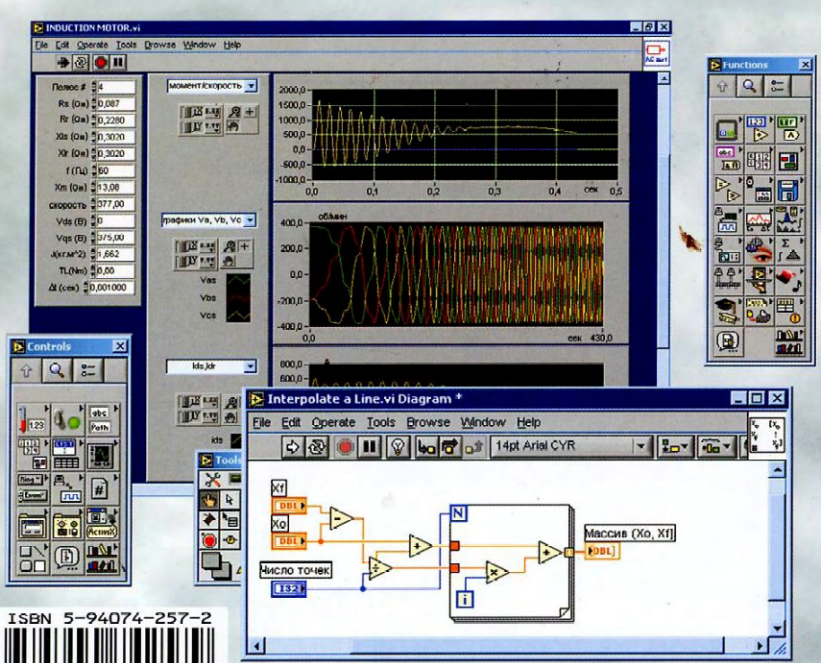
NATIONAL INSTRUMENTS™
ПриборКомплект
ДМК ПРЕСС ИЗДАТЕЛЬСТВО

КОМПАКТ-ДИСК К КНИГЕ

LabVIEW® для всех



ISBN 5-94074-257-2

ДЖЕФФРИ ТРЕВИС

Джеффри Тревис

LabVIEW
для всех

LabVIEW®

для всех

▲ Джеффри Тревис

Перевод Клушина Н. А.
Под редакцией Шаркова В. В., Гурьева В. А.

ПриборКомплект

ДМК Пресс
издательство

Москва, 2005

УДК 004.94
ББК 32.973.26-018.2
Т66

Тревис Дж.
Т66 LabVIEW для всех / Джеффри Тревис : Пер. с англ. Клушин Н. А. – М. : ДМК Пресс ; ПриборКомплект, 2005. – 544 с. : ил.

ISBN 5-94074-257-2

«LabVIEW для всех» – базовый курс по основам графического программирования в инженерной среде LabVIEW. Пакет LabVIEW формализует этап создания алгоритма работы прибора, описывая этот алгоритм в виде блок-схемы. В книге отражены все этапы создания виртуального прибора: регистрация сигнала, обработка, отображение.

На компакт-диске, приложенном к книге, представлена демонстрационная версия программы LabVIEW и приведено большое количество примеров.

Издание предназначено для инженеров и студентов технических вузов.

ББК 32.973.26-018.2
УДК 004.94

ISBN 0-13-065096-X (англ.)

ISBN 5-94074-257-2 (рус.)

СОДЕРЖАНИЕ

Предисловие

LabVIEW или *Laboratory Virtual Instrument Engineering Workbench* (Среда разработки лабораторных виртуальных приборов) представляет собой среду графического программирования, которая широко используется в промышленности, образовании и научно-исследовательских лабораториях в качестве стандартного инструмента для сбора данных и управления приборами. LabVIEW – мощная и гибкая программная среда, применяемая для проведения измерений и анализа полученных данных. LabVIEW – многоплатформенная среда: вы можете использовать ее на компьютерах с операционными системами Windows, MacOS, Linux, Solaris и HP-UX. Персональные компьютеры являются более гибкими инструментами, чем традиционные измерительные приборы, поэтому создание собственной программы на LabVIEW, или *виртуального прибора* (ВП), является довольно несложным делом, а интуитивно понятный пользовательский интерфейс в среде LabVIEW делает разработку программ и их применение весьма интересным и увлекательным занятием.

Концепция LabVIEW сильно отличается от последовательной природы традиционных языков программирования, предоставляя разработчику легкую в использовании графическую оболочку, которая включает в себя весь набор инструментов, необходимых для сбора данных, их анализа и представления полученных результатов. С помощью графического языка программирования LabVIEW, именуемого G (Джей), вы можете программировать вашу задачу из графической блок-диаграммы, которая *компилирует* алгоритм в машинный код. Являясь превосходной программной средой для бесчисленных применений в области науки и техники, LabVIEW поможет вам решать задачи различного типа, затрачивая значительно меньше времени и усилий по сравнению с написанием традиционного программного кода.

За пределами лабораторий

LabVIEW находит применение в самых разнообразных сферах человеческой деятельности. В соответствии со своим названием он первоначально использовался в исследовательских лабораториях, да и в настоящее время является наиболее популярным программным пакетом как в лабораториях фундаментальной науки (например, Lawrence Livermore, Argonne, Batelle, Sandia, Jet Propulsion Laboratory, White Sands и Oak Ridge в США, CERN в Европе), так и в отраслевых промышленных

лабораториях. Все более широкое применение LabVIEW находит в образовании – в университетских лабораторных практикумах – особенно по предметам электротехники, механики и физики.

Распространение LabVIEW за пределами лабораторий пошло по всем направлениям: вверх (на борту космических аппаратов), вниз (на подводных лодках) и по горизонтали (от буровых установок в Северном море до промышленных предприятий в Новой Зеландии). В связи с ростом возможностей Internet сфера применения LabVIEW стала расширяться не только в географическом, но и в виртуальном пространстве (cyberspace). Все большее число разработчиков создает виртуальные приборы, допускающие удаленное управление и наблюдение через Internet. Измерительные системы на основе виртуальных приборов отличаются своей многофункциональностью, гибкостью и низкой стоимостью как с точки зрения оборудования, так и с точки зрения затрат времени на разработку. Нужно ли удивляться, что они стали столь популярны?

Расширяющийся мир виртуальных приборов

Пожалуй, лучшим способом объяснить причины столь широкого (можно сказать, лавинообразного) распространения пакета LabVIEW будет обобщение способов его использования. Во всех видах человеческой деятельности существуют области, где не обойтись без определенных видов измерений – очень часто это температурные измерения, например в печах, холодильниках, парниках, технологических помещениях и даже... в кастрюле с супом. Кроме температуры, часто измеряют давление, силу, пространственное смещение, механическое напряжение, *pH* и т.д. – список огромный! Сейчас персональные компьютеры проникли практически во все сферы жизнедеятельности. LabVIEW ускоряет внедрение компьютера в измерительные системы – и не только потому, что облегчает проведение измерений, он также дает возможность проанализировать измеренные величины, отобразить их на графиках и в отчетах и при желании опубликовать.

После *измерения* и *анализа* какой-либо величины следующим логическим шагом часто является *управление*, то есть изменение определенных параметров в зависимости от полученных результатов. Например, измерив температуру объекта, можно включить устройство для его охлаждения либо нагрева. И вновь LabVIEW значительно облегчает решение этой задачи: *мониторинг* и *управление процессами* являются основными функциями этого программного продукта. Управление процессами может быть прямым или осуществляется через специальные *программируемые логические контроллеры* (programmable logical controllers – PLC), что принято называть *диспетчерским управлением и сбором данных* (supervisory control and data acquisition – SCADA).

Итоги

В этой книге вы познакомитесь с описанием различных применений LabVIEW. Они приведены непосредственно со слов разработчиков в весьма краткой форме, но дают представление о сущности приложения. Примеры подобраны большей частью

из тех областей промышленности, где пакет наиболее популярен – автоматизированное тестирование электронных компонентов, производство полупроводников, медицинского оборудования, испытания автомобильной техники и автоматизация технологических процессов. Из огромного перечня успешных применений LabVIEW мы выбрали следующие, наиболее яркие примеры:

- стимулирование сердечной активности;
- определение мест утечки водорода на космических кораблях;
- управление процессом изготовления мороженого;
- моделирование энергетических систем для анализа качества электроснабжения;
- управление питанием молодых страусов;
- исследование воздействия физических упражнений на лабораторных крыс;
- управление вращением сервомоторов и шаговых двигателей;
- проверка электронных схем в компьютерах и других электронных устройствах;
- имитация движения в системах виртуальной реальности;
- дистанционное (через Internet) управление и обратная связь с дирижаблями, наполненными гелием.

Цели данной книги

Книга «LabVIEW для всех» призвана помочь вам быстро и просто начать работу в среде LabVIEW, что скоро сделает вас разработчиком-экспертом. В книге приводится много примеров и упражнений для демонстрации различных методик программирования, предлагаются дополнительные источники информации о LabVIEW и описываются сферы его наиболее эффективного использования. Вы можете открыть, просмотреть, запустить или модифицировать любой виртуальный прибор, приведенный на компакт-диске, сопровождающем эту книгу. CD также содержит демонстрационную версию LabVIEW, возможности которой практически совпадают с возможностями коммерческой версии, но ограниченную по времени работы 30 днями.

Предполагается, что вы уже обладаете навыками работы с компьютером и базовыми знаниями о его операционной системе. Если этих знаний у вас недостаточно, то вам следует ознакомиться с руководствами пользователя компьютера и закрепить полученные знания на практике. Например, вы должны знать, как пользоваться системой меню программ и операционной системы, как открывать и сохранять файлы, как создавать резервные копии ваших данных (*backup copies*) и как пользоваться графическим указателем – мышью.

После прочтения этой книги и выполнения упражнений вы сможете легко и быстро реализовывать следующие (и еще многие другие) операции:

- создавать программы LabVIEW, именуемые *виртуальными приборами* (ВП);
- использовать разнообразные способы отладки программ;

- применять как встроенные функции LabVIEW, так и библиотечные ВП;
- создавать и сохранять собственные ВП, чтобы использовать их в качестве виртуальных подприборов – подпрограмм (ВПП);
- создавать оригинальные графические интерфейсы пользователя;
- сохранять свои данные в файлы и отображать их на графиках;
- создавать программы, применяющие интерфейсы канала общего пользования (GPIB) и последовательного порта RS-232;
- создавать приложения, использующие встраиваемые платы ввода/вывода (*plug-in DAQ boards*);
- использовать встроенные функции анализа для обработки данных;
- повышать скорость и эффективность ваших LabVIEW-программ;
- применять расширенные методики программирования с применением локальных и глобальных переменных и узлов свойств;
- публиковать данные в Internet с помощью HTML-публикации LabVIEW или технологии DataSocket;
- использовать LabVIEW для создания измерительных и управляющих приложений.

Книга «LabVIEW для всех» послужит хорошим стартовым руководством для создания оригинальных программ обработки и анализа данных. Книга разделена на две части: «Основы» и «Дополнительные разделы».

В первой части содержится 9 глав, которые научат вас основам программирования на языке G в LabVIEW. Вторая часть поделена на 6 глав, призванных усовершенствовать ваши навыки работы с LabVIEW и познакомить с полезными методиками и стратегиями оптимизации алгоритмов. Мы предлагаем вам тщательно и подробно изучить первую часть – «Основы» – для приобретения базовых знаний; а затем, особенно если у вас мало времени, просмотреть материал второй части и выбрать для чтения тот материал, который действительно понадобится в практической работе.

В обеих частях материал излагается таким образом, чтобы облегчить изучение LabVIEW:

- в разделах «Обзор», «Задачи» и «Основные термины» описываются основные концепции, изучаемые в данной главе;
- прочие разделы содержат обсуждение представленных тем;
- разделы «Упражнения» служат для усвоения и закрепления информации на практике;я
- разделы «Итоги» суммируют теоретические знания, полученные при изучении главы.

Часть I. Основы

Глава 1 описывает пакет LabVIEW и знакомит с некоторыми его особенностями и сферами применения.

Глава 2 содержит обзор концепции виртуального прибора: каким образом осуществляется сбор экспериментальных данных, управление приборами и анализ

данных с помощью LabVIEW. Вы также познакомитесь с историей развития LabVIEW и несколькими примерами его использования в технике.

В главе 3 вы узнаете о среде разработки LabVIEW, включая составные части ВП, об окне контекстной помощи, меню, инструментах, палитрах и виртуальных подприборах.

В главах 4 и 5 рассказывается об основах графического программирования в среде LabVIEW: об использовании элементов управления и индикации (числовых, логических и строковых); о соединении объектов при помощи проводников данных, создании, редактировании, отладке и сохранении виртуальных приборов, создании подприборов и документировании работы. Здесь также будет объяснено, почему язык G считается языком программирования потока данных (dataflow programming language).

Глава 6 описывает базовые структуры программирования: цикл по условию, цикл с фиксированным числом итераций, сдвиговые регистры, структуру варианта, структуру последовательности и узел Формула. В ней также рассказывается, как осуществить привязку ВП ко времени (timing).

В главе 7 вы узнаете, как пользоваться двумя важными структурами данных – массивами и кластерами, а также функциями LabVIEW для работы с ними.

В главе 8 подробно рассматриваются разнообразные типы графиков в LabVIEW, а также их использование для информативного динамического отображения данных. Здесь также описывается тип данных «осциллограмма» (waveform).

Глава 9 посвящена строковым типам данных, функциям обработки строк, таблиц. Здесь вы также научитесь сохранять и считывать данные из файла, используя подпрограммы верхнего уровня файлового ввода/вывода LabVIEW.

Часть II. Дополнительные сведения о LabVIEW

В главе 10 подробно рассматривается процесс сбора данных, а также работа с интерфейсами канала общего пользования (КОП) и последовательного порта RS-232. Вы познакомитесь с теорией этих процессов и описанием некоторого необходимого оборудования вместе с кратким руководством по аббревиатурам и сокращениям, принятым в измерительной технике. В этой главе также описывается процедура установки программного обеспечения аппаратных средств сбора данных.

Глава 11 посвящена использованию LabVIEW для получения данных при помощи многофункциональных плат ввода/вывода (plug-in DAQ boards); здесь кратко рассматриваются способы коммуникации с другим измерительным оборудованием при помощи КОП и протокола RS-232.

В главе 12 изучаются некоторые дополнительные возможности, такие как локальные и глобальные переменные, узлы свойств, преобразование типов данных, работа с ActiveX, библиотеками DLL и т.д.

Глава 13 показывает, каким образом конфигурируется поведение и вид ВП при помощи настройки его опций и как осуществить доступ к элементам управления передней панели с помощью клавиатуры.

Глава 14 описывает коммуникационные возможности LabVIEW и содержит разделы о публикации данных в Internet, об обмене информацией при помощи протокола

DataSocket и взаимодействии через стандартные протоколы типа TCP/IP. В ней также содержится введение в механизм сервера виртуального прибора.

В главе 15 излагаются некоторые полезные методики программирования. Их вы можете использовать для оптимизации и совершенствования программ, в том числе для повышения их быстродействия, снижения объема потребляемой памяти, более легкой переносимости на другие платформы.

В главе 16 вы научитесь хорошему стилю программирования в LabVIEW, получите несколько полезных советов – например, как улучшить внешний вид ВП путем импорта рисунков и как использовать редактор элементов управления (Control Editor).

В конце книги приводится список терминов и приложение, в котором описываются некоторые ресурсы, призванные помочь в работе с пакетом LabVIEW. Дается перечень дополнительных программных библиотек, которые существенно расширяют возможности LabVIEW.

Условные обозначения

Полужирным шрифтом обозначается название виртуального прибора, функций, пунктов меню и палитры, а также входные и выходные параметры ВП. Например, «Выберите функцию **Узел свойств** из палитры **Управление приложением**».

Курсивом помечаются основные термины или важные идеи. Например, «*Ссылка на элемент управления* является объектом, который указывает на элемент управления или отображения в LabVIEW и позволяет управлять его свойствами».

Шрифтом `Courier` обозначается текст или символы, которые вы вводите с клавиатуры, а также файлы и пути их размещения. С помощью этого шрифта отображаются и фрагменты кода, примеры программирования, примеры синтаксиса, сообщения и ответы на сообщения, которые будут появляться на экране компьютера. Например, «В текстовом окне введите `c:\data\datafile.txt` в качестве имени файла».

Внимание. Этот знак отмечает информацию, на которую нужно обратить особое внимание.

Осторожно! Указывает на место возможных ошибочных действий или на специальную информацию, предостерегающую от серьезной ошибки.

Полезный совет. Предлагает вашему вниманию советы и подсказки, как сделать что-то более эффективно.

Замечание об указании путей к файлам

В зависимости от платформы используются различные соглашения об указании путей к файлам. Например, в ОС Windows принято записывать путь в виде `X:\LabVIEW\Mine.llb\Bingo.vi`. Тот же путь в ОС MacOS 9.x выглядел бы как `Hard Drive Name:LabVIEW:Mine.llb:Bingo.vi`. На Linux-машине у вас будет что-то вроде `/usr/labview/mine.llb/bingo.vi`. В данной книге вместо точного обозначения пути применяется директория LabVIEW по умолчанию, где вы можете найти пример, о котором идет речь. Для облегчения написания ссылки мы будем использовать стандарт обозначения операционной системы Windows; если вы работаете с ОС MacOS 9.x или Linux, задействуйте в качестве разделителей двоеточие или обратный слэш.

