РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 6 1994

УДК 681.3.06

А. Ю. Бржазовский, А. Н. Ермаков, С. А. Кулагин, С. Л. Мушер, О. В. Сердюков, А. И. Тимошин, В. И. Шагаева

(Новосибирск)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ СИСТЕМ УПРАВЛЕНИЯ АВТОМОБИЛЬНЫМИ ДВИГАТЕЛЯМИ

Приведен краткий обзор современного состояния алгоритмического и программного обеспечений для систем управления автомобильными двигателями внутреннего сгорания. Указаны основные проблемы реализации алгоритмов для серийных бортовых контроллеров. Описаны принципы разработки ядра операционной системы реального времени для инструментальной системы разработки алгоритмов управления двигателями внутреннего сгорания (краткое изложение результатов представлено в [1]). Результаты работы могут применяться также в других системах управления сложными физическими объектами в реальном времени.

Введение. Проблемы защиты окружающей среды и истощение природных ресурсов привели к качественно новому подходу в разработке двигателей внутреннего сгорания (далее ДВС) для серийных автомобилей. Еще в 1983 году правительством США были приняты нормы, которым должен удовлетворять любой автомобиль, находящийся на территории этой страны. Аналогичные требования существуют и в Европе. В ближайшее время произойдет переход к новым нормам «США-93», еще более ужесточающим требования к допустимым параметрам автомобильных двигателей.

Новым шагом в автомобилестроении стал переход к двигателям с так называемым «электронным впрыском топлива». Такие двигатели экономичнее и экологически чище обычных (карбюраторных), но требуют применения высокоточных бортовых вычислительных средств, называемых бортовыми контроллерами или электронными блоками управления (ЭБУ).

Первые бортовые контроллеры серийных автомобилей строились в основном на базе 8-разрядных микропроцессоров, поскольку выполняли относительно простые функции — управление зажиганием и электронным карбюратором. Переход на двигатели со впрыском топлива означает значительное увеличение объема вычислений. В таких устройствах либо используют 16-разрядные микроконтроллеры, либо, как фирма "General Motors", два 8-разрядных. Пока не существует промышленного бортового контроллера, полностью удовлетворяющего нормам «США-93», но прослеживается тенденция к применению в таких устройствах современных 32-разрядных микроконтроллеров [2—4].

В нашей стране также проводятся исследования в области управления ДВС. В частности, в настоящее время на АО «АвтоВАЗ» реализуется проект по созданию ЭБУ для серийных отечественных автомобилей. Авторы принимают активное участие в разработке программного обеспечения (ПО) для ЭБУ в рамках этого проекта. В данной работе изложены постановка задачи, принципы построения и основные характеристики реализованной на этих принципах операционной системы реального времени.

Для выяснения принципов построения программного обеспечения рассмотрим круг задач, стоящих перед ЭБУ и особенно перед ПО.

Электронный блок управления. Обычно ЭБУ выполняется в виде единого блока, содержащего один или несколько процессорных элементов, и схем сопряжения с датчиками и исполнительными устройствами. ЭБУ осуществляет управление исполнительными устройствами ДВС (катушкой зажигания и форсунками для впрыска топлива) на основе данных, получаемых от набора датчиков, среди которых: датчик угла поворота коленчатого вала двигателя (сигнал этого датчика, частота которого может достигать 6 кГц, является главным синхронизирующим сигналом для ЭБУ); датчики температур окружающего воздуха и охлаждающей жидкости; датчик расхода воздуха (частота сигнала до 10 кГц), на его основе рассчитывается необходимый объем топлива; датчик количества кислорода в выхлопных газах (вырабатывающий сигнал обратной связи в цепи управления топливоподачей); датчик детонации (формирующий сигнал обратной связи в цепи управления моментом зажигания).

Как правило, каждый импульс сигналов от датчика угла поворота коленчатого вала двигателя и датчика расхода воздуха является сигналом запроса прерывания микропроцессора, т. е. внешним событием для ПО, установленного в электронном блоке управления. Точность формирования временных характеристик выходных сигналов ЭБУ достигает 10 мкс.

