9.2. OPC сервер

Стандарт OPC разработан международной организацией OPC Foundation, членами которой являются более 400 фирм, работающих в области средств автоматизации и измерительной техники. Основателями организации являются фирмы Fisher-Rosemount, Rockwell Software, Opto 22, Intellution и Intuitive Technology. Первая версия OPC стандарта была выпущена в 1998 г. [OLE]. В совет директоров OPC Foundation в 2008 году входили представители Siemens AG, Emerson Process Management, Yokogawa, Honeywell, Rockwell Automation, ICONICS.

9.2.1. Обзор стандарта OPC

Главной целью стандарта OPC явилось обеспечение возможности совместной работы (интероперабельности) средств автоматизации, функционирующих на разных аппаратных платформах, в разных промышленных сетях и производимых разными фирмами. До разработки OPC стандарта SCADA пакет нужно было адаптировать к каждому новому оборудованию индивидуально. Существовали длинные списки "поддерживаемого оборудования", очень сложной была техническая поддержка. При модификации оборудования нужно было вносить изменения во все драйверы, каждый из которых поддерживал протокол обмена только с одной клиентской программой. Число таких драйверов доходило до сотен.

После появления стандарта OPC практически все SCADA-пакеты были перепроектированы как OPC-клиенты, а каждый производитель аппаратного обеспечения стал снабжать свои контроллеры, модули ввода-вывода, интеллектуальные датчики и исполнительные устройства стандартным OPC сервером. Благодаря появлению стандартизации интерфейса стало возможным подключение любого физического устройства к любой SCADA, если они оба соответствовали стандарту OPC. Разработчики получили возможность проектировать только один драйвер для всех SCADA-пакетов, а пользователи получили возможность выбора оборудования и программ без прежних ограничений на их совместимость.

Стандарт OPC относится только к интерфейсам, которые OPC сервер предоставляет клиентским программам. Метод же взаимодействия сервера с аппаратурой (например, с модулями ввода-вывода), стандартом не предусмотрен и его реализация возлагается полностью на разработчика аппаратуры. Поэтому стандарт OPC может быть использован не только для взаимодействия SCADA с "железом", но и для обмен данными с любым источником данных, например, с базой данных или с GPS приемником.

OPC сервер как средство взаимодействия с техническим устройством может быть использован при разработке заказных программ на C++, Visual Basic, VBA и т. п. В этих задачах OPC сервер используется как Microsoft DCOM объект, от которого он отличается только стандартизацией обозначений и специфическими терминами из области промышленной автоматизации. Применение OPC сервера при разработке заказных программ позволяет скрыть от разработчика

всю сложность общения с аппаратурой, представляя простой и удобный метод доступа к аппаратуре через интерфейсы COM-объекта.

Стандарт OPC состоит из нескольких частей:

- OPC DA (OPC Data Access) - спецификация для обмена данными между клиентом (например SCADA) и аппаратурой (контроллерами, модулями ввода-ввода и др.) в реальном времени;
- **OPC Alarms & Events** (A&E) - спецификация для уведомления клиента о событиях и сигналах тревоги, которые посылаются клиенту по мере их возникновения. Этот сервер пересылает аварийные сигналы, действия оператора, информационные сообщения, результаты контроля состояния системы;
- OPC HDA (Historical Data Access) - спецификация для доступа к предыстории процесса (к сохраненным в архиве данным). Сервер обеспечивает унифицированный способ доступа с помощью DCOM технологии. Обеспечивает чтение, запись и изменение данных;
- **Batch** - спецификация для особых физико-химических технологических процессов обработки материалов, которые не являются непрерывными. В таких процессах выполняется загрузка нескольких видов сырья в определенных пропорциях согласно рецепту, устанавливаются режимы обработки, а после выполнения цикла обработки и выгрузки готового материала загружается новая партия сырья. OPC сервер выполняет обмен между клиентом и сервером рецептами, характеристиками технологического оборудования, условиями и результатами обработки;
- **OPC Data eXchange** - спецификация для обмена данными между двумя OPC DA серверами через сеть Ethernet;
- **OPC Security** - спецификация, которая определяет методы доступа клиентов к серверу, которые обеспечивают защиту важной информации от несанкционированной модификации;
- **OPC XML-DA** - набор гибких, согласующихся друг с другом правил и форматов для представления первичных данных с помощью языка XML, веб технологий и сообщений SOAP (см. раздел "Архитектура автоматизированной системы".);
- **OPC Complex Data** - дополнительные спецификации к OPC DA и XML-DA, которые позволяют серверам работать со сложными типами данных, такими как бинарные структуры и XML-документы;
- **OPC Commands** - набор программных интерфейсов, который позволяет OPC клиентам и серверам идентифицировать, посылать и контролировать команды, исполняемые в техническом устройстве (в контроллере, модуле ввода-вывода);
- **OPC Unified Architecture** - принципиально новый набор спецификаций, который уже не базируется на DCOM технологии, подробнее см. раздел "Спецификация OPC UA".

Из перечисленных спецификаций в России широко используются только две: OPC DA и реже - OPC HDA.

9.2.2. OPC DA сервер

Сервер OPC DA является наиболее широко используемым в промышленной автоматизации. Он обеспечивает обмен данными (запись и чтение) между клиентской программой и физическими устройствами. Данные состоят из трех полей: значение, качество и временная метка. Параметр качества данных позволяет передать от устройства клиентской программе информацию о выходе

измеряемой величины за границы динамического диапазона, об отсутствии данных, ошибке связи и другие.

Существует четыре стандартных режима чтения данных из OPC сервера:

- **синхронный режим**: клиент посылает запрос серверу и ждет от него ответ;
- **асинхронный режим**: клиент отправляет запрос и сразу же переходит к выполнению других задач. Сервер после выполнения функции запроса посылает клиенту уведомление и тот забирает предоставленные данные;
- **режим подписки**: клиент сообщает серверу список тегов, значения которых сервер должен отправлять клиенту только в случае их изменения. Для того, чтобы шум данных не был принят за их изменение, вводится понятие "мертвой зоны", которая слегка превышает максимально возможный размах помехи;
- **режим обновления данных**: клиент вызывает одновременное чтение всех активных тегов. Активными называются все теги, кроме обозначенных как "пассивные". Такое деление тегов уменьшает загрузку процессора обновлением данных, принимаемых из физического устройства.