Что нового во втором издании книги

«LabVIEW для всех» оказалась первой книгой, предназначенной для начинающих пользователей. С момента ее выхода несколько издательств, в том числе Prentice Hall, выпустили более дюжины книг по соответствующей тематике. Второе издание «LabVIEW для всех» переработано под усовершенствованную версию среды LabVIEW 6i и имеет следующие особенности:

- исправлены и обновлены «экранные снимки» (screenshots) приборов, панелей и т.д., обновлены примеры и упражнения с учетом нового интерфейса LabVIEW 6i;
- обновлен текст, описывающий свойства LabVIEW 6;
- обновлен подраздел о сборе данных и управлении приборами с указанием особой роли оболочки NI Measurement and Automation Explorer (MAX);
- добавлена глава, посвященная коммуникационным возможностям LabVIEW, в которой описываются Internet-технологии, DataSocket, сервер виртуального прибора, публикация в Internet;
- обсуждаются новые возможности LabVIEW: тип данных «осциллограмма», построение трехмерных графиков, генерация отчетов и т.д.

Инструкции по установке LabVIEW

Если вы располагаете полной версией LabVIEW и испытываете трудности по ее установке, вы можете изучить документацию, идущую в комплекте с программным продуктом, либо обратиться в службу технической поддержки National Instruments. Если к настоящему моменту у вас нет LabVIEW, используйте демонстрационную версию LabVIEW, которая содержится на компакт-диске, сопровождающем эту книгу. Срок действия демо-версии – 30 дней.

Для работы с упражнениями вам понадобится каталог `Everyone` с компакт-диска. Лучше всего скопировать содержимое этого каталога на жесткий диск вашего

компьютера с тем, чтобы иметь возможность сохранять результаты самостоятельной работы.

Минимальные требования к системе для работы со средой LabVIEW

Основной исполняемый модуль демо-версии LabVIEW такой же, как и у полной версии, так что системные требования практически одинаковы.

Windows XP/2000/NT/Me/9x

- при использовании Windows NT 4.0 обязательно наличие Service Pack 3 или выше;
- минимум 32 Мб ОЗУ (рекомендуется 64 Мб);
- 65 Мб свободного дискового пространства при минимальной установке LabVIEW, 200 Мб при полной установке[1];
- процессор класса Pentium-166 и выше.

MacOS

- MacOS версии 7.6.1 или выше;
- 32 Мб ОЗУ (рекомендуется 64 Мб);
- 100 Мб свободного дискового пространства при минимальной установке LabVIEW, 225 Мб при полной установке[1];
- процессор PowerPC.

Linux

- Linux-ядро версии 2.0.x или выше;
- дистрибутив Linux с библиотекой GNU C Library версии 2.0.5 или выше (glibc2 или libc.so.6), например:
  - RedHat Linux 5.0 или выше;
  - SuSE Linux 6.0 или выше;
  - SuSE Linux 5.3 с установленной библиотекой shlibs6-98.9.25-0 RPM;
  - Caldera Open Linux 1.3 или выше;
  - Debian Linux 2.0 или выше;
- 32 Мб ОЗУ (рекомендуется 64 Мб);
- размер своп-файла 32 Мб;
- 65 Мб свободного дискового пространства при минимальной установке LabVIEW, 150 Мб при полной установке[1];
- сервер XWindows System;
- процессор Pentium-166 или выше.

[1]Для установки драйверов устройств потребуется дополнительное место.

Sun

- Solaris версии 2.5.1 или выше;
- ОЗУ, своп-файл, диск: то же;
- сервер XWindows System;
- процессор SPARC (станция Sun SPARC).

HP-UX

- HP-UX версии 10.20 или выше;
- ОЗУ, своп-файл, диск: то же;
- сервер XWindows System;
- процессор PA-RISC (станция Hewlett Packard 9000 Series 700).

Общие требования

LabVIEW использует специальный каталог для хранения временных файлов (temporary files). Некоторые из этих файлов могут иметь достаточно большой размер, поэтому следует зарезервировать еще несколько мегабайтов на жестком диске под каталог временных файлов. По умолчанию LabVIEW применяет в качестве такого каталога в Windows папку, заданную переменной окружения TEMP, а в MacOS создает временные файлы внутри системной Корзины (Trash Can).

Приобретение LabVIEW

Если вы хотите приобрести LabVIEW, обратитесь в региональное представительство National Instruments:

Российская Федерация, 119361, г. Москва,
ул. Озерная, 42, офис 1101
Тел./факс: +7 (095) 783-6851, 783-6852
Электронная почта: ni.russia@ni.com
Internet: http://ni.com/russia
http://www.labview.ru

Благодарности

Как и моя предыдущая книга, «Internet-приложения в LabVIEW» («Internet Applications in LabVIEW»), книга, которую вы держите в руках, создана при содействии множества моих друзей и коллег.

Бернард Гудвин (Bernard Goodwin) – редактор этой книги в издательстве Prentice Hall, убедивший меня написать третью по счету рукопись для Prentice Hall. Его поддержка сделала эту книгу реальностью.

Лиза Уэллс (Lisa Wells) – мой друг и соавтор в первом издании, которая была терпелива и мудра в изнурительном процессе написания книги. Выражаю ей свою

благодарность за поддержку и оригинальный материал, предоставленный для первого издания. Без ее творческого вклада эта книга определенно не имела бы того успеха у читателей, который мы наблюдаем.

Рави Маравар (Ravi Marawar) – сотрудник National Instruments, который не только предоставил много исходного материала для книги, включая иллюстрации и даже аппаратуру ввода/вывода сигналов, но также оказал неоценимое содействие при оформлении и издании книги. Хочу выразить ему свою признательность.

Мои коллеги по компании Rayodyne: Билл Шварц (Bill Schwarz), Майкл Такер (Michael Tucker), Джордж Кастл (George Castle), Шерри Арнольд (Sherry Arnold), Холи Тондр (Holley Tondre), Стивен Мэттисон (Steven Mattison), Грэм Клагли (Graeme Cloughley), Кевин Шмайссер (Kevin Schmeisser) и остальные участники Стаи Крыс (Rat Pack) – вы, ребята, лучшая команда, с которой стоит работать. Благодарю вас за воодушевление и поддержку.

Моя благодарность и признательность также моим друзьям: Полу и Уне Дэвис (Paul and Una Davis), Трэвису и Лесли Хайнс (Travis and Leslie Hines), Дэвиду Тэйлору (David Taylor), Тристану Д'Артаньяну (Tristan D'Artangan), Джастину и Эмили Харгрэйв (Justin and Emily Hargrave), Джиму и Мишель Хасбрук (Jim and Michelle Hasbrouck), Вивьену Бадилло (Vilian Badillo) и другим друзьям из Hope Chapel.

И наконец, моя сердечная благодарность и любовь членам семьи: жене Стефани (Stephanie) и очаровательным детям Мэйв (Maeve), Эйдан (Aidan) и Рэчел (Rachel). Моя глубокая благодарность также Автору авторов, Иисусу Христу, с коим пребывает свобода и жизнь.

Основы

Обзор

Добро пожаловать в мир LabVIEW! В этой главе вы познакомитесь с основными концепциями LabVIEW, с его возможностями и с тем, каким образом он сделает вашу жизнь проще.

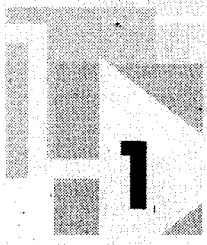
ЗАДАЧИ

- Ответить на вопрос: «Что же такое LabVIEW?»
- Изучить, что понимается под терминами «графический язык программирования» и «программирование потока данных»
- Внимательно рассмотреть вводные примеры
- Получить представление о среде разработки LabVIEW

ОСНОВНЫЕ ТЕРМИНЫ

- LabVIEW
- Виртуальный прибор (ВП)
- Поток данных (Dataflow)
- Графический язык программирования
- Лицевая панель (Front panel)
- Блок-диаграмма (Block diagram)
- Иконка (Icon)
- Соединительная панель (Connector Pane)
- Инструментальная линейка (Toolbar)
- Палитра (Palette)
- Иерархия (Hierarchy)

Что же такое LabVIEW?



1.1. Что такое LabVIEW и что он может для меня сделать?

Прежде чем вы приступите к изучению пакета, возможно, вам будет интересно узнать, что такое LabVIEW, что вы можете с ним делать и что он может сделать для вас. LabVIEW (Laboratory Virtual Instrument Engineering Workbench – среда разработки лабораторных виртуальных приборов) является *средой программирования*, с помощью которой вы можете создавать приложения, используя графическое представление всех элементов алгоритма, что отличает ее от обычных *языков программирования*, таких как C, C++ или Java, где программируют, используя текст. Однако LabVIEW представляет собой значительно большее, чем просто алгоритмический язык. Это среда разработки и исполнения приложений, предназначенная для исследователей – ученых и инженеров, для которых программирование является лишь частью работы. LabVIEW функционирует на компьютерах, работающих под управлением всех распространенных операционных систем: Windows, MacOS, Linux, Solaris и HP-UX.

Мощный графический язык программирования LabVIEW позволяет в сотни раз увеличить производительность труда. Создание законченного приложения с помощью обычных языков программирования может отнять очень много времени – недели или месяцы, тогда как с LabVIEW требуется лишь несколько часов, поскольку пакет специально разработан для программирования различных измерений, анализа данных и оформления результатов. Так как LabVIEW имеет гибкий графический интерфейс и прост для программирования, он также отлично подходит для моделирования процессов, презентации идей, создания приложений общего характера и просто для обучения современному программированию.

Измерительная система, созданная в LabVIEW, имеет бóльшую гибкость по сравнению со стандартным лабораторным прибором, потому что она использует многообразие возможностей современного программного обеспечения. И именно

вы, а не изготовитель оборудования, определяете функциональность создаваемого прибора. Ваш компьютер, снабженный встраиваемой измерительно-управляющей аппаратной частью, и LabVIEW составляют полностью настраиваемый *виртуальный прибор* для выполнения поставленных задач. С помощью LabVIEW допустимо создать неооходимый тип виртуального прибора при очень малых затратах по сравнению с обычными инструментами. При необходимости вы можете внести в него изменения буквально за минуты.

Рис. 1.1. Листовая печь зонной плавки создана в аэрокосмической промышленности США и предназначена для высокотемпературной обработки полупроводниковых материалов в условиях микрогравитации на борту самолета NASA KC-135, двигающегося по специальной параболической траектории. Ее работа автоматизирована с помощью LabVIEW через промышленный компьютер на базе Macintosh

LabVIEW создан для облегчения работы по программированию ваших задач. Для этой цели имеется расширенная библиотека функций и готовых к использованию подпрограмм, которые реализуют большое число типичных задач программирования и тем самым избавляют вас от рутинной возни с указателями, распределением памяти и прочего шаманства, присущего традиционным языкам программирования. В LabVIEW также содержатся специальные библиотеки виртуальных приборов для *ввода/вывода данных со встраиваемых аппаратных* средств (data acquisition – DAQ), для работы с *каналом общего пользования* (КОП, General Purposes Interface Bus – GPIB), управления устройствами через последовательный порт *RS-232*, программные компоненты для анализа, представления

и сохранения данных, взаимодействия через сети и Internet. Библиотека *анализа* (Analysis) содержит множество полезных функций, включая генерирование сигнала, его обработку, различные фильтры, окна, статистическую обработку, регрессионный анализ, линейную алгебру и арифметику массивов.

Благодаря своей графической природе LabVIEW – это пакет эффективного отображения и представления данных. Выходные данные могут быть показаны в любой форме, какую вы пожелаете. Диаграммы, графики стандартного вида, а также оригинальная пользовательская графика (user-defined graphics) составляют лишь малую часть возможных способов отображения выходных данных.

Программы LabVIEW легко портировать на другие платформы: вы можете создать приложение на Macintosh, а затем запустить его в Windows, для большинства приложений практически ничего не меняя в программе. Вы увидите, что приложения, созданные на LabVIEW, качественно улучшают работу во многих сферах деятельности человека – как в автоматизации технологических процессов, так и в биологии, сельском хозяйстве, психологии, химии, физике, образовании и множестве других.

1.1.1. Потоки данных и язык графического программирования

Разработка приложений в среде LabVIEW отличается от работы в средах на основе C или Java одной очень важной особенностью. Если в традиционных алгоритмических языках программирование основано на вводе текстовых команд, *последовательно* образующих *программный код*, в LabVIEW используется *язык графического программирования*, где алгоритм создается в графической иконной форме (pictorial form), образующей так называемую *блок-диаграмму* (block-diagram), что позволяет исключить множество синтаксических деталей. Применяя этот метод, вы можете сконцентрировать внимание лишь на программировании потока данных; упрощенный синтаксис теперь не отвлекает вас от анализа самого алгоритма. На рис. 1.2 и 1.3 показан простой пользовательский интерфейс LabVIEW и реализующий его код.

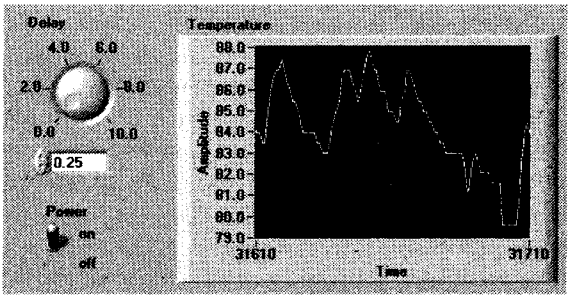


Рис. 1.2. Интерфейс пользователя

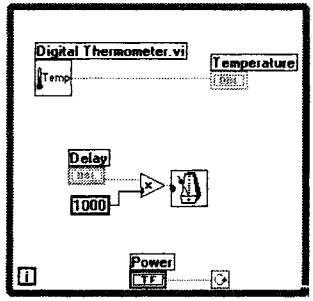


Рис. 1.3. Графический код

В LabVIEW используется терминология, рисунки иконок и основные идеи, знакомые ученым и инженерам. Этот язык базируется на графических символах, а не на тексте для описания программируемых действий. Основополагающий для LabVIEW принцип *потока данных* (dataflow), согласно которому функции выполняются лишь тогда, когда они получают на вход необходимые данные, однозначно определяет порядок исполнения алгоритма. Вы можете освоить LabVIEW при небольшом или даже отсутствующем опыте традиционного программирования, хотя знание его принципов было бы весьма полезным.

1.1.2. Как работает LabVIEW

Программы LabVIEW называются *виртуальными приборами* (ВП, virtual instruments – VI), так как они функционально и внешне подобны реальным (традиционным) приборам. Однако они столь же подобны программам и функциям на популярных языках программирования, таких как C или Basic. Здесь и далее мы будем называть программы LabVIEW *виртуальными приборами* или *ВП*, причем вне зависимости от того, соотносится их вид и поведение с реальными приборами или нет.

Виртуальный прибор состоит из трех основных частей:

- *лицевая панель* (Front Panel) представляет собой интерактивный пользовательский интерфейс виртуального прибора и названа так потому, что имитирует лицевую панель традиционного прибора. На ней могут находиться ручки управления, кнопки, графические индикаторы и другие *элементы управления* (controls), которые являются средствами ввода данных со стороны пользователя, и *элементы индикации* (indicators) – выходные данные из программы. Пользователь вводит данные, используя мышь и клавиатуру, а затем видит результаты действия программы на экране монитора;
- *блок-диаграмма* (Block Diagram) является исходным программным кодом ВП, созданным на языке графического программирования LabVIEW, G (Джей). Блок-диаграмма представляет собой реально исполняемое приложение. Компонентами блок-диаграммы являются: *виртуальные приборы более низкого уровня*, *встроенные функции* LabVIEW, *константы* и *структуры управления* выполнением программы. Для того чтобы *задать поток данных* между определенными объектами или, что то же самое, *создать связь* между ними, вы должны нарисовать соответствующие *проводники* (wires). Объекты на лицевой панели представлены на блок-диаграмме в виде соответствующих *терминалов* (terminals), через которые данные могут поступать от пользователя в программу и обратно;
- для того чтобы использовать некоторый ВП в качестве подпрограммы (подприбора) в блок-диаграмме другого ВП, необходимо определить его *иконку* (icon) и *соединительную панель* (connector). Виртуальный прибор, который применяется внутри другого ВП, называется *виртуальным*

подприбором (ВПП, SubVI), который аналогичен *подпрограмме* в традиционных алгоритмических языках. Иконка является однозначным *графическим представлением* ВП и может использоваться в качестве *объекта* на блок-диаграмме другого ВП. Соединительная панель представляет собой механизм передачи данных в ВП из другой блок-диаграммы, когда он применяется в качестве подприбора – ВПП. Подобно аргументам и параметрам подпрограммы, соединительная панель определяет входные и выходные данные виртуального прибора.

