УДК 621.391

ВЫБОР ОПЕРАЦИОННОЙ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ СИСТЕМЫ УПРАВЛЕНИЯ ВПЧА

Карасев А.В., Егунова А.И., Егунов О.И., Таланов М.В.
ГОУВПО НИ «Мордовский государственный университет им. Н. П. Огарева», г. Саранск
Тел. +7(834-2) 290602

Аннотация. Рассмотрены вопросы взаимодействия системы управления асинхронным двигателем и ВПЧА. Выполнен анализ существующих операционных систем реального времени и выбирается ОСРВ, соответствующая заявленным требованиям.
Ключевые слова: система управления, протокол взаимодействия, операционные системы реального времени.

Постановка задачи

Система управления ВПЧА имеет иерархическую структуру и предназначена для сбора, обработки и отображения данных, получаемых с подчиненных модулей. Система управления состоит из Головной станции управления преобразователем – промышленного компьютера с ОС реального времени. В его задачи входит задание режимов работы всех подчиненных модулей ЦСУ: модуля управления инвертором, модулями управления трехфазного тиристорного выпрямителя, измерительными модулями, модулями защиты; обеспечение регулирования технологических параметров нагрузки, сбор и хранение диагностических данных, связь с более высокими уровнями, например, через Интернет или по специальным линиям связи; отображение данных в удобной для оператора форме.

Для взаимодействия головной станции с модулями разработаны протоколы обмена данными между блоками СУ. Головная станция (HOST) соединена с модулями по кольцевой топологии, где данные передаются последовательно в одном направлении от одного модуля другому. Промышленный компьютер инициирует любую передачу данных, а модули являются ведомыми и не могут самостоятельно инициировать обмен, за исключением дейтаграмм об ошибках.

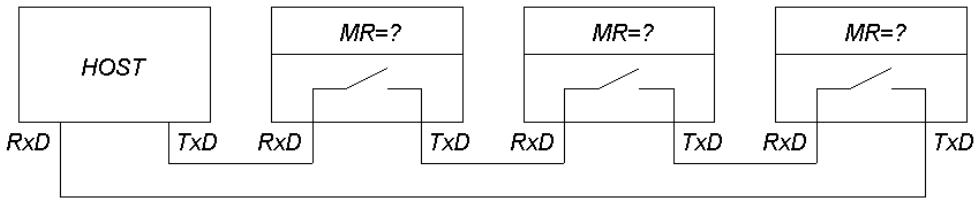

Рис.1. Схема соединения модулей

Все модули оснащены микроконтроллером (МК) и взаимодействуют с каналом передачи данных посредством модуля UART. Для обеспечения связанности в кольце в станции имеются кроме модуля UART, два дополнительных буфера. Данные буферы производят физическую коммутацию канала передачи данных, осуществляя подключение передатчика UART МК к каналу передачи данных, или поддержание связанности кольца.

Промышленный компьютер, выполняющий функции головной станции управления преобразователем, построен на одноядерном чипсете Mobile Intel 945 с использованием 2ГБ оперативной памяти. Обмен информацией с регистрирующими и измеряющими модулями выполняется по 8-ми последовательным портам, имеющим каждый FIFO-буффер емкостью в 512 байт. Параметры с контроллеров модулей измерений на входе и выходе выпрямителя передаются в головную станцию через 10 мс на последовательные порты, работающие с би-

товой скоростью 19 200 bit/s. Модули управления вентильным блоком выпрямителя и инвертора взаимодействуют с промышленным компьютером на скорости в 57 600 bit/s.

Операционная система реального времени должна гарантировать успешность работы любой программы в зависимости не только от её логической правильности, но и от времени, за которое она выполнила заданные функции. Если система не может удовлетворить временным ограничениям, должен быть зафиксирован сбой в её работе. Заданные условия временной обработки массива параметров, поступающих с датчиков системы управления ВПЧА по восьми COM – портам в промышленный компьютер, предъявляют повышенные требования к выбору операционной системы реального времени.

