Современные принципы построения программного обеспечения бортовых и наземных программно-аппаратных комплексов обработки данных

к.ф.-м.н. Иванов Денис Владимирович

ОАО «Корпорация «Русские Системы»
105066, г. Москва, ул. Старая Басманная, 19/16
тел. (495)261-06-76, факс. (495)261-97-57, email: denis@rusys.ru

Реферат

В настоящей работе рассматриваются современные принципы построения программного обеспечения бортовых и наземных программно-аппаратных комплексов обработки данных. Необходимость создания новых подходов к разработке программного обеспечения обусловлена имеющимися тенденциями в развитии подобных систем, включающих перераспределение функций между программным и аппаратным обеспечением, повышение степени ответственности и усложнение математического (алгоритмического) обеспечения, необходимость переноса алгоритмической составляющей на различные аппаратные платформы. После анализа указанных тенденций в работе рассматривается структурная схема модульного программного комплекса, обладающая необходимыми свойствами. Кроме того, анализируется необходимость использования инженерных редакторов алгоритмов работы разрабатываемых систем и средств управления объемом и формой отображения информации. В завершении работы представлена стратегия разработки и верификации логики работы комплекса соответствующими специалистами без использования штатного аппаратного обеспечения и без привлечения разработчиков программного обеспечения к данному процессу. Рассматриваемые подходы апробированы на практике и подтвердили эффективность их использования при разработке современных программно-аппаратных решений.

1. Введение

Современные тенденции в развитии авиационной техники показывают, что роль бортовых и наземных программно-аппаратных комплексов (ПАК), использующих передовые технологии в области цифровой микроэлектроники, стремительно возрастает. При этом отличительными особенностями процесса создания таких комплексов являются следующие свойства.

Перераспределение функций между программным и аппаратным обеспечением. В силу необходимости снижения времени разработки новых ПАК и снижения затрат на всех этапа их жизненного цикла, а также в связи с повышением интеллекта подобных систем, все большую функциональная нагрузка перекладывается разработчиком на программное обеспечение, оставляя за аппаратной составляющей лишь низкоуровневые операции. Преимущества такого подхода очевидны как для разработчика, так и для эксплуатанта, который получает возможность последующей функциональной модернизации изделия без замены аппаратной части, однако технологический цикл разработки и эксплуатации таких систем требует новых подходов, отвечающих новым потребностям.

Повышенная степень ответственности. Наряду с высокими требованиями по надежности и корректности работы, возлагаемыми на программно-аппаратные средства, относящиеся к классу военной техники, расширение функциональных возможностей одновременно возлагает и большую ответственность на системы подобного рода. В качестве примера можно привести создание цифровых регуляторов двигателей с полной ответственностью, что означает отсутствие средств страховки в виде гидромеханической системы регулирования на случай программного или аппаратного сбоя регулятора. Другим примером служит создание в Российской Федерации

бортовой активной системы безопасности полетов (БАСБП), которой впервые в мире доверено выключение летчика из контура управления самолетом и автоматическое осуществление оперативных мер по предотвращению развития критической ситуации в случае принятия решения о неработоспособности или неадекватности его действий.

Усложнение математического (алгоритмического) обеспечения. Расширение функциональных возможностей и обеспечение современных требований, предъявляемых к авиационным комплексам, обуславливает создание все более сложных алгоритмов работы бортовых систем и автоматизированных средств наземного обслуживания. При этом, зачастую, эффективность таких алгоритмов не может быть обоснована теоретически или полностью подтверждена в ходе ограниченного срока испытаний. Таким образом, реализация экспертного метода оценки логики работы системы, а также возможность оперативного уточнения алгоритмов работы комплекса на всех этапах его жизненного цикла, становится залогом его эффективной эксплуатации.

Независимость программного обеспечения от аппаратной платформы. С повышением интеллекта программно-аппаратных комплексов, логика их работы, в конечном итоге реализованная в виде программного обеспечения, становится независимой от аппаратной платформы, на которой выполняется это программное обеспечение. В качестве иллюстрации данного процесса можно рассмотреть следующий пример.

Рассмотрим автоматизированную контрольно-проверочную аппаратуру АКПА-30, предназначенную для наземной проверки цепей каналов энергетического и информационного взаимодействия самолетов с авиационными средствами поражения. В ее состав входят блоки имитации подвески (БИП30), обеспечивающие прием и имитацию полного спектра необходимых сигналов, блок имитации видеосигнала (БИВ30), предназначенный для проверки видеотрактов, и Пульт АКПА-30, задачи которого включают управление работой всей системы по каналу информационного обмена, а также хранение результатов контроля и отображение их оператору (см. Рис. 1). Работы с объектом контроля проводятся по алгоритмам, хранящимся в Пульте АКПА-30 и обеспечивающим максимально-возможную глубину контроля технического состояния с определением неисправности с точностью до клеммы на соответствующих разъемах. Подтверждение соответствия таких алгоритмов реальным циклограммам работы изделий, полноты и глубины контроля, является результатом длительного и крайне затратного цикла испытаний, проводящихся установленным порядком.