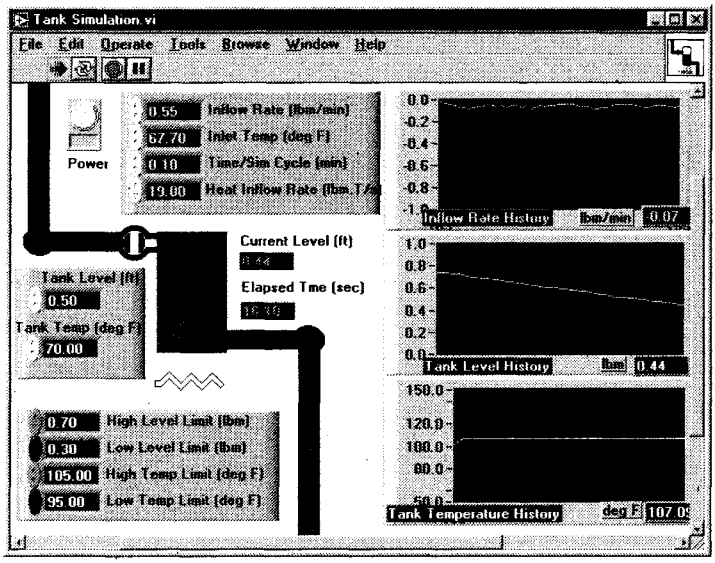


Рис. 1.4

Виртуальные приборы являются *иерархическими* и *модульными* (modular). Вы можете использовать их как самостоятельные приложения (top-level programs), так и в качестве виртуальных подприборов. Согласно этой логике, LabVIEW следует концепции *модульного программирования* (modular programming). Вначале вы разделяете большую прикладную задачу на ряд простых подзадач. Далее создаете виртуальные приборы для выполнения каждой из подзадач, а затем объединяете эти ВП на блок-диаграмме прибора более высокого уровня, который выполняет прикладную задачу в целом.

Технология модульного программирования очень хороша, потому что вы можете работать с каждым ВПП по отдельности, что облегчает отладку приложения. Более того, ВПП низкого уровня часто выполняют задачи, типичные для нескольких приложений, и поэтому могут использоваться независимо во многих отдельных приложениях.

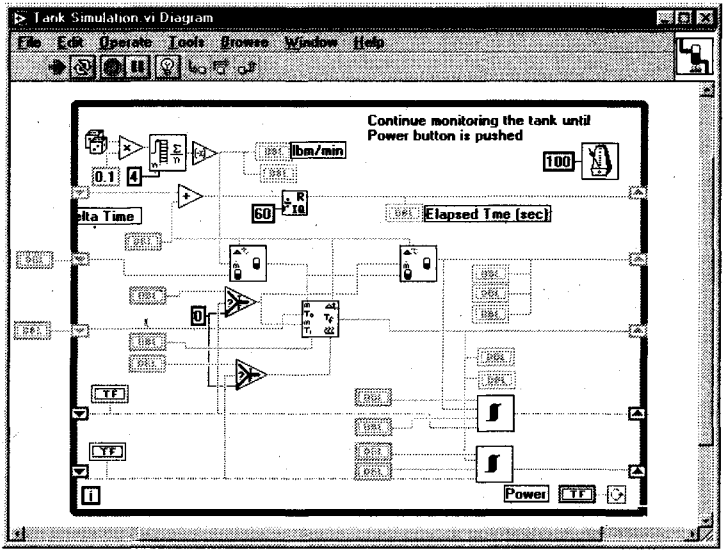


Рис. 1.5

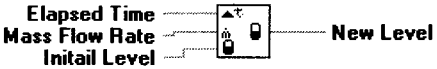


Рис. 1.6.

В табл. 1.1 приведен ряд основных терминов LabVIEW и их общепринятых эквивалентов для традиционных языков программирования.

Таблица 1.1. Термины LabVIEW и их эквиваленты для традиционных языков программирования

LabVIEW	Традиционные языки программирования
Виртуальный прибор (ВП)	Программа
Функция	Функция или метод
Виртуальный подприбор (ВПП)	Подпрограмма, объект
Лицевая панель	Интерфейс пользователя
Блок-диаграмма	Программный код
G или LabVIEW	C, C++, Java, Basic и др

1.2. Демонстрационные примеры

Хорошо, на этом этапе довольно чтения. Чтобы почувствовать, как работает LabVIEW, надо открыть и запустить несколько готовых примеров на LabVIEW.

Пользуетесь ли вы полной или демонстрационной версией LabVIEW, просто запустите ее! Убедитесь, что у вас есть доступ к каталогу `Everyone` на компакт-диске или на жестком диске, как указано во введении, – этот каталог содержит все примеры и решения упражнений этой книги. При запуске LabVIEW появится начальное диалоговое окно. Для открытия примера (или любого другого существующего ВП) выберите опцию **Открыть ВП** (Open VI) и укажите место расположения интересующего ВП на диске.

По умолчанию во всей книге манипуляции мышью сопровождаются щелчком ее левой клавиши (если на вашей мыши более чем одна), кроме случаев, когда мы явно предложим щелкать правой. На компьютерах Macintosh вам придется щелкать мышью, удерживая нажатой клавишу <command> на клавиатуре. Вообще клавиша <Ctrl> на ПК с Windows эквивалентна клавише <command> Macintosh, клавише <meta> на станциях Sun и <alt> на компьютерах с Linux/UNIX/HP.

1.2.1. Упражнение 1.1: демонстрация измерения температуры

Откройте и запустите ВП под названием **Temperature System Demo.vi** следующим образом:

1. Запустите LabVIEW.
2. Выберите **Открыть** (Open) из меню **Файл** (File) или щелкните мышью по кнопке **Открыть ВП** (Open VI) в начальном диалоговом окне LabVIEW.
3. Откройте каталог (или папку) `Everyone` двойным щелчком мыши. Выберите библиотеку `CH1.LLB` (обратите внимание: библиотека в LabVIEW, отмеченная расширением .llb, является виртуальным каталогом, включающим только виртуальные приборы). Просмотреть содержимое такой библиотеки вы можете только через LabVIEW – для операционной системы она является единичным файлом. Затем откройте **Temperature System Demo.vi** (если у вас полная версия LabVIEW, вы также можете обнаружить этот пример в `examples/apps/tempsys.llb`). После загрузки вы увидите лицевую панель **Демонстрация измерения температуры** (Temperature System Demo), показанную на рис. 1.7. На ней находятся цифровые элементы управления, логические (булевские)

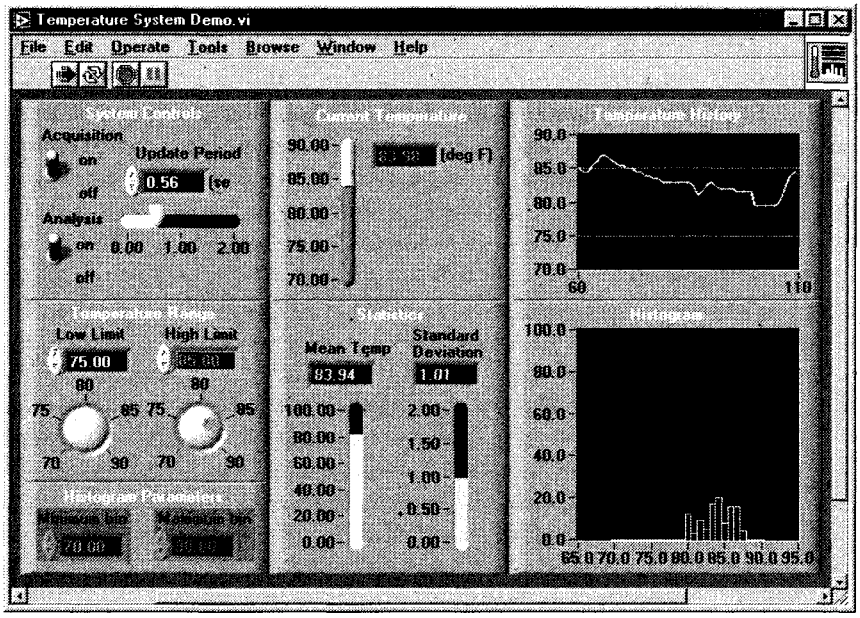


Рис. 1.7 Окно лицевой панели ВП ***Temperature System Demo***

переключатели, регуляторы в виде ползунков и ручек, кнопки, диаграммы, графики и индикатор температуры.


Run button


Run button (active)


Abort button

4. Запустите ВП, щелкнув по кнопке **Запуск** (Run). Кнопка поменяет внешний вид, указывая, что ВП выполняется. *Инструментальная линейка* (Toolbar), представляющая собой ряд иконок в верхней части экрана, также меняется, поскольку функции редактирования не могут быть использованы во время работы ВП.

   Обратите внимание, что кнопка **Прервать** (Abort) на инструментальной линейке становится активной. Вы можете нажать ее, чтобы экстренно прервать выполнение ВП.

   Прибор **Temperature System Demo.vi** имитирует приложение отслеживания (мониторинга) температуры. Виртуальный прибор измеряет температуру и показывает мгновенные отсчеты на температурном индикаторе, а временную зависимость – на графике. Хотя отсчеты температуры в этом примере генерируются случайным образом, вы легко можете изменить программу и измерять реальную температуру с помощью реальных аппаратных средств. Ползунковый регулятор **Период опроса** (Update Period) задает частоту получения новых отсчетов температуры. LabVIEW также показывает верхний и нижний пределы температур на графике; вы можете изменить эти пределы, используя ручки **Диапазон**

Температур (Temperature Range). Если текущая величина температуры выходит за установленные пределы, то рядом с термометром загораются светодиоды.

Виртуальный прибор работает до тех пор, пока вы не щелкнете мышью по переключателю **Опрос** (Acquisition), переведя его в состояние *выключено* (off). Вы также можете включать и выключать режим анализа данных. Секция **Статистика** (Statistics) показывает текущее среднее и стандартное отклонение, а на графическом индикаторе **Гистограмма** (Histogram) вычерчивается частота появления каждого значения температуры.

Изменение значений

Operational tool

5. Используйте курсор, который во время работы виртуального прибора работает как *инструмент управления* («палец»), для того, чтобы изменить величины верхнего и нижнего температурных пределов. Выделите прежнее значение двойным щелчком мыши, затем введите новое значение и щелкните мышью на кнопке ввода данных, которая расположена рядом с кнопкой запуска на линейке инструментов. Можно также, удерживая нажатой кнопку мыши, переместить ручку управления на нужное значение.

Enter button

6. Измените значение, задаваемое ползунковым регулятором **Период опроса** (Update Period). Наведите указатель инструмента управления на ползунок, нажмите кнопку мыши и, не отпуская, переместите ползунок в новое положение. Установить значение ползункового элемента управления при помощи инструмента управления можно, щелкнув по выбранной точке на отсчетной линейке для мгновенного перемещения ползунка в это положение либо щелкая мышью по кнопкам прокрутки для последовательного изменения значения (значение увеличивается, если щелкать по кнопке прокрутки «стрелка вверх», и уменьшается, если щелкать по «стрелке вниз»). С той же целью можно щелкнуть мышью в поле цифрового индикатора (digital display) данного регулятора и ввести нужное число с клавиатуры.

Даже когда значение на индикаторе какого-либо управляющего элемента изменилось, это еще не значит, что новая величина поступила в программу. Для подтверждения ввода необходимо нажать кнопку **Ввод** на панели инструментов, <ctrl>+<enter> на клавиатуре или щелкнуть мышью в любой свободной точке окна

7. Попробуйте регулировать и другие элементы управления подобным образом.
8. Остановите виртуальный прибор щелчком по переключателю **Опрос** (Acquisition).

Исследование блок-диаграммы

Блок-диаграмма, изображенная на рис. 1.8, представляет собой завершенное приложение на LabVIEW. В данный момент вы не обязаны понимать назначение и взаимосвязь всех элементов блок-диаграммы – о них речь пойдет в дальнейшем; наша задача состоит в ознакомлении с блок-диаграммой. Если вы уже понимаете эту диаграмму, у вас не возникнет никаких проблем при прочтении первой части книги.

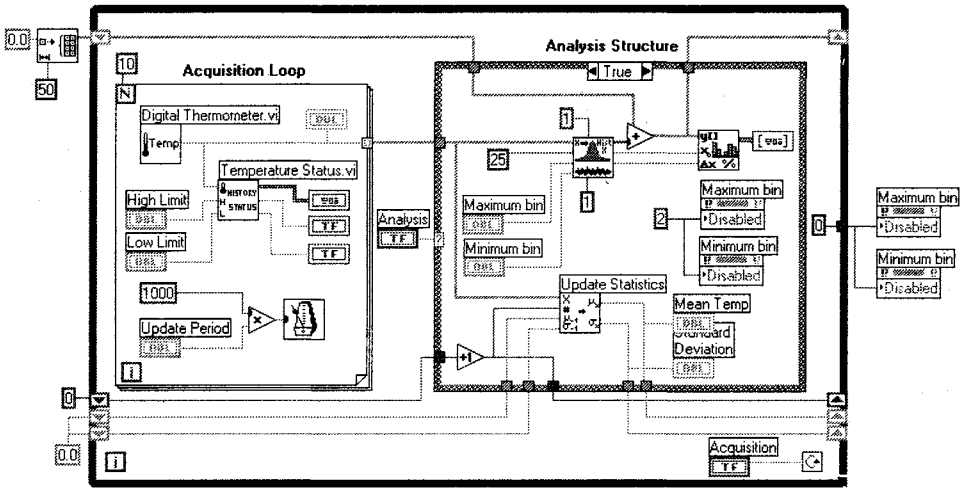


Рис 1.8

9. Откройте блок-диаграмму ВП **Демонстрация измерения температуры** (Temperature System Demo.vi), выбрав пункт **Показать диаграмму** (Show Diagram) из меню **Окно** (Window).
10. Изучите различные объекты в окне диаграммы. Не пугайтесь их большого количества: все они будут объяснены шаг за шагом далее в книге.
11. Используя пункт **Показать контекстную справку** (Show Context Help) из меню **Справка** (Help), откройте плавающее окно контекстно-ориентированной справки. Посмотрите, как изменяется окно контекстной справки при наведении курсора на различные объекты блок-диаграммы. Если объект является встроенной функцией LabVIEW или ВПП, в этом окне появится также описание ее входных и выходных контактов.

Иерархия

Сила LabVIEW заключена в иерархической структуре его программ – виртуальных приборов. Создав ВП, вы можете использовать его как подприбор (ВПП)

в блок-диаграмме ВП более высокого уровня и создавать столько уровней иерархии, сколько пожелаете. Чтобы оценить эту гибкость, посмотрите на подприборы, содержащиеся в ВП **Демонстрация измерения температуры**.

History status

12. Откройте подприбор **Температурное состояние** (Temperature Status) двойным щелчком по его иконке. При этом вызывается лицевая панель, изображенная на рис. 1.9.

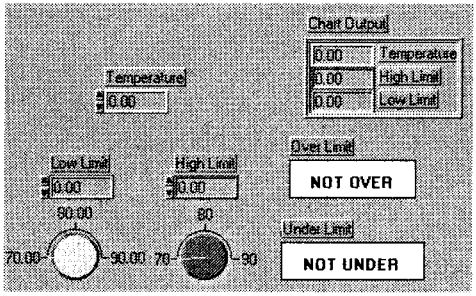


Рис. 1.9

Иконка и соединительная панель

Иконка и соединительная панель обеспечивают графическое представление ВП и определение его параметров, которые необходимы в случае использования данного ВП в качестве подприбора или функции в другом виртуальном приборе. Они расположены в верхнем правом углу окна лицевой панели. Иконка графически представляет виртуальный прибор на блок-диаграммах других виртуальных приборов, в то время как терминалы соединительной панели показывают места подсоединения проводников входных и выходных данных. Эти терминалы аналогичны параметрам подпрограммы или функции в традиционных языках программирования. Для каждого элемента управления или индикатора на лицевой панели необходим один терминал, через который он передает данные в виртуальный прибор (или получает данные от него). Иконка находится в правом верхнем углу лицевой панели, поверх соединительной панели, до тех пор, пока вы не выберете опцию просмотра соединительной панели.

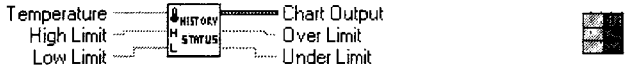


*Рис. 1.10. Иконка **Temperature Status** и соединительная панель*

Используя виртуальные подприборы, вы делаете блок-диаграммы модульными и более управляемыми. Модульность упрощает создание, понимание и отладку ВП. Кроме того, допустимо создать один ВПП, чтобы реализовать функцию, вызываемую из нескольких различных виртуальных приборов.

Теперь запустите ВП высокого уровня, чтобы его окно и окно ВПП **Температурное состояние** (Temperature Status) были видимыми. Обратите внимание, как изменяются значения в ВПП каждый раз, когда его вызывает главная программа.

13. Выберите пункт **Закрыть** (Close) из меню **Файл** для ВПП **Температурное состояние**. Никаких изменений сохранять не нужно.
14. Выберите пункт **Закрыть** из меню **Файл** для ВП **Демонстрация измерения температуры** (Temperature System Demo.vi), не сохраняя никаких изменений.

Выбор опции **Закрыть** на панели блок-диаграммы закроет только окно блок-диаграммы. Выбор опции **Закрыть** на лицевой панели закрывает весь ВП: и лицевую, и диаграммную панели.

