



Сервер VXI-11 на платформе QNX Neutrino

Сергей Минеев, Сергей Фомин

Обсуждается архитектура клиент-сервер применительно к распределённым измерительным системам. Показана роль спецификации VXI-11 при создании встроенного программного обеспечения контрольно-измерительных приборов. Содержится практическое руководство по созданию программного обеспечения, позволяющего управлять приборами по сети Ethernet в соответствии с требованиями специализированного стандарта.

ВВЕДЕНИЕ

Программно-управляемые измерительные приборы находят всё более широкое применение в контрольно-измерительных комплексах, испытательных стендах и различных автоматизированных системах. Уже многие годы действуют международные стандарты на аппаратное и программное обеспечение интерфейсов связи программно-управляемой аппаратуры с управляющими компьютерами: IEEE 488.1, IEEE 488.2, *VXIplug&play* 4.X, VXI-11.X и др. На сегодняшний день наиболее распространённым приборным интерфейсом является IEEE 488. Им оснащается подавляющее большинство аппаратуры таких фирм, как Agilent Technologies, Tektronix, Rohde&Shwartz, ННИПИ «Кварц» и др. С помощью специальных интерфейсных плат и таких программных продуктов, как NI VISA (National Instruments) или AG VISA (Agilent Technologies), достаточно просто организовать обмен данными между управляющим компьютером и приборами, подключёнными к шине IEEE 488, что уже давно оценено российскими и зарубежными разработчиками контрольно-измерительных систем.

На рубеже XX и XXI веков невысокая пропускная способность шины IEEE 488 и другие принципиальные ограничения заставили производителей измерительной аппаратуры оснащать свои изделия дополнительными высокоскоростными интерфейсами Ethernet 10/100/1000 Гбит/с, USB, IEEE 1394. Вновь выпускаемое оборудование всё

чаще совсем не поддерживает интерфейс IEEE 488, но благодаря реализации производителями требований таких стандартов, как *VXIplug&play* 4.X и VXI-11.X, сохраняется преемственность способов взаимодействия программ пользователей с измерительной аппаратурой.

К сожалению, отечественные фирмы-производители, в своё время хорошо освоившие шину IEEE 488, не торопятся оснащать свои приборы современными скоростными последовательными интерфейсами. Причина такой неторопливости не связана со сложностью интеграции в приборы аппаратной части интерфейсов, а кроется в проблемах логической совместимости со спецификациями VXI-11.X и *VXIplug&play* 4.X. То есть техническая возможность оснастить прибор интерфейсом, например Ethernet 100Base-T, есть, но для того чтобы иметь возможность управлять данным прибором так же, как и приборами других производителей, необходимо реализовать и разместить в приборе программное обеспечение, соответствующее опубликованным спецификациям консорциумов VXI и *VXIplug&play*.

Существенно сократить затраты, связанные с внедрением в приборостроительной отрасли новых связанных интерфейсов, можно с помощью встраиваемой микропроцессорной техники. Предпосылками этого являются снижение стоимости компактных, быстродействующих микропроцессорных плат, уже оснащённых необходимыми интерфейсами, и появление мульти-

платформенных встраиваемых операционных систем. Как в таком окружении организовать обмен данными между измерительным прибором и управляющим компьютером в соответствии со стандартным протоколом? Ответу на этот вопрос и посвящена данная статья.

АРХИТЕКТУРА КЛИЕНТ-СЕРВЕР ДЛЯ РАСПРЕДЕЛЁННЫХ ИЗМЕРИТЕЛЬНЫХ СИСТЕМ

По мере совершенствования вычислительной техники и средств коммуникации всё чаще контрольно-измерительные комплексы рассматриваются как распределённые многопроцессорные системы (рис. 1). Измерительные средства, входящие в состав таких комплексов, управляются встроенными микропроцессорами, а обмен данными с управляющими узлами осуществляется посредством каналов Ethernet или USB. За счёт открытости и распространённости применяемых связанных интерфейсов упрощаются разработка, сопровождение, расширение таких комплексов, а ограничения на пространственное размещение аппаратных средств практически отсутствуют.