По стандарту POSIX 1003.1 ОСРВ определяется как: «Реальное время в операционных системах — это способность операционной системы обеспечить требуемый уровень сервиса в определённый промежуток времени». Существуют операционные системы мягкого и жесткого реального времени, однако система аппаратного управления должна работать с операционной системой жесткого реального времени. Она характеризуется минимальными задержками реакции на событие и выполняет заданные процессом задачи функции в строго указанное время. Жесткие системы не допускают задержек, так как они могут привести к непредсказуемым последствиям, потери результатов, авариям или большим финансовым потерям. В них предусмотрена обработка критических ситуаций, позволяющая прерывать или блокировать медленные процессы, реализуя выполнение требований надежности и готовности остальной части системы.

ОСРВ строятся по монолитной, уровневой (слоевой) и клиент-серверной архитектурам. Монолитная архитектура обладает плохой предсказуемостью взаимодействия модулей сложного ядра в работе с аппаратурой системы. Уровневая архитектура позволяет работать с аппаратурой не только через ядро, но и обращаться напрямую. Архитектура «клиент-сервер» выносит сервисы ОС в виде серверов на уровень пользователя и выполняет микроядром функции диспетчера сообщений между клиентскими пользовательскими программами и серверами – системными сервисами. Такая архитектура позволяет повысить надежность, обеспечить хорошую масштабируемость и повышенную отказоустойчивость.

В настоящий момент широкое распространение получили QNX, RTX для Windows и RTKernel для DOS.

QNX — коммерческая POSIX-совместимая операционная система реального времени, предназначенная преимущественно для встраиваемых систем. Считается одной из лучших реализаций концепции микроядерных операционных систем.

Это юниксоподобная, POSIX-совместимая, многозадачная, многопользовательская, многопоточная, встраиваемая, легко и гибко масштабируемая операционная система реального времени и в отличие от Linux и BSD не требует пересборки ядра - любые модули могут подключаться и отключаться на лету. Кроме того, ее производительность высока, несмотря на то, что в ней используется защищенная модель памяти, а не принцип единого адресного пространства, характерный для ОСРВ. Есть открытая версия GNU GPL аналогичной операционной системы реального времени под названием ChorusOS (возможно, что QNX является подражанием Chorus) разработанная фирмами Sun Microsystems, Alcatel и ABB. Гостехкомиссия сертифицировала защищенную операционную систему реального времени QNX 4.25 как изделие КПДА.00002-01 по 3 классу защиты от несанкционированного доступа (НСД) и 2 уровню контроля отсутствия недекларированных возможностей (НДВ) за № 906 ФСТЭК России. В сертификате изделие определено как защищённое средством вычислительной техники (СВТ), которое может быть использовано при создании автоматизированных систем (АС), имеющих класс защищённости до 1Б включительно. В настоящее время ФСТЭК России продлила действие сертификата соответствия № 906 до 30.05.2013 и подтвердила соответствие программных обновлений соответствующим классам защиты. В новой версии добавлена поддержка графических приложений Photon и в состав изделия включены модули Photon Runtime, а так же средства поддержки русского языка SWD CyrPack и Photon Cyrillic Supplement. С 2005 года стартовал процесс сертификации следующей ЗОСРВ на базе техно-

логий QNX6 "Нейтрино" и завершился утверждением изделия КПДА.10964-01, с определением соответствия 3 классу защиты от несанкционированного доступа (НСД3).

Система QNX состоит из небольшого ядра (микроядра) и набора взаимодействующих процессов. Как показано на рис. 1, система не имеет иерархической структуры, ее организация скорее напоминает "спортивную команду", в которой игроки (процессы), имеющие равную значимость, взаимодействуют друг с другом и со своим "ведущим игроком" (ядром).

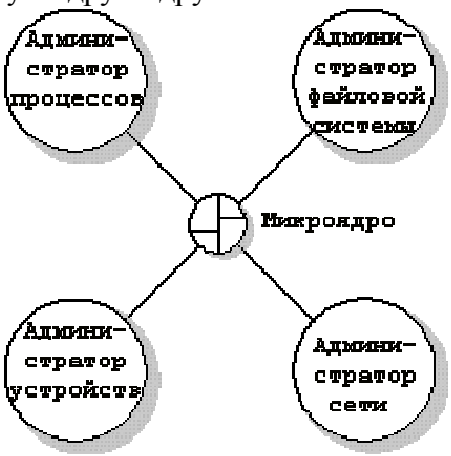

Рис. 2. Микроядро системы QNX координирует работу системных администраторов.

