УДК 004.021

ВСТРОЕННЫЕ СРЕДСТВА КОНТРОЛЯ БОРТОВОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ ПОД УПРАВЛЕНИЕМ ОПЕРАЦИОННОЙ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ КАК ИТЕРАТИВНЫЙ АГРЕГИРОВАННЫЙ ОБЪЕКТ

© 2012 Н. А. Долбня

ОАО «Ульяновское конструкторское бюро приборостроения»

В статье описан метод организации встроенных средств контроля под управлением операционной системы жесткого реального времени. Встроенные средства контроля представляются как итеративный агрегированный объект. Объект базируется на паттерне проектирования «Итератор». Показаны преимущества такого подхода, а также детали способа его программной реализации.

Встроенные средства контроля, паттерн проектирования, агрегированный объект, операционная система реального времени.

В большинстве современных авиационных систем, построенных по принципу ИМА (интегрированная модульная авионика), с использованием встраиваемых решений на основе применения ОСРВ (операционных систем реального времени), разработчикам приходится решать круг типовых задач:

- загрузки прикладных программ во флэш-память главного модуля;
- начальную инициализацию контроллеров главного модуля;
- сбор данных с периферийных по отношению к главному модулю контроллеров;
- обработка прерываний от периферийных контроллеров;
- встроенный контроль узлов как главного модуля, так и периферийных.

Загрузка в оперативную, а затем и во флэш-память главного модуля файлов образа ядра ОСРВ и ее корневой файловой системы, а также файлов прикладных программ, проверка целостности этих файлов и прочие операции цехового обслуживания традиционно выполняются посредством начального загрузчика, поставляемого в комплекте с главным модулем системы. Такие загрузчики универсальны, так как содержат все драйвера, необходимые для работы с флэш - памятью модуля, с сетевыми протоколами загрузки файлов.

Начальная инициализация контроллеров главного модуля также выполняется средствами начального загрузчика или, в случае особых требований к времени старта ОСРВ, средствами драйверов ОСРВ. Так как на базе одного процессорного модуля могут выпускаться несколько модификаций вычислителей или индикаторов, а стоимость портирования ОСРВ довольно высока, то ОСРВ является специфичной только для главного процессорного модуля. Поэтому инициализацией периферийных контроллеров занимаются уже прикладные программы посредством обращения к драйверам контроллеров главного модуля.

Сбор данных от периферийных контроллеров по этой же причине также является прерогативой прикладных программ, поскольку чаще всего сбор данных и их предварительная обработка являются специфичными для конкретного исполнения вычислителя или индикатора.

Обработка прерываний выполняется также средствами ОСРВ и прикладных программ в зависимости от того, что является источником прерываний – контроллер главного модуля или периферийный.

Задача же организации встроенного контроля вычислителя и всей системы, в состав которой он входит, с одной стороны, зависит от конкретного исполнения вычислителя, с другой стороны, имеет много общего во всех его исполнениях.

С точки зрения разработчика программного обеспечения для встраиваемой ОСРВ весь модуль с

периферийными устройствами представляет собой древовидную модель, корневым элементом которого является центральный процессор главного процессорного модуля, дочерними элементами которого являются как контроллеры оконечных устройств главного модуля, так и его интерфейсные контроллеры, обеспечивающие доступ к оконечным устройствам уже всего вычислителя или индикатора.

В качестве главного процессорного модуля используется либо ПЛИС, либо полноценный микрокомпьютер, например, на базе процессора PowerPC, имеющий в своем составе контроллеры Ethernet, RS-232 и т.п., которые могут рассматриваться в качестве оконечных устройств главного модуля.

В качестве интерфейсного контроллера главного модуля может выступать любой контроллер интерфейсной мультиплексированной шины, например LocalPlus Bus, на которой расположены все оконечные устройства вычислителя или индикатора. В общем случае как число интерфейсных шин, так и глубина ветвления описанной древовидной модели может отличаться. Важно лишь то, что драйверы интерфейсных шин реализованы как часть ОСРВ (согласно общепринятым требованиям к ОСРВ это должен быть статически слинкованный с ядром код) и формат обращения к этим драйверам специфичен только для ОСРВ и не зависит от исполнения вычислителя или индикатора на ее базе. Специфика же обращения непосредственно к оконечным устройствам вычислителя заключена уже в прикладной программе либо явно, либо в виде библиотеки, скрывающей детали реализации доступа к оконечным устройствам и делающей прикладную программу при условии соблюдения стандарта POSIX максимально переносимой на уровне исходных кодов.