В каждом из этих режимов данные могут читаться либо из кэша OPC сервера, либо непосредственно из физического устройства. Чтение из кэша выполняется гораздо быстрее, но данные к моменту чтения могут устареть. Поэтому сервер должен периодически освежать данные с максимально возможной частотой. Для уменьшения загрузки процессора используют параметр частоты обновления, которая может быть установлена для каждой группы тегов индивидуально. Кроме того, некоторые теги можно сделать пассивными, тогда их значения не будут обновляться данными из физического устройства.

Запись данных в физическое устройство может быть выполнена только двумя методами: синхронным и асинхронным и выполняется сразу в устройство, без промежуточной буферизации. В синхронном режиме функция записи выполняется до тех пор, пока из физического устройства не поступит подтверждение, что запись выполнена. Этот процесс может занимать много времени, в течение которого клиент находится в состоянии ожидания завершения функции и не может продолжать выполнение своей работы. При асинхронной записи клиент отправляет данные серверу и сразу продолжает свою работу. После окончания записи сервер отправляет клиенту соответствующее уведомление.

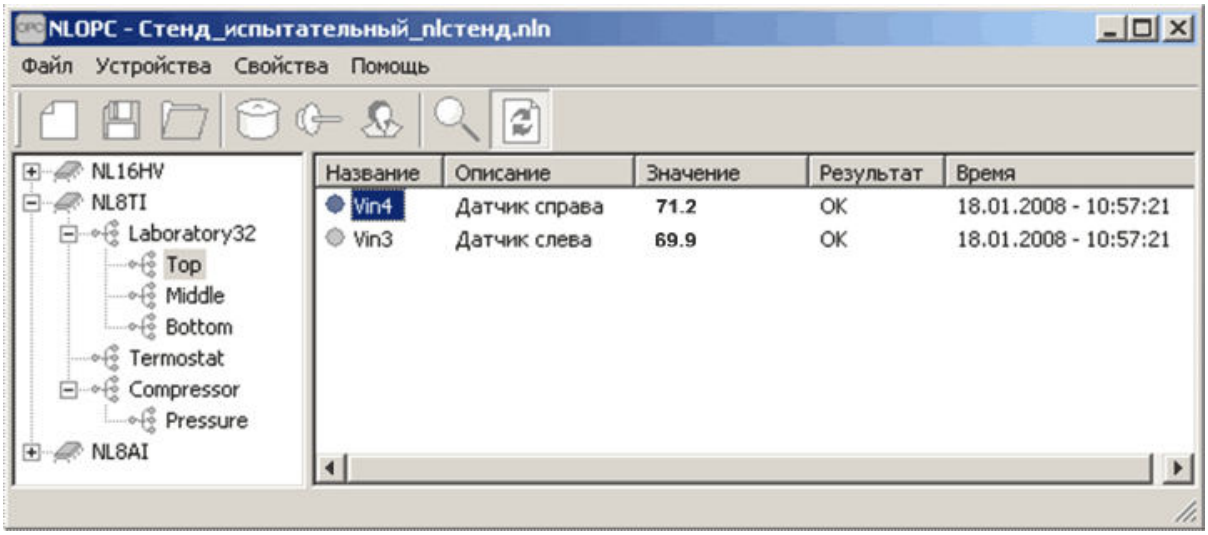

Рис. 9.1. Пример диалогового окна OPC сервера NLopc фирмы RealLab!

OPC DA сервер может иметь (не обязательно) пользовательский интерфейс, который позволяет выполнять любые вспомогательные функции для облегчения работы с оборудованием. Например, OPC сервер NLopc фирмы Reallab! позволяет, помимо обмена данными со SCADA, выполнять следующие полезные функции:

- поиск подключенного к промышленной сети оборудования;
- установку параметров оборудования (имени, адреса, скорости обмена данными, периода сторожевого таймера, наличие контрольной суммы и др.);
- создавать иерархическое представление имен тегов;
- наблюдать значения тегов;
- управлять правами доступа к OPC серверу.

В соответствии со стандартом, OPC сервер во время инсталляции автоматически регистрируется в реестре Windows. Запуск сервера осуществляется так же, как любой другой программы или автоматически из клиентской программы.

На рис. 9.1 показано диалоговое окно OPC сервера NLopc. Сервер позволяет выполнить поиск физических устройств, подключенных к COM-порту компьютера. На рис. 9.1 окно сервера слева показывает, что к компьютеру подключены три модуля ввода: NL16HV, NL8TI и NL8AI. Для удобства представления измеряемых величин (тегов) на объекте автоматизации имена тегов могут быть составными и путь к тегу может быть представлен в виде дерева, как показано на рис. 9.1. Имя выделенного на рисунке тега выглядит как "NL8TI.Laboratory32.Top.Vin4". Все имена и их структура задаются с помощью средств окна OPC сервера.