1.2.2. Упражнение 1.2: пример измерения частотной характеристики

Этот пример измеряет частотную характеристику некоего «черного ящика». Функциональный генератор подает синусоидальный сигнал на вход «черного ящика», а цифровой мультиметр измеряет напряжение на его выходе.

В этом примере в качестве «черного ящика» взят полосовой фильтр, который пропускает сигналы только в определенном интервале частот)

Хотя данный виртуальный прибор использует ВПП, лишь имитирующие работу генератора и цифрового мультиметра, при небольшой модификации ВП реальные приборы могут быть легко подсоединены к реальному «черному ящику» для получения реальных данных. Вам потребуются лишь ВПП для управления сбором данных, передачей данных и команд по КОП и последовательному порту для ввода или вывода реальных данных вместо использования имитированных.

Откройте виртуальный прибор, запустите его и понаблюдайте за работой.

1. Выберите пункт **Открыть** (Open) из меню **Файл**, чтобы открыть виртуальный прибор, или щелкните мышью по кнопке **Открыть ВП** (Open VI), если у вас на мониторе исходное диалоговое окно LabVIEW.

2. Выберите каталог EVERYONE, а в нем – библиотеку CH1.LLB. Затем дважды щелкните по виртуальному прибору **Frequency Response.vi**. (Если у вас полная версия LabVIEW, этот пример находится в директории examples/apps/freqresp.llb.) Появится лицевая панель, изображенная на рис. 1.11.

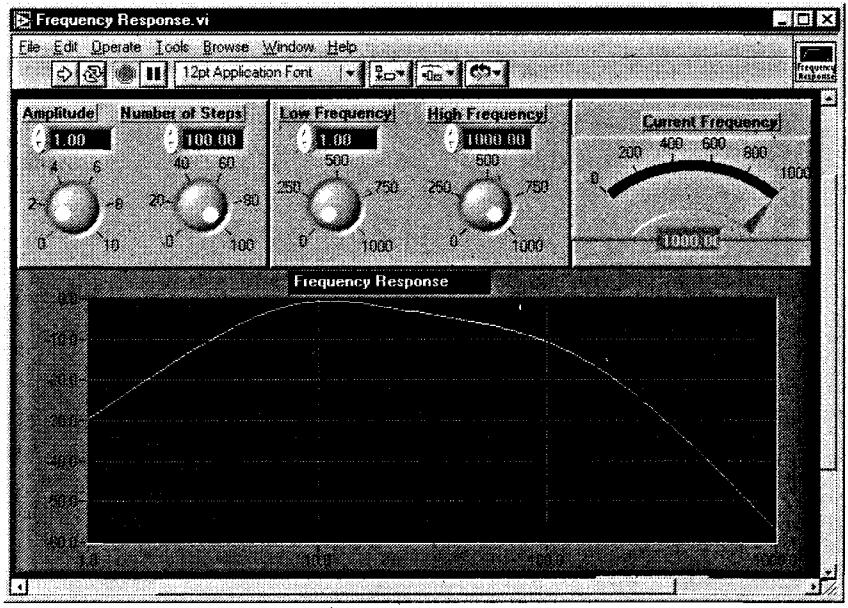


Рис. 1.11


Run button

3. Запустите ВП щелчком по кнопке **Пуск** (Run). Вы можете установить амплитуду входного синусоидального сигнала и число шагов, необходимых виртуальному прибору для определения частотной характеристики с помощью соответствующих элементов управления, и запустить виртуальный прибор с новыми параметрами. Также разрешается задать пределы изменения частоты генератора с помощью ручек управления **Нижняя частота** (Low Frequency) и **Верхняя частота** (High Frequency). Поэкспериментируйте с этими настройками, и вы увидите, какое воздействие они оказывают на выходные данные с «черного ящика».
4. Откройте и изучите блок-диаграмму, выбрав опцию **Показать диаграмму** (Show Diagram) из меню **Окно** (Window).
5. Закройте ВП, выбрав пункт **Закрыть** (Close) из меню **Файл**.

Эти упражнения помогут вам понять идею и организацию программирования в LabVIEW. Используя LabVIEW, вы увидите, что создание серьезного приложения (и его отладка) осуществляется буквально одним щелчком! Читайте дальше и учитесь!

1.3. Итоги

LabVIEW является мощным и гибким программным пакетом для получения, обработки и анализа данных. Для создания программ, называемых *виртуальными приборами (ВП)*, в LabVIEW применяется язык графического программирования. Пользователь взаимодействует с программой через *лицевую панель*. Каждой лицевой панели соответствует *блок-диаграмма*, которая является исходным кодом виртуального прибора. LabVIEW имеет много встроенных функций для облегчения процесса программирования; компоненты связываются между собой *проводниками*, определяющими пути потока данных в пределах блок-диаграммы.

В каталоге EVERYONE приложенного к книге компакт-диска вы найдете решения ко всем последующим упражнениям. Мы надеемся, что наличие готовых решений не помешает вам поработать над задачами самостоятельно!

1.4. Дополнительные упражнения

1.4.1. Упражнение 1.3: более изящные примеры

В данном упражнении вы познакомитесь с примерами программ, которые поставляются вместе с LabVIEW.

1. Из меню **Справка** (Help) выберите вкладку **Примеры** (Examples). Это активизирует систему помощи LabVIEW в разделе примеров. Щелчок мышью по выбранному примеру загружает указанный виртуальный прибор в LabVIEW.


Run button

3. Запустите программу щелчком мыши по кнопке **Пуск** (Run).
4. После запуска выберите пункт **Показать диаграмму** (Show Diagram) из меню **Окно** (Window), чтобы посмотреть на алгоритм работы данного ВП.
5. Теперь просмотрите и запустите другие виртуальные приборы, чтобы разобраться в системе программирования LabVIEW и понять, что вы можете сделать с ее помощью. Хотя все примеры направлены на обучение, вам следует обратить внимание на несколько довольно интересных разделов: **Демонстрации** (Demonstrations), **Измерительные примеры** (Measurement Examples) и **Интерфейсы ввода/вывода** (I/O Interfaces). Просмотрите все виртуальные приборы, которые вызвали интерес, – вы научитесь многому, даже просто наблюдая, как они работают. Вы также можете свободно изменять и использовать эти примеры

в собственных разработках (учтите, однако, что сохранять модифицированные копии нужно в другом каталоге, чтобы не уничтожить существующие встроенные примеры).

6. Когда вы завершите работу с примером, выберите пункт **Закрыть** (Close) из меню **Файл**, чтобы закрыть каждый ВП. Если сделанные в процессе работы изменения не потребуются вам в дальнейшем, можете не сохранять их.

Обзор

Виртуальные приборы – настоящая находка для любой современной лаборатории. Виртуальный прибор состоит из компьютера, специального праграммного продукта, встраиваемой в компьютер платы, выполняющей те же функции, что и традиционный измерительный прибор; такое же имя носят программы LabVIEW. Поскольку функциональность виртуальных приборов программируется пользователем, они обладают исключительной гибкостью и эффективностью. В этой главе рассказывается о том, как с помощью LabVIEW научить компьютер обмениваться информацией с окружающим миром: проводить измерения, управлять внешними приборами и пересылать данные на другие компьютеры. Сначала это будет очень краткое введение: подробно изучить процедуры ввода/вывода аналоговых и цифровых данных, коммуникации с внешним электроизмерительным оборудованием и сетевые технологии LabVIEW вы сможете во второй части книги. Дополнительно вы познакомитесь с историей развития самого пакета LabVIEW.

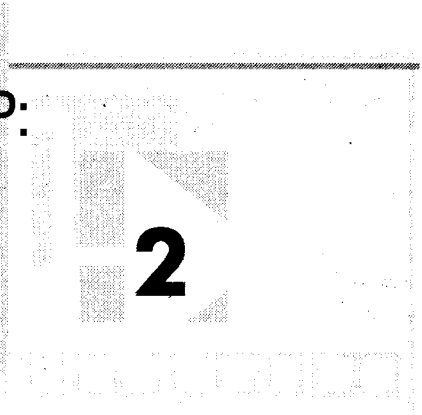
ЗАДАЧИ

- Изучить основы сбора данных и управления приборами
- Описать компоненты типичной системы для сбора данных или управления внешним прибором по каналу общего пользования
- Получить представление о стандартных интерфейсных портах компьютера: последовательном, параллельном и USB
- Рассмотреть функции обработки и анализа введенных данных
- Познакомиться с аппаратными платформами автоматизации PXI и VXI
- Увидеть, как LabVIEW позволяет программировать обмен данными между компьютерами
- Получить представление о некоторых дополнительных библиотеках, расширяющих возможности LabVIEW

ОСНОВНЫЕ ТЕРМИНЫ

- Сбор данных (Data acquisition – DAQ)
- Канал общего пользования – КОП (General Purpose Interface Bus – GPIB)
- Стандарт IEEE 488
- Последовательный порт (Serial port; RS-232)
- PXI
- VXI
- Работа в сети (Networking)
- Библиотека динамических связей (Dynamic Link Library – DLL)
- Узел кодового интерфейса – УКИ (Code Interface Node – CIN)
- ActiveX
- Дополнительная библиотека (Toolkit)

ВИРТУАЛЬНЫЙ ПРИБОР: ПОДКЛЮЧЕНИЕ КОМПЬЮТЕРА К РЕАЛЬНОМУ МИРУ



2.1. Эволюция LabVIEW

В 1983 году компания National Instruments начала поиски способов сокращения времени, необходимого для программирования измерительных систем. В результате появилась концепция виртуального прибора LabVIEW – сочетания интуитивного пользовательского интерфейса лицевой панели с передовой методикой блок-диаграммного программирования, позволяющего создавать эффективные измерительные системы на основе графического программного обеспечения.

Первая версия LabVIEW увидела свет в 1986 году. Она была предназначена только для компьютеров Macintosh. Несмотря на то что Macintosh довольно редко использовались в задачах измерений и автоматизации, графическая оболочка их операционной системы MacOS наилучшим образом соответствовала технологии LabVIEW. Довольно скоро и другие наиболее распространенные операционные системы перешли на графический пользовательский интерфейс и начали поддерживать эту технологию.

К 1990 году разработчики National Instruments полностью переделали LabVIEW, сочетая новые компьютерные технологии с анализом отзывов пользователей. И, что более важно, вторая версия LabVIEW включала компилятор, делающий скорость исполнения виртуальных приборов сравнимой со скоростью выполнения программ, созданных на языке программирования С. Патентное ведомство США (United States Patent Office) выдало National Instruments несколько патентов, признающих новизну технологии LabVIEW.

С появлением новых графических операционных систем, подобных MacOS, компания National Instruments перенесла ставшую уже признанной технологию LabVIEW на другие платформы – персональные компьютеры и рабочие станции. В 1992 году благодаря новой открытой архитектуре появились версии LabVIEW для Windows и Sun.

Третья версия LabVIEW появилась в 1993 году сразу для трех операционных систем: Macintosh, Windows и Sun. Виртуальные приборы версии 3, разработанные на одной из платформ, могли без изменений запускаться на другой. Эта межплатформенная совместимость дала пользователям возможность выбора платформы разработки и уверенность, что созданные ВП будут функционировать и на других платформах (обратите внимание, что это было реализовано за пару лет до появления Java). В 1994 году список платформ, поддерживающих LabVIEW, увеличился и стал включать Windows NT, Power Macs, рабочие станции Hewlett Packard и в 1995 году – Windows 95.

LabVIEW 4 была выпущена в 1996 году и обеспечивала большую гибкость оболочки среды разработки, которая позволила пользователям создавать программы, подходящие типу их деятельности, уровню опыта и навыкам разработки. Кроме этого, LabVIEW 4 включала в себя мощные инструменты редактирования и отладки более совершенных измерительных систем, в том числе обмен данными на основе OLE-технологии и распределенных средств исполнения.

Версии LabVIEW 5 и 5.1 (в 1999 году) продолжают наращивать возможности системы: появляется встроенный Internet-сервер, подсистема динамического программирования и управления (сервер виртуального прибора), интеграция с ActiveX и особый протокол для упрощения обмена данными через Internet – *DataSocket*. Также введена долгожданная возможность *отката действий пользователя* (undo), которая уже присутствовала в большинстве компьютерных программ.

Вышедшей в 2000 году новой версии – LabVIEW 6 (известной также как 6i) – сделали «подтяжку лица»: в нее встроили новый комплект объемных элементов управления и индикации, поскольку в то время компьютерная индустрия обнаружила, что внешний вид программного продукта имеет весьма серьезное значение (чему способствовало появление систем Apple iMac и G4). В LabVIEW 6 воплотилась очень серьезная работа по обеспечению как простого и интуитивного интерфейса среды программирования (особенно для непрограммистов!), так и поддержки множества передовых технологий программирования, например объектно-ориентированного программирования, многопоточности (multithreading), распределенных вычислений (distributed computing) и т.д. И пусть простота графической оболочки LabVIEW не вводит вас в заблуждение: LabVIEW – это инструмент, который является достойным соперником систем программирования C++ или Visual Basic, да еще и превосходит их в удобстве работы, как отмечают тысячи пользователей.

В версии LabVIEW 6.1, вышедшей в 2001 году, было введено *событийно-управляемое* (event-oriented) программирование, удаленное управление LabVIEW через Internet и другие улучшения.

Совершенно особой разновидностью LabVIEW, на которую следует обратить внимание, является *LabVIEW RT*. RT означает *Real Time – реальное время*. LabVIEW RT представляет собой совокупность аппаратного и программного обеспечения, которая позволяет выделять части кода LabVIEW и загружать их для выполнения на отдельном контроллере, работающем под управлением собственной операционной системы реального времени. Таким образом гарантируется, что

выделенные участки LabVIEW-приложения будут выполняться в точно определенные моменты времени, даже если Windows «зависнет» и компьютер перестанет работать.

LabVIEW является мощным инструментом программирования, пригодным для решения практически любых задач, например компьютерного моделирования, тем не менее он чаще всего используется для сбора экспериментальных данных и управления приборами и установками, и поэтому содержит множество виртуальных приборов, разработанных специально для этой цели. Например, LabVIEW может управлять *встраиваемыми многофункциональными устройствами сбора данных* (plug-in data acquisition – DAQ), которые предназначены для ввода и/или вывода аналоговых и цифровых сигналов. Например, вы можете совместно использовать многофункциональные платы и LabVIEW для мониторинга температуры, формирования управляющих сигналов для экспериментальной установки или определения частоты неизвестного сигнала. LabVIEW также обеспечивает передачу команд и данных по каналу общего пользования (КОП) или через стандартный последовательный порт компьютера. Канал общего пользования часто применяется для взаимодействия с осциллографами, сканерами и мультиметрами, а также для дистанционного управления подобными приборами. С помощью программного обеспечения LabVIEW допустимо управлять сложными измерительными системами стандарта VXI, приборами с сетевым интерфейсом Ethernet или через порт USB. Получив со встроенной платы или внешнего прибора массив данных, вы можете использовать множество содержащихся в LabVIEW виртуальных приборов анализа для всесторонней обработки этих данных и их преобразования.

Часто полезен обмен данными не только с измерительными приборами, но и с другими программными продуктами и удаленными компьютерами. В LabVIEW встроены функции, которые упрощают этот процесс, поддерживая несколько сетевых протоколов, вызов внешнего программного кода или динамических библиотек (DLL) и автоматизацию ActiveX. Оставшаяся часть этой главы посвящена обсуждению типичных задач, для решения которых и был создан LabVIEW.

2.2. Что такое сбор данных

Сбор, или ввод/вывод данных (Data Acquisition – DAQ), упрощенно можно определить как процесс измерения реального сигнала, например электрического напряжения, и передачи этой информации в компьютер для обработки, анализа, преобразования и хранения. На рис. 2.1 показаны компоненты типичной системы сбора данных. Человек научился преобразовывать большинство физических явлений в сигналы, которые можно измерять: скорость, температура, влажность, давление, текучесть, pH, пространственное положение, радиоактивность, интенсивность света и т.д. Датчики (иногда говорят «измерительные преобразователи» или «сенсоры») воспринимают действие физических явлений и преобразуют их в электрические сигналы согласно определенным пропорциям. Например, *термопара* (thermocouple) преобразует температуру в электрическое напряжение, которое может быть измерено при помощи аналого-цифрового преобразователя (АЦП).

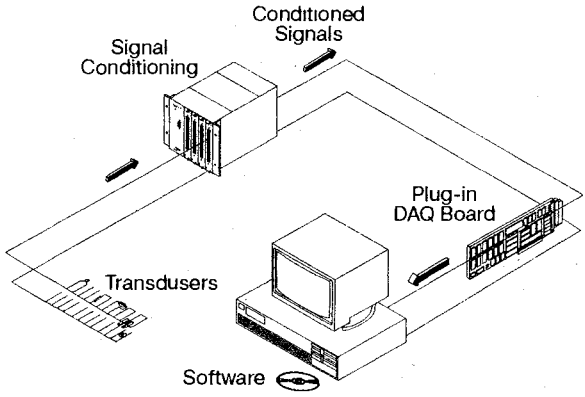


Рис. 2.1. Типичная система сбора данных

Другими примерами датчиков служат тензометрические датчики (strain gauges), расходомеры (flowmeters) и датчики давления, которые измеряют силу, скорость потока и давление соответственно. В каждом случае электрический сигнал напрямую связан с явлением, которое воспринимается датчиком.