Встроенные средства контроля вычислителя представляют собой совокупность исполняемого кода, который традиционно классифицируют на следующие составляющие:

- стартовый контроль;
- текущий контроль;
- расширенный контроль.

Стартовый контроль выполняется каждый раз при подаче питания на процессорный модуль и отвечает за корректность функционирования внутренних и оконечных устройств процессорного модуля при их инициализации. Специфика используемой ОСРВ, в частности, использование виртуального режима адресации памяти, диктует необходимость размещения кода, отвечающего за стартовый контроль центрального процессора, оперативной и флэш-памяти, в начальный загрузчик, часть которого функционирует еще в физическом режиме адресации памяти. Остальная часть стартового контроля может располагаться в исполняемом коде прикладной программы, в частности, в библиотеке, скрывающей детали реализации доступа прикладной программы к оконечным устройствам вычислителя.

Текущий контроль целиком является частью кода прикладной программы и выполняется параллельно основной прикладной программе. Особенности реализации текущего контроля с одной стороны определяются архитектурой программного обеспечения вычислителя. Это означает, что он может быть вынесен в отдельную программу и даже в одну из отдельных виртуальных машин, механизм которых поддерживается многими современными ОСРВ такими, как LynxOS-178, VxWorks и др. Такая архитектура может быть вызвана различиями в уровнях критичности текущего контроля и прикладного программного обеспечения согласно распространенному в авионике стандарту DO-178B.

С другой стороны, если текущий контроль и прикладная программа, несущая основную функциональную нагрузку, обладают одинаковым уровнем критичности и текущий контроль выполняется итеративно через довольно большие промежутки времени, вполне допустимой считается организация текущего контроля в виртуальном адресном пространстве основной программы, то есть, в виде одного с ней процесса. Такой подход позволяет прикладной программе и текущему контролю использовать единую статическую библиотеку для доступа к оконечным

устройствам вычислителя, что не только экономит имеющуюся в составе главного процессорного модуля флэш-память, но дает возможность избежать двойственности определения одних и тех же данных, если у исполняемого кода программы и текущего контроля будут свои независимые версии библиотеки. Динамически же загружаемая единая библиотека доступа к оконечным устройствам не решает проблему, так как ее использование является крайне нежелательным согласно общей идеологии ОСРВ.

Расширенный контроль выполняется автономно, при остановленной или завершенной основной программе. В таких условиях в отличие от ситуации с текущим контролем, где необходимо учитывать возможность вынужденного монопольного использования аппаратных ресурсов основной задачей, а также жесткие временные характеристики, связанные с использованием ОСРВ, возможности расширенного контроля ничем не ограничены.

Действия же, предусмотренные на случай выявления одной из составляющих частей встроенного контроля аппаратного сбоя, определяются требованиями системного уровня к программному обеспечению вычислителя. Реализация обработки таких ситуаций целиком расположена в исполняемом коде функциональной прикладной программы, а значит, в общем случае не привязана к библиотеке доступа к оконечным устройствам вычислителя.

Таким образом, размещение исполняемого кода встроенного контроля и исполняемого кода библиотеки для доступа к оконечным устройствам вычислителя в едином программном модуле (несмотря на то, что в общем случае архитектурно это разные компоненты) является вполне оправданным.

Основное свойство встроенных средств контроля с точки зрения архитектуры – его итеративный характер. Поскольку в подавляющем большинстве случаев контролируемые оконечные устройства представляют собой набор регистров и в общем случае диапазон памяти, отображенной на пространство адресов интерфейсной шины, действия при контроле каждого устройства можно легко разбить на несколько шагов так, чтобы выполнение каждого шага занимало достаточно мало времени.

Как только такое разбиение на этапе проектирования программного обеспечения вычислителя будет выполнено, весь контроль как компонент начинает представлять собой агрегированный объект с массивом структур, содержащих унифицированное представление о каждом устройстве, и унифицированным набором точек входа:

- инициализация контроля указанного устройства;
- итерация по всем устройствам.