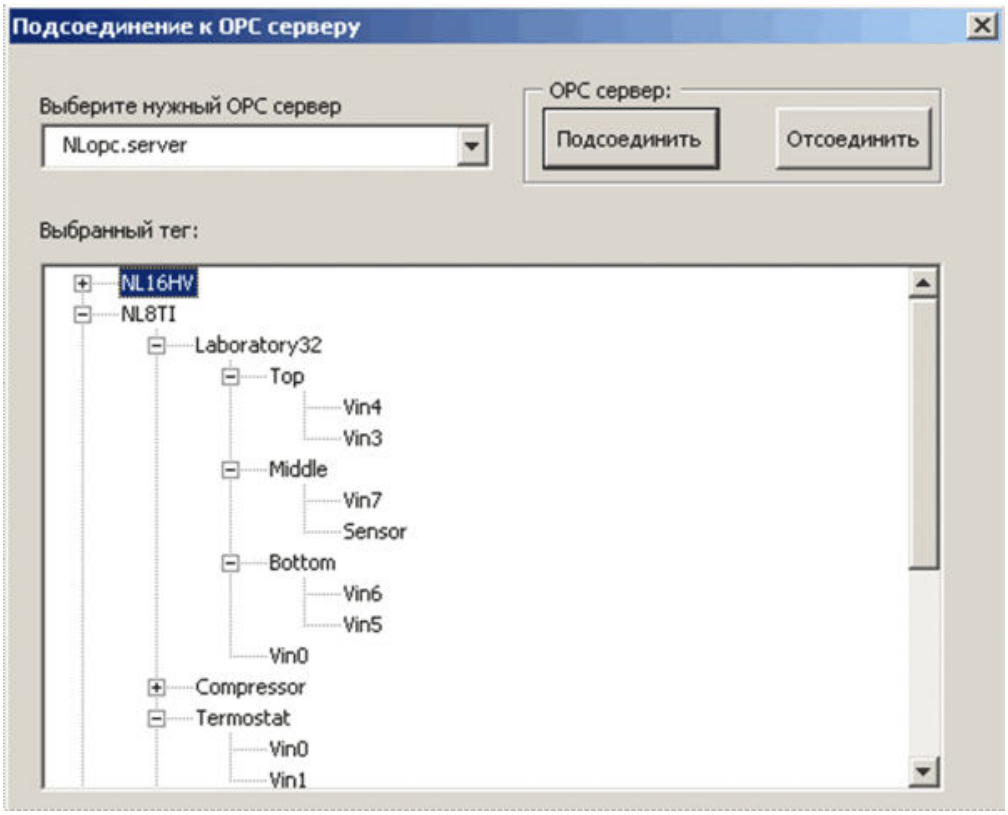

Рис. 9.2. Пример диалогового окна навигатора тегов OPC клиента

При использовании OPC клиента (например, SCADA), имена тегов, доступные через OPC сервер, представляются в аналогичной форме в окне навигатора тегов (рис. 9.2). Клиент показывает все OPC серверы, установленные на компьютерах, доступных по сети Ethernet, и позволяет использовать все теги этих серверов.

Пример архитектуры систем, включающих OPC серверы и OPC клиенты, показаны на рис. 9.3 и рис. 9.4. В качестве OPC клиента может выступать программа на языке C++ (например, SCADA-пакет) или программа на языке Visual Basic, VBA, Delphi или любая другая программа, поддерживающая внедрение COM-объектов (рис. 9.3). Программа на языке C++ взаимодействует с OPC сервером через интерфейс OPC Custom, а программа на Visual Basic, VBA, Delphi - через интерфейс автоматизации OPC Automation. OPC сервер и OPC клиенты могут работать только на компьютерах и контроллерах с операционными системами, поддерживающими технологию DCOM (например, Windows XP и Windows CE).

OPC сервер подключается к физическим устройствам любым способом; эти способы стандартом не предусмотрены. Например, сервер NLopc использует для каждого физического устройства свой драйвер (рис. 9.3).

Клиентская программа и OPC сервер могут быть установлены на одном и том же компьютере, как показано на рис. 9.3, или на разных компьютерах сети Ethernet (рис. 9.4). При наличии нескольких компьютеров каждый из них может содержать OPC сервер и подключенные к нему физические устройства. В такой системе любой OPC клиент с любого компьютера может обращаться к любому OPC серверу, в том числе к расположенному на другом компьютере сети. Это достигается благодаря технологии DCOM, использующей удаленный вызов процедур (**RPC** - Remote Procedure Call). Например, SCADA на рис. 9.4 может обратиться за данными к модулю ввода-вывода по пути,

указанному на рис. 9.4 штриховой линией. Обратим внимание, что компьютеры и контроллеры в такой архитектуре могут работать с разными промышленными сетями. Обмен данными с ПЛК, работающими с ОС Windows CE, выполняется точно так, как с компьютерами.

При использовании оборудования разных производителей на компьютере (контроллере) может быть установлено несколько OPC серверов разных производителей, однако OPC сервер монопольно занимает COM-порт компьютера (поскольку непрерывно выполняет обновление данных), поэтому количество портов должно быть равно количеству OPC серверов. Для наращивания количества COM портов можно использовать преобразователи интерфейса USB в RS-232. К разным портам компьютера могут быть подключены разные промышленные сети. В этом случае OPC серверы используются в качестве межсетевых шлюзов.

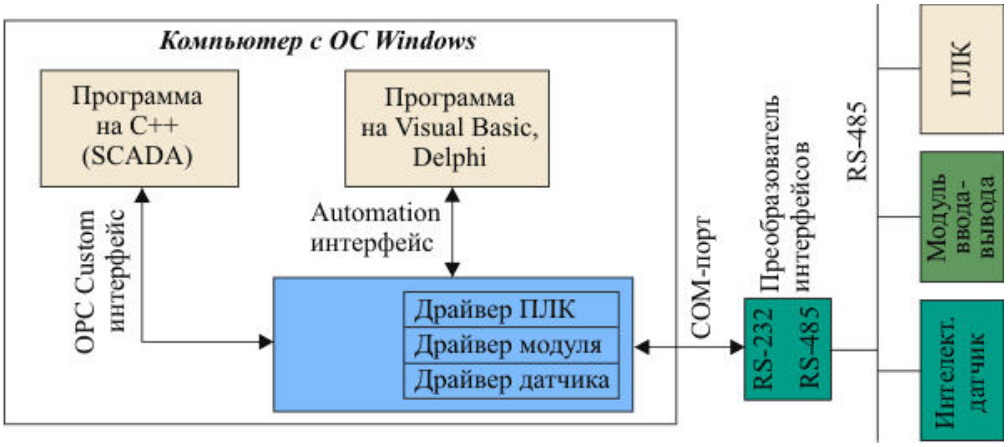

Рис. 9.3. Простой пример взаимодействия прикладных программ и физических устройств через OPC сервер на одном компьютере