Для взаимодействия с датчиками LabVIEW управляет многофункциональными платами ввода/вывода, чтобы считать аналоговые входные сигналы или сформировать аналоговые выходные сигналы, считать и записать цифровые сигналы, может также запрограммировать встроенные в DAQ-платы счетчики для измерения частоты сигналов или генерации последовательности импульсов и т.д. Например, аналоговый входной сигнал (электрическое напряжение) поступает с датчика на установленную в компьютер плату ввода/вывода, которая преобразует напряжение в код и отправляет эту информацию в память для обработки, хранения и других операций.

Не все датчики физических величин имеют форму выходных сигналов, которую плата сбора данных может воспринять непосредственно. Поэтому зачастую требуется *согласование сигнала* (signal conditioning), осуществляемое специальными модулями, также поставляемыми National Instruments. Например, вы хотите ввести и проанализировать сигнал очень высокого напряжения (скажем, молнию) – тогда не забудьте позаботиться о гальванической развязке или изоляции сигнала: в подобном случае ошибки обойдутся очень дорого! Модули согласования сигнала выполняют множество функций: усиление, линеаризация, фильтрация, изолирование и т.п. Не все, но многие измерительные задачи требуют согласования сигнала, поэтому следует обратить внимание на специфику задачи и технические характеристики применяемых датчиков и измерительных преобразователей, чтобы избежать потенциальных ошибок. Кроме того, иногда неправильные данные могут

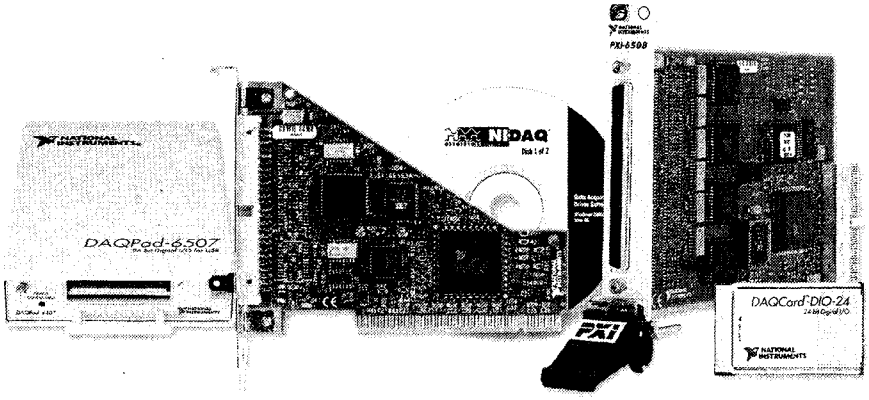


Рис 2 2 Платы и модули ввода/вывода сигналов выпускаются National Instruments в различных модификациях

быть даже хуже выхода оборудования из строя! Шум, нелинейность, перегрузки и т.д. способны безнадежно исказить сигнал и тут LabVIEW вряд ли поможет. Преобразование сигнала очень часто является не вспомогательной, а обязательной задачей, поэтому вначале следует изучить измерительную часть, а уж потом приступать к программированию.

Для получения данных в лаборатории с использованием технологии виртуальных приборов понадобится многофункциональная плата ввода/вывода (DAQ-плата), компьютер с установленной средой LabVIEW и драйверами применяемой платы сбора данных, а также соединение датчика с платой при помощи, например, терминального блока, макетной платы, кабеля или провода. Может также потребоваться оборудование для согласования сигнала – в зависимости от особенностей задачи.

Например, вы хотите измерить температуру. Нужно подключить датчик температуры к каналу аналогового ввода на DAQ-плате компьютера (для этого типа измерений часто требуется предварительное согласование сигнала). Затем, используя *виртуальные приборы сбора данных* (DAQ VIs) LabVIEW, легко снять показания выбранного канала, ввести их в память, отобразить на экране монитора, записать в файл и проанализировать по заданному алгоритму.

Виртуальные приборы сбора данных LabVIEW предназначены только для работы с платами сбора данных National Instruments. Если вы применяете оборудование других фирм, получите у них драйвер под LabVIEW (если таковой имеется) либо используйте DLL-библиотеки или внешний программный код для вызова функций задействованного оборудования в LabVIEW.

2.3. Что такое КОП

Канал общего пользования (КОП, General Purpose Interface Bus – GPIB) был разработан компанией Hewlett Packard в конце 1960 года для обеспечения связи между компьютерами и измерительными приборами. Под каналом понимают способ соединения, с помощью которого компьютеры и приборы обмениваются данными и командами. Канал общего пользования обеспечил необходимые спецификации и протокол для управления процессом передачи. Институт инженеров электротехники и электроники (Institute of Electrical and Electronic Engineers) в 1975 году утвердил GPIB (КОП) в качестве стандарта, который стал известен как *стандарт IEEE 488*. Первоначальной целью создания КОП было обеспечение компьютерного управления устройствами тестирования и измерения. Однако использование КОП довольно быстро расширилось до таких областей, как осуществление связи между компьютерами и управление универсальными измерительными приборами, сканерами и осциллографами.

КОП является цифровой 24-разрядной параллельной шиной. Шина состоит из восьми линий данных (data lines), пяти линий управления шиной (bus management lines) – ATN, EOI, IFC, REN, SRQ, трех линий квитирования (handshaking) и восьми заземленных линий. КОП использует параллельную 8-битовую асинхронную побайтовую схему передачи данных. Другими словами, байты целиком последовательно передаются по шине со скоростью, определяемой самым медленным устройством на шине. Поскольку по КОП данные передаются байтами (1 байт = 8 бит), то пересылаемая информация, или *сообщения* (messages), часто представляются в виде символов ASCII. Вы можете использовать КОП для связи с приборами и устройствами, если компьютер оборудован встроенной платой контроллера КОП (или подключен к выносному модулю КОП, рис. 2.3) и на него установлены соответствующие драйверы.

Допустимо подключить к одной шине КОП несколько компьютеров и приборов. Каждое устройство, в том числе плата-контроллер, должно иметь свой уникальный адрес КОП в диапазоне от 0 до 30, чтобы источник и приемник данных могли однозначно определяться этим номером. Адрес 0 обычно соответствует плате контроллера шины КОП. Приборы, размещаемые на шине, используют адреса от 1 до 30. КОП имеет один контроллер (обычно это плата в компьютере), который осуществляет управление шиной. Для того чтобы передать команды управления и данные, контроллер устанавливает адрес *источника сообщений* (Talker) и одного или нескольких *приемников* (Listeners). Затем строковые данные пересылаются по шине от источника к приемнику (приемникам). Виртуальные приборы КОП в LabVIEW автоматически выполняют процедуры адресации и другие функции управления шиной, избавляя вас от утомительного низкоуровневого программирования. На рис. 2.4 изображена типичная КОП-система.

Использование КОП является одним из способов ввода данных в компьютер, принципиально отличающимся от сбора данных DAQ-платами, несмотря на то, что в обоих случаях используются встраиваемые платы. По специальному протоколу

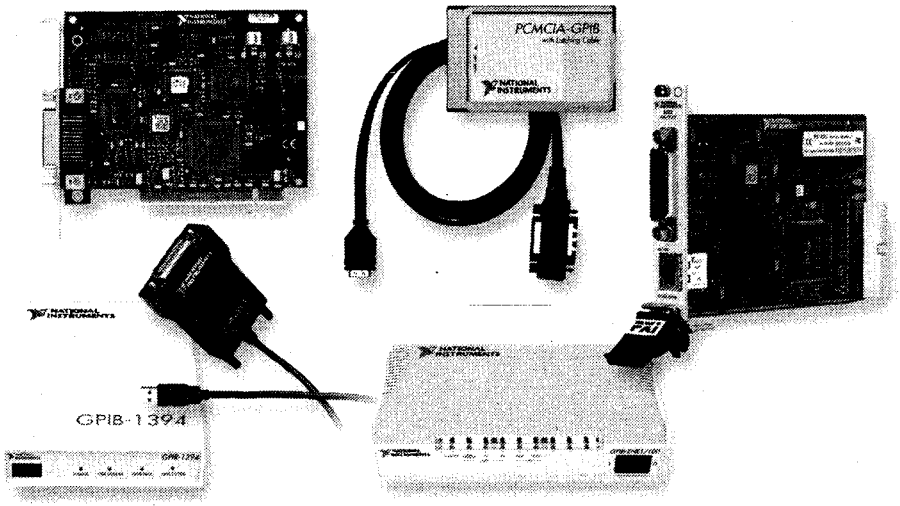


Рис. 2.3 Платы и модули КОП, выпускаемые National Instruments

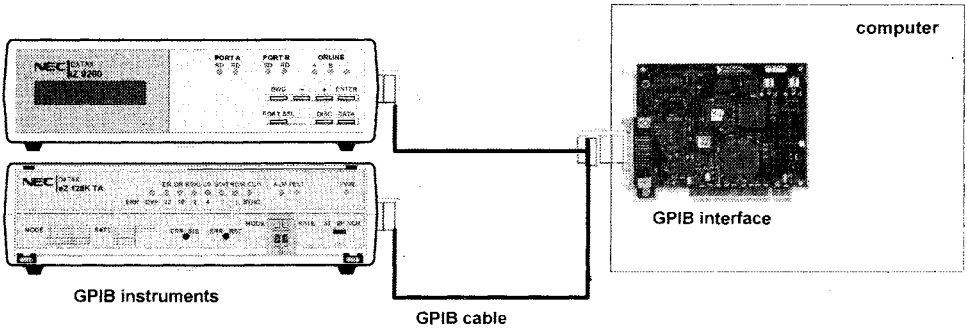


Рис. 2.4. Типичная система с КОП

КОП предписывает другому устройству или прибору выдать полученные им данные, в то время как функции сбора данных подразумевают подключение сигнала напрямую к многофункциональной плате ввода/вывода в компьютере.

Чтобы задействовать КОП как часть измерительной системы, понадобится плата или внешний модуль контроллера КОП, интерфейсный кабель, компьютер с LabVIEW, а также прибор, совместимый с протоколом IEEE 488, с которым будет осуществляться коммуникация (или другой компьютер, также имеющий плату КОП). Кроме того, необходимо установить на компьютере драйвер

КОП в соответствии с указаниями, приведенными в LabVIEW или в руководстве по работе с платой.

Виртуальные приборы КОП в LabVIEW предназначены только для работы с платами КОП National Instruments. Если вы применяете контроллеры других фирм, получите у них драйвер под LabVIEW (если таковой имеется), либо используйте DLL-библиотеки или внешний программный код для вызова функций задействованного оборудования в LabVIEW. Как и в случае с платами ввода/вывода (DAQ), это может оказаться непростой задачей!

Мы более подробно обсудим сбор данных и работу с КОП в главах 10 и 11.

2.4. Связь через последовательный порт

Другим популярным средством обмена информацией является последовательный интерфейс. С помощью этого интерфейса осуществляется передача данных между компьютерами или связь компьютера с периферийным устройством типа программируемого прибора путем использования встроенного *последовательного порта* (стандарты RS-232 и RS-422). При последовательном соединении передатчик посылает 1 бит информации за единицу времени через единственную линию связи на приемник. Вы можете пользоваться этим методом, когда скорость передачи данных невелика или когда необходимо передать информацию на большие расстояния. Данный метод является более медленным и менее надежным по сравнению с КОП, но в этом случае вам не нужна плата-контроллер в компьютере и прибору не требуется совместимость со стандартом IEEE 488. На рис. 2.5 изображена типичная система с использованием последовательной передачи данных.

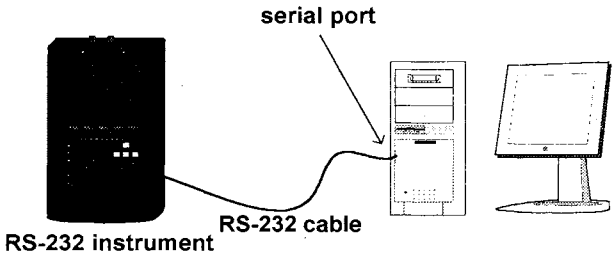


Рис 2.5. Типичная система с последовательным интерфейсом

Последовательная связь весьма удобна, поскольку большинство персональных компьютеров имеют один или два встроенных последовательных порта, позволяющих посылать и получать данные без приобретения какой-либо специальной

аппаратной части. Хотя в настоящее время большинство компьютеров имеют также встроенные порты *универсальной последовательной шины* (Universal Serial Bus – USB), протокол обмена по этой шине является более сложным и ориентирован на работу с периферийными устройствами компьютера, а не на связь. Последовательный протокол (RS-232, 422 и 485) считается устаревшим по сравнению с USB, но по-прежнему широко используется во многих промышленных устройствах.

Большинство приборов с КОП также имеют встроенные последовательные порты. Однако, в отличие от КОП, к последовательному порту можно подключить лишь одно устройство, что ограничивает его применимость для многих задач. Серьезным недостатком связи через последовательный порт является очень маленькая скорость и отсутствие возможностей проверки ошибок. Но последовательная связь имеет свою сферу применения (в силу простоты и дешевизны), поэтому в LabVIEW встроены виртуальные приборы для такого протокола – VISA Serial, – которые содержат готовые к использованию функции для работы с последовательным портом. Если у вас имеется кабель и устройство с последовательным портом, значит, есть все необходимое для создания системы последовательной связи!

2.5. Применения в реальном мире: почему мы анализируем?

Как только данные поступили в компьютер, вы начинаете их каким-либо образом обрабатывать. Модемы, высокоскоростные процессоры цифровых сигналов с плавающей точкой приобретают все большую значимость в системах реального времени и в аналитических системах. Вот лишь некоторые сферы применения LabVIEW для анализа информации: обработка биомедицинских данных, распознавание и синтез речи, цифровая обработка звука и изображений.

Необходимость использования функций и библиотек анализа в лабораторных исследованиях очевидна: необработанные данные, полученные с плат ввода/вывода или приборов с интерфейсом КОП, редко сразу содержат полезную информацию. Чаще всего приходится сначала преобразовывать сигнал, убирать шумовые искажения, корректировать аппаратные ошибки или компенсировать воздействие на сигнал факторов окружающей среды – температуры, давления, влажности. На рис. 2.6 приведен исходный сигнал, наглядно показывающий необходимость функций обработки и анализа.

Анализируя и обрабатывая оцифрованный сигнал, вы можете извлечь полезную информацию из шума и представить ее в более понятной форме, чем исходные данные. Обработанные данные должны выглядеть примерно так, как показано на рис. 2.7.

Метод программирования блок-диаграмм LabVIEW и широкий спектр виртуальных приборов анализа позволяют упростить разработку аналитических приложений. Образец блок-диаграммы на рис. 2.8 иллюстрирует концепцию анализа с помощью LabVIEW.

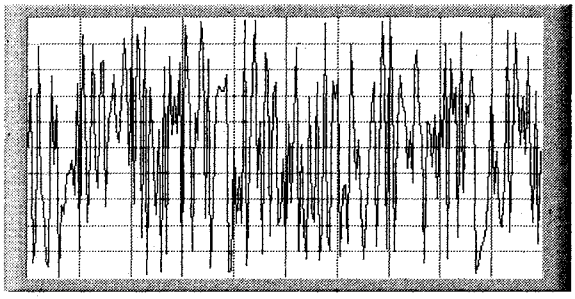


Рис. 2.6

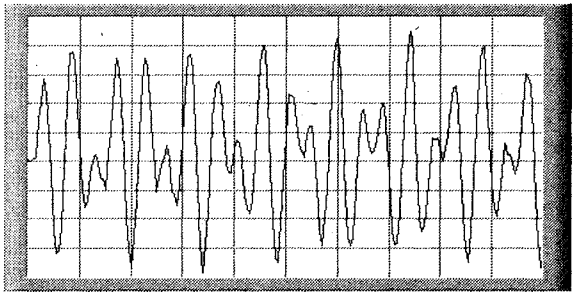


Рис. 2.7

Поскольку функции анализа LabVIEW представляют распространенные методики анализа данных в виде отдельных виртуальных приборов, допустимо соединять их для реализации более сложных алгоритмов. Вместо того чтобы заботиться о правильности реализации типовых алгоритмов обработки и анализа, что неизбежно при написании программ на традиционных языках программирования, в LabVIEW можно сконцентрироваться исключительно на вашем оригинальном алгоритме. Библиотека виртуальных приборов анализа LabVIEW является достаточно мощной для создания опытными разработчиками сложных приложений с использованием цифровой обработки сигналов (Digital Signal Processing – DSP), цифровых фильтров, статистической обработки данных или численных методов и в то же время все реализованные функции достаточно просты для новичков, впервые столкнувшихся со сложными вычислениями.

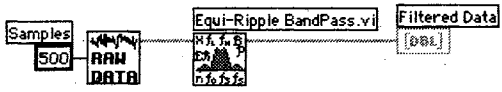


Рис. 2.8

Виртуальные приборы анализа эффективно обрабатывают массивы данных, представленных в цифровом виде. Эти приборы применяют в следующих областях обработки данных:

- генерация сигналов;
- цифровая обработка сигналов;
- цифровая фильтрация;
- сглаживающие окна;
- статистический анализ;
- подбор кривой по точкам и аппроксимация;
- линейная алгебра;
- численные методы;
- анализ, основанный на измерениях.