Спецификация VXI-11 [1] консорциума VXI определяет механизмы и логику взаимодействия приборов, оснащённых интерфейсом Ethernet, с управляющим компьютером. Хотя данная спецификация и выпущена под эгидой консорциума VXI, но к шине VXIbus никакого отношения не имеет и может применяться (и применяется) к любым приборам, управление которыми сводится к выдаче команд и опера-

циям чтения/записи символьных буферов (аналогично IEEE 488.1 и/или IEEE 488.2).

Определим прибор (вернее, логический блок прибора, «обращённый» наружу через Ethernet) как приборный сервер (Network Instrument Server), а программу, выполняющуюся на управляющем компьютере, как клиент прибора (Network Instrument Client).

Стек протоколов, задействованных при обмене информацией между приборным сервером и приборным клиентом, описывается в таблице 1. Два самых нижних уровня стека протоколов должны быть реализованы драйвером сетевого ввода/вывода и никакой специфики, связанной с рассматриваемой задачей, не имеют. Сеансовый уровень определён как ONC/RPC (Open Network Computing / Remote Procedure Call), а внешнего представления данных — XDR (External Data Representation). Наиболее известная реализация данных спецификаций — пакет Sun RPC фирмы Sun Microsystems [2], который входит в состав дистрибутивов большинства операционных систем семейства Unix. Для семейства Windows доступны коммерческие реализации ONC/RPC + XDR.

Прикладной уровень подробно описан в спецификации VXI-11 [1], где определены интерфейсы вызываемых посредством RPC функций и правила реализации внутренней логики приборного сервера и его клиента. Всего определены три канала (Channels) для информационного взаимодействия: основной (Core), прекращения (Abort) и прерываний (Interrupt). Распределение процедур по каналам показано в таблице 2.

Вызов всех процедур через RPC осуществляется по инициативе приборного клиента (каналы Core и Abort). Исключением является `device_intr_srqr()` — эта

Таблица 1
Стек протоколов, задействованных при обмене информацией между приборным сервером и приборным клиентом

Уровень	Протокол
Прикладной	VXI-11
Внешнего представления данных	XDR
Сеансовый	ONC/RPC
Транспортный	TCP
Сетевой	IP

Таблица 2

Распределение процедур по каналам

Функция	Канал	Назначение
<code>create_link()</code>	Core	Устанавливает соединение с прибором
<code>device_write()</code>	Core	Отправляет сообщение прибору
<code>device_read()</code>	Core	Считывает ответное сообщение прибора
<code>device_readstb()</code>	Core	Считывает статусный байт прибора
<code>device_trigger()</code>	Core	Запускает прибор
<code>device_clear()</code>	Core	Сбрасывает прибор в исходное состояние
<code>device_remote()</code>	Core	У прибора отключается локальное (с панели) управление
<code>device_local()</code>	Core	У прибора включается локальное (с панели) управление
<code>device_lock()</code>	Core	Прибор блокируется
<code>device_unlock()</code>	Core	Прибор разблокируется
<code>create_intr_chan()</code>	Core	Создает канал прерываний
<code>destroy_intr_chan()</code>	Core	Ликвидирует канал прерываний
<code>device_enable_srqr()</code>	Core	Прибору разрешается/запрещается запрашивать обслуживание
<code>device_docmd()</code>	Core	Прибору выдается команда
<code>destroy_link()</code>	Core	Ликвидирует соединение с прибором
<code>device_abort()</code>	Abort	Прекращает исполнение прибором текущей команды
<code>device_intr_srqr()</code>	Interrupt	Устройство запрашивает обслуживание

процедура вызывается по инициативе сервера, в этом случае происходит обмен ролями между сервером и клиентом.

Для успешной работы RPC на стороне сервера должен быть запущен демон (скрытая от пользователя служебная программа) `portmap` или `grcbind` [2]. Для успешной работы клиента наличие процессов `portmap` или `grcbind` необязательно.

Взаимодействие приборного сервера и клиента начинается с вызова клиентом процедуры `create_link()`. При первом вызове данной процедуры сервер должен открыть свободный порт,

запустить процесс, где данный порт будет прослушиваться. Номер прослушиваемого порта должен быть возвращён клиенту в качестве порта канала Abort. Кроме номера порта, функцией `create_link()` должны быть возвращены код ошибки (0 — успех), идентификатор соединения (произвольное число) и максимально допустимое число байтов для сообщений устройству. При последующих вызовах `create_link()` значения возвращаемых параметров должны быть тождественны значениям при первом вызове.