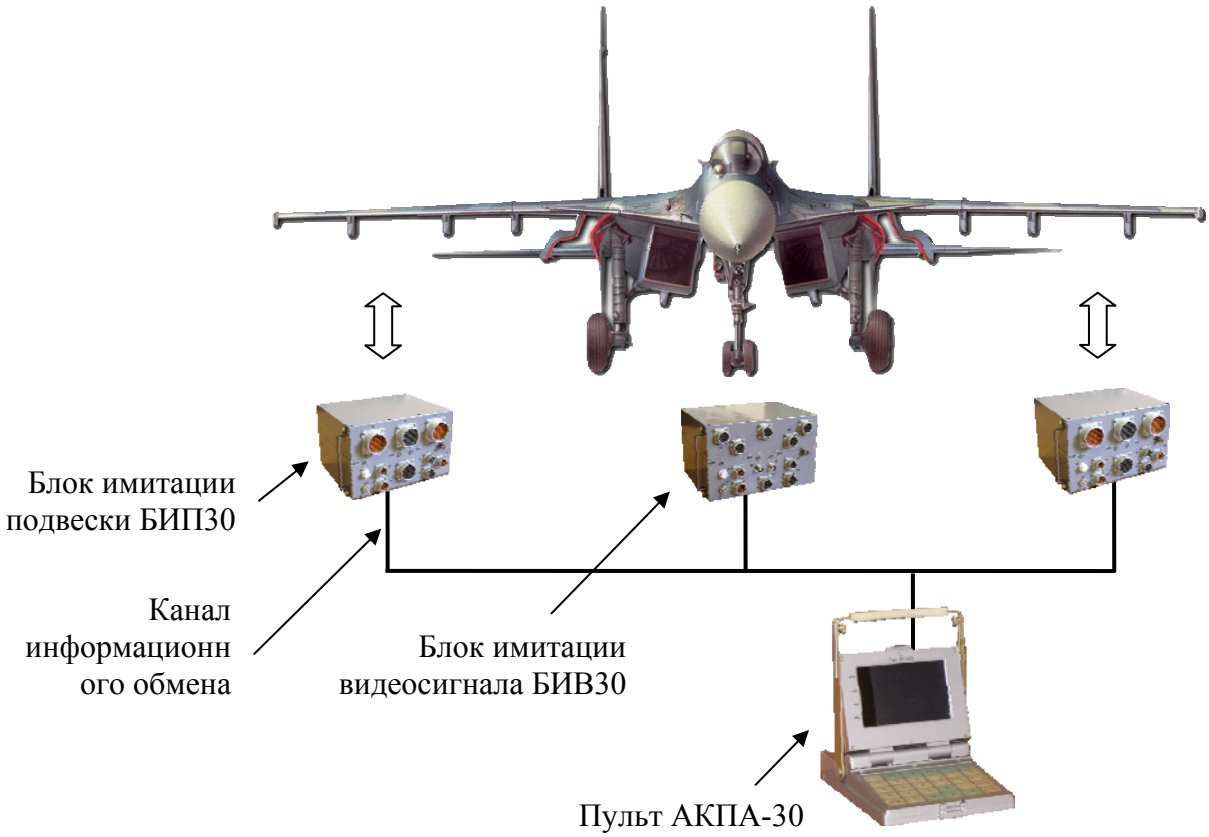

Рис. 1. Автоматизированная контрольно-проверочная аппаратура АКПА-30.

Пульт АКПА-30, как и вся аппаратура АКПА-30, выдерживает воздействие механических и климатических факторов в соответствии с требованиями ГОСТ РВ 20.39.304-98 для аппаратуры классификационной группы 1.10, в частности может эксплуатироваться в жестких условиях при температурах от -40 до +60 С°. При этом Пульт АКПА-30 в полном объеме выполняет задачи по управлению комплексом, в частности реализует логику работы автоматизированной системы, осуществляет отображение необходимой информации пользователю и прием от него управляющих команд, обеспечивает накопление результатов работы для их документального оформления и последующего анализа. Тем не менее, в ходе применения комплекса в стационарных условиях, например, при поиске и устранении неисправностей в ТЭЧ, возможна потребность в наличие некоторых дополнительных функций.

Для обеспечения дополнительных функций в аппаратуре АКПА-30 предусмотрена возможность замены Пульта портативным компьютером на базе процессора класса Intel Pentium или аналогичного. Однако, с точки зрения процесса контроля данный компьютер, являясь несовместимым с Пультом АКПА-30 на уровне исполняемого кода программного обеспечения, обязан обеспечить 100% соответствие реализации разработанных и испытанных для него алгоритмов. Таким образом, заменяя Пульт на компьютер (см. Рис. 2), пользователь должен быть гарантирован, что обе аппаратные платформы обеспечат одинаковый процесс и результат с точки зрения логики их работы.

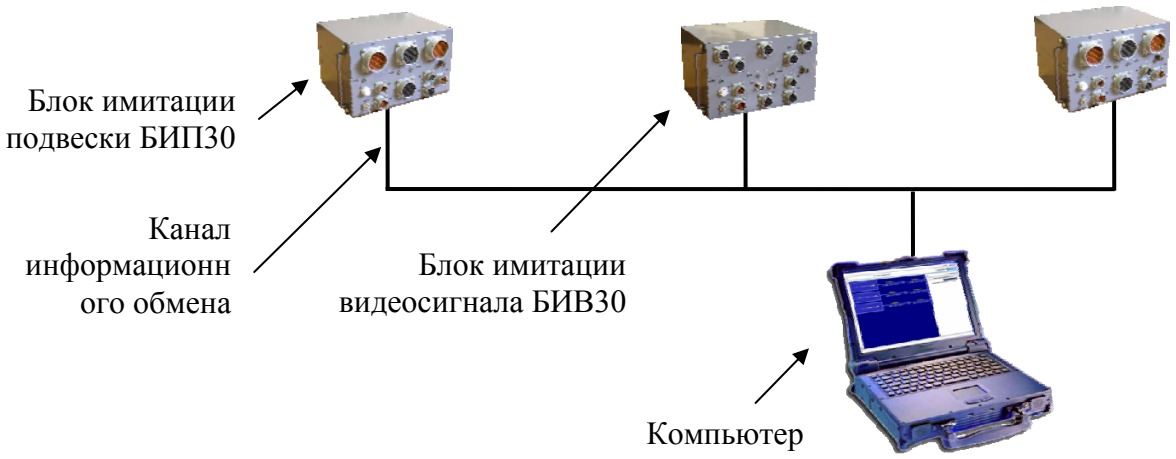

Рис. 2. Аппаратура АКПА-30 с управляющим устройством в виде компьютера.