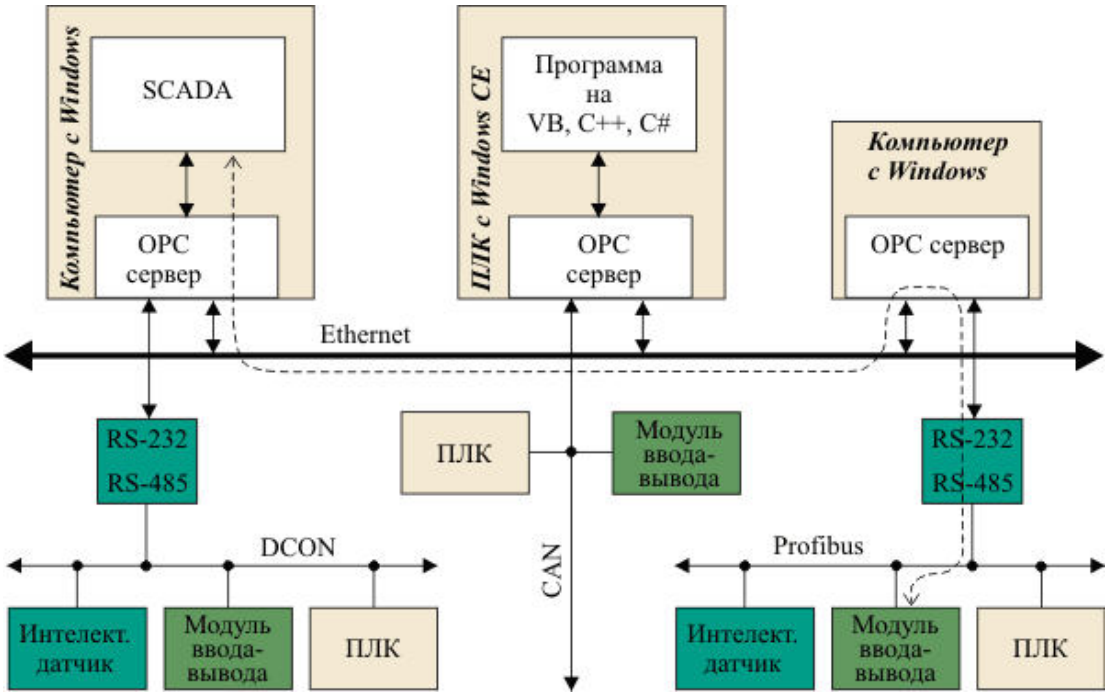

Рис. 9.4. Пример применения OPC технологии для сетевого доступа к данным в системах автоматизации

9.2.3. OPC HDA сервер

Целью **OPC HDA сервера** [OPC] (сервера предыстории процесса) является предоставление клиентской программе единого интерфейса для обмена данными с любыми хранилищами данных, в качестве которых может выступать нестандартный файл с данными, стандартная СУБД, OPC DA сервер или другой OPC HDA сервер. Стандарт распространяется только на интерфейсы для взаимодействия HDA сервера с клиентскими программами и не устанавливает способов получения или хранения данных.

Спецификация OPC HDA устанавливает стандарт на интерфейсы COM объекта и методы его использования. Структура сервера и методы взаимодействия с клиентами полностью аналогичны общей идеологии OPC и описанному выше OPC DA в частности. Например, OPC клиент может подсоединяться к нескольким OPC HDA серверам разных производителей и быть установлен на разных компьютерах в сети Ethernet.

Существует два типа HDA серверов:

- простой сервер данных предыстории для построения графиков (трендов);
- сервер для хранения данных в упакованном виде с возможностью их обработки и анализа. К функциям обработки и анализа данных относятся нахождение среднего, минимального и максимального значения и др.

Работа с данными заключается в чтении, записи или изменении данных.

9.2.4. Спецификация OPC UA

Несмотря на огромный успех и всеобщее признание, практика выявила следующие недостатки OPC технологии [Wikipedia, Lange]:

- доступность только на операционных системах семейства Microsoft Windows;
- связь с технологией DCOM, исходные коды которой являются закрытыми. Это не позволяет решать вопросы надежности ПО, а также выявлять и устранять возникающие программные отказы;
- бывают проблемы конфигурирования, связанные с DCOM;
- неточные сообщения DCOM о прерываниях связи;
- неприспособленность DCOM для обмена данными через интернет;
- неприспособленность DCOM для обеспечения информационной безопасности.

В связи с этим в 2006 году OPC Foundation предложил новую стандартную спецификацию для обмена данными в системах промышленной автоматизации [OPC], получившую название "OPC Unified Architecture" - "OPC с унифицированной архитектурой", которая рассматривается как OPC стандарт нового поколения.

Стандарт OPC UA устанавливает методы обмена сообщениями между OPC сервером и клиентом, не зависящие от аппаратно-программной платформы, от типа взаимодействующих систем и сетей. OPC UA обеспечивает надежную и безопасную коммуникацию, противодействие вирусным атакам, гарантирует идентичность информации клиента и сервера.

В новом стандарте используется понятие объекта, под которым понимается физический или абстрактный элемент системы. Примерами объектов могут быть физические устройства, включающие их системы и подсистемы. Датчик температуры, к примеру, может быть представлен как объект, который включает в себя значение температуры, набор параметров сигнализаций и границы их срабатывания. Объект, по аналогии с объектно-ориентированным программированием, определяется как экземпляр класса, а класс рассматривается как тип данных. Объекты включают в себя переменные, события и методы.

OPC UA использует несколько различных форматов данных, основными из которых являются бинарные структуры и XML документы. Формат данных может быть определен поставщиком OPC сервера или стандартом. Для работы с произвольными форматами клиент может запросить у сервера информацию об описании этого формата. Во многих случаях используется автоматическое распознавание формата данных во время их передачи.

OPC UA обладает высокой робастностью* данных и уведомлений о событиях. Робастность обеспечивается механизмом быстрого обнаружения ошибок коммуникации и восстановления данных.

Серверы могут иметь доступ как к текущим, так и архивированным данным, к событиям и аварийным сигналам. OPC UA может быть внедрен в различные коммуникационные протоколы, а данные могут быть закодированы способами, оптимальными по соотношению эффективности и переносимости на другие платформы.

Архитектура, ориентированная на сервисы

Основным отличием OPC UA от OPC является отказ от технологии COM и DCOM фирмы Microsoft и переход к архитектуре **SOA** (Service Oriented Architecture - "Архитектура, ориентированная на сервисы") с целью обмена информацией и обеспечения совместимости с множеством различных аппаратно-программных платформ. Под сервисом в OPC UA понимается некоторая функциональность, заключенная в программном компоненте, который может быть транспортирован от сервера к клиенту или обратно и вызван удаленно. Вызов сервиса аналогичен вызову метода в языках объектно-ориентированного программирования. Интерфейс между клиентом OPC UA и сервером определяется как набор сервисов. Основным принципом SOA является независимость от программной технологии, от вычислительной платформы, от языков программирования, от конкретных приложений, а также организация сервисов как слабосвязанных компонентов для построения систем. Сервисы включают в себя средства для обеспечения информационной безопасности.

Благодаря построению сервера OPC UA на основе сервисов появилась возможность изменять размер (масштабировать) сервера для его использования на платформах с разными вычислительными ресурсами: для встроенных приложений может быть использован сокращенный набор сервисов, для корпоративных сетевых серверов - полный набор.