На ЭБУ возлагается задача такого управления ДВС, при котором обеспечиваются: выполнение экологических норм на состав выхлопных газов, сокращение расхода топлива и увеличение коэффициента полезного действия двигателя, улучшение мощностных и динамических характеристик двигателя (по сравнению с неинтеллектуальными регуляторами), диагностика аппаратного окружения ДВС.

Для решения указанных задач ЭБУ содержит программное обеспечение, реализующее соответствующие алгоритмы управления (АУ) ДВС. Разработка и оптимизация алгоритмов являются задачей специалистов по двигателям, но взгляд со стороны позволяет выделить некоторые основные характеристики АУ, во многом определяющие структуру программного обеспечения ЭБУ.

Алгоритмы управления ДВС. Сложность АУ обусловлена в первую очередь высокой частотой внешних событий (до 10 кГц), так как точность поддержания заданных характеристик ДВС, т. е. точность управления, во многом определяется тем, насколько быстро ЭБУ отслеживает изменение внешних условий. Например, идеальным вариантом был бы перерасчет параметров сигналов зажигания и впрыска топлива 2 раза за оборот коленчатого вала двигателя, т. е. каждые 5 мс. Этот период не так велик, как может показаться, из-за большого объема вычислений параметров этих сигналов, а также наличия других сигналов, которые рассчитываются реже, но все же занимают часть процессорного времени.

Большие трудности вносит и отсутствие точного аналитического описания ДВС. Алгоритмы традиционно создаются на базе интуитивного понимания принципов работы двигателя, постоянно изменяются и с ростом требований к точности управления постепенно усложняются.

В настоящее время имеются схемы алгоритмов как для формирования угла опережения зажигания, так и для управления топливоподачей. Рассмотрим вкратце некоторые из этих схем.

ДВС обычно рассматривается как ряд подсистем, каждая с конечным числом внутренних состояний (режимов). В каждом состоянии осуществляется соответствующая логика управления.

Различают как минимум три подсистемы: зажигания, подачи воздуха и топливоподачи.

Подсистема зажигания включает режим запуска двигателя, режим холостого хода и так называемые рабочие режимы. Процесс управления зажиганием заключается в управлении углом опережения зажигания (УОЗ) и временем накопления энергии в катушке зажигания. Оптимальный по экономичности, токсичности и мощности УОЗ определяется по параметрам (оборотам коленчатого вала, нагрузке двигателя, температуре охлаждающей жидкости

и т. д.) с индивидуальной поцилиндровой коррекцией по детонации. Управление временем накопления предназначено для получения энергии, достаточной для формирования искрового разряда в цилиндрах двигателя с минимизацией рассеиваемой мощности в силовом модуле зажигания.

В разных режимах перед подсистемой зажигания ставятся различные задачи. При запуске двигателя это задача набора двигателем заданного количества оборотов. В режиме холостого хода целью управления является поддержание стабильных оборотов двигателя, что достигается взаимодействием подсистем зажигания и управления подачей воздуха. Подсистема управления подачей воздуха обеспечивает поддержание заданных оборотов двигателя с точностью до 30—50 об./мин; более тонкая регулировка производится углом опережения зажигания.

В рабочем режиме критерием оптимальности УОЗ является развитие двигателем максимального крутящего момента.

Во всех режимах системы зажигания происходит корректировка угла опережения индивидуально в каждом цилиндре с целью снижения уровня детонации двигателя.

Подсистема подачи воздуха обеспечивает увеличение подачи воздуха при запуске двигателя и его прогреве, грубую стабилизацию оборотов холостого хода, а также предотвращает самопроизвольную остановку двигателя.

Топливоподача осуществляется посредством программного управления топливными форсунками, впрыскивающими топливо в цилиндры двигателя. Время открытого состояния форсунки задает количество впрыскиваемого топлива. Требуемое количество топлива определяется по ряду параметров (потребляемое количество воздуха, положение дроссельной заслонки, нагрузка и т. д.) с учетом состояния двигателя с коррекцией по составу выхлопных газов с целью достижения оптимальных характеристик двигателя в зависимости от режима работы.