2.6. Немного о PXI и VXI

Две другие аппаратные платформы, о которых вам следует узнать, – PXI и VXI.

PXI (compactPCI eXtension for Instrumentation – расширение шины CompactPCI для использования в инструментальных системах) представляет собой модульную аппаратную платформу, активно использующую возможности шины CompactPCI (модификация шины PCI) и программных технологий Microsoft Windows. Типичная конфигурация включает в себя *PXI шасси* (chassis), в котором

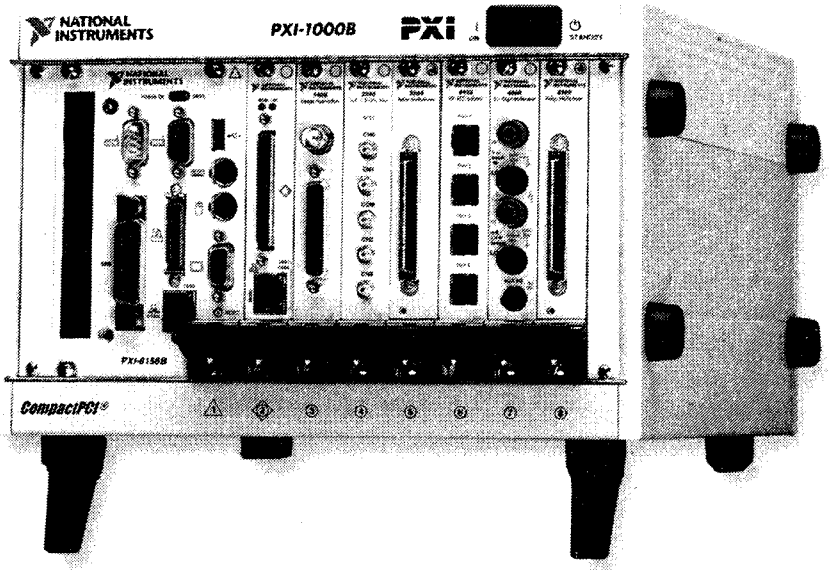


Рис. 2.9. Промышленный компьютер платформы PXI

находится собственный персональный компьютер – *контроллер* (controller) – и дополнительные слоты для установки любых типов измерительных модулей: аналогового ввода, ввода изображений, звука, релейных, интерфейсов КОП и VXI и т.д. Компактность, надежность и гибкость делает эту инструментальную платформу привлекательной для большого числа применений. Очень удобно и эффективно программировать измерительные системы с PXI при помощи LabVIEW. Кроме того, вы можете использовать версию LabVIEW для работы в жестком реальном времени (LabVIEW RT) на контроллере PXI для создания более надежной и устойчивой системы.

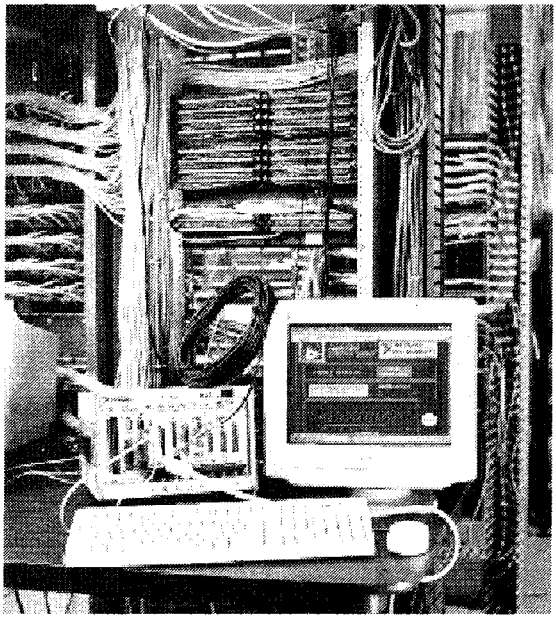


Рис. 2.10. Система PXI

VXI (VMEbus eXtension for Instrumentation – расширение шины VME для использования в инструментальных системах) – другой стандарт оборудования для модульных измерительных систем (instrument-on-a-card). Появившийся в 1987 году и основанный на шине VME (IEEE 1014), VXI является более высокой по классу и более дорогостоящей системой, чем PXI. VXI представляет собой базовый аппаратный блок (mainframe) со слотами, содержащими модульные инструменты на съемных платах. Многие фирмы предлагают целый набор приборов и базовых блоков различной вместимости. Кроме этого, вы можете использовать в VXI-системе модули стандарта VME. VXI широко применяется при традиционных измерениях и в оборудовании автоматического тестирования (automated test equipment – ATE). VXI также успешно используется при сборе и анализе данных

в научно-исследовательских и промышленных разработках, где требуется очень большое количество измерительных и управляющих каналов (сотни и тысячи).

Марка *VXIplug&play* используется для обозначения продукции VXI, которая имеет стандартизированные характеристики, дополнительные к базовым. Приборы, совместимые с VXIplug&play, обеспечиваются стандартизированной программной и драйверной поддержкой и унифицированной процедурой установки оборудования, что повышает эффективность их работы и облегчает задачу создания программ. Программная поддержка VXI в LabVIEW полностью совместима с требованиями VXIplug&play.

2.7. Коммуникации

В некоторых сферах использования виртуальных приборов необходимо обмениваться данными с другими программами на этом же компьютере или в локальной сети. Во многих случаях было бы удобно передать информацию через Internet и предоставить другим пользователям возможность получать данные с вашей системы или даже управлять ею через Internet.

LabVIEW имеет встроенные возможности (такие, как Internet-сервер и инструмент Internet-публикации) и ряд функций, которые делают названные задачи достаточно простыми. Эти виртуальные приборы обеспечивают коммуникации через локальную сеть или Internet. LabVIEW использует протокол DataSocket для передачи данных по сети, вызывает и создает библиотеки динамических связей (DLL) или внешний программный код, а также поддерживает автоматизацию ActiveX. Используя дополнительный набор инструментов – *Enterprise Connectivity Toolset*, LabVIEW может также взаимодействовать с большинством баз данных SQL, таких как Oracle, SQL Server и Access.

2.7.1. Подключение к Internet

LabVIEW имеет несколько встроенных инструментов, которые обеспечивают доступ к вашим виртуальным приборам и данным через Internet. С помощью Internet-сервера LabVIEW вы разрешаете другим людям просматривать лицевую панель вашего виртуального прибора (без какого-либо дополнительного программирования).

Применяя *Enterprise Connectivity Toolset*, вы можете связываться с удаленными системами через электронную почту, по протоколам FTP и Telnet, а также использовать более мощные возможности Internet.

Проигрыватель LabVIEW (LabVIEW Player) является бесплатной утилитой, которую можно загрузить с сервера http://ni.com/labview/. Посредством LabVIEW Player легко использовать (то есть наблюдать и запускать, но не редактировать и создавать) виртуальные приборы без необходимости установки или приобретения полной версии. Проигрыватель LabVIEW – удобное средство «обмена» виртуальными приборами через Internet, подобно тому как документы в формате Adobe PDF способны открыть любые пользователи, на компьютере которых установлен Adobe Acrobat Reader.

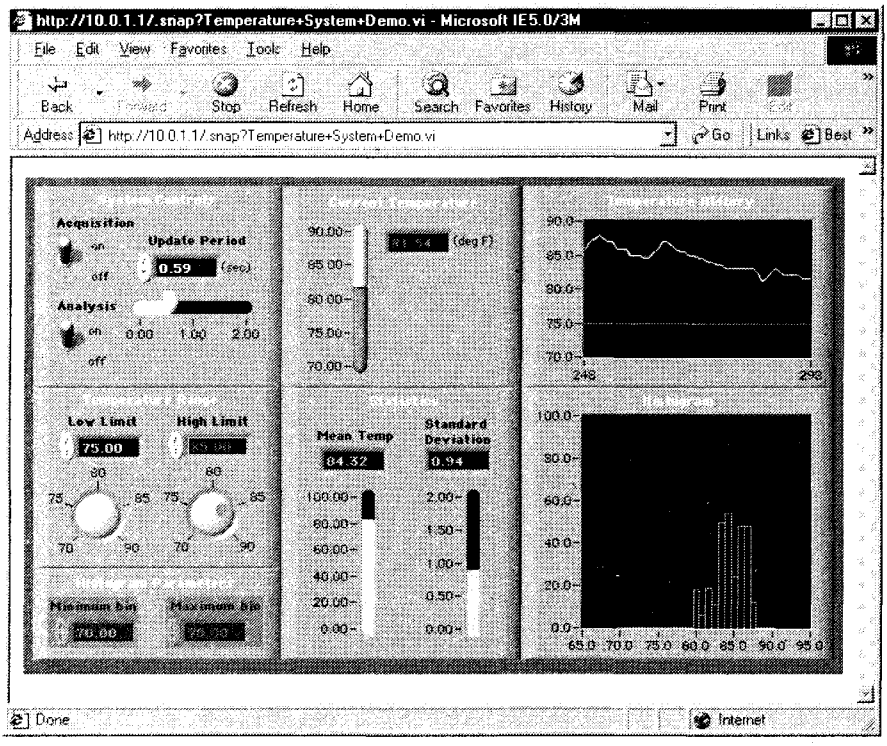


Рис. 2.11. Изображение лицевой панели ВП в Internet Explorer, сгенерированное Web-сервером LabVIEW

2.7.2. Работа в сети

В нашем понимании *работа в сети* (networking) означает взаимодействие между множеством процессов, которые обычно (но не всегда) исполняются на различных компьютерах. Взаимодействие (коммуникация) может происходить как в *локальной вычислительной сети* (Local Area Network – LAN), так и в Internet. Одним из основных применений программных продуктов при работе в сети является разрешение одному или нескольким приложениям использовать возможности (службы) другого приложения. В дополнение к средствам Internet-публикации в LabVIEW допустимо задействовать сетевые функциональные возможности для связи с другими приложениями или другой программой LabVIEW.

Для осуществления взаимодействия между процессами должен применяться общий язык связи, называемый протоколом. LabVIEW предлагает следующие протоколы:

- DataSocket – протокол связи, запатентованный корпорацией National Instruments, для обмена данными через сеть. Преимуществом его является простота в использовании;

- TCP/IP (основной протокол большинства сетей, включая Internet);
- UDP.

Кроме этих сетевых протоколов, независимых от типа операционной системы, LabVIEW также поддерживает некоторые устаревшие и поэтому редко используемые протоколы, такие как DDE (Windows), AppleEvents и PPC (MacOS).

2.7.3. ActiveX

ActiveX, технология компании Microsoft, является архитектурой, основанной на компонентах, для создания приложений, которые могут взаимодействовать друг с другом. ActiveX базируется на ранних технологиях, таких как OLE. Использование ActiveX позволяет одному приложению разделять права доступа к части программного кода (или компоненту) с другим приложением. Например, поскольку Microsoft Word является компонентом ActiveX, вы можете вставлять документ Word (и управлять им) в другую программу, совместимую с ActiveX, например в виртуальный прибор LabVIEW. LabVIEW поддерживает автоматизацию ActiveX и способен содержать в себе компоненты ActiveX. Если вам непонятно, о чем пошла речь, не волнуйтесь. ActiveX достаточно сложный расширенный инструмент. Более подробно о нем вы узнаете в главе 14.

2.7.4. Библиотеки динамических связей и узел кодового интерфейса

Благодаря своей гибкости LabVIEW может вызывать и создавать процедуры внешнего кода, или *библиотеки динамических связей* (Dynamic Link Library – DLL), и интегрировать эти процедуры в исполняемые программы. Библиотека динамических связей представляет собой набор функций, к которым приложение обращается во время выполнения программы, а не при компиляции. В LabVIEW также присутствует специальная структура блок-диаграммы, называемая *узлом кодового интерфейса* (Code Interface Node – CIN), служащая для присоединения обычного текстового программного кода к виртуальному прибору. LabVIEW вызывает исполняемый код во время работы узла, передает в него входные данные и возвращает данные после исполнения кода в блок-диаграмму. Аналогично используется функция *вызов библиотечной функции* (Call Library Function) для вызова DLL при работе в Windows. Из LabVIEW вы также можете скомпилировать свои виртуальные приборы в библиотеку динамических связей, которую будут применять другие приложения и системы программирования (например, C++).

Большинство приложений не нуждается в использовании CIN или DLL. Хотя компилятор LabVIEW обычно создает код, который достаточно быстро выполняется при решении ряда задач, CIN и DLL являются полезными инструментами для решения критических по времени задач, которые требуют большого количества манипуляций с данными, или при условии, что у вас уже много разработанного кода на традиционных языках программирования. Они также нужны для решения задач,

например выполнения системных утилит, для которых в LabVIEW не предусмотрено функций.

2.8. Набор дополнительных инструментов LabVIEW

Вы можете задействовать следующие специальные библиотеки дополнительных функций для LabVIEW с целью увеличения гибкости и эффективности в ряде специфических областей применения. Наиболее широко используемыми библиотеками и инструментами являются следующие:

- Application Builder;
- Enterprise Connectivity Toolset;
- Internet Toolkit;
- Database Connectivity Toolset;
- SPC (Statistical Process Control) Toolkit;
- Motion Control Toolkit;
- Sound & Vibration Analysis Toolset.

Многие библиотеки инструментов поставляются компанией National Instruments; другие можно приобрести у фирм, чаще всего входящих в Альянс National Instruments. Если у вас есть очень специфическая задача и вы хотите узнать, не решена ли она уже кем-либо, обращайтесь в техническую поддержку National Instruments или на многочисленные информационные форумы пользователей LabVIEW (см. приложение).

2.9. Итоги

Встроенные в LabVIEW функции облегчают взаимодействие с внешними устройствами, поэтому нет необходимости писать специальные программы. Виртуальные приборы LabVIEW способны работать с различными типами оборудования для сбора данных или обмена информацией, такими как: встраиваемые многофункциональные платы ввода/вывода, платы КОП, последовательный порт компьютера и аппаратная часть PXI и VXI. Допустимо использовать платы ввода/вывода (DAQ), управляемые LabVIEW, для ввода или формирования аналоговых сигналов, цифровых сигналов, а также осуществлять операции со счетчиками-таймерами. LabVIEW также может взаимодействовать с приборами через КОП (если компьютер оснащен платой контроллера КОП) или управлять инструментальными системами стандартов PXI и VXI. Если у вас нет специальной аппаратной части, то LabVIEW свяжется с внешними устройствами через последовательный порт компьютера.

Функции анализа в LabVIEW облегчают обработку и преобразование данных в компьютере. Вместо рутинной реализации сложных алгоритмов или написания собственного низкоуровневого кода вы просто задействуете необходимые встроенные функции LabVIEW.

Вы можете использовать возможности Internet для публикации лицевых панелей ваших виртуальных приборов в Internet, обмена виртуальными приборами при использовании Проигрывателя LabVIEW, а также взаимодействовать с другими программами и компьютерами в сетях посредством таких протоколов, как DataSocket или TCP/IP. LabVIEW поддерживает технологию ActiveX для взаимодействия с другими программами, а также может вызывать и создавать библиотеки динамических связей.

Если вы хотите расширить возможности LabVIEW, приобретите дополнительные библиотеки виртуальных приборов для выполнения специфических задач.

В следующей главе вы научитесь основам программирования на LabVIEW. Так что будьте готовы создавать виртуальные приборы!

Обзор

В этой главе вы откроете для себя среду LabVIEW и изучите, как три основные составляющие виртуального прибора – лицевая панель, блок-диаграмма и иконка/соединительная панель – работают вместе. Когда эти главные компоненты собраны и настроены, ваш виртуальный прибор готов к работе как самостоятельное приложение или в качестве подприбора в составе иного ВП. Вы познакомитесь с оболочкой среды разработки LabVIEW – всплывающими и ниспадающими меню, плавающими палитрами и подпалитрами, панелью инструментов, а также с системой помощи. В завершение мы обсудим значимость и эффективность виртуальных подприборов и причины, по которым их следует использовать.

ЗАДАЧИ

- Научиться пользоваться лицевой панелью, блок-диаграммой и иконкой/соединительной панелью
- Увидеть разницу между элементами управления и индикаторами
- Научиться распознавать на блок-диаграмме терминалы элементов управления и индикаторов
- Понять идею программирования потока данных
- Познакомиться с системой меню в LabVIEW
- Научиться пользоваться панелью инструментов, палитрой инструментов, палитрой элементов управления и индикации, палитрой функций и подпалитрами
- Понять, почему система справки и помощи может стать самым важным союзником
- Понять, что такое виртуальный подприбор и почему он так полезен
- Выполнить упражнения, чтобы почувствовать, как работает LabVIEW

ОСНОВНЫЕ ТЕРМИНЫ

- Элемент управления (Control)
- Индикатор (Indicator)
- Проводник (Wire)
- Виртуальный подприбор (SubVI)
- Терминал (Terminal)
- Узел (Node)
- Поток данных (Dataflow)
- Всплывающее меню (Pop-up menus)
- Панель инструментов (Toolbar)
- Палитра (Palette)
- Подпалитра (Subpalette)
- Окно помощи (Help window)

СРЕДА LabVIEW: СОЗДАНИЕ СВОЕГО РАБОЧЕГО МЕСТА

3

3.1. Лицевые панели

Говоря простым языком, *лицевой панелью* (front panel) называется окно, через которое пользователь взаимодействует с программой. Когда вы запускаете виртуальный прибор, лицевая панель должна быть открыта для того, чтобы можно было ввести данные в выполняющуюся программу. С другой стороны, лицевая панель является окном просмотра результатов выполнения ВП. На рис. 3.1 показана типичная лицевая панель виртуального прибора LabVIEW.