После того как соединение установлено, клиент может вызывать любые другие процедуры каналов Core и Abort. Обязательными для реализации на стороне приборного сервера являются процедуры `device_write()`, `device_read()`, `device_lock()`, `device_unlock()`. Остальные процедуры могут быть не реализованы, то есть при вызове возвращать код ошибки 8 (operation not supported). Примечательно, что в спецификациях [3, 4, 5] определены рекомендуемые шаблоны поведения приборов. Шаблоны представляют собой комплекты правил (ограничений) для реализации процедур таким образом, чтобы поведение приборов соответствова-

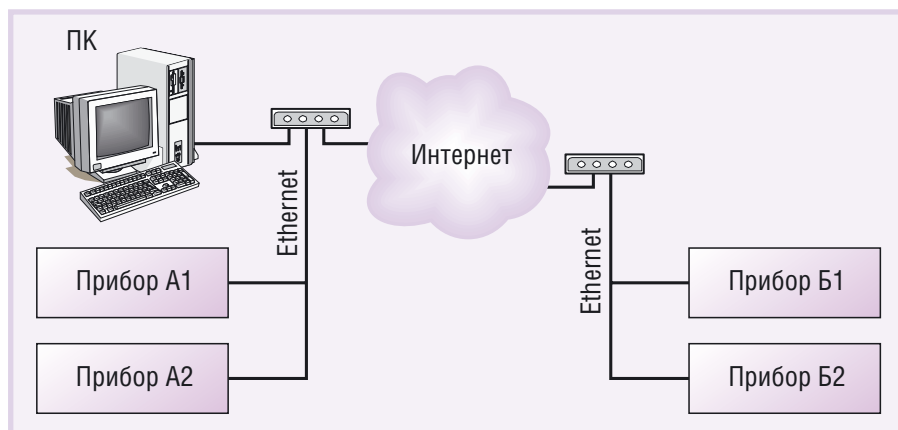


Рис. 1. Распределённый многопроцессорный контрольно-измерительный комплекс

Листинг 1

Генератор сигналов Agilent Technologies 33220A	Микропроцессорная плата с QNX Momentics 6.3 (запущена программа-демон portmap [2])
<pre>C:\tmp>ntrpcinfo 192.168.10.94 Getting RPC information for: 192.168.10.94 ===== Program: **DONE** Program ID : 395180 Version: 1 Port: 1024 Protocol: TCP ===== Program: **DONE** Program ID : 395183 Version: 1 Port: 1024 Protocol: TCP</pre>	<pre>C:\tmp>ntrpcinfo 192.168.10.81 Getting RPC information for: 192.168.10.81 ===== Program: **DONE** Program ID : 100000 Version: 4 Port: 111 Protocol: TCP ===== Program: **DONE** Program ID : 100000 Version: 3 Port: 111 Protocol: TCP ===== Program: **DONE** Program ID : 100000 Version: 2 Port: 111 Protocol: TCP Program: **DONE** ===== Program ID : 100000 Version: 4 Port: 111 Protocol: UDP ===== Program: **DONE** Program ID : 100000 Version: 3 Port: 111 Protocol: UDP ===== Program: **DONE** Program ID : 100000 Version: 2 Port: 111 Protocol: UDP</pre>

ло спецификациям VXIbus [3], IEEE 488.1 [4] и IEEE 488.2 [5].

По завершении работы с прибором клиент должен ликвидировать соединение посредством вызова процедуры `destroy_link()`. После того как успешно отработала данная процедура, работа с прибором возможна только после вызова `create_link()`.

