

УДК 621.9-114 + 004.383.4

Л. И. Мартинова, канд. техн. наук, доц.,

Г. М. Мартинов, д-р техн. наук, проф.,

МГТУ "Станкин"

book@ncsystems.ru

Организация межмодульного взаимодействия в распределенных системах ЧПУ. Модели и алгоритмы реализации*

Систематизированы требования к межмодульному обмену в системах ЧПУ с распределенной архитектурой. Обозначены места и определены функции коммуникационной среды. Исследованы модели проектов OSACA и OCEAN с применением стандартов COM, CORBA, MMS. Раскрыты некоторые аспекты создания математического обеспечения и выработаны рекомендации для построения распределенной системы ЧПУ с открытой модульной архитектурой.

Ключевые слова: ЧПУ, коммуникационная среда, COM, CORBA, MMS, распределенная архитектура, реальное время

Введение

Ключевая особенность современных распределенных систем числового программного управления — использование принципа открытой архитектуры, регламентирующего и стандартизирующего только описание принципа действия системы и ее конфигурации, что позволяет собирать ее из отдельных узлов и деталей, разработанных и изготовленных независимыми фирмами-производителями. В результате появляется возможность строить, модернизировать и расширять системы наиболее экономичным способом. Открытость архитектуры подразумевает, что любые новые протоколы передачи данных между системами и аппаратные решения базируются на общепринятых стандартах с опубликованными спецификациями, в то время как модули системы могут быть созданы на основе различных технических средств и могут функционировать на разных платформах.

Хотя мы обычно и имеем дело с закрытым математическим обеспечением для систем ЧПУ, но его организация поддается анализу, в связи с чем хотелось бы обратить внимание разработчиков на ряд интересных решений, которыми можно было бы

* Работа выполнена по Госконтракту № П1901 от 26 мая 2010 г. на проведение НИР в рамках ФЦП "Научные и научно-педагогические кадры инновационной России" на 2009—2013 годы.

воспользоваться в практике создания математического обеспечения ЧПУ. Другая цель заключается в информировании заинтересованной инженерной аудитории относительно реальной архитектуры систем ЧПУ, поступающих в нашу страну по импорту.

Модели открытой системы ЧПУ. Был предпринят ряд международных проектов *OMAC (Open Modular Architecture Controls)*, *OSACA (European Open System Architecture for Controls within Automation Systems)*, *OSEC (Japan Open System Environment for Controller Architecture)* и *IROFA (Japan International Robotics and Factory Automation)*, которые в различной степени затрагивали проблемы распределенного компьютерного управления технологическими системами и ставили целью создание модели открытой системы ЧПУ. Несмотря на то, что конечная цель не была достигнута, результаты проектов можно считать достаточно успешным хотя бы потому, что появились многочисленные производители, которые практически приняли важные технические наработки [1].

В наиболее общем виде модель системы ЧПУ может быть представлена как многоуровневая виртуальная машина (рис. 1, см. третью сторону обложки).

В этой модели выделены:

- *уровень терминала* (интерфейс оператора, приложения) с машинным масштабом времени работы всех его компонентов;
- *уровень задач управления*, работающих в реальном времени;
- *уровень объектов управления на станке* (следящие приводы, электроавтоматика).

Каждому уровню такой машины могут быть сопоставлены свои модели. Достаточно полный комплекс таких моделей представлен на рис. 2.

В отличие от традиционного представления [2] виртуальной машины здесь разделены платформа и прикладная часть математического обеспечения. В прикладной части обозначены модели пользователя, модели приложений и модели услуг. Следует заметить, что международные проекты были нацелены именно на создание пополняемого резерва моделей прикладной части математического обеспечения.