Ядро является "сердцем" любой операционной системы. В некоторых системах на ядро возложено такое количество функций, что, по сути дела, оно само является полной операционной системой. В системе QNX ядро является действительно ядром и имеет небольшой размер_-_менее 8 Кбайт. На ядро системы QNX возложено выполнение только двух основных функций:

— передача сообщений (ядро реализует передачу всех сообщений между всеми процессами во всей системе);
— планирование (планировщик является частью ядра и подключается каждый раз, когда процесс меняет свое состояние в результате появления сообщения или прерывания).

В отличие от процессов само ядро никогда не планируется к выполнению. Управление передается ядру только в результате прямого вызова ядра либо из процесса, либо по аппаратному прерыванию.

Все функции, выполняемые операционной системой QNX, за исключением функций ядра, реализуются стандартными процессами. В типичной конфигурации системы QNX имеются следующие системные процессы:

— Администратор процессов (Proc);
— Администратор файловой системы (Fsys);
— Администратор устройств (Dev);
— Сетевой администратор (Net).
— Системные процессы и процессы пользователя

Системные процессы практически ничем не отличаются от любого процесса пользователя: у них нет специального или скрытого интерфейса, недоступного процессу пользователя.

Именно такая архитектура обеспечивает системе QNX неограниченную расширяемость. Поскольку большинство функций QNX выполняется стандартными системными процессами, то расширить операционную систему совсем не сложно: достаточно написать и включить в систему программу, реализующую новую функцию ОС.

Действительно, грань между операционной системой и прикладными программами весьма условна. Единственным принципиальным отличием системных процессов от прикладных является то, что системные процессы управляют ресурсами системы, предоставляя их прикладным процессам.

Драйверы устройств - это процессы, которые избавляют операционную систему от необходимости иметь дело со всеми особенностями работы аппаратного обеспечения.

Поскольку драйверы выполняются как стандартные процессы, то добавление нового драйвера в систему QNX не влияет на работу других компонентов операционной системы. Единственное изменение, которое происходит в среде QNX - это запуск нового драйвера.

В типичной многозадачной среде, при одновременном выполнении нескольких процессов реального времени, операционная система должна обеспечивать возможность взаимодействия процессов друг с другом.

Связь между процессами (interprocess communication - IPC) является ключом к разработке приложений, представляющих собой набор взаимосвязанных процессов, в котором каждый процесс выполняет одну строго определенную функцию.

В QNX реализован простой, но мощный набор IPC-возможностей, благодаря которому значительно упрощается разработка приложений, представляющих набор взаимодействующих процессов. IPC основан на принципе передачи сообщений от одного процесса другому, при этом формат сообщений зависит только от данных взаимодействующих процессов.

Синхронизация выполнения нескольких процессов происходит посредством сообщений. При передаче, получении и выдаче ответа на сообщения процессы изменяют свое состояние, что определяет время и продолжительность их выполнения. Располагая информацией о состоянии и приоритетах процессов, ядро может максимально эффективно планировать их выполнение, используя все доступные ресурсы центрального процессора. Этот метод передачи сообщений используется глобально во всей системе.

Для приложений реального времени требуется зависимая форма IPC, т.к. процессы, входящие в состав подобных приложений, сильно взаимосвязаны. Механизм передачи сообщений, реализованный в системе QNX, позволяет обеспечить высокую надежность работы приложений.

QNX изначально разрабатывалась как сетевая операционная система. В некоторых аспектах сеть QNX скорее похожа на универсальную вычислительную машину (mainframe), чем на набор микрокомпьютеров. В частности, благодаря большому набору ресурсов, которые могут быть доступны любому приложению. Но в отличие от универсальной машины QNX обеспечивает высокореактивную среду, поскольку любой узел сети может задействовать требуемые ему вычислительные мощности.

Например, программы, управляющие работой устройств ввода/вывода, работающие в реальном времени, могут потребовать больше ресурсов, чем другие, менее критичные ко времени, приложения (текстовый процессор). Сеть QNX достаточно реактивна для того, чтобы поддерживать оба типа приложений одновременно: QNX позволяет сконцентрировать вычислительную мощность там, где и когда это необходимо без ущерба для параллельной обработки данных.