С другой стороны, бортовые вычислительные комплексы, например, бортовая активная система безопасности полетов (БАСБП), обладают достаточной производительностью и объемами памяти для управления системой блоков, входящих в состав АКПА-30, по каналу информационного обмена, для реализации алгоритмов контроля и хранения результатов проводимых работ. При этом взаимодействие с оператором может осуществляться через штатные для самолета многофункциональные индикаторы (МФИ) и многофункциональные кнопки (МВК), расположенные в кабине. Использование бортовых вычислителей для управления контрольно-проверочной аппаратурой, а МФИ и МФК в качестве интерфейсных устройств, позволяет обеспечить проведение работ по контролю технического состояния непосредственно из кабины самолета, что существенно снижает временные затраты, критичные, в частности, для оперативных видов подготовки. Тем не менее, при переносе логики работы АКПА-30 из специализированных управляющих устройств (Пульт, компьютер) в бортовой комплекс, необходимо гарантировать идентичность исполнения алгоритмов контроля и формы получаемого результата.

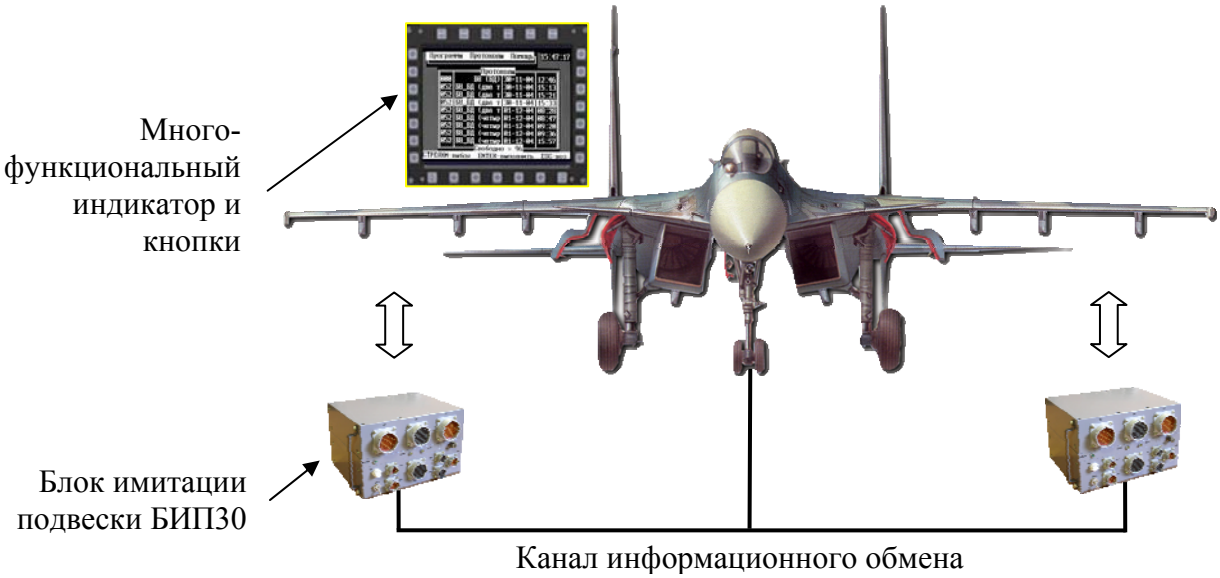

Рис. 3. Замена управляющего устройства системы АКПА-30 на бортовой вычислитель.

В рассматриваемом примере три различных по характеристикам, элементной базе и общему программному обеспечению вычислительных устройства – специализированный Пульт АКПА-30, персональный компьютер и бортовая активная система безопасности – реализуют одинаковые алгоритмы наземной проверки цепей каналов энергетического и информационного взаимодействия самолетов с авиационными средствами поражения. При этом понятие алгоритма – логики работы системы в целях обеспечения соответствующих требований по назначению – не привязаны к какой-либо конкретной аппаратной платформе управляющего устройства.

Таким образом, в условиях указанных выше тенденций при создании современных авиационных комплексов, а именно, повышение роли и ответственности программного обеспечения, постоянное усложнение логики его работы и формирование потребности переноса алгоритмической составляющей на различные аппаратные платформы, возникает необходимость в новом взгляде на принципы построения программного обеспечения.

В данной работе рассматриваются новые подходы к структуре и процессу разработки программного обеспечения, обеспечивающие эффективную реализацию современных требований на этапе испытаний и эксплуатации.

2. Структура программного обеспечения

Любой программно-аппаратный комплекс, решающий в масштабе реального времени возложенную на него задачу, характеризуется наличием следующих функциональных процессов (см. Рис. 4):

- Прием данных из внешних источников – каких-либо датчиков, средств взаимодействия с оператором, других систем;
- Обработка и анализ поступающей в систему информации, возможно с формированием промежуточных данных, принятие соответствующих решений;
- Передача данных внешним потребителям – средствам отображения информации, исполнительным механизмам, другим системам.

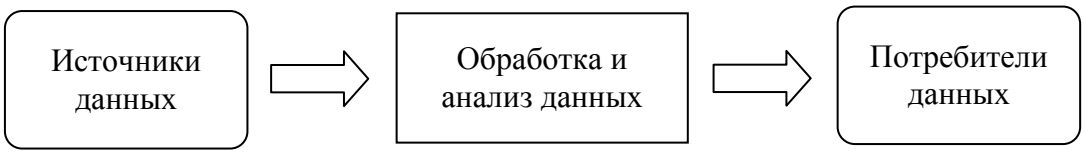

Рис. 4. Функциональная схема системы обработки и анализа данных.

Для системы, реализующей подобные функциональные процессы можно определить следующие понятия.

Перечень параметров – совокупность атрибутивной информации о параметрах (входных, промежуточных, выходных, системных), используемых системой в процессе своей работы.

Срез параметров – есть процесс сопоставления параметрам из перечня их значений на какой-либо (например, текущий) момент времени.

Менеджер времени – управляющая подсистема программного обеспечения комплекса, реализующая выбранную стратегию распределения временного ресурса для выполнения требуемых задач.