СОЗДАНИЕ СЕРВЕРА VXI-11 НА ПЛАТФОРМЕ QNX

Одним из способов оснащения прибора интерфейсом Ethernet является применение готовой микропроцессорной платы, поддерживающей данный интерфейс. Компактные и функциональные устройства данного типа выпускаются фирмами Octagon Systems, Fastwel, Advantech и др. Абстрагироваться от аппаратной реализации микропроцессорных плат можно, выбрав в качестве операционной системы RTOS QNX Neutrino. Основные предпосылки такого выбора: близость QNX Neutrino к операционным системам Unix и, как следствие, наличие реализации ONC/RPC, а также широкая номенклатура поддерживаемых аппаратных платформ.

Итак, имеем процессорную плату с интерфейсом Ethernet и развёрнутым пакетом QNX Momentics 6.3 (среда разработки). Кроме серверной части, понадобится клиент — пусть его роль бу-

дет выполнять ПЭВМ под управлением операционной системы Windows XP. На клиентском вычислительном узле должны быть развёрнуты программы AG VISA (библиотека ввода/вывода для взаимодействия с приборами доступна по адресу http://adn.tm.agilent.com/index.cgi?CONTENT_ID=1035&LAST_CONTENT_ID=747) и NTRpcInfo (аналог Unix-утилиты `rpcinfo`, доступна по адресу <http://www.securitylab.ru/software/232914.php>).

Попробуем посмотреть на реальный прибор и наш сервер со стороны клиента. В качестве прибора, уже поддерживающего спецификацию VXI-11, возьмём генератор сигналов Agilent Technologies 33220A. Пусть IP-адрес прибора будет 192.168.10.94, а создаваемого сервера — 192.168.10.81. Воспользуемся утилитой NTRpcInfo для получения информации о механизме и процедурах RPC на данных сетевых узлах (листинг 1). Из этого эксперимента видно, что в генераторе сигналов зарегистрированы две программы, процедуры которых можно вызвать посредством ONC/RPC по протоколу TCP. Согласно спецификации [1] номер 395183 имеет канал Core, то есть все процедуры, определённые для данного канала, могут быть вызваны удалённо. Программа под номером 395180 в спецификациях VXI-11 не определена. Для разрабатываемого же сервера ка-

налы Core и Abort не определены (что совершенно естественно), а эксперимент показывает наличие полноценной службы распределения портов ONC/RPC (зарезервированные для такой службы идентификатор 100000 и порт 111), поддерживающей одновременно 3 версии спецификации ONC/RPC на базе протоколов TCP и UDP. Тот факт, что программ с идентификатором 100000 нет в генераторе, означает, что полноценного распределителя портов там тоже нет, а развёрнута только урезанная реализация, рассчитанная на работу только с двумя программами (395183 и 395180).

ГЕНЕРАЦИЯ ONC/RPC-ЗАГЛУШКИ

Первое, что необходимо сделать на пути к реализации сервера VXI-11, — это создать файлы-заглушки серверной части. Роль заглушки — ожидание запросов со стороны клиента на запуск процедур с известной сигнатурой и по приходу таких запросов вызов этих процедур с передачей им полученных от клиента параметров. Так как интерфейсы RPC-процедур различны, то и заглушка для каждого нового интерфейса должна быть сгенерирована заново.

Интерфейсы процедур каналов Core, Abort и Interrupt определены в спецификации VXI-11 с использованием нотации RPCL (RPC Language). Необходимо скопировать спецификацию каналов Core и Abort в файл с расширением «.x» (спецификацию канала Interrupt копировать не надо, так как данная программа должна поддерживаться клиентом, а не сервером). В состав операционных систем семейства Unix обычно входит утилита `grscgen`, которая предназначена для генерации кода заглушек RPC; есть такая утилита и в QNX Momentics. Пусть спецификации каналов Core и Abort помещены в файл `Device.x`. Попробуем сгенерировать заглушку средствами QNX Momentics:

```
$rpcgen -C /tmp/Device/Device.x
cannot find any C preprocessor (cpp)
```

Весьма неожиданный результат: утилита `grscgen` есть, а сгенерировать заглушку с её помощью нельзя. К сожалению, в составе QNX Momentics 6.3 отсутствует препроцессор `glibc`, и утилита `grscgen` неработоспособна. Поэтому для генерации придётся воспользоваться какой-либо другой Unix-системой (так как сгенерировать код заглушки нужно только один раз, то это не должно вызвать особых затруднений), например FreeBSD или Linux.