Компания VenturCom создала расширение RTX(Real Time Extention) для операционной системы Windows (NT, NT Embedded, 2000, XP, XP Embedded). Расширение RTX позволяет создавать приложения с детерминированным и очень малым временем реакции на внешние события и, кроме того, обеспечивает пользователя средствами и утилитами для построения и выполнения программ реального времени, а также средствами для измерения и "тонкой" настройки производительности, как аппаратных, так и программных средств. RTX глубоко интегрировано в ядро Windows и для обеспечения необходимых функций использует сервис Windows и WIN32 API.

Расширения реального времени добавляют к Windows специфическую для реального времени функциональность, а именно:

- Процессы реального времени, управляемые собственным планировщиком. Этот планировщик работает уже по всем правилам планировщиков реального времени и использует алгоритм вытеснения по приоритетам. Кроме того, процессы реального времени имеют преимущество перед стандартными процессами Win32, вытесняя их. Процессы реального времени имеют совсем иную, по

сравнению со стандартными процессами Windows, степень надёжности и специфическую функциональность.

- Взаимодействие процессов реального времени и стандартных процессов Win32 друг с другом.
- Программный интерфейс RTAPI для процессов реального времени, реализующий развитый набор средств, характерный для программных интерфейсов (API) операционных систем реального времени.
- Одно и то же приложение может использовать как стандартные функции Win32, так и специфические функции API реального времени (RTAPI), что позволяет выделять критические участки кода приложений Windows и контролировать время и надёжность их выполнения.
- Контроль над работоспособностью и временем реакции системы. "Зависания" стандартных приложений Windows или "крах" системы не приводят к "зависанию" приложений реального времени.
- Работа с быстрыми часами и таймерами высокого разрешения.
- Прямой доступ к памяти и физическим устройствам. и т.д.