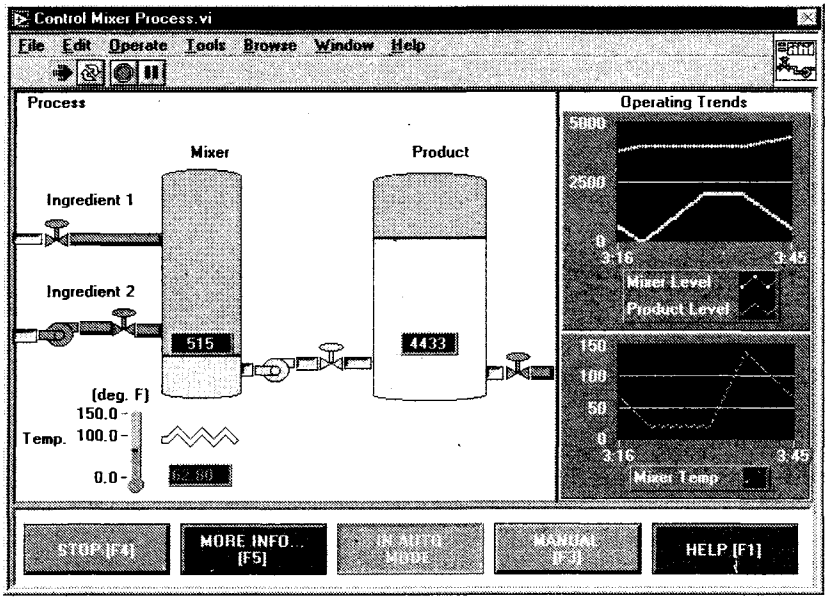


Рис. 3.1. Лицевая панель ВП LabVIEW

3.1.1. Элементы управления и индикаторы

Лицевая панель в основном состоит из совокупности *элементов управления* (controls) и *индикаторов* (indicators). Элементы управления имитируют типичные органы управления, которые имеются у любого измерительного прибора, например кнопки и переключатели. Элементы управления позволяют пользователю ввести данные; они передают данные в блок-диаграмму виртуального прибора. Индикаторы отображают выходные данные, являющиеся результатом выполнения программы. С помощью нижеприведенных аналогий вам будет проще понять суть элементов управления и индикаторов виртуального прибора:

Элементы управления = данные, вводимые пользователем = терминалы-источники данных

Индикаторы = данные, выводимые пользователю = приемники данных

Эти понятия абсолютно не взаимозаменяемы, так что хорошенько разберитесь с их различием.

Вы перетаскиваете элементы управления и индикаторы на лицевую панель, сначала выбрав их в *подпалитре* плавающей палитры **Элементы управления** (Controls), а затем разместив в нужном месте. После появления объекта на лицевой панели вы можете легко поменять его размер, форму, положение, цвет и другие атрибуты.

3.2. Блок-диаграммы

Окно *блок-диаграммы* содержит исходный графический код виртуального прибора LabVIEW. Блок-диаграмма LabVIEW соответствует строкам текста в обычных языках программирования вроде C или Basic – это такой же реально исполняемый код. Конструирование блок-диаграммы осуществляется путем соединения между собой объектов, выполняющих определенные функции. В этом разделе мы рассмотрим различные компоненты блок-диаграммы: *терминалы* (terminals), *узлы* (nodes) и *проводники данных* (wires).

Простой виртуальный прибор, изображенный на рис. 3.2, вычисляет сумму двух чисел. Его блок-диаграмма на рис. 3.3 содержит примеры терминалов, узлов и проводников данных.

3.2.1. Терминалы данных

Когда вы помещаете элемент управления или индикатор на лицевую панель, LabVIEW автоматически создает на блок-диаграмме соответствующий *терминал*. По умолчанию нельзя удалить с блок-диаграммы терминал, который соответствует элементу управления или индикатору, хотя можно попытаться это сделать. Терминал исчезнет лишь тогда, когда вы удалите соответствующий элемент управления или индикатор с лицевой панели.

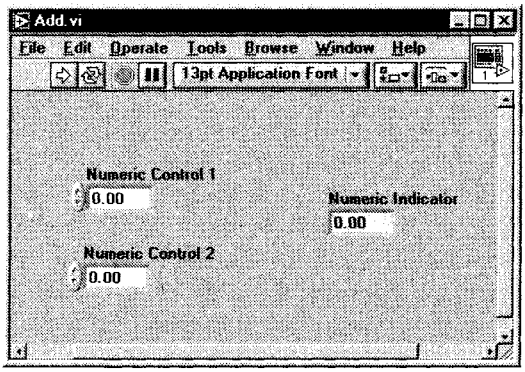


Рис. 3.2

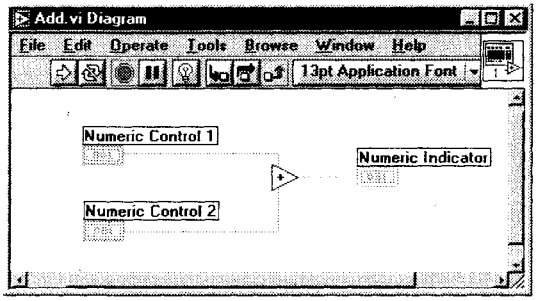


Рис. 3.3

Рамка терминалов элементов управления выделена жирной линией, в то время как граница терминалов индикаторов обозначена тонкой линией (рис. 3.4). Очень важно не путать эти два типа терминалов, поскольку они функционально абсолютно различны (управляющий элемент = ввод, индикатор = вывод, поэтому нельзя заменить один другим).

Вы можете рассматривать терминалы как порты ввода и вывода блок-диаграммы или как источники и приемники данных. Данные, которые вы вводите в элемент управления **Numeric Control 1** (см. рис. 3.3) выходят с лицевой панели и поступают в блок-диаграмму через терминал **Numeric Control 1**. Затем данные с терминала

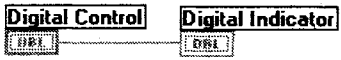


Рис. 3.4

Numeric Control 1 по проводнику поступают на вход функции **Сложить** (Add). Аналогично данные поступают на сумматор и со второго терминала элемента управления. После того как функция Сложить выполнит вычисления, она создаст новое значение на своем выходе. Это значение поступит на терминал **Числовой индикатор** (Numeric Indicator) и будет выведено на лицевую панель, где его может наблюдать пользователь.

3.2.2. Узлы данных

Узел данных – это просто обобщающее название любого исполняемого элемента программы. Узлы аналогичны операторам, функциям и подпрограммам в традиционных языках программирования. Функции **Сложить** (Add) и **Вычесть** (Subtract) представляют один вид узла. Другим видом узла является *структура* (structure), которая может выполнять код циклически или по условию, точно так же, как циклы и условные конструкции в традиционных языках программирования. LabVIEW содержит и специальные типы узлов, например *узел Формула* (Formula Node), предназначенный для работы со сложными математическими формулами и выражениями.

3.2.3. Проводники данных

Виртуальный прибор LabVIEW представляет собой единое целое за счет *проводников данных* (wires), соединяющих узлы и терминалы. Проводники являются каналами прохождения данных от терминала-источника к одному или нескольким терминалам-приемникам. Если вы попытаетесь присоединить к проводнику более чем один источник или вообще ни одного источника, то LabVIEW «не одобрит» ваших действий, и проводник станет *поврежденным* (broken).

Принцип соединения источников и приемников проводниками объясняет, почему управляющие элементы и индикаторы не могут заменять друг друга. Управляющие элементы – это источники, а индикаторы – приемники данных.

Каждый проводник имеет свой стиль и цвет в зависимости от типа данных, проходящих по нему. Блок-диаграмма на рис. 3.3 показывает стиль проводника для числовой скалярной величины (тонкая сплошная линия). В табл. 3.1 приведены несколько типов проводников и соответствующие им типы передаваемых данных. Простой совет: чтобы не перепутать тип данных, подберите их по цвету и стилю проводника.

Таблица 3.1. Основные стили проводников на блок-диаграммах

	Скаляр (Scalar)	(1D Array)	(2D Array)	Цвет
Число с плавающей точкой				Оранжевый
Целое число				Синий
Логическое				Зеленый
Строка				Розовый

3.2.4. Программирование потока данных – движение вместе с потоком

Поскольку LabVIEW не является текстовым языком программирования, его код не может выполняться «строка за строкой». Принцип, который управляет выполнением программы LabVIEW, называется *потоком данных* (dataflow). Говоря проще, код узла выполняется только тогда, когда данные поступили на все его входные терминалы; по окончании работы узел передает данные на свои выходные терминалы, и данные немедленно поступают от источника на терминалы следующих приемников. Принцип потока данных сильно отличается от метода *потока управления* (control flow) в текстовых языках программирования, где инструкции выполняются в той последовательности, в которой они написаны. К этому различию следует привыкнуть. Таким образом, если традиционный поток управления осуществляется при помощи инструкций, обработка потока данных управляется самими данными, то есть *зависит от данных* (data dependent).

3.3. Иконка и соединительная панель

Если ваш виртуальный прибор работает в качестве *виртуального подприбора* (subVI), то его элементы управления и индикаторы получают и возвращают данные в тот ВП, который их вызвал. *Иконка* (icon) ВП однозначно ассоциируется с этим подприбором на блок-диаграмме другого ВП. Иконка может представлять собой изображение, или небольшое текстовое описание ВП, или то и другое вместе.

Соединительная панель (connector) виртуального прибора, по сути, является почти тем же, что и список параметров функций языков C и Pascal; терминалы соединительной панели действуют как параметры ввода/вывода данных подприбора. Каждый терминал соответствует собственному элементу управления или индикатору на лицевой панели. Во время вызова подприбора его входные параметры копируются на подключенных элементах управления, и подпрограмма выполняется. По завершении выполнения подприбора информация индикаторов копируется на терминалах выходных параметров.

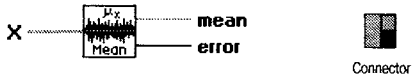


Рис. 3.5. Иконка и соответствующая соединительная панель

Каждый ВП по умолчанию имеет иконку, которая отображается в верхнем правом углу лицевой панели и окна блок-диаграммы. Иконка, принятая по умолчанию, показана на рис. 3.6.

Соединительная панель ВП спрятана под иконкой. Доступ к ней осуществляется путем выбора опции **Показать соединительную панель** (Show Connector) всплывающего меню иконки на лицевой панели (о всплывающем меню мы подробнее поговорим позднее). Если вы вызываете соединительную панель в первый раз, то

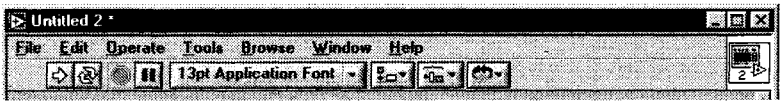


Рис. 3.6. Область иконки

LabVIEW предложит модель панели, имеющей по одному терминалу для каждого элемента управления и индикатора лицевой панели. Прежде чем выйти из области редактирования соединительной панели, вы можете выбрать различные модели панели и назначить до 28 терминалов.

3.3.1. Упражнение 3.1: начало работы

Вы получили достаточно информации для начала работы с программой. Запустите LabVIEW. Теперь вы пройдете через все этапы создания простого ВП, который генерирует случайное число и графически отображает его значение на развертке осциллограммы. В следующей главе все этапы создания прибора рассматриваются более подробно. Сейчас же постарайтесь привыкнуть к изучаемой среде программирования.

Если вы используете полную версию LabVIEW, запустите ее. Теперь вы готовы к созданию первого виртуального прибора.

Если у вас оценочная версия LabVIEW, продолжайте работать с ней, поскольку оценочная версия LabVIEW почти не имеет ограничений по созданию и редактированию виртуальных приборов. Только помните, что ваш ВП будет работать не более 5 минут, а спустя 30 дней LabVIEW вообще перестанет функционировать.

1. Во время запуска в диалоговом окне LabVIEW щелкните мышью на опции **Новый ВП** (New VI). На экране появится лицевая панель с названием **Untitled 1**. Перейдите к палитре **Элементы управления** (Controls) и щелкните мышью по кнопке **Графики** (Graph), чтобы войти в подпалитру **Графики** (Graph). Если палитру **Элементы управления** не видно, выберите **Показать палитру элементов управления** (Show Controls Palette) из меню **Окно** (Windows). Убедитесь также, что лицевая панель активизирована, в противном случае вы увидите палитру **Функции** (Functions) вместо палитры **Элементы управления**. В подпалитре **Графики** выберите **Развертка осциллограммы** (Waveform Chart). Во время прохождения курсором по иконкам в палитре **Элементы управления** выбранная кнопка или имя иконки появляется в верхней части палитры, как это показано на рис. 3.7 и 3.8.

Вы увидите контур графического индикатора, «удерживаемый» курсором. Установите курсор в желаемом месте на лицевой панели и щелкните мышью – график появится точно в этом месте. Если вы хотите его переместить, то выберите инструмент **Перемещение** (Positioning) из палитры **Инструменты** (Tools), затем переместите диаграмму на новое место. Если палитру **Инструменты** не видно, то выберите **Показать палитру инструментов** (Show Tools Palette) из меню **Окно**.

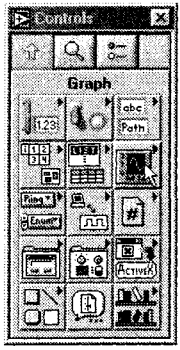


Рис. 3.7

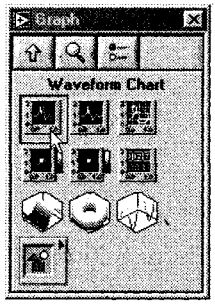


Рис. 3.8

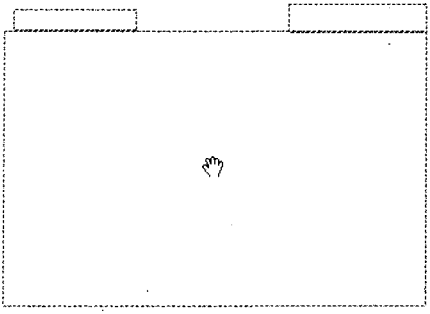


Рис. 3.9

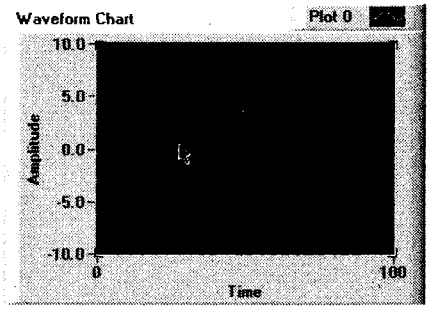


Рис. 3.10

2. Вернитесь к палитре **Элементы управления**, щелкнув по стрелке **На палитру верхнего уровня** (Up to Owning Palette) в подпалитре **Графики** (эта стрелка находится в верхнем левом углу всех палитр управления). В палитре **Элементы управления** выберите подпалитру **Логические** (Boolean), затем укажите элемент **Вертикальный переключатель** (Vertical Toggle Switch).
   Установите его рядом с графическим индикатором, как показано на рис. 3.13.


Operating Tool

3. Выберите инструмент управления в палитре **Инструменты**.
   Теперь измените масштаб графика. Выделите число 10, дважды щелкнув по нему инструментом управления. Напечатайте `1.0` и щелкните по кнопке ввода, которая появится на панели инструментов в верхней части окна.


Enter Button

4. Переключитесь в блок-диаграмму путем выбора пункта **Показать блок-диаграмму** (Show Diagram) из меню **Окно**. Вы обнаружите на ней два терминала (рис. 3.15).

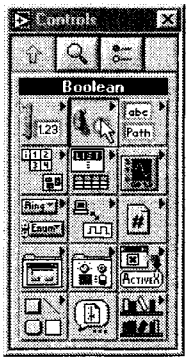


Рис. 3.11

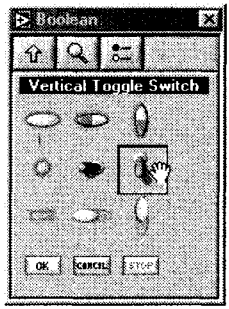


Рис. 3.12

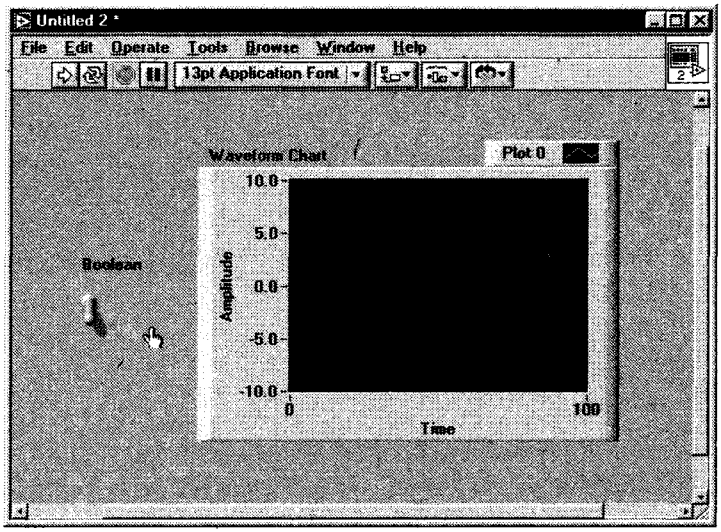


Рис. 3.13

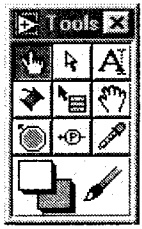


Рис. 3.14

Рис. 3.15

5. Теперь поместите терминалы внутрь цикла по условию, чтобы обеспечить повторение выполнения сегмента вашей программы. Перейдите в подпалитру **Структуры** (Structures) в палитре **Функции** и выберите **Цикл по условию** (While Loop). Убедитесь, что окно блок-диаграммы активизировано. В противном случае вы увидите палитру **Элементы управления** вместо палитры **Функции**.