Подсистема топливоподачи должна обеспечивать принудительное обогащение топливной смеси в режимах запуска и ускорения, а также выполнение экологических норм в стационарных режимах, для чего осуществляется коррекция вычисленного количества топлива по показаниям датчика кислорода. Также в режиме торможения обычно происходит принудительное обеднение топливной смеси вплоть до полной отсечки топлива в целях экономии.

Необходимо отметить обязательное наличие в составе АУ средств самообучения. Благодаря им удается осуществлять более точный предварительный расчет момента зажигания и длительности впрыска, тем самым происходит значительное сокращение времени последующей коррекции этих параметров цепями обратной связи (по детонации и составу выхлопных газов соответственно), что, в свою очередь, уменьшает время, в течение которого работа двигателя не удовлетворяет установленным нормам. Средства самообучения позволяют ПО адаптироваться к изначальному разбросу характеристик двигателей, их изменению в процессе эксплуатации, равно как к изменениям условий эксплуатации. Аппаратно эти средства выполняются в виде энергонезависимой памяти, содержащей набор интегральных поправочных коэффициентов, корректирующих УОЗ, количество впрыскиваемого топлива и подачу воздуха.

Нормы «США-93» содержат требования обязательного наличия средств самодиагностики, назначением которых являются обнаружение и фиксация в реальном времени неисправностей аппаратного окружения, а также перевод в этом случае систем управления на резервные режимы работы, позволяющие автомобилю добраться до места ремонта с незначительным ухудшением характеристик ДВС.

Выше перечислены лишь основные принципы управления ДВС. Но даже из вышеизложенного можно сделать ряд выводов:

— АУ отличает высокая сложность, поэтому реализующее их ПО будет достаточно громоздким;

— различные разделы АУ часто изменяются с целью достижения более высоких показателей;

— АУ, а также реализующее их ПО легко и логично разбиваются на ряд независимых или слабосвязанных подсистем или процессов.

Принципы построения программного обеспечения. ПО для первых бортовых контроллеров разрабатывалось по принципу последовательного программирования, т. е. АУ реализовывались в виде единой программы (обычно на языке Ассемблер), которая затем заносилась в ПЗУ и устанавливалась на ЭБУ. При таком подходе в ЭБУ устанавливается программное обеспечение, непосредственно необходимое для управления ДВС, тем самым максимально используются такие ресурсы электронного блока управления, как процессорное время и объемы памяти (ОЗУ и ПЗУ). Однако имеется и ряд существенных недостатков: сложность отладки ПО и изменения ПО при изменении АУ, сложность адаптации ПО при смене «окружения» ДВС, практическая неосуществимость переноса ПО на другой тип процессора.

До тех пор пока объемы ПО были невелики, эти недостатки не проявляли себя в полной мере. Но алгоритмы, реализующие нормы «США-93» или других стандартов аналогичного класса, существенно отличаются от ранее применяемых более сложной внутренней структурой и значительным увеличением объема вычислений. Бурное развитие микроэлектроники приводит к появлению еще более мощных процессоров, позволяющих реализовывать все более сложные АУ. Аппаратная независимость программного обеспечения и его переносимость выступают на передний план. С увеличением объема ПО также обостряется проблема обеспечения его надежности, т. е. способности выполнять возложенные на него функции в любых ситуациях. Многие ситуации встречаются чрезвычайно редко и практически не поддаются моделированию.

Все это вынуждает применять другие методы построения программного обеспечения. Один из таких методов давно применяется в других областях науки и техники, но, насколько известно авторам, не использовался в ПО для ЭБУ ДВС. Суть этого метода в приложении к данной проблеме можно сформулировать следующим образом:

1. Применение специализированных операционных систем (ОС) реального времени, что позволяет создать универсальную среду для работы АУ. Эта среда не зависела бы от аппаратного окружения процессора и его типа и взяла бы на себя часть работы, не связанной напрямую с управлением ДВС, например, интерфейс с аппаратным окружением, синхронизацию ветвей алгоритмов, распределение процессорного времени между модулями АУ в соответствии с их приоритетами.

2. Реализация АУ ДВС либо на языках программирования высокого уровня, либо с помощью алгоритмических языков, специально разработанных для задач управления, что позволяет свести до минимума зависимость ПО от типа процессора.