Сервисы OPC UA делятся на логические группы [OPC]:

- сервисы безопасных каналов;
- сервисы сессий взаимодействия приложений по инициативе пользователя;

- сервисы для управления узлами**. Позволяют клиентам добавлять, модифицировать или удалять узлы в адресном пространстве;
- сервисы видимости узлов, позволяющие задавать индивидуальные наборы видимых узлов для разных клиентов;
- сервисы атрибутов позволяют модифицировать атрибуты узлов;
- сервисы методов, которые вызывают функции, исполняемые элементами системы;
- сервисы для мониторинга узлов в режиме подписки. Эти сервисы периодически контролируют переменные, атрибуты и события, а также генерируют уведомления при наступлении заданных условий;
- сервисы для осуществления подписки и публикации уведомлений.

Независимость от COM, DCOM

Отказ от DCOM стал возможен благодаря появлению новых транспортных механизмов, основанных на SOAP, XML, HTTP (см. раздел "Промышленные сети и интерфейсы") и сервисах (см. [Ивьен]). Благодаря им OPC UA позволяет осуществить безопасную и надежную доставку информации и объединить в одном сервере функциональность OPC DA, OPC HDA и OPC A&E серверов.

Стандарт OPC UA не предназначен для замены существующих OPC спецификаций, а дополняет и расширяет их возможности [Lange].

Безопасность

Для обеспечения информационной безопасности в OPC UA используются стандартные **Web сервисы** безопасности, такие как WS-Security, WS-Trust или WS-SecureConversation [Lange]. Диапазон возможностей средств безопасности простирается от простой аутентификации с помощью пароля и обмена цифровыми подписями до полного шифрования передаваемых сообщений.

OPC сообщения в стандарте UA передаются с помощью сообщений SOAP в виде XML текста. Поскольку кодирование и декодирование текстового формата занимает довольно много времени, стандарт предусматривает альтернативный способ представления информации в виде бинарного файла.

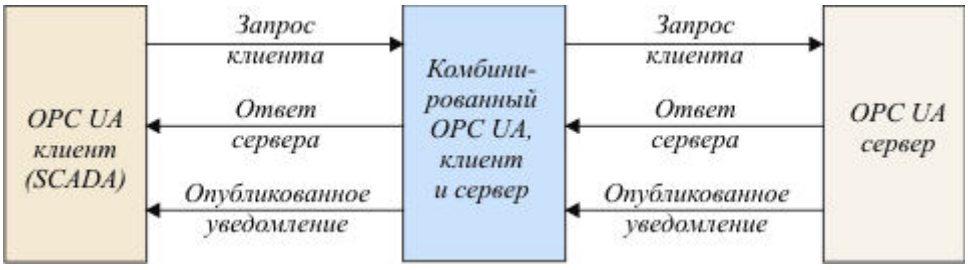

Рис. 9.5. OPC UA клиент и сервер могут быть скомбинированы в одном приложении для взаимодействия с другими OPC UA клиентами и серверами

Достоинства нового стандарта

Основными достоинствами OPC UA являются [Lange]:

- реализация на языке программирования ANSI C для обеспечения переносимости на другие платформы, включая встраиваемые системы. Доступны также версии на .NET и Java (от OPC Foundation);
- ориентация на сервисы вместо ориентации на объекты, что позволяет использовать OPC UA на любых компьютерах, встраиваемых системах, в коммуникаторах и т.п. которые используют веб-сервисы;
- позволяет осуществить масштабирование OPC UA, т.е изменение объема программы в зависимости от вычислительных ресурсов процессора и требуемой функциональности. Может быть выполнена также компиляция в виде однопоточного или многопоточного приложения;
- поддержка надежного и современного транспортного механизма SOAP на базе XML с применением HTTP протокола;
- обеспечение хорошей информационной безопасности;
- конфигурируемый таймаут для каждого сервиса;
- использование открытых стандартов World Wide Web Consortium (**W3C**) вместо закрытого стандарта COM/DCOM.

Концепция системы на базе OPC UA

Система на базе OPC UA может содержать множество клиентов и серверов. Каждый клиент может работать параллельно с несколькими серверами и каждый сервер может обслуживать нескольких клиентов. Пользовательское приложение (например, SCADA) может создавать комбинированные группы клиентов и серверов для ретрансляции сообщений, которыми оно обменивается с другими клиентами и серверами, как показано на рис. 9.5.

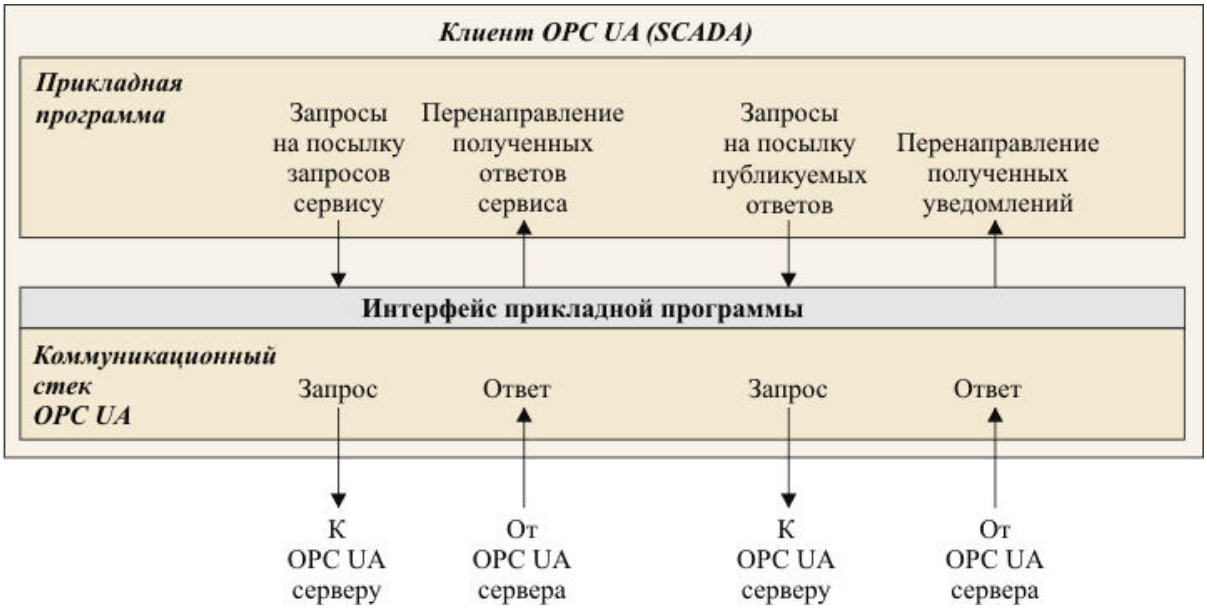

Рис. 9.6. Структура клиентской программы в стандарте OPC UA

Клиентом при взаимодействии с OPC сервером является прикладная программа, например, SCADA. Структура клиента показана на рис. 9.6. Клиентская программа выполняет запросы сервисов OPC сервера через внутренний интерфейс, который является изолирующей прослойкой

между программой и коммуникационным стеком. Коммуникационный стек конвертирует запросы клиентской прикладной программы в сообщения для вызова необходимого сервиса, которые посылает серверу. После получения ответа на запросы коммуникационный стек передает их в клиентскую программу.