Ядро системы – компонент программного обеспечения комплекса, реализующий общие для всех систем функции – поддержка текущего среза параметров и управление временем.

Системный сервис – компонент программного обеспечения комплекса, обеспечивающий взаимодействие системы с аппаратными ресурсами. Такие сервисы могут включать файловую систему, систему распределения динамической памяти, графическую подсистему, доступ к портам ввода/вывода, доступ к системному времени и прерываниям таймера, и т.п.

Программный модуль (далее, модуль) – компонент программного обеспечения комплекса, реализующий специфичное для конкретной системы требования по ее назначению.

Программный интерфейс (далее, интерфейс) – стандартизированный на уровне системы перечень функций, с помощью которых компоненты системы пользуются ее ресурсами – системными сервисами, срезом параметров, менеджером времени и т.п.

На Рис. 5 показана общая структура программного обеспечения комплекса, характеризующаяся разделением его на взаимодействующие компоненты по их функциональному назначению с учетом приведенных выше определений.

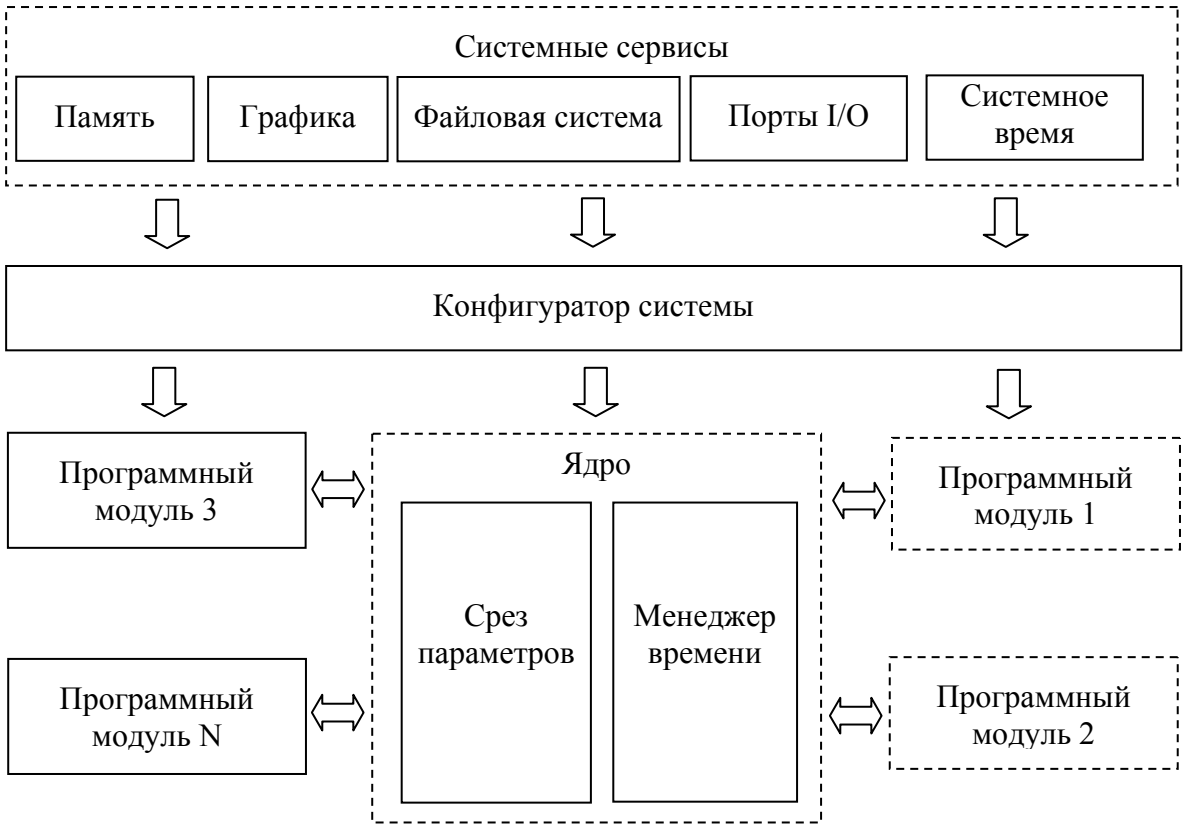

Рис. 5. Структурная схема программного комплекса.

Конфигуратор системы является зависящим от платформы исполняемым приложением, получающим управление при ее запуске. (Например, на платформе с операционной системой Windows это exe файл, тогда как ядро и внешние модули являются динамически линкуемыми библиотеками dll). Функция конфигуратора заключается в инициализации ядра и передаче ему управления. Кроме того, конфигуратор предоставляет интерфейсы к системным сервисам ядру и через него всем модулям системы.

Ядро системы и ряд модулей (показаны пунктирной границей на Рис. 5), в случае необходимости, разрабатываются переносимыми на уровне исходного текста программ на языке ANSI C, что обеспечивает идентичность их работы на различных платформах. Требование кросс-платформенности для ядра обязательно, а для модулей формируется архитектором (разработчиком) конкретной системы по назначению. Очевидно, что свойство переносимости между платформами на уровне исходного кода требует от разработчиков модулей выполнения дополнительных правил при написании исходного текста программы.

Примерами конкретных платформ, под которые может быть реализована программная система указанной структуры, являются

- IBM-совместимый персональный компьютер с операционной системой Windows;
- IBM-совместимый ПК с ОС типа Linux (включая также и МСВС);
- Комплекс на базе специализированного вычислителя (например, типа Багет) как с операционной системой общего назначения (в том числе реального времени), так и без нее;

- Комплекс на базе цифрового сигнального процессора (например, типа TMS), возможно без операционной системы общего назначения;

Важно отметить, что для работы программной системы предлагаемой структуры требуется только реализация используемых системных сервисов, как связующей с аппаратными ресурсами компоненты. Такие сервисы могут быть реализованы на средствах, предоставляемых операционной системой общего назначения (например, Windows или Linux), однако, в случае отсутствия такой ОС, сервисы могут быть реализованы в составе прикладной системы.