Преимущества такого подхода общеизвестны: процесс разработки и отладки ПО становится простым и стандартизованным; при использовании многозадачных ОС оказывается проще обеспечить параллельность работы ветвей алгоритмов, так как это свойство заложено в самой ОС; повышение надежности ПО вследствие того, что применение языков высокого уровня уменьшает вероятность ошибок; аппаратная независимость и переносимость ПО.

К недостаткам этого подхода можно отнести накладные расходы, вносимые ОС, т. е. суммарное время, расходуемое процессором на выполнение кода ОС, что, естественно, требует применения более мощных процессоров, чем это необходимо. Кроме того, происходит некоторое увеличение размера ПО вследствие применения языков высокого уровня.

Подобный подход к разработке ПО для ЭБУ ранее, по-видимому, не применялся. По нашему мнению, на это были три причины:

1) недостаток производительности 8-разрядных микропроцессоров отодвигает проблему переносимости на второй план;

2) появление более мощных 16-разрядных микропроцессоров могло бы решить эту проблему, однако простым решением оказалось применение микропроцессоров, совместимых с ранее используемыми; это позволяет перенести ПО с 8- на 16-разрядные микропроцессоры почти без изменений;

3) возможно, что подобные разработки ведутся в мире, но данная информация не является общедоступной.

С появлением на рынке мощных 32-разрядных микроконтроллеров MC6833x фирмы "Motorola" стало возможным применять операционные системы реального времени в серийных автомобильных контроллерах.

На основе анализа существующих АУ ДВС можно сформулировать требования, которым должна удовлетворять ОС для ЭБУ.

1. АУ ДВС легко и логично разбиваются на ряд независимых или слабо связанных между собой подсистем. Поэтому для их реализации желательна многозадачная исполнительная система. Реализация независимых подсистем в виде отдельных процессов облегчает разработку ПО, повышает его надежность, упрощает модификацию и перенос на другие микропроцессоры.

Алгоритм планирования задач такой ОС может быть самым простым — предварительным планированием по фиксированным приоритетам, когда более приоритетная задача остается исполняемой до тех пор, пока сама не завершит работу или не появится задача с большим приоритетом. Этого достаточно для реализации существующих АУ ДВС, так как весь круг задач известен на момент создания системы и имеется возможность изначального распределения приоритетов.

2. Требования высокой точности управления и высокая частота внешних событий (6—10 кГц) заставляют применять только системы реального времени, причем основным критерием применимости конкретной ОС является минимальность объема накладных расходов, вносимых операционной системой в обработку прерываний. Задержка обработки внешних прерываний, т. е. сумма максимально возможного времени запрета прерываний и времени обработки прерываний, осуществляемой ОС, не должна превышать 5—10 мкс, чтобы не занимать более 10—20 % доступного процессорного времени (числа взяты из расчета двух источников прерывания с частотами 6 и 10 кГц). Этот параметр настолько важен, что ради его минимизации можно пожертвовать многими другими возможностями ОС.

3. Необходима развитая подсистема ввода/вывода. Стандартизация программного интерфейса с периферийными устройствами делает ПО аппаратнонезависимым, что облегчает процесс его адаптации при переходе на другую аппаратную конфигурацию. При этом система ввода/вывода не должна существенно замедлять обмен с периферийными устройствами.

4. Требуются средства межзадачной коммуникации и синхронизации. К обязательным средствам можно отнести поддержку общей области памяти, через которую процессы могут обмениваться информацией или пользоваться общими данными. Желательны также механизмы сигналов и семафоров.

5. ОС не должна требовать много памяти, так как системные ресурсы ЭБУ ограничены. Предварительные оценки показывают, что ЭБУ, содержащий свыше 64 Кбайт ПЗУ или свыше 8 Кбайт ОЗУ, будет слишком дорогим для массового производства. Большая же часть доступной памяти приходится на ПО, реализующее АУ. Поэтому желательно, чтобы ОС занимала не больше 4—8 Кбайт ПЗУ и 2—4 Кбайт ОЗУ.

6. ОС должна быть расширяемой, чтобы ее конфигурацию можно было менять в зависимости от решаемой задачи, так как по мере усложнения АУ может появиться необходимость внесения в ОС новых функций, например поддержки многопроцессорных систем.