Структура сервера OPC UA представлена на рис. 9.7. Модули ввода-вывода, ПЛК, интеллектуальные устройства и программы, которые могут поставлять данные через OPC сервер, обозначены на рис. 9.7 как "реальные объекты". Серверное приложение представляет собой программную реализацию функций, которые должен выполнять сервер. Взаимодействие OPC UA сервера с клиентом выполняется через интерфейс прикладной программы (рис. 9.7), путем отправления запросов и получения ответов.

Адресное пространство OPC сервера представляет собой множество узлов, доступных клиентской программе с помощью сервисов OPC UA. "Узлы" в адресном пространстве используются, чтобы представить реальные объекты, их определения и перекрестные ссылки. В адресном пространстве выделяется подпространство узлов, которые сервер делает "видимыми" для клиента. Видимые узлы организуются в виде иерархической структуры, для удобства навигации их клиентской программой.

Обмен данными между клиентом и сервером может выполняться как путем получения мгновенных ответов на запросы, так и по схеме "издатель-подписчик". Во втором случае клиентская программа осуществляет "подписку" на получение определенных данных, которые сервер должен будет предоставить по мере их появления. Для реализации режима подписки сервер осуществляет непрерывный контроль (мониторинг) узлов и соответствующих им реальных объектов с целью обнаружения изменений. При обнаружении изменений в данных, событиях или аварийных сигналах (алармах) сервер генерирует уведомление, которое передается клиенту по каналу подписки.

OPC UA допускает обмен между двумя серверами. Для этого один из серверов выступает в роли клиента, второй - в роли сервера. Таким образом можно соединить несколько серверов цепочкой, при этом каждый из них будет выступать с одной стороны цепочки в качестве клиента, с другой стороны - в качестве сервера, как показано на рис. 9.5.

Для защиты уже сделанных инвестиций в OPC на базе DCOM организация OPC Foundation разработала стратегию перехода на новую технологию с применением " **UA-оболочки**", которая допускает обмен данными между старыми и новыми продуктами. Такая оболочка позволяет, например, DCOM OPC серверу работать с OPC UA клиентом, и наоборот.

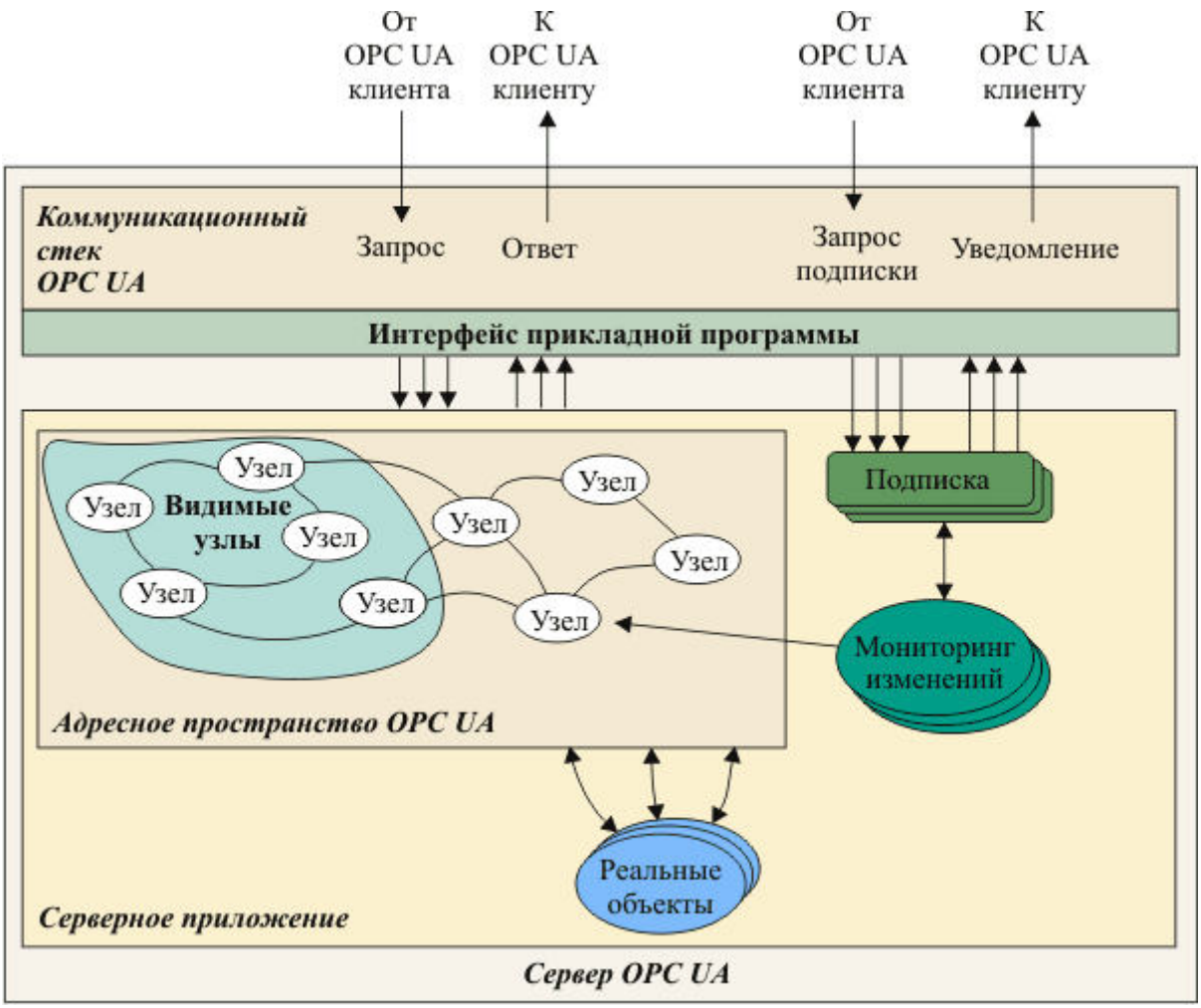

Рис. 9.7. Структура сервера в стандарте OPC UA

---

*Робастность - нечувствительность данных к помехам, сбоям, вычислительным ошибкам.