Термин «объект» в данном случае совершенно не накладывает никаких ограничений на используемый язык программирования, так как даже в процедурном языке программирования его можно определить в виде структуры.

Причем точки входа определены для всего контроля в целом. Фактически это один из модулей библиотеки доступа к оконечным устройствам вычислителя.

Первая точка входа определяет, будет ли оконечное устройство, которому соответствует описывающая структура с номером dev_index, задействовано и каким именно образом (битовые флаги ctrl_flags, значение которых для каждого устройства трактуется в зависимости от его специфики):

int ctrl_initizlize(enum DEVICES dev_index, unsigned long ctrl_flags);

Вторая точка входа – просто вызов очередной итерации по очередному устройству:

int ctrl_iterate(void);

Обе точки входа возвращают значение кода ошибки или ее отсутствия. Структура унифицированного описания каждого устройства имеет вид:

typedef unsigned char (*CTRL_ITER)(void*);

```
typedef unsigned char (*CTRL_INIT)(void*);

typedef struct ctrl_node
{
 // контролируется ли узел
 char in_use;
 unsigned long ctrl_flags;
 // символьное имя узла
 char *node_name;
 // номер текущей итерации
 unsigned long iter_index;
 // сообщение об ошибке
 char *error_message;
 // инициализация контроля узла
 CTRL_INIT node_init;
 // итератор для узла
 CTRL_ITER node_iter;
} CTRL_NODE;
```

где node_init - указатель на функцию инициализации контроля по устройству, node_iter – указатель на функцию итерации по контролируемому устройству. Реализации этих функций в силу своей специфики находятся в соответствующих модулях для доступа к устройствам.

Основной смысл такого подхода в том, что функциональной прикладной программе достаточно вызвать унифицированный инициализатор описанного объекта, который вызовет соответствующую реализацию инициализатора конкретного устройства. В зависимости от состояния флага инициализатор может выполнить свою часть стартового контроля. Затем в процессе работы прикладная программа вызывает функцию-итератор контроля, которая каждый раз переходит к очередному устройству и далее вызывает конкретную реализацию итератора конкретного устройства, выполняющего очередной шаг в контроле устройства.

Разработчику библиотеки доступа к оконечным устройствам нужно лишь реализовать для каждого устройства инициализатор и итератор. Общая реализация самого объекта, выполненного с использованием паттерна проектирования «итератор», будет оставаться неизменной, что избавляет разработчиков от необходимости каждый раз заново проектировать архитектуру встроенного контроля конкретного исполнения вычислителя.

Такая унифицированная организация средств контроля практически не сказывается на производительности программного обеспечения вычислителя, так как реализация инициализатора и итератора по-прежнему не является полностью универсальной с точки зрения исполняемого кода.

Независимость от аппаратной составляющей вычислителя позволяет сделать его даже частью ОСРВ и придать ему статус RSC (Reusable Software Component) после процесса сертификации первой же версии программного обеспечения вычислителя, что существенно сократит временные затраты на проектирование и разработку программного обеспечения в последующих версиях.

INTERNAL CONTROLS BOARD COMPUTER SYSTEM RUNNING REAL TIME OPERATION SYSTEM AS AN ITERATIVE AGGREGATE

© 2012 N. A. Dolbnya

PJSC “Ulyanovsk Instrumental Manufacturing Design Bureau”

The article refers to the method of organization of built-in control board computer systems running real time operating system. Built-in controls are considered as an aggregate object is an iterative. This aggregated object is implemented based on the design pattern "iterator". This section describes the advantages and details of the software implementation of this approach.

Built-in control board computer system, design pattern, aggregated object, real-time operation system.

Информация об авторе

Долбня Николай Алексеевич, начальник тематической конструкторской бригады, ОАО «Ульяновское конструкторское бюро приборостроения». E-mail: dolbnya_na@ukbp.ru. Область научных интересов: автоматизация проектирования бортовых информационных систем.

Dolbnya Nikolay Alexeevich, chief of thematic design team of Ulyanovsk Instrumental Manufacturing Design Bureau. E-mail: dolbnya_na@ukbp.ru. Area of research: embedded systems computer aided development.