Принципиальным также является свойство переносимости ядра и ряда модулей на уровне исходного кода между платформами, в том числе и не заявленными как поддерживаемые в исходных требованиях при их формировании. Такой подход, хотя и требует дополнительных затрат на этапе разработки, обеспечивает гарантию идентичности процесса управления работой системы и логики ее работы на любой платформе, реализуя тем самым независимость программного обеспечения, а именно алгоритмической его части, от конкретной аппаратуры.

3. Технология разработки алгоритмов

С учетом сложившейся тенденции по усложнению математического (алгоритмического) обеспечения во вновь создаваемых программно-аппаратных комплексах обостряется ряд вопросов в части процесса разработки программного обеспечения, а именно:

- Невозможность на этапе планирования четко передать описание требуемой логики работы программного обеспечения системы в виде технического задания от специалиста к разработчику программного обеспечения. Это обусловлено двумя факторами: (1) специалист, за редким исключением, не владеет языками программирования и современными подходами в части разработки программного обеспечения; (2) во многих случаях, на этапе первоначального планирования алгоритм работы системы определен в общих чертах и может быть уточнен только в процессе его разработки.
- Необходимость длительной отладки логики работы системы, реализуемой на уровне программного обеспечения, что чаще всего возможно сделать только при наличии как минимум макетного образца соответствующей аппаратной части. В таком случае время разработки всей системы фактически складывается из времени разработки и изготовления макетного (опытного) образца и времени разработки программного обеспечения.

Таким образом, при разработке алгоритмов необходимо решить две задачи: (1) функционально разделить задачи специалиста, формирующего логику работы изделия, и разработчика программного обеспечения в целях обеспечения независимости их работы; (2) обеспечить возможность разработки и предварительной верификации алгоритмического обеспечения вне зависимости от фактического наличия создаваемой аппаратной платформы.

Для решения поставленных задач разработаны, так называемые, инженерные редакторы алгоритмов в виде прикладного программного обеспечения для персонального компьютера. С помощью таких редакторов специалист (инженер) без привлечения разработчика программного обеспечения создает алгоритм работы разрабатываемой им системы, используя понятные ему структуру и термины. Примеры формы представления алгоритма для специалиста в таком редакторе показаны на Рис. 6, Рис. 7 и Рис. 8.

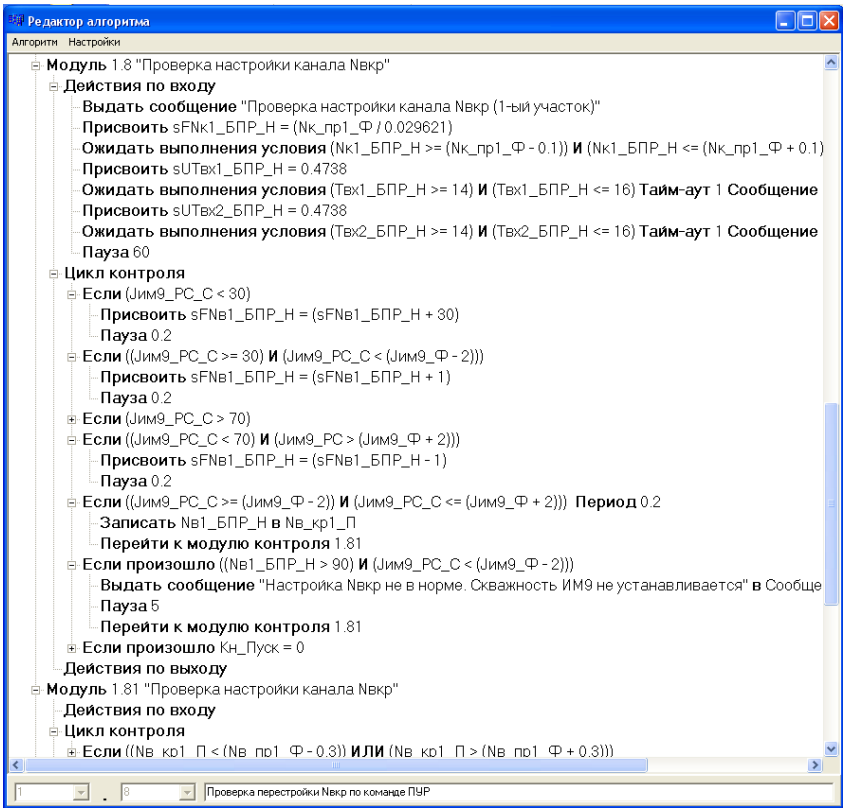

Рис. 6. Пример формы представления алгоритма в виде дерева операций.

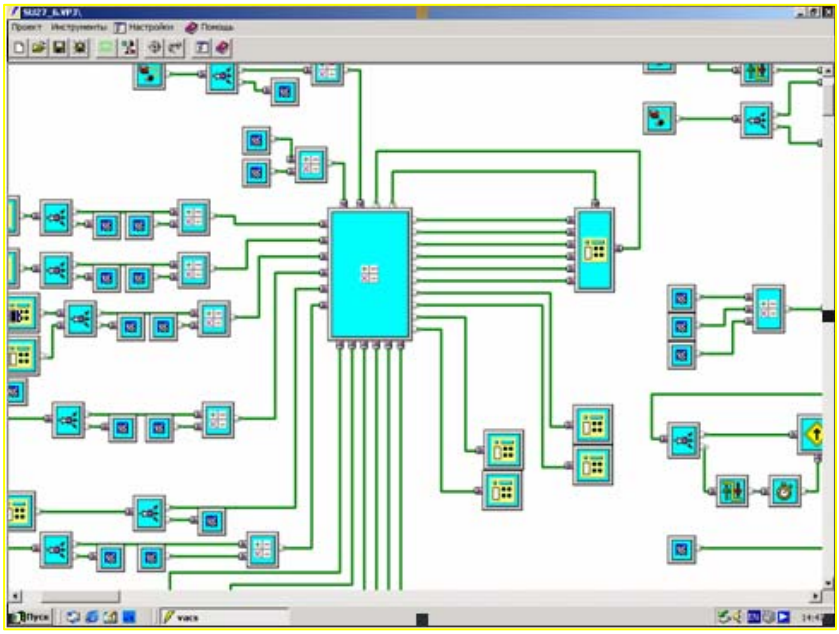

Рис. 7. Пример формы представления алгоритма в виде графа процесса.