**Узел - представление объектов, их определений и перекрестных ссылок в адресном пространстве сервера OPC UA.

9.2.5. OPC DA сервер в среде MS Excel

Microsoft **Excel** является хорошей средой для разработки несложных систем автоматизации благодаря тому, что он содержит удобный пользовательский интерфейс для обработки данных, для построения графиков, вставки рисунков, выполнения анимации и т. п. [Sanchez]. Он содержит также встроенные элементы управления (раскрывающиеся списки, "радиокнопки", "чек-боксы" и т. п. [Кузьменко]). Excel позволяет сделать специализированный пользовательский интерфейс, не уступающий по дизайну профессиональным SCADA-пакетам. Особенно удобным является простое сохранение выполненной работы в файл.

Недостатком MS Excel является низкое быстродействие при записи данных в ячейки таблицы и отсутствие в VBA возможности выполнения задач в нескольких параллельных потоках. VBA включает в себя Visual Basic и дополнительные функции, обеспечивающие работу с приложением, например, с ячейками MS Excel или с параграфами MS Word. Мы рекомендуем использовать MS Excel только для применений, использующих не более 10...30 тегов при периоде их опроса не

менее 1 с. Однако это касается только операций в ячейках и динамического обновления графиков и не распространяется на обработку данных средствами языка VBA.

Возможность работы MS Excel (и других приложений Microsoft Office) с OPC сервером обеспечивается благодаря тому, что Visual Basic, входящий в состав MS Excel, поддерживает технологию **Automation** (**OLE Automation**) фирмы Microsoft. Суть Automation состоит в том, что она позволяет одному приложению (клиенту) использовать объекты другого приложения (сервера). Применительно к задаче промышленной автоматизации это означает, что приложение-клиент может использовать объекты OPC сервера, который, в свою очередь, может получать данные из физического устройства или записывать их в него.

Automation позволяет использовать в среде MS Excel также объекты других приложений, например, ActiveX объекты, объекты MS Word, Outlook Express и др. Интерфейс Automation позволяет использовать OPC сервер с любыми другими приложениями, имеющими встроенный Visual Basic for Application.

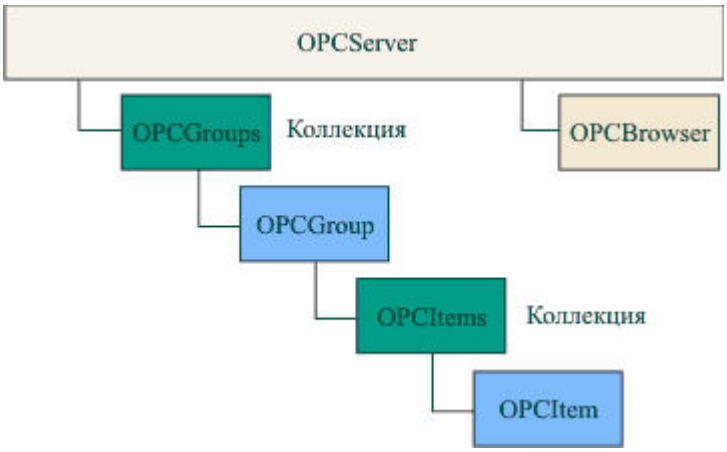

Рис. 9.8. Иерархия объектов интерфейса Automation

Объекты, доступные в MS Excel через интерфейс Automation, видны в списке объектов "Object Browser" в меню "View" Microsoft Visual Basic, который открывается из пункта меню Сервис/Macros/ Редактор Visual Basic в Excel 2003 и более старых версиях, или в меню "Разработчик/Visual Basic" в MS Excel 2007.

Для подключения OPC сервера к MS Excel сначала необходимо инсталлировать на компьютере OPC сервер. Стандартный OPC сервер должен регистрироваться в реестре Windows автоматически. Затем нужно открыть пункт меню Tools/References в редакторе Visual Basic, найти в списке "Available References" и отметить галочкой ссылку на имя OPC сервера. После этого в списке объектов "Object Browser" появляется список объектов OPC сервера и их можно использовать по общим правилам программирования на Visual Basic. Более подробно о работе с объектами OLE Automation можно ознакомиться в руководствах по программированию, например, [Кузьменко].

На рис. 9.8 показана иерархия объектов, предоставляемых интерфейсом OPC Automation. Описание свойств и методов каждого объекта можно найти в документах [Data, OLE]. Значение данных, считанных из устройства ввода, параметр качества данных и временная метка хранятся в объекте OPCItem. Несколько OPCItem могут составлять коллекцию [Кузьменко]. Коллекция - это

термин VBA, она представляет собой группу связанных объектов, каждый из которых называется элементом коллекции и может быть вызван по его индексу [Кузьменко]. Каждая коллекция имеет свойство Count, которое позволяет подсчитать количество элементов.

Объект OPCServer представляет собой экземпляр OPC сервера. Этот объект должен быть создан до того, как будут установлены ссылки на другие объекты. Он содержит коллекцию OPCGroups и создает объект OPCBrowser, который используется в OPC клиенте для построения навигатора тегов.

Рассмотрим пример чтения значения тега из OPC сервера в ячейку листа MS Excel. Сначала необходимо создать модуль в Visual Basic, входящем в состав MS Excel. В заголовке модуля объявляются переменные *Server* и *Group* как имеющие тип *OPCServer* и *OPCGroup* соответственно, с областью видимости Public (строки 1 и 2 в листинге 1):

Листинг 1. Подсоединение к OPC серверу

```
1  Public Server As OPCServer
2  Public Group As OPCGroup
3  Sub Connect()
4      If Server Is Nothing Then
5          Set Server = New OPCServer
6      End If
7      If Group Is Nothing Then GoTo noGroup
8      Set Group = Nothing
9  noGroup:
10     Server.Connect "NLopc.server"
11     Set Group = Server.OPCGroups.Add("RLDA")
12End Sub
```

Первым делом нужно создать связь MS Excel с OPC сервером. Для этого используется процедура *Connect()* (строка 3). Для того, чтобы не создавать новые объекты, если они уже существуют, используется проверка существования объекта *Server* (строка 4) и если он не существует, то создается новый оператором
New (строка 5).