Модели проекта OSACA. В архитектуре проекта *OSACA (Open Systems Architecture Controls within Automation Systems)*, архитектура открытых систем управления для автоматизации) был сделан акцент на разработке коммуникационной среды, поддерживающей взаимодействие прикладных модулей между собой и с системной платформой [3, 4]. Концепция проекта базировалась на положении, в соот-

ветствии с которым стандартизации подлежит лишь внешнее поведение прикладных модулей, поскольку заложенное в них технологическое "know-how" должно быть защищено, а разработчикам математического обеспечения должна быть предоставлена свобода программирования модулей в их собственной манере. Поэтому усилия авторов проекта были сосредоточены вокруг тщательного специфицирования прикладных интерфейсов *API (Application Programming Interface)* на стыке приложений и их инфраструктуры. Возможность использования практически любой платформы позволяет *OSACA*-приложениям выстраивать систему на базе одно- или двухкомпьютерного решения (рис. 3).

В системную платформу в *OSACA*-архитектуре интегрированы операционная система, коммуникационная среда и средства конфигурации, которые используются для построения топологии математического обеспечения из доступных модулей в целях достижения заданной функциональности. Доступ к системной платформе осуществляется через *API*. Интерфейс *API* должен допустить использование переносимых модулей, в том числе и от разных разработчиков.

В силу объектно-ориентированного построения математического обеспечения прикладные модули получили наименование архитектурных объектов *AO (Architectural Objects)*. Единственным средством информационного обмена между архитектурными объектами как в пределах одной вычислительной среды, так и за ее пределами в распределенной системе служит коммуникационная среда. Стандартные протоколы коммуникационной среды обеспечивают единообразные форматы данных и фиксированные наборы сообщений [5]. Система протоколов производна по отношению к базовой архитектуре открытых систем *OSI (Open Systems Interconnection)*. Она представлена двумя крупными уровнями (рис. 4, см. третью сторону обложки):

- уровнем транспорта сообщений *MTS (Message Transport System)*, эквивалентным четырем нижним слоям 1–4 архитектуры *OSI*;

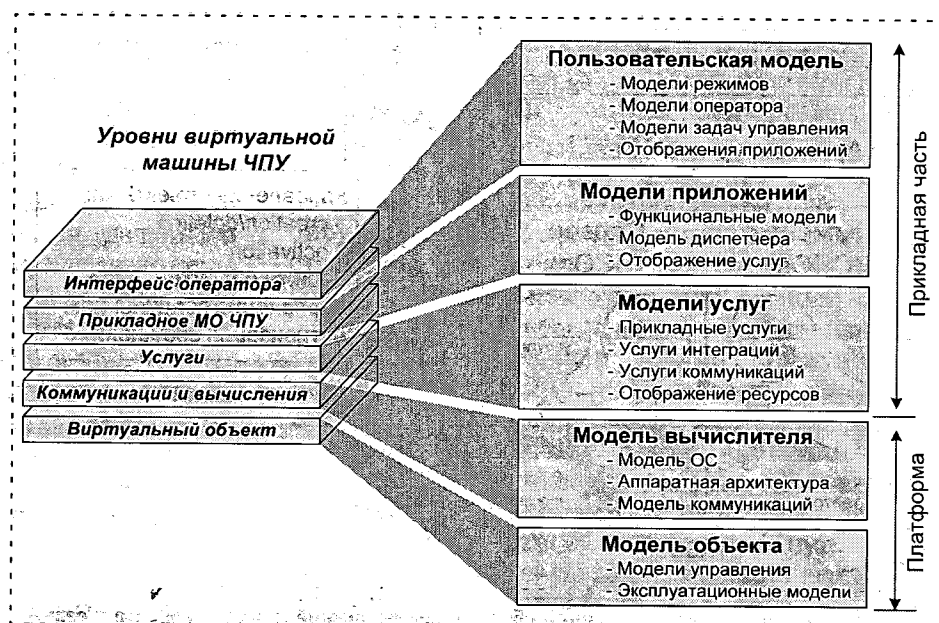


Рис. 2. Иерархический комплекс моделей системы ЧПУ

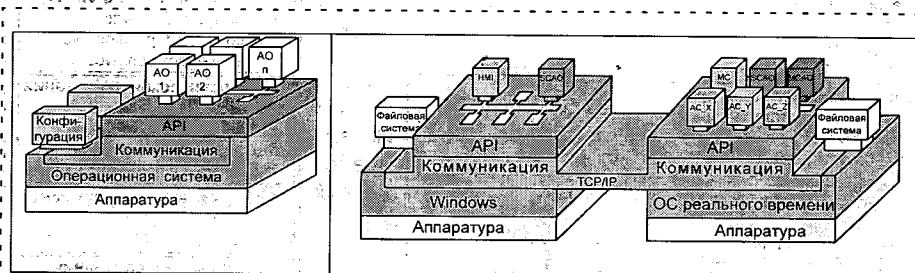


Рис. 3. Архитектура *OSACA*:

AO — *Architectural Object*, архитектурный объект; *HMI* — *Human Machine Interface*, интерфейс оператора; *MCAO* — *Master configuration AO*, ведущий конфигурационный архитектурный объект; *SCAO* — *Slave Configuration AO*, ведомый конфигурационный архитектурный объект

- уровнем прикладного сервиса *ASS (Application Services System)*, эквивалентным трем верхним слоям 5–7 архитектуры *OSI*.

Уровень *MTS* предлагает сервис транспорта произвольных сообщений, с предварительной установкой соединения; между архитектурными объектами *AO*. Этот уровень может быть адаптирован к любым существующим механизмам обмена информацией (рис. 5, см. третью сторону обложки).

Уровень *ASS* имеет дело с прикладным протоколом, выстроенным на основе клиент-серверных отношений с использованием объектно-ориентированного подхода. В серверном архитектурном объекте любая информация, данные или услуги, которые доступны извне, привязаны к коммуникационному объекту *CO (Communication Object)*. С точки зрения клиента, сервер представляет собой набор коммуникационных объектов, доступных для передачи и получения сообщений с помощью услуг уровня *ASS* (рис. 6). Архитектурные объекты могут совмещать функции клиента и сервера.

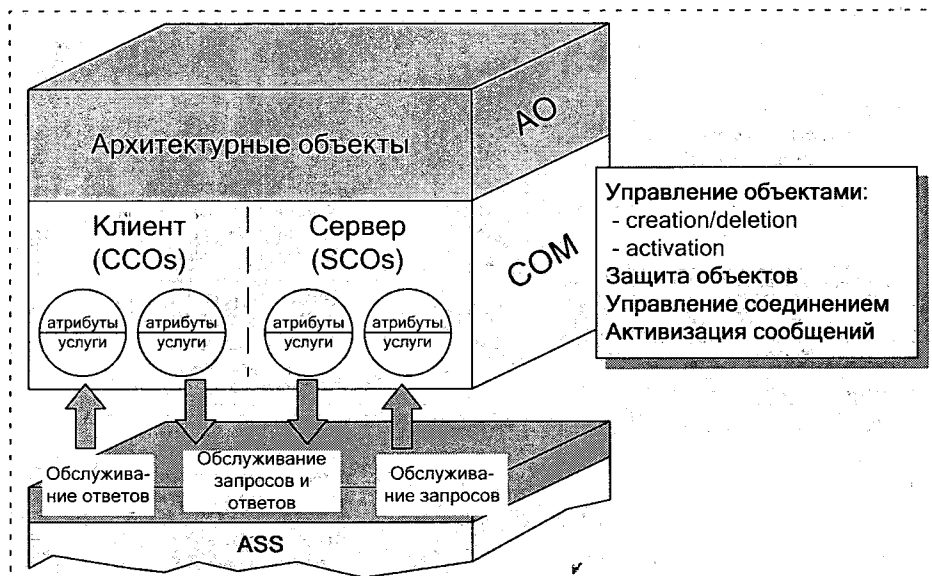


Рис. 6. Интерфейс коммуникационного объекта, совмещающий функции клиента и сервера

