs/2001/CNS01_Session_ F3-Moreaux.pdf
[17] ISO/IEC 15408: 1999, Information Technology—Security Techniques—Evaluation Criteria for IT Security. www.iso.org
[18] G. Kuhn, “VxWorks Secure Architecture.” Technical paper, Wind River
[19] DO-178B, “Software Considerations in Airborne Systems and Equipment Certification.” www.rtca.org
[20] “Safety Management Requirements for Defense Systems,” Parts 1 & 2. Interim Defence Standard 00-56, Issue 3, December 17, 2004, UK Ministry of Defense. www.dstan.mod.uk/
[21] FAA Draft Notice, N8110 RSC. www.faa.gov
[22] Raytheon WAAS Customer Success Story. www.windriver.com/news/press/pr.html?ID=393