Поскольку в качестве центрального элемента будущего ЭБУ в силу ряда причин выбирался микроконтроллер MC68333 (MC68332), то был проделан анализ существующих ОС реального времени для указанных микроконтроллеров. К сожалению, ни одна из них не пригодна в качестве ОС для управления ДВС. Большинство из них не проходят по критериям требуемого объема памяти и максимальной задержки обработки прерываний [5—10].

В силу этого было принято решение по написанию собственной операционной системы реального времени, удовлетворяющей вышеперечисленным требованиям. Эта система была названа MTX (multi-task executive — многозадачная исполнительная система), и в настоящий момент под ее управлением

работает ПО аппаратно-программного комплекса, являющегося прототипом будущего ЭБУ.

Ниже приводятся краткое описание основных идей, реализованных в МТХ, и его характеристики, а на рисунке представлена блок-схема системы.

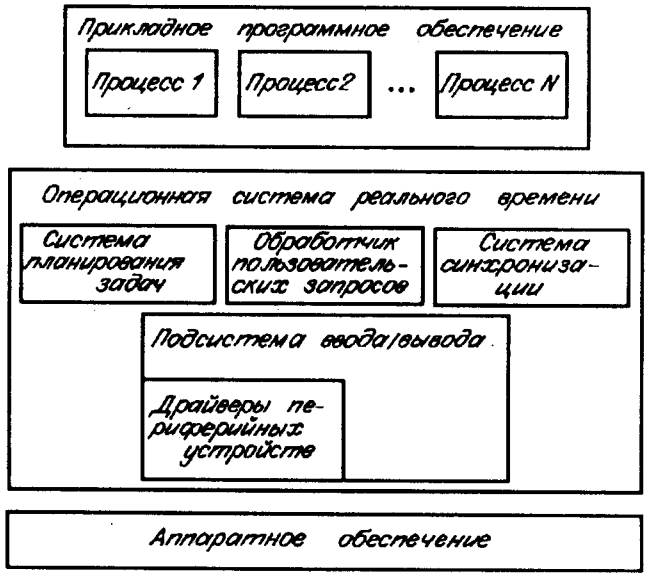
Разработанный монитор является простой, небольшой по объему и быстрой многозадачной исполнительной системой реального времени. Он предназначен для использования в системах на базе микропроцессоров MC680x0 или микроконтроллеров MC683xx фирмы "Motorola". Основная задача МТХ заключается в распределении системных ресурсов (процессорного времени, ОЗУ, периферийных устройств) между задачами, в синхронизации задач, а также в минимизации времени реакции на внешние события (прерывания).

Приведем для иллюстрации основные характеристики созданной системы (все времена измерялись на MC68332, 16 МГц): время задержки обработки прерывания 4 мкс; максимальное время запрета прерываний внутри ядра МТХ 2 мкс; время переключения задач 20 мкс; максимальное количество одновременно исполняющихся процессов 255; требуемый объем памяти: 3 Кбайт ПЗУ или ОЗУ для кода, 0,5 Кбайт ОЗУ для системных переменных плюс до 1 Кбайт для таблицы векторов прерываний плюс по 100 байт на каждую исполняемую задачу.

Для разработки прикладного программного обеспечения, работающего для МТХ, возможно использование в качестве кросс-средств стандартных для операционной системы OS-9 компилятора С, ассемблера и сборщика задач. Кроме того, МТХ имеет возможность работать на одном процессоре одновременно с OS-9, что позволяет создавать и отлаживать программы, используя мощь этой операционной системы, а затем переносить их на микроконтроллер.

В МТХ реализован алгоритм приоритетного обслуживания задач (preemptive priority scheduling). По этому алгоритму исполняется та задача, которая имеет самый высокий приоритет. Текущая задача может быть отложена только при появлении более приоритетной задачи. Такой алгоритм минимизирует накладные расходы, вносимые ядром ОС, но имеет недостаток, заключающийся в том, что процесс, став исполняемым, может заблокировать выполнение остальных менее приоритетных процессов. В принципе это недопустимо в многопользовательских ОС, но для встраиваемых систем вполне приемлемо, так как блокировки задач можно избежать написанием ПО соответствующим образом и корректным распределением приоритетов между процессами.