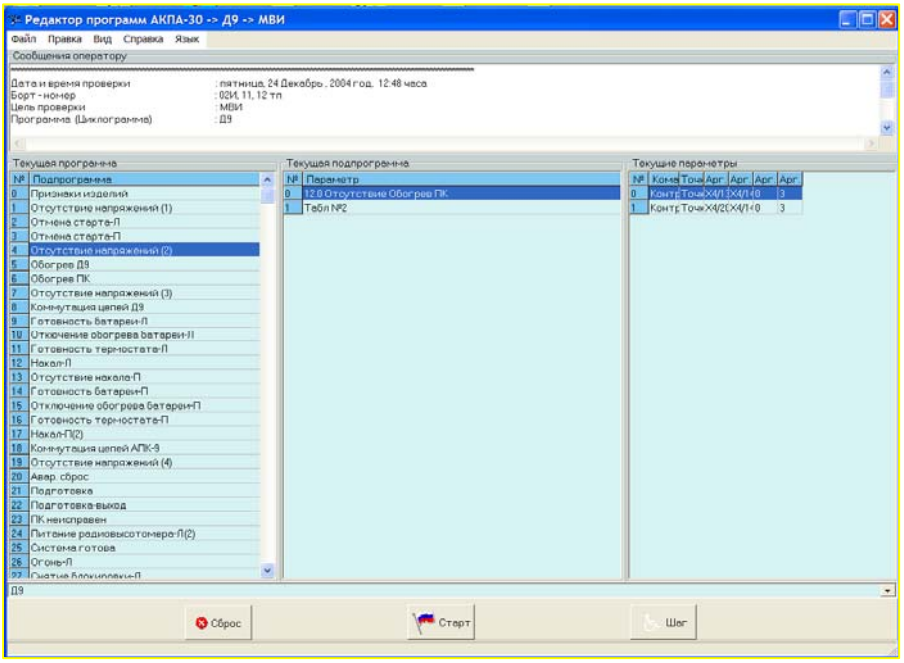

Рис. 8. Пример формы представления алгоритма в виде таблиц операций.

Роль разработчика программного обеспечения в таком процессе сводится к разработке такой прикладной программы, которая помимо взаимодействия со специалистом удобным для него способом обеспечивает автоматическую трансляцию результатов его труда в исходные коды кросс-платформенного модуля на языке ANSI C. Компиляция исходных текстов полученного модуля соответствующим компилятором обеспечивает его исполнения на требуемой аппаратной платформе, построенной по изложенному выше принципу.

Для документального оформления алгоритмов в соответствии с требованиями ГОСТ (например, [1,2]) может также использоваться инженерный редактор, обеспечивающий эту функцию в автоматическом режиме.

4. Управление отображением информации

Во многих программно-аппаратных комплексах, в особенности, в наземных автоматизированных средствах контроля, одной из важнейших задач является отображение оператору параметрической и другой информации в реальном масштабе времени. При этом, поскольку объем такой информации может быть достаточно большим, а эргономические требования к виду представления информации часто носят субъективный характер, то необходимо иметь инструмент по оперативной настройке объема и формы отображения информации «под заказчика», который бы позволял эффективно формировать «информационное поле» без привлечения специалистов по разработке программного обеспечения. При этом может использоваться следующая идеология:

- Специалист описывает элементы отображения для системы отображения информации (например, монитора ПК) в текстовой форме использую специализированный язык описания (см. Рис. 9);
- Текстовое описание формата отображения транслируется технологическим программным обеспечением в исходный код самостоятельного модуля программного комплекса;
- С помощью штатных компиляторов исходный код модуля отображения, сформированный на языке ANSI C или C++, компилируется в исполняемый файл на соответствующей платформе;

- Полученный исполняемый файл встраивается в программный комплекс и обеспечивает отображения необходимой информации (см. Рис. 10) в установленной специалистом, а не разработчиком программного обеспечения, форме.