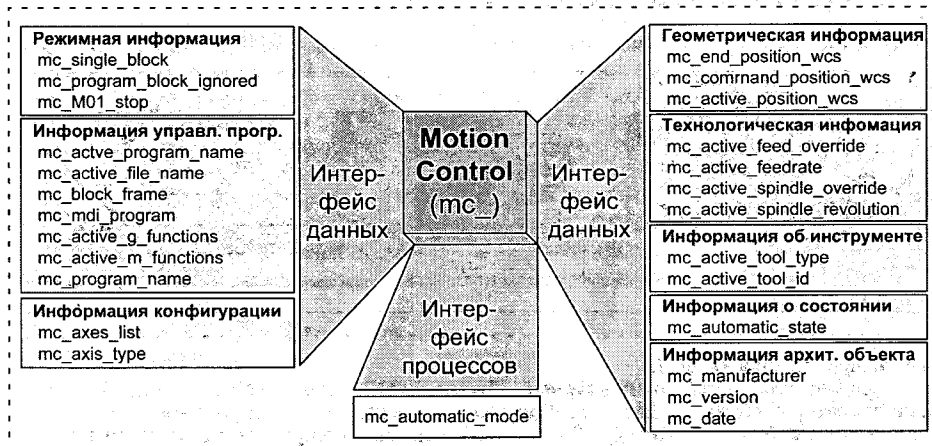


Рис. 7. Диаграмма коммуникационных объектов в интерфейсе ядра системы ЧПУ

Существует фиксированный набор классов коммуникационных объектов; наиболее важными из которых являются класс переменной (*variable*) для записи и чтения данных и класс процесса (*process*) для переключения состояний в конечных автоматах. Дополнительный класс события (*event*) служит для передачи инициативных событий и сообщений. Эти коммуникационные объекты создаются в каждом архитектурном объекте и регистрируются в менеджере коммуникационных объектов *COM* (*Communication Object Manager*); который взаимодействует с уровнем *ASS*. Один серверный коммуникационный объект *SCO* одновременно доступен различным клиентским коммуникационным объектам *CCO*.

Разделение транспортных и прикладных услуг позволяет разработчику математического обеспечения ЧПУ полностью сосредоточиться на решении специфических задач управления.

Выделяют пять типов архитектурных объектов, соответствующих основным задачам управления:

- интерфейс оператора, или терминальная задача, *MMC* (*Man Machine Control*);
- управление автоматикой, или логическая задача, *LC* (*Logic Control*);
- ядро ЧПУ, или геометрическая задача, *MC* (*Motion Control*);
- управление следящими приводами *AC* (*Axis Control*);
- управление рабочим процессом, или технологическая задача, *PC* (*Process Control*).

На рис. 7 представлен структурированный набор коммуникационных объектов, определенных для архитектурного объекта "Ядро ЧПУ".

Коммуникационная среда с использованием архитектуры *CORBA*. В сложных распределенных программных системах управления достаточно широкое распространение получила "архитектура брокеров объектных запросов" *CORBA* (*Common Object Request Broker Architecture*) [6].

Для доставки сообщения от клиента к серверу и получения ответных результатов необходим системный компонент, отвечающий за выполнение подобных функций. В архитектуре *CORBA* такой компонент называется брокером объектных запросов *ORB*

(*Object Request Broker*), в функции которого, в том числе, входит передача данных в машинно-независимом формате от клиента серверу и от сервера клиенту. Кроме того, *ORB* отвечает за правильное указание сетевого адреса объекта-сервера. В архитектуре *CORBA* каждой объектной реализации сопоставлена уникальная объектная ссылка *object reference*, которая используется клиентом для указания брокеру *ORB*. Простейшие схемы взаимодействия брокеров *ORB* (*Object Request Broker*) показаны на рис. 8.

Важнейшими понятиями являются *Skeleton*, *Stub* и язык описания интерфейсов *IDL* (*Interface Definition Language*), который определяет интерфейсы объектов независимо от способа и языка реализации самого объекта. При разработке конкретных приложений на базе *ORB* *IDL*-описания интерфейсов используемых объектов транслируются в наборы функций доступа к *ORB*, которые затем связываются с исполняемым модулем. *Stub* (заглушка) —

набор функций на языке программы-клиента, который сгенерирован из IDL-описания интерфейса объекта-клиента; *Skeleton* (заготовка) — набор функций, который сгенерирован на основе IDL-описания интерфейса объекта-сервера, используется ORB для вызова метода объекта-сервера, запрошенного объектом-клиентом. На стороне клиента объект-*stub* играет роль локального представителя удаленного объекта. В коде, генерируемом в *stub*, заложены знания о том, что требуется сделать для обращения к методу удаленного объекта. На стороне сервера объект-*skeleton* служит посредником при доступе к удаленному объекту по правилам его системы программирования, при этом *skeleton* распознает удаленность обращения. Объектная реализация (*Servant*) выполняет операции, сформулированные в IDL-интерфейсе; *servant* может представлять один или несколько объектов.