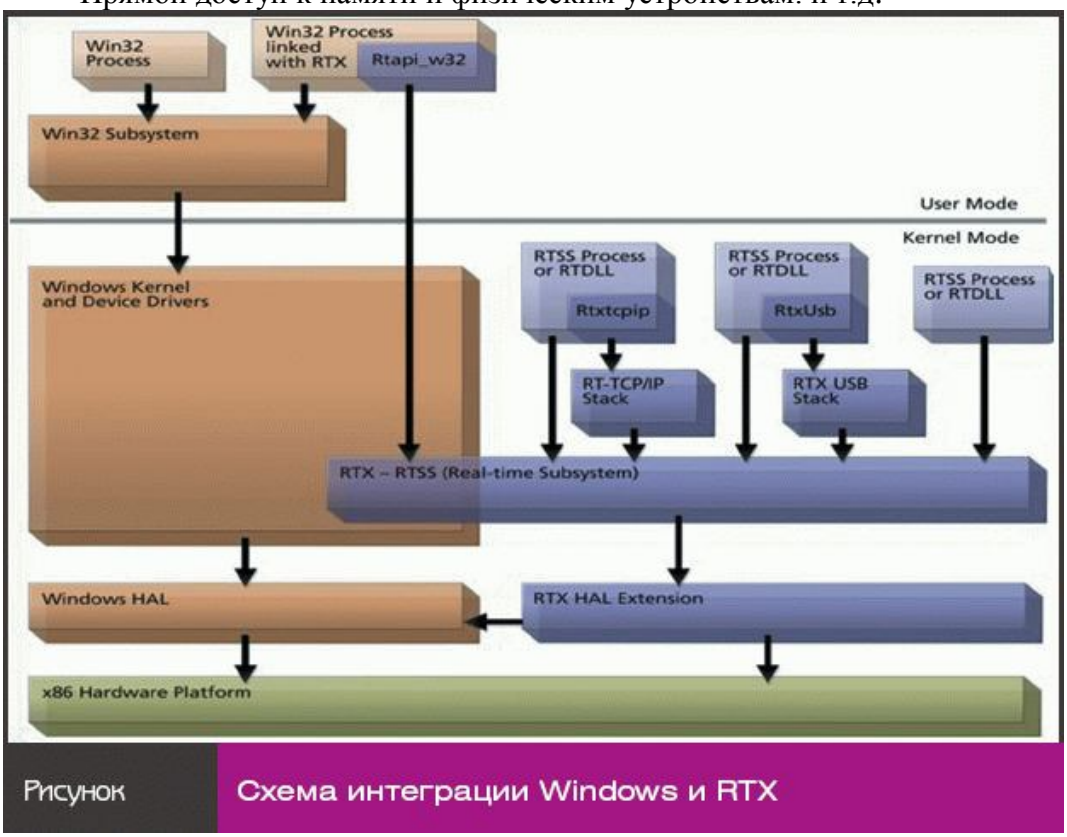

Рис. 3. Схема интеграции Windows и RTX

Продукт RTX состоит из пакета разработчика RTX SDK, предназначенного для создания собственных приложений для работы в среде RTX и подсистемы исполнения RTX Runtime, для обслуживания приложений RTX. Подсистема RTX Runtime устанавливается на целевые системы, где предполагается запуск RTX-приложений. Пакет разработчика интегрирован со средой разработки Visual Studio от компании Microsoft и также включает в себя подсистему реального времени, поэтому разрабатываемые приложения RTX можно запускать и отлаживать непосредственно на локальном месте.

Подсистема, реализующая функции системы реального времени, работает «рядом» с Windows. Обычные задачи Windows выполняются в недетерминированной среде, как и раньше. Один из вариантов реализации подобного механизма — разделение ресурсов между Windows и подсистемой реального времени. Такая система очень хорошо работает в много-

ядерных архитектурах, где под операционную систему Windows и под систему реального времени отводятся отдельные ядра.

RTX тесно интегрируется с Windows (рис. 2), при этом RTX дополняет стандартный HAL Windows своим расширением, RTX HAL Extension. На этом уровне, кроме организации доступа к аппаратуре, обрабатываются прерывания от таймера подсистемы реального времени. Непосредственно функциональность реального времени реализует слой RTSS, Real-Time SubSystem. Это ядро всей подсистемы реального времени. Здесь находится свой планировщик, который оперирует выполнением задач реального времени и предоставлением ресурсов задачам Windows-среды. Фактически любая задача RTSS имеет более высокий приоритет, нежели любая задача, выполняющаяся в Windows. Также этот слой полностью реализует API реального времени, Real-Time API — RTAPI, на основе которого создаются приложения подсистемы RTSS. Приложения реального времени, RTSS Process, выполняются на уровне ядра Windows и имеют те же привилегии и ограничения, что и драйверы устройств.

RTKernel это расширение реального времени для MS-DOS, работающий как мощный планировщик многозадачного режима. RTKernel была создана для разработчиков программного обеспечения, желающих осуществлять профессиональный процесс управления приложениями на компьютерах с DOS или встроенных системах. Особые внимание было уделено удобному использованию и отслеживанию жестких временных характеристик исполнения. ОСРВ очень компактно (около 16k код, 6k данных) и обеспечивает программистов основными инструментами, необходимыми для разработки эффективных приложений реального времени. RTKernel предлагает библиотеки, которые могут быть использованы в прикладных программах и ряд функций и процедур для управления задачами, семафоры, почтовые ящики, прерывания и т. д. Все RTKernel-задачи выполняются в рамках одной программы. RTKernel-приложение состоит из одного EXE-файл, содержащий ядро и необходимых драйверов. EXE-файл может быть выполнен на любом компьютере под управлением DOS. Хотя применение RTKernel позволяет использовать возможности многозадачности в приложениях реального времени. Система обладает следующими возможностями:

- выполнение неограниченного количества задач одновременно;
- под задачу достаточно отвести 500 байт;
- целевое переключение выполняется между задачами с учетом их приоритетов (всего определено 64 приоритета) приблизительно через 6 микросекунд для 486 процессоров с частотой 33 МГц;
- использует планировщик выполнения процессов по времени;
- обработка аппаратных прерываний и различное отрабатывание прерываний по скорости (от 0,1 до 55 мс);
- поддерживает до 36 последовательных портов и FIFO буфер в 16550;
- выполняется на любом компьютере.

Для разработчиков RTKernel-предоставляет возможности использования задач на C / C + + или функции и процедуры Паскаля. При инициализации программы, работает только одна основная программа, которая является главной задачей. Хотя программа содержит систему реального времени, он ведет себя как "нормальная", последовательная программа в процессе функционирования. Главная задача может запускать дополнительные задачи с использованием ядра API вызовов. Эти задачи в дальнейшем выполняться параллельно с основной задачей. Каждая задача имеет приоритет между 1 и 64 и свой собственный стек. Большее число указывает на больший приоритет. Все переменные, объявленные в данной процедуре задачи или функции, используют стек, выделенный для этой задачи. Все задачи могут получить доступ к глобальным данным без ограничений.