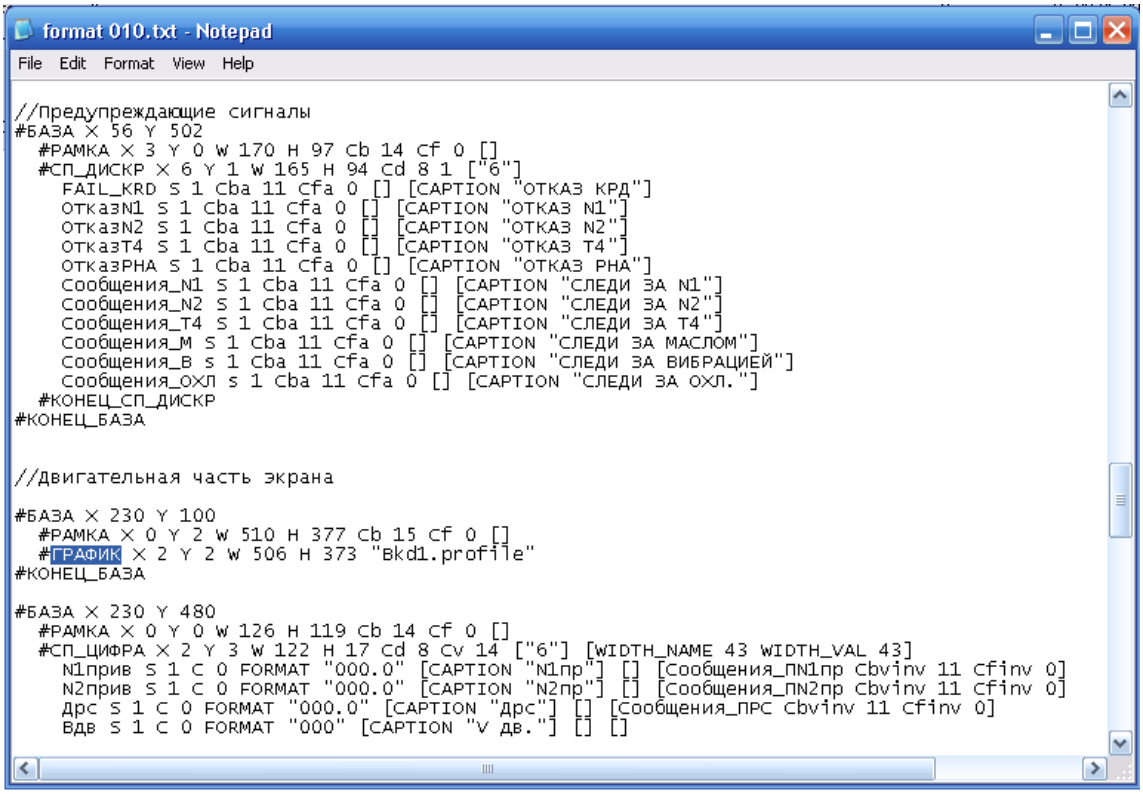

Рис. 9. Пример текстового описания формата отображения.

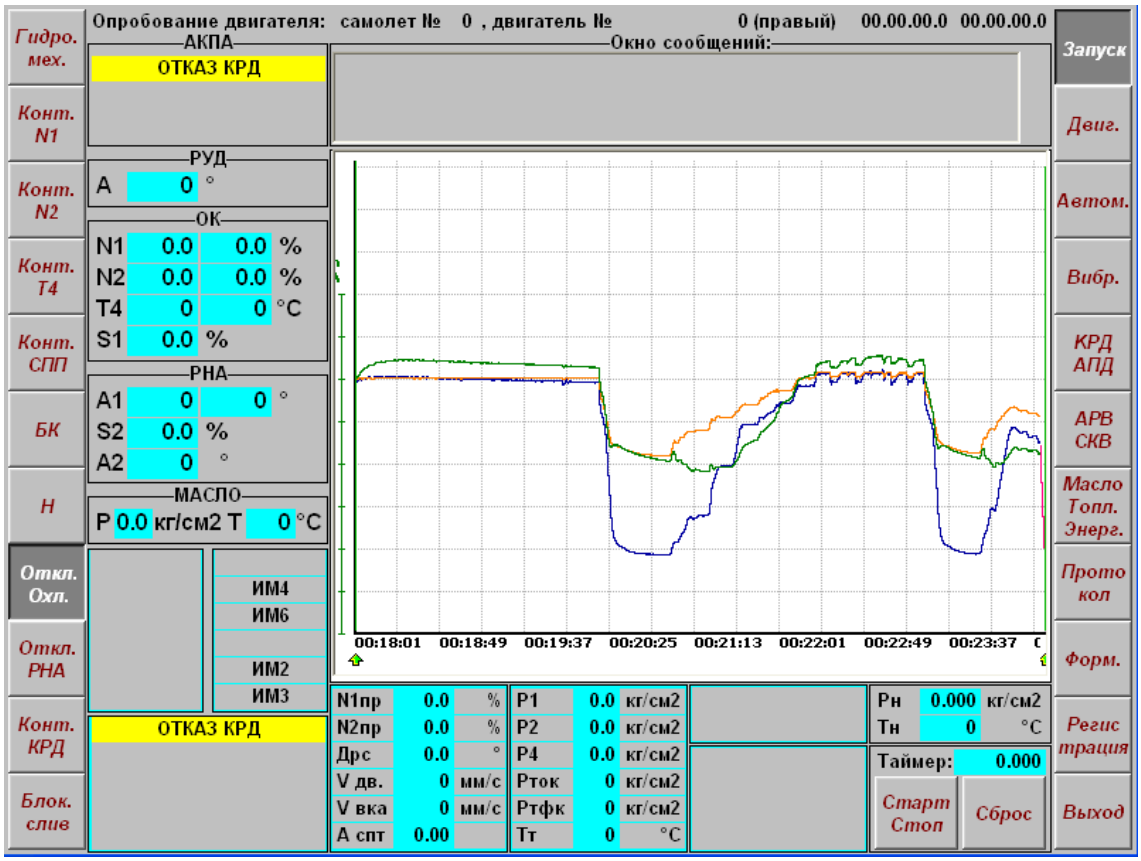

Рис. 10. Пример формата отображения параметров в соответствии с текстовой настройкой.

Важно отметить, что, руководствуясь указанной выше идеологией, специалист имеет возможность изменения объема параметрической информации и способа ее отображения без привлечения разработчика ПО. Такой подход позволяет оптимизировать процесс разработки «информационного поля» для штатных систем, а также обеспечить возможности оперативного управления отображением информации в условиях стенда.

5. Технология стендовой отладки алгоритмов

Процесс разработки логики работы сложных программно-аппаратных комплексов чаще всего является достаточно длительным по отношению к времени разработки всего комплекса и на практике требует вовлечения различных специалистов. Традиционно, отладка и верификация подобных алгоритмов осуществляется при наличии макетных или опытных образцов, для которых имитируются различные входные условия с помощью технологической стендовой аппаратуры и контролируется корректность получаемого результата. Безусловно, такой этап необходим в процессе разработки, однако для его сокращения во времени необходимо обеспечить возможность предварительного формирования и отладки логики работы изделия без фактического наличия штатной аппаратуры.