Взаимодействие брокеров ORB архитектуры CORBA поддерживается "универсальным межброкерным протоколом" GIOP (*General Inter-ORB Protocol*). Универсальность протокола состоит в том, что он не зависит от конкретной сетевой транспортной среды и может быть отображен на любой транспортный протокол, поддерживающий виртуальные соединения. Так, отображение GIOP на протокол TCP/IP называют "межброкерным Internet-протоколом" IIOP (*Internet Inter-ORB Protocol*). Назначение протоколов GIOP/IIOP заключается в том, чтобы поддержать сети брокеров в рамках или вне Internet. Клиент-серверная структура взаимодействия брокеров ORB в реальном времени показана на рис. 9.

Адаптивная коммуникационная среда ACE (*Adaptive Communication Environment*) представляет собой объектно-ориентированную структуру с открытыми кодами, которая располагает коммуникационными шаблонами [7]. Шаблоны и компоненты, разработанные в среде ACE, привели к созданию коммуникационной среды для брокеров ORB, получившей наименование TAO (*The ACE ORB*). Система TAO представляет собой "промежуточное" (*middleware*) математическое обеспечение для CORBA, которое

позволяет клиентам вызывать операции в распределенных объектах, не заботясь об их расположении, языке программирования, операционной среде, коммуникационных протоколах, внутренних связях и аппаратной поддержке [8].

Компоненты TAO структурированы и оптимизированы для повышения быстродействия и масштабируемости [9]. Коммуникационная среда представляет собой многоуровневую среду, траектории сообщений через которую могут различаться при "collocated"- и "uncollocated"-взаимодействии.

Среда TAO предлагает два типа коммуникации между компонентами: для "collocated"-компонентов, размещенных в едином адресном пространстве; для "uncollocated"-компонентов (локальных или удаленных), размещенных в разных адресных пространствах. В первом случае существует возможность сократить коммуникационный путь за счет обхода уровней маршallingа, маршрутизации, демультимплексирования и др., что, конечно же, существенно повышает быстродействие. Во втором