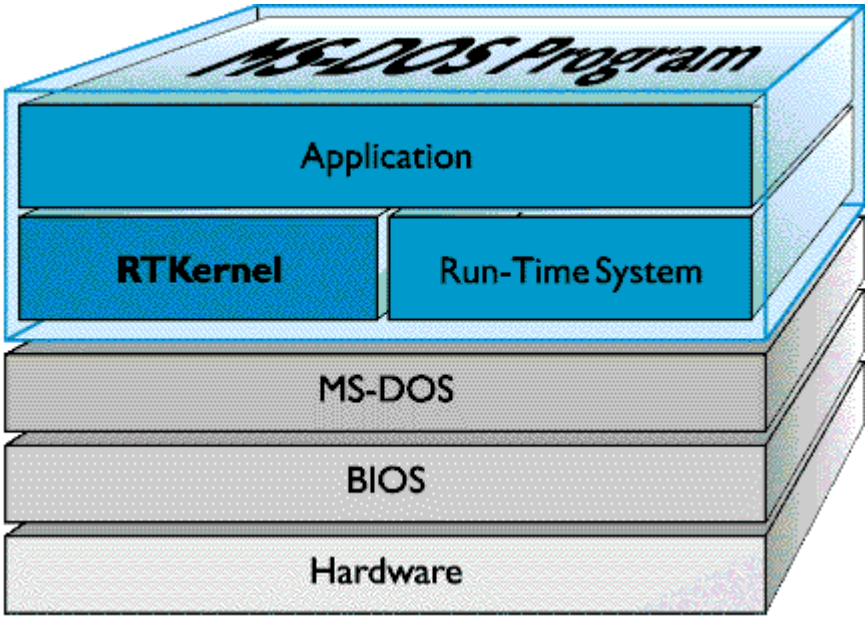

Рис. 4. RTKernel Структура программы

Головная станция системы управления должна обеспечивать выполнение таких функций как: начальная установка рабочих параметров при старте системы, сбор значений параметров работы асинхронного двигателя со всех модулей измерений (например, напряжение, температура и ток), оценивать полученные параметры на принадлежность к заданному диапазону, обрабатывать аварийные ситуации. В параллельном режиме должны выполняться программные интерфейсы взаимодействия с диспетчерами станции, как в непосредственном, так и в удаленном режиме, передавая необходимые данные через интернет.

Для выполнения задач сбора больших объемов данных и управления подходят ОСРВ Qnx и RTKernel, но организация графически удобного интерфейса и работа в сети хорошо реализуется в Qnx и RTX Windows. По результатам представленных возможностей более подходит для реализации перечисленных функций ОСРВ Qnx.

Литература

1. Защищённая операционная система реального времени QNX (изделие КПДА.00002-01). http://www.kpda.ru/Solutions/KPDA.
2. ОСи реального времени: QNX в вопросах. Дата: 08.04.2003. Хакер. http://www.xakep.ru/post/18109/
3. QNX. Материал из Википедии — свободной энциклопедии. http://ru.wikipedia.org/wiki/QNX
4. Операционная система реального времени. Материал из Википедии — свободной энциклопедии. http://ru.wikipedia.org/wiki/Операционная_система_реального_времени.
5. Зыль С. Операционная система реального времени QNX: от теории к практике. — 2-е изд. — СПб.: БХВ-Петербург, 2004. — 192 с.
6. Зыль С. QNX Momentics. Основы применения. — СПб.: БХВ-Петербург, 2004. — 256 с.
7. Дорогов А. Ю. Синхронизация и взаимодействие программных потоков в операционной среде реального времени: Учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2007. — 64 с.
8. А. Исаев, Л.Акиншин.Windows IT Pro : Windows изнутри. Система реального времени или Windows? — http://www.osp.ru/win2000/2009/12/13001334/
9. RTKernel - Real-Time Multitasking Kernel for DOS. — http://www.on-time.com/rtkernel-dos.htm