В случае построения программного обеспечения с учетом принципов, рассматриваемых в данной статье, такой технологический процесс становиться возможным. При этом, специалист, разработав логику работы системы с использованием инженерного редактора, автоматически получает на выходе исходный текст модуля, реализующего необходимый алгоритм. Далее, данный модуль компилируется под технологическую стендовую платформу (например, Windows или Linux) и погружается в соответствующий программный комплекс. Кроме того, этот комплекс может включать следующие модули (см. Рис. 11):

- Модуль имитации параметров, назначением которого является формирование на срезе параметров различных наборов значений для проверки всех режимов работы штатного алгоритма. При этом данные значения могут формироваться вручную либо задаваться из ранее записанных данных регистрации работы реальных систем. Например, для отладки работы бортового комплекса могут использоваться данные бортовых регистраторов, воспроизводимые в масштабе реального времени, что позволяет в точности реконструировать реальные полетные ситуации.
- Модуль отображения параметров состояния системы, позволяющий отобразить значения внутренних параметров системы. Такие значения часто необходимы для анализа корректности работы алгоритма специалистом, однако информация о них часто избыточна в эксплуатации. Такое отображение может быть сформировано с помощью автоматизированных средств управления отображением информации, упомянутых выше.

При наличии в программном комплексе указанных выше модулей для имитации различных ситуаций работы системы и отображения внутренней информации для отладки, возникает возможность в качестве стенда использовать персональный компьютер без специализированной аппаратной части для целей верификации логики работы разрабатываемых изделий.

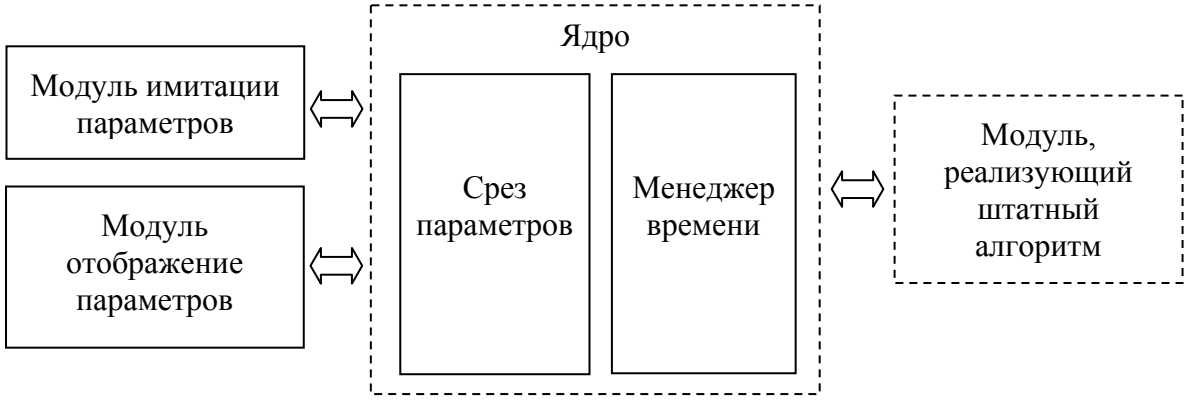

Рис. 11. Структура программного комплекса для стендовой отладки алгоритмов.

После отладки на указанном стенде программный модуль, реализующий логику работы изделия, переносится на уровне исходного текста программы на языке ANSI C на соответствующую штатную программно-аппаратную платформу, которых может быть несколько типов, как указывалось во введении. При этом указанная технология разработки гарантирует, что алгоритм работы системы будет совпадать на всех используемых платформах.

6. Заключение

В настоящей работе сформулированы современные тенденции в разработке программно-аппаратных комплексов, включающие

- Перераспределение функций между программным и аппаратным обеспечением;
- Повышенная степень ответственности;
- Усложнение математического (алгоритмического) обеспечения;
- Независимость программного обеспечения от аппаратной платформы.

С учетом указанных выше тенденций предлагаются современные принципы разработки программного обеспечения бортовых и наземных программно-аппаратных комплексов. Данные принципы включают:

- Наличие модульной архитектуры программного обеспечения, включающей системные сервисы, ядро и модули, реализующие логику работы системы исходя из ее назначения. При этом ядро и часть модулей разрабатывается на языке ANSI C с обеспечением их переносимости на различные платформы.
- Наличие инженерного редактора для разработки алгоритмов работы системы специалистом без привлечения разработчика программного обеспечения. При этом сформированный алгоритм из высокоуровневого языка, удобного для работы специалиста, автоматически транслируется в исходный код и далее компилируется в виде модуля под необходимую аппаратную платформу.
- Наличие технологических программных средств управления объемом информации и способом ее воспроизведения на средствах отображения информации, как в штатных комплексах, так и в условиях стенда. При этом настройка режимов отображения должна также осуществляться специалистом без участия разработчика программного обеспечения.
- Обеспечение возможности разработки, отладки и верификации логики работы бортового или наземного комплекса средствами персонального компьютера без привлечения штатной аппаратуры (макетных, опытных или стендовых образцов). При этом технология разработки обеспечивает наличие гарантии идентичности работы таких алгоритмов на различных платформах, как стендовых, так и штатных.

Рассмотренные принципы создания программного обеспечения широко используются и апробированы на практике в ходе разработки современных программно-аппаратных комплексов бортового и наземного применения в ОАО «Корпорация «Русские Системы». Принимая во внимание накопленный положительный опыт, внедрение и широкое использование указанных в настоящей работе подходов позволяет радикально снизить сроки и стоимость процесса разработки, одновременно качественно расширив возможности по экспертному созданию и верификации алгоритмического обеспечения. Кроме того, применение рассмотренных принципов позволяет сократить время и затраты на доводку систем и их адаптацию под модернизированный объект за счет оперативной доработки или изменения логической части программного обеспечения силами специалистов по конкретному комплексу.

7. Список использованной литературы

1. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
2. ГОСТ Р 51904-2002. Программное обеспечение встроенных систем. Общие требования к разработке и документироанию.