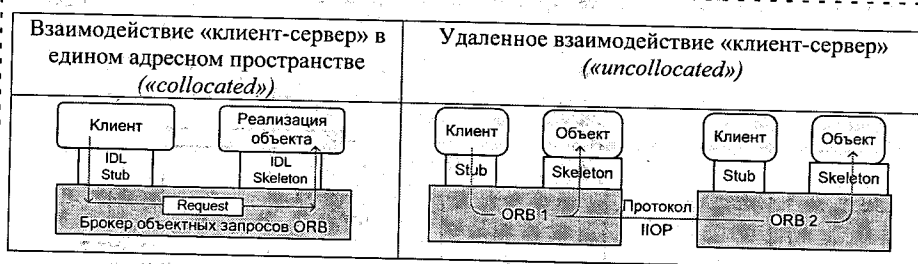


Рис. 8. Простейшие схемы взаимодействия брокеров ORB

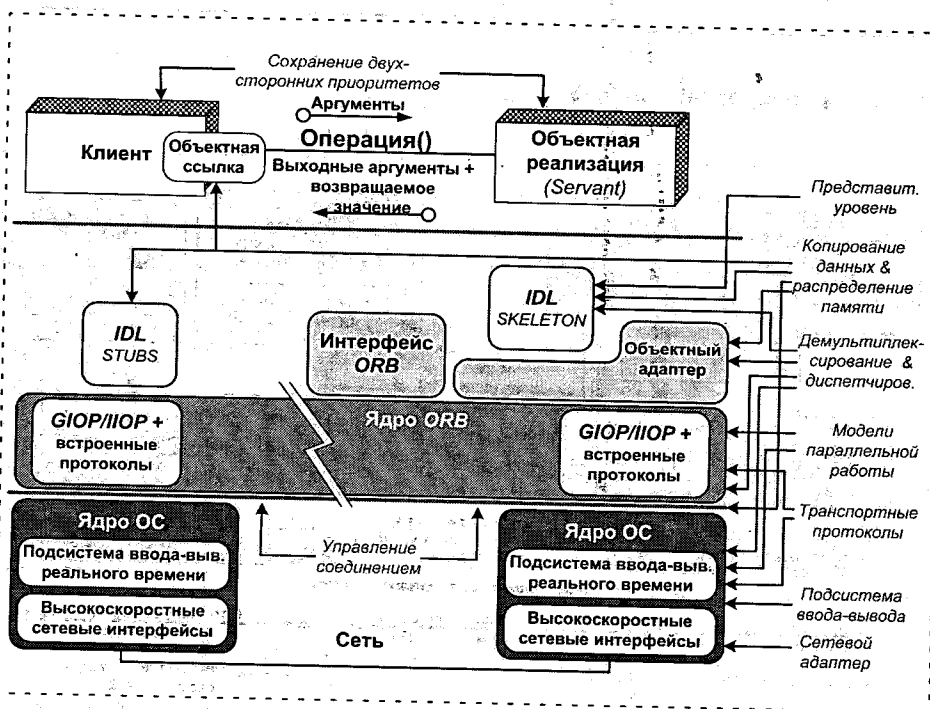


Рис. 9. Взаимодействие брокеров в реальном времени

случае можно использовать другую возможность повышения быстродействия на основе модификации сетевой подсистемы *Linux* под названием *RTnet*. *RTnet* предоставляет собой сеть жесткого реального времени поверх стандартного протокола *IP* (сеть разработана компанией *Embedix Inc.*; [10], известной как *Lineo*). Сеть работает с ядром *Linux* с расширением *RTLlinux*; с использованием прикладного интерфейса реального времени *RTAI (Real Time Application Interface)*.

В системах управления пространства задач реального времени и пользовательских приложений обычно обособлены и требуют межпространственной коммуникации с помощью быстрого и надежного транспорта. Для этого разработан встроенный *TAO*-протокол с применением разделяемой памяти *SHMIOP (Shared Memory Inter-ORB Protocol)*, использующий *RTAI* [11].

Объединение моделей *CORBA* и *MMS*. Интересное решение представлено в [12], где предлагается объединить *CORBA* с объектно-ориентированным представлением *MMS (ISO 1990)*.

Спецификация производственных сообщений *MMS (Manufacturing Message Specification)* представ-

ляет собой прикладной уровень протокольного стека *ISO/OSI*. Функции уровня позволяют организовать удаленное управление и мониторинг для таких компьютеризованных объектов, как станки, роботы и другие автоматические устройства. С позиций *MMS* компьютеризованные объекты представляют собой виртуальные производственные устройства *VMD (Virtual Manufacturing Devices)*, которые оказывают услуги удаленным пользователям (*Users*) или клиентам. Коммуникация выстроена на базе интерактивных транзакций (*Association*) и реализуется посредством обменных сообщений *PDU (Protocol Data Units)* стандартного формата. Коммуникация по большей части носит характер подтверждаемых услуг (*positive, negative, error*), запрошенных удаленным пользователем. Существует также небольшой набор неподтверждаемых услуг в виде сообщений сервера (например, уведомление об изменении статуса *VMD*).

Виртуальное устройство *VMD* служит абстракцией реальной удаленной системы. Это устройство представляет собой объект, который инкапсулирует атрибуты (*identity, status, ...*), а также и методы (*services, услуги*). Для *VMD* разработан некоторый набор классов объектов, наиболее известные из которых:

- домены (*Domains*), представляющие собой загружаемые наборы ресурсов пользователя (данные и код);
- программные вызовы (*Program Invocations*), представляющие собой исполняемые программы (подпрограммы) пользователя;
- переменные и наборы переменных (*Variables, Sets of variables*), принадлежащие *VMD* или домену;
- объекты-события (*Event related objects*).