Для обеспечения независимости прикладного ПО от аппаратного окружения в состав МТХ была включена подсистема ввода/вывода (ПСВВ), состо-



ящая из двух уровней. Нижний уровень ПСВВ составляют драйверы внешних устройств, реализующие стандартный интерфейс обмена с устройствами для верхних уровней. Верхний уровень ПСВВ определяется как аппаратным окружением, так и спецификой прикладной задачи. Он ответствен за своевременное обновление данных, необходимых для прикладного ПО, что осуществляется как по запросам сверху, так и асинхронно.

При разработке подсистемы ввода/вывода основное внимание было уделено минимизации накладных расходов при взаимодействии с периферийными устройствами. С этой целью в качестве основного механизма обмена с устройствами ввода/вывода применялся механизм ассоциированных областей памяти. Суть этого метода заключается в том, что прикладному ПО на стадии инициализации передается от ПСВВ набор адресов, по которым в памяти расположены данные, поступившие от конкретных внешних устройств. Прикладное ПО затем косвенно по полученному адресу обращается к ассоциированной области и передает или получает данные. Ассоциированной областью может быть как действительный регистр периферийного устройства, так и область ОЗУ, в которую периодически заносится обновленная информация. В последнем случае обновление информации в ассоциированной области является функцией верхнего уровня ПСВВ. При таком подходе накладные расходы на ввод/вывод либо полностью отсутствуют, либо минимальны, так как ПСВВ функционирует вне контекста какой-либо задачи.

МТХ содержит простые средства обеспечения коммуникации и синхронизации между задачами. В настоящий момент реализованы механизм сигналов и поддержка общей области памяти.

Конечно, МТХ значительно уступает другим операционным системам реального времени по богатству сервисных возможностей. Подобная «ограниченность» была введена специально с целью минимизации таких критических характеристик, как объем требуемой памяти, время реакции на прерывания и объем накладных расходов, вносимый ОС.

Кроме того, изначально заложенная в МТХ способность к расширению, которая является следствием модульного принципа построения ядра, позволяет либо по мере необходимости наращивать мощность ядра, либо исключать ненужные для данного приложения свойства, уменьшая тем самым размер ядра и экономя системные ресурсы.

Заключение. В качестве примера реализации изложенных принципов разработано ядро многозадачной операционной системы реального времени МТХ для микропроцессоров семейства MC68000. Его отличает малое время реакции на внешние прерывания (около 5 мкс на MC68332, 16 МГц) и малые размеры (3 Кбайта кода и 1—1,5 Кбайта системных переменных).

Разработанное ядро операционной системы реального времени МТХ в первую очередь предназначено для конкретного применения — поддержки АУ ДВС. Однако никакие свойства ядра не являются специализированными для данного типа приложений. Реальная область применения, на наш взгляд, значительно шире — встроенные микроконтроллеры для управления физическими и технологическими объектами или установками в «жестком» реальном времени.

СПИСОК ЛИТЕРАТУРЫ

1. Serdyukov O. V., Kulagin S. A., Ermakov A. N. et al. High performance instrumentation for a car electronics design // Мат-лы конф. VVconex94 "VMEbus and VXIbus SYSTEMS in INDUSTRY and RESEARCH", 31 мая — 3 июня 1994 г.—М., Россия, 1994.
2. Коуб Дж. Рост мощности бортовой электроники // Автомобильная промышленность США.—1991.—№ 7.
3. Айверсен У. Р. Развитие автомобильной электроники в США // Электроника.—1989.—№ 19.
4. Гош Дж. Стремление Западной Европы не отстать в развитии автомобильной электроники // Там же.
5. OS-9 Advanced System Manual: Microware Systems Corp., 1989.

6. VxWorks Programmer's Guide. Wind River Systems. Inc.
7. VxWorks — партнер UNIX в системах реального времени // Открытые системы, 1993.
8. VRTXsa User's Guide: Ready Systems, 1993.
9. VRTX32 User's Guide: Ready Systems, 1993.
10. Pohjanpalo H. MROS-69K a memory resident operating system for MC68000 // Software.— 1981.—11, N 8.

Поступила в редакцию 12 июля 1994 г