Аналогично поступают с объектом *Group* (строка 7). Если объект не существует, то переходят к метке *noGroup*:, за которой вызывается метод *Connect* объекта *Server*, который осуществляет соединение с сервером *"NLopc.server"* (в этом примере использован OPC сервер Nopc, www.RealLab.ru). Далее создается новая объектная переменная *Group* и добавляется методом *Add* в коллекцию *OPCGroups* с любым именем, например, "RLDA". Если объект *Group* существует, то его сначала удаляют (строка 8), чтобы не проверять, откуда он появился, затем создают заново (строка 10, 11).

После выполнения процедуры листинга 1 можно прочитать данные из OPC сервера. Пример чтения данных иллюстрирует листинг 2.

Предположим, что нужно считать данные тега, указанного на рис. 9.1. Его имя присваивают переменной *tagname* (строка 7). Далее имя тега нужно добавить к коллекции *OPCItems* с помощью метода *AddItem* (строка 8). Второй переменной после имени тега является номер тега в коллекции.

После этого объектную переменную anItem ассоциируют с объектом Item, имеющим порядковый номер 1, под этим номером в коллекцию OPCItems был добавлен наш тег.

Объект anItem имеет метод Read, который позволяет прочесть параметры интересующего нас тега: значение Value, качество Quality и временную метку TimeStamp (строка 10).

После чтения этих переменных (строка 10) их значения записываются ячейки листа Excel с именем Лист1, находящиеся на пересечении 10-й, 11-й и 12-й строки с 5-м столбцом. Функция DoEvents необходима для того, чтобы Excel мог обрабатывать другие события, происходящие в системе, когда процедура Read() используется в цикле.

После получения данных от OPC сервера тег нужно удалить из коллекции. Для этого находят его указатель с помощью метода ServerHandle (строка 15), затем удаляют из коллекции OPCItems методом Remove. Далее удаляют ассоциацию объектной переменной anItem с реальным объектом (строка 17).

Листинг 2. Чтение данных из OPC сервера

```
1  Sub Read()
2      If Server Is Nothing Then Exit Sub
3      If Group Is Nothing Then Exit Sub
4      Dim serverHandles(1) As Long, Errors() As Long
5      Dim tagname As String, anItem As OPCItem
6      Dim Value, Quality, TimeStamp As Variant
7      tagname = "NL8TI.Laboratory32.Top.Vin4"
8      Group.OPCItems.AddItem tagname, 1
9      Set anItem = Group.OPCItems.Item(1)
10     anItem.Read OPCCache, Value, Quality, TimeStamp
11     Лист1.Cells(10, 5).Value = Value
12     Лист1.Cells(11, 5).Value = Quality
13     Лист1.Cells(12, 5).Value = TimeStamp
14     DoEvents
15     serverHandles(1) = anItem.ServerHandle
16     Group.OPCItems.Remove 1, serverHandles, Errors
17     Set anItem = Nothing
18End Sub
```

Упрощенный интерфейс EasyAccess

Описанная процедура работы с OPC сервером хороша, когда на основе Visual Basic делается удобная заказная программа и нужно использовать все богатство возможностей OPC сервера. Однако для быстрого получения данных при проведении несложного эксперимента применение OPC сервера неоправданно сложно. В этих случаях можно использовать упрощенный COM объект **EasyAccess** [Денисенко], подробно описанный на вебсайте RealLab!. Для того, чтобы считать из OPC сервера значение тега, например, "NL8TI.Laboratory32.Top.Vin4" (см. листинг 3), достаточно после объявления переменных создать объект функцией "CreateObject" (строка 5) и сразу после этого получить значение тега (строка 7). Строка 8 записывает это значение на лист Excel в ячейку (7, 6).

Листинг 3. Применение упрощенного интерфейса EasyAccess

```
1Sub OnReadTag()
2    Dim obj As Object
3    Dim val As Variant
4    Dim tag As String
5    Set obj = CreateObject("NLopc.console")
6    tag = "NL8TI.Laboratory32.Top.Vin4"
7    obj.GetTagValue tag, val
8    Лист1.Cells(7, 6).Value = val
9End Sub
```

Для записи данных в физическое устройство вместо функции obj.GetTagValue используют obj.SetTagValue.

Построение графиков, обработка полученных данных выполняются обычными средствами Visual Basic [Кузьменко].

9.2.6. Применение OPC сервера с Matlab и LabVIEW

В **MatLab**, начиная с версии 7, входит пакет OPC Toolbox, поддерживающий OPC серверы версии v2.05a. Он содержит также блоки Simulink для обмена данными с OPC серверами. Аналогично, LabVIEW, начиная с версии 7, также имеет встроенную поддержку стандарта OPC.

Более ранние версии MatLab и LabVIEW не поддерживают OPC, однако поддерживают OLE Automation. Для работы с такими программами можно использовать объект OPCWrapper.Service, разработанный RealLab! (www.RealLab. ru) и представляющий собой буфер, преобразующий стандартные вызовы OPC в стандартные вызовы OLE Automation. Для использования объекта OPCWrapper.Service сначала создают соединение с ним с помощью строки (в среде MatLab)

wrapper = actxserver(' OPCWrapper.Service'),

затем подключают объект к серверу:

invoke(wrapper,'OpcOpenServer','NLopc.Server',''),

где вместо "NLopc. Server" можно использовать имя любого OPC сервера.

Значение тега, например, "NL8TI.Vin1" читают с помощью функции invoke из пакета MatLab:

value = invoke(wrapper,'OpcReadItem','NL8TI.Vin1').

Запись в физическое устройство дискретного значения "1" тега "NL8TI.Bit.Dout0" выполняется аналогично:

invoke(wrapper,'OpcWriteItem','NL8TI.Bit.Dout0',1).

Отсоединить объект от сервера можно следующим образом:

invoke(wrapper, 'OpcCloseServer').

После этого соединение с объектом OPCWrapper.Service необходимо закрыть:

release(wrapper).

Примеры работы пакета MatLab с физическими устройствами можно найти на сайте www.RealLab.ru, а также в работе [Grega].

9.1. развитие программных средств автоматизации

9.3. системы программирования на языках мэк 61131-3