Шина *CORBA* вполне может быть приспособлена для коммуникации между архитектурными объектами *AO*. Для этого потребуется служба межобъектной коммуникации *ORB*, работающая в реальном времени.

В качестве примера на рис. 10 представлена распределенная система ЧПУ на базе персонального компьютера. Здесь используются следующие обозначения: *HMI (Human Machine Interface)* — интерфейс оператора; *MCM (Motion Control Manager)* — диспетчер каналов; *MC (Motion Controls)* —

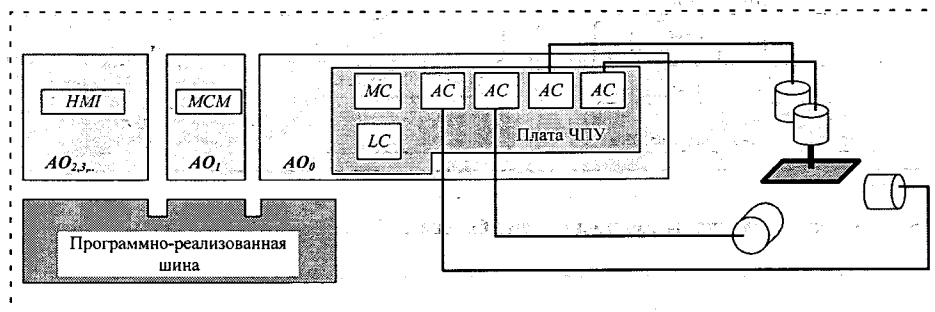


Рис. 10. Архитектура системы ЧПУ

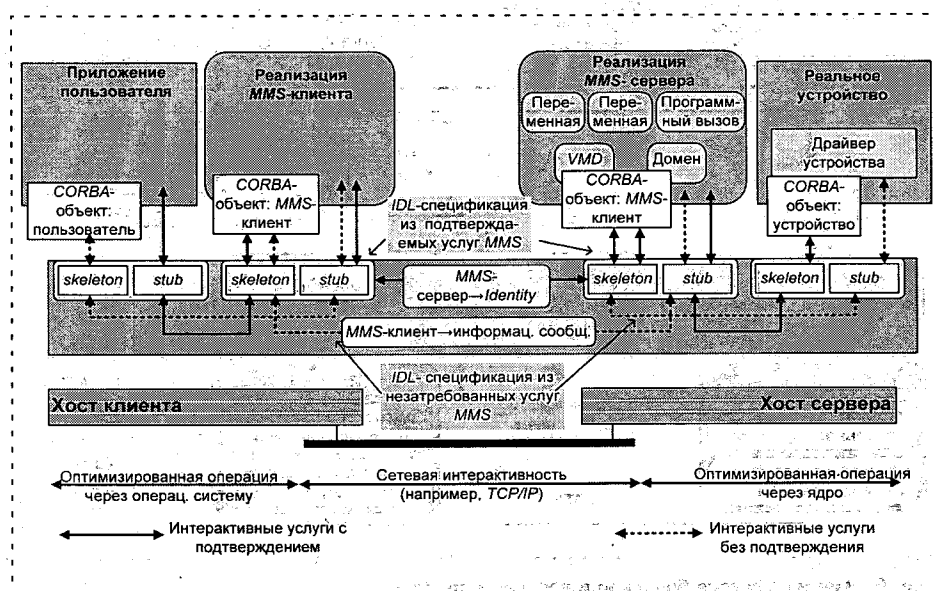


Рис. 11. Принципы наложения протоколов *MMS* на шину *CORBA*

ядро ЧПУ (интерпретатор-интерполятор); *AC* (*Axis Controls*) — контроллер осевого перемещения; *SC* (*Spindle Controls*) — контроллер шпинделя; *LC* (*Logic Controller*) — контроллер автоматики; *AO* (*Architectural Object*) — модуль системы. Коммуникация между удаленным терминалом и локальной системой ЧПУ поддержана соответственно стандарту *MMS*. Архитектурные объекты *AO* дополнительно декомпозированы на функциональные блоки: интерфейс оператора *HMI*; диспетчер каналов *MCM*; ядро ЧПУ (интерпретатор-интерполятор) *MC*; контроллер осевого перемещения *AC*; контроллер шпинделя *SC*; контроллер автоматики *LC*.

На рис. 11 показана структура наложения протоколов *MMS* на шину *CORBA*. Такое решение позволяет отделить терминальную часть системы ЧПУ от модулей реального времени в распределенной системе управления, объединяющей различные системы ЧПУ между собой в единой производственной среде. Возникает возможность использовать несколько терминалов с одним удаленным модулем реального времени или использовать один терминал с несколькими модулями реального времени.

Заключение. Модели, представленные в проектах *OSACA* и *OCEAN* в стандартах *CORBA* и *MMS*, дают достаточно материала для разработки эффективного математического обеспечения открытой системы ЧПУ. Они в особенности полезны при создании системы ЧПУ "с нуля", когда груз предыдущих

решений не оказывает давления на разработчиков. Эти модели полезны и в методическом плане, поскольку трудно представить что-нибудь более наглядное для объяснения принципов работы современной системы ЧПУ в целом.

Список литературы

1. Григорьев С. Н., Мартинов Г. М. Перспективы развития распределенных гетерогенных систем ЧПУ децентрализованными производствами // Автоматизация в промышленности. 2010. № 5. С. 4–8.
2. Соколкин В. Л., Мартинов Г. М. Системы числового программного управления: учеб. пособие. М.: Логос, 2005. 296 с.
3. Sperling W., Lutz P. Designing applications for an OSACA control // Proc. of the Intern. Mechanical Engineering Congress and Exposition (The ASME Winter Annual Meeting). Dulles/USA, November 16–21, 1997. 5 p.
4. www.osaca.org/documentation_and_software/demo_software.htm
5. Мартинов Г. М. Открытая система ЧПУ на базе общей магистральной // Автомобильная промышленность. 1997. № 4. С. 31–34.
6. www.omg.org/technology/documents/spec_catalog.htm
7. www.theaceorb.com/product/abouttao.html
8. Schmidt D. C., Deshpande M., O’Ryan C. Operating System Performance in Support of Real-time Middleware // Proc. of the 7th IEEE Workshop on Object-oriented Real-time Dependable Systems. San Diego, CA, January, 2002. P. 1–9.
9. www.theaceorb.com/downloads/index.html
10. www.embedix.com/
11. www.aero.polimi.it/~rtai/
12. Boissier R., Gressier-Soudan E., Laurent A., Seinturier L. Enhancing numerical controllers, using MMS concepts and a CORBA-based software bus. // International Journal of Computer Integrated Manufacturing, Publisher Taylor & Francis. 2001. Vol. 14. № 6 (November). P. 560–569.

УДК 681.768.2.5

И. И. Дунин-Барковский, канд. техн. наук, доц.,
Московский государственный
технологический университет "СТАНКИН"
igordbar@hotmail.com

Обработка информации, изображений и управление в мехатронной системе с использованием универсального контроллера на основе FPGA

Рассматривается подход к построению универсального контроллера для обработки информации в мехатронных системах оптического контроля на основе устройств программируемой логики высокой степени интеграции FPGA, позволяющий реализовать обработку информации с очень высоким темпом и высокой степенью параллелизма, а также, в случае необходимости, эффективно совместить аппаратную обработку с программной.

Ключевые слова: мехатронная система, FPGA, программируемая логика, обработка изображений, трехмерные измерения

Введение

Прогресс в современных устройствах программируемой логики высокой степени интеграции FPGA позволяет строить на их основе сложные многофункциональные устройства, в частности, устройства, способные управлять сложными мехатронными системами и обрабатывать необходимую для этого информацию в реальном масштабе времени. Потребность в таких устройствах ощущается все сильнее, особенно в современных технологиях электронного производства [1, 5].

Функции, которые должны быть возложены на контроллер, представлены на рис. 1. Некоторые из них, такие как управление движением, ввод-вывод управляющих сигналов и управление последовательностью операций рабочего цикла системы, относятся к управлению механической подсистемы мехатронной системы. Функции ввода и обработки изображений, а также управления освещением являются частью активной системы технического зрения, являющейся сенсорной подсистемой мехатронной системы.