   Курсор изменит вид и превратится в маленькую иконку цикла. Теперь охватите терминалы DBL и TF: нажмите и удерживайте кнопку мыши во время перемещения курсора от верхнего левого угла в нижний правый угол, охватывая объекты, которые вы хотите поместить в цикл. При освобождении кнопки мыши пунктирная линия – след перемещения курсора – трансформируется в цикл по условию. Сделайте поле цикла больше, чтобы внутри было некоторое свободное пространство.

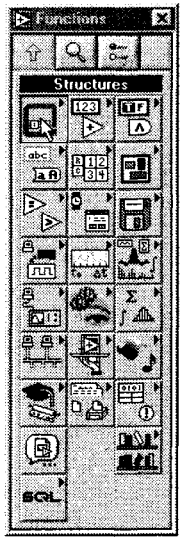


Рис. 3.16

6. Перейдите к палитре **Функции** и выберите опцию **Случайное число (0-1)** – Random number (0-1) – в подпалитре **Числовые** (Numeric). Поместите его внутри цикла по условию.

   Цикл по условию является особой структурой LabVIEW, которая повторяет код, находящийся внутри его границ, до тех пор, пока считывает значение ЛОЖЬ. Это своего рода эквивалент цикла `Do-While` в обычном языке программирования. В главе 6 вы более подробно узнаете об этой структуре.

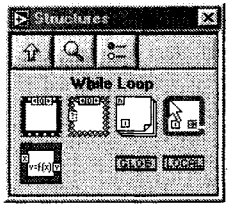


Рис. 3.17

Positioning Tool

7. Выберите инструмент перемещения из палитры **Инструменты** и расположите объекты на вашей блок-диаграмме таким образом, чтобы они выглядели аналогично предыдущей блок-диаграмме.

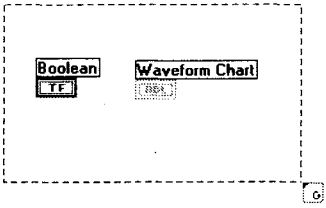


Рис. 3.18

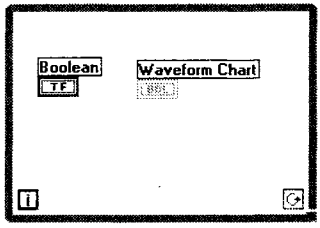


Рис. 3.19

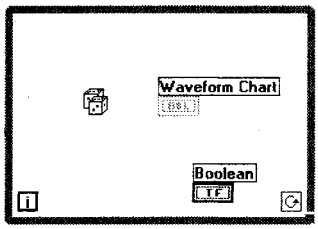


Рис. 3.20

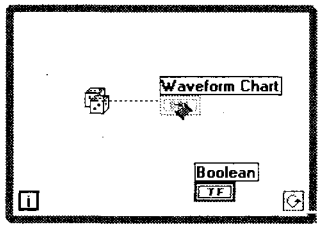


Рис. 3.21

Wiring Tool

8. Теперь выберите инструмент соединения («катушка») из палитры **Инструменты**. Щелкните мышью один раз на терминале **Случайное число (0-1)**, переместите курсор на терминал DBL и еще раз щелкните мышью (рис. 3.22). Теперь две иконки будут соединены сплошной оранжевой линией. Если вы сделали что-то неправильно, выделите проводник или его часть с помощью инструмента перемещения, затем нажмите клавишу <delete>, чтобы удалить его. Теперь соедините терминал **Boolean TF** с терминалом условия выхода из цикла. Цикл начнет выполняться, если переключатель на лицевой панели находится в состоянии ИСТИНА (положение «вверх»), и остановится, если переключатель перейдет в состояние ЛОЖЬ (положение «вниз»).

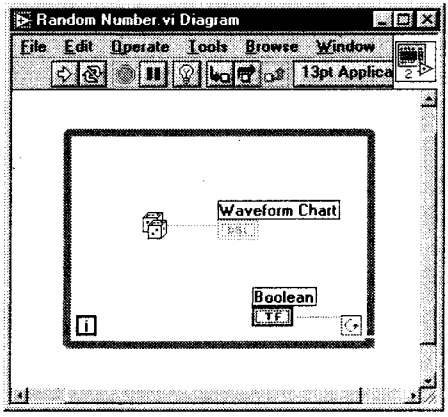


Рис. 3.22

Operating Tool

Run Button

9. Вы почти готовы к запуску программы. Вначале вернитесь к лицевой панели, выбрав пункт **Показать панель** из меню **Окно.** Используя инструмент управления, переведите переключатель в положение «вверх». Щелкните мышью по кнопке запуска, чтобы запустить программу. Вы увидите последовательность случайных чисел, непрерывно вычерчиваемых на

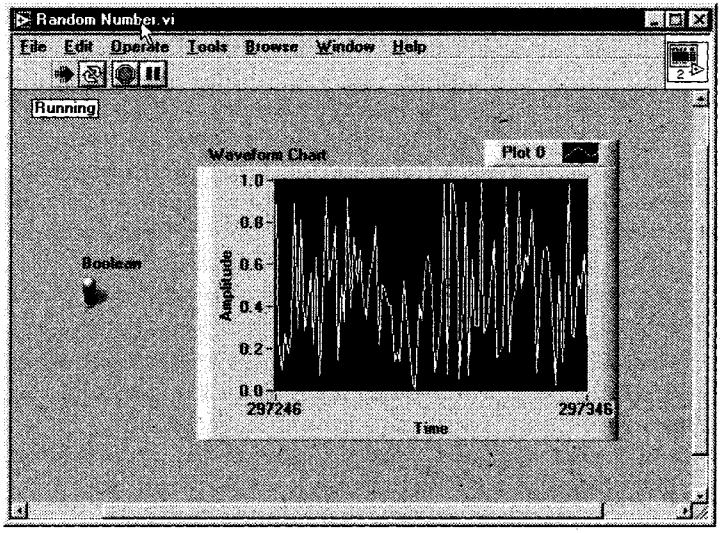


Рис. 3.23

графике. Если хотите остановить выполнение программы, то щелкните по переключателю, переведя его в нижнее положение.

10. Создайте директорию MYWORK в директории LabVIEW. Сохраните ваш ВП в директории MYWORK, выбрав **Сохранить** (Save) из меню **Файл** (File) и указав точное место для сохранения. Назовите его **Random Number.vi**.

Сохраняйте все промежуточные результаты в папке MYWORK. Позже вы легко сможете их найти.

Если вы находитесь в затруднении при создании прибора, вспомните, что решение всех упражнений данной книги имеется в директории EVERYONE или на сопутствующем компакт-диске.

Поздравьте себя – вы только что написали свою первую программу LabVIEW.

3.4. Выпадающее меню

Имейте в виду, что возможности LabVIEW многообразны. Данная книга не ставит перед собой задачи изучения всех особенностей работы с LabVIEW (это заняло бы еще не одну тысячу страниц). Мы пытаемся ускорить ваше обучение и представить обзор функций ВП, которые можно использовать. Если вы захотите

узнать больше о каком-либо предмете, рекомендуем просмотреть другие книги о LabVIEW, посетить семинары или зайти на сайт ni.com/labview (см. приложение).

В LabVIEW есть два типа меню: выпадающее и контекстное (всплывающее). Вы пользовались некоторыми из них во время выполнения последнего упражнения и в дальнейшем будете часто обращаться к ним при создании других программ. В этом разделе мы лишь кратко рассмотрим выпадающее меню. Во время объяснений полезно просмотреть меню на компьютере и даже немного поэкспериментировать.

Панель меню в верхней части окна виртуальных приборов содержит несколько выпадающих меню. Если щелкнуть мышью по какому-либо элементу панели, то меню появляется под панелью. Выпадающие меню содержат несколько пунктов, таких как **Открыть** (Open), **Сохранить** (Save), **Копировать** (Copy), **Вставить** (Paste), которые являются общими для многих приложений. Здесь мы рассмотрим некоторые основные функции меню. Позже вы более подробно изучите дополнительные возможности работы с меню.

Многие меню содержат сокращенные комбинации клавиш, которыми вы при желании можете пользоваться. Для этого нажмите соответствующую кнопку в сочетании с клавишей <control> на PC, клавишей <command> в Mac, клавишей <meta> в Sun и клавишей <alt> в HP.

Многие разделы меню содержат комбинации клавиш, расположенные справа от соответствующих команд. Возможно, вы захотите пользоваться ими вместо команд меню.

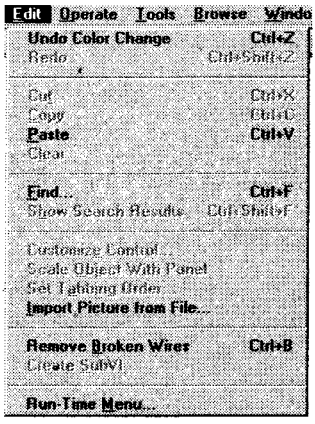


Рис. 3.24

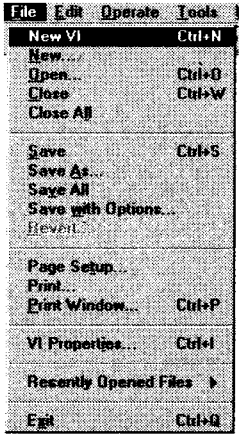


Рис. 3.25

Меню Файл

Вызовите меню **Файл** (File). Оно содержит несколько универсальных команд, таких как **Сохранить** и **Печать** (Print). Также с помощью этого меню вы можете создать новые виртуальные приборы или открыть уже существующие.

Меню Правка

Посмотрите на меню **Правка** (Edit). Оно включает несколько универсальных команд типа **Отменить** (Undo), **Удалить** (Cut), **Копировать** (Copy) и **Вставить** (Paste), что дает возможность редактировать содержимое окна ВП. Также допустимо искать объекты с помощью команды **Найти** (Find) и удалять неисправные проводники с блок-диаграммы.

Меню Управление

Запуск или остановка программы производится с помощью меню **Управление** (Operate), хотя обычно для этой цели применяют кнопки панели инструментов. Вы также можете изменить значения, принятые по умолчанию в программе, управлять функционированием опций печати и регистрации по окончании выполнения и переключаться между режимом запуска и режимом редактирования.

Меню Инструменты

Меню **Инструменты** (Tools) осуществляет доступ к встроенным и дополнительным инструментам и облегчает их работу с функциями LabVIEW, такими как **Measurements & Automation Explorer**, где вы конфигурируете устройства ввода/вывода, или **Средства публикации в Интернете** (Web Publishing Tool), предназначенные для создания HTML-страниц из LabVIEW. Разрешается просматривать и изменять огромное количество параметров в меню **Опции** (Options) LabVIEW.

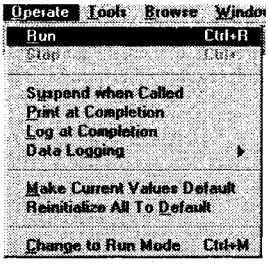


Рис. 3.26

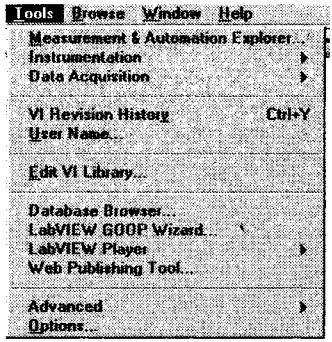


Рис. 3.27

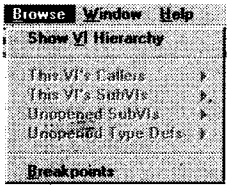


Рис. 3.28

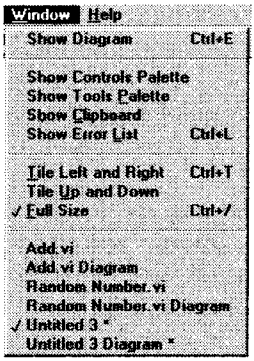


Рис. 3.29

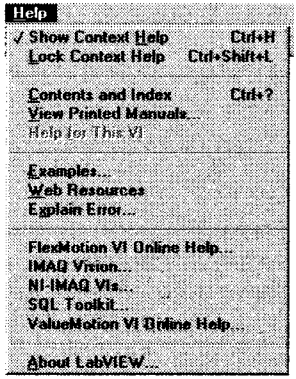


Рис. 3.30

Меню Просмотр

Меню **Просмотр** (Browse) содержит функции для упрощения работы с большими наборами виртуальных приборов. Вы можете увидеть иерархию виртуальных приборов, определить все подприборы виртуального прибора и посмотреть состояние контрольных точек в процессе отладки программы.

Меню Окно

Вызовите меню **Окно** (Windows). С его помощью можно переключаться между окном лицевой панели и окном блок-диаграммы, показывать перечень ошибок и палитру инструментов, располагать окна так, чтобы видеть их одновременно, и переключаться между виртуальными приборами. Допустимо заставить палитры появиться, если они были закрыты. Кроме того, вы можете увидеть информацию о ВП и историю его создания.

Меню Справка

Используя меню **Справка** (Help), легко показать, скрыть или закрепить окно контекстной помощи. Вы также можете получить информацию о LabVIEW в режиме online и просмотреть ее в окне **About LabVIEW**.

3.5. Плавающие палитры

LabVIEW имеет три часто используемые палитры, которые помещают в любое удобное место на экране: палитра **Инструменты,** палитра **Элементы управления** и палитра **Функции**. Вы можете перемещать их, щелкнув мышью по их названию. Они закрываются так же, как и другие окна в операционной системе. Чтобы вернуть их, используйте функцию **Показать палитру ...** (Show ... Palette) в меню **Окно**.

3.5.1. Палитры Элементы управления и Функции

Вам часто придется пользоваться палитрой **Элементы управления**, поскольку именно здесь выбираются элементы управления и индикаторы для расположения их на лицевой панели, и еще чаще – палитрой **Функции**, так как в ней сосредоточены функции и структуры, необходимые для создания ВП.

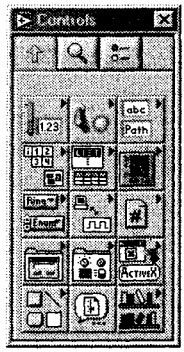


Рис. 3.31

Палитры **Функции** и **Элементы управления** уникальны во многих отношениях. *Палитра **Элементы управления** видна только тогда, когда активизировано окно лицевой панели, а палитру **Функции** можно увидеть, когда вы работаете на блок-диаграмме.* Обе палитры имеют подпалитры, содержащие необходимые объекты. Если провести курсором по кнопкам подпалитр в палитрах **Элементы управления** и **Функции**, можно заметить, что название подпалитры появляется в верхней части окна.

Если щелкнуть мышью по кнопке, появляется соответствующая подпалитра, которая замещает предыдущую активную палитру. Чтобы выбрать объект в подпалитре, щелкните мышью по этому объекту, а затем щелкните мышью в любом месте на лицевой панели или блок-диаграмме, чтобы разместить его здесь. Имена объектов подпалитры появляются при проведении над ними курсора. Для того чтобы вернуться к предыдущей палитре, необходимо щелкнуть мышью по стрелке в верхнем левом углу каждой палитры. Вы можете осуществить поиск любого специфического объекта в палитре, щелкая мышью по иконке с изображением лупы, а также отредактировать собственные палитры, щелкая мышью по кнопке **Опции** (Options).

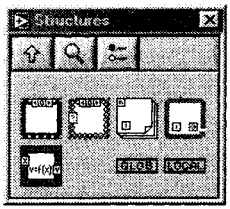


Рис. 3.32. Кнопки палитры (слева направо): ***В главную палитру, Поиск, Опции***

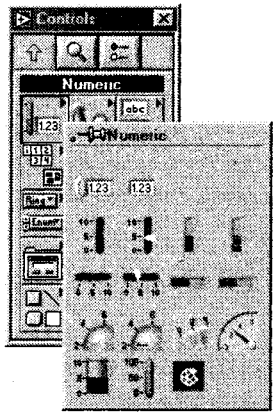


Рис. 3.33

Существует еще один способ управления палитрами, который для некоторых покажется проще. Вместо замещения работающей палитры подпалитрой вы можете пройтись по подпалитрам в иерархическом порядке путем нажатия правой кнопки мыши (Windows) или удерживая клавишу <command> (MacOS) и нажимая на соответствующие кнопки палитры (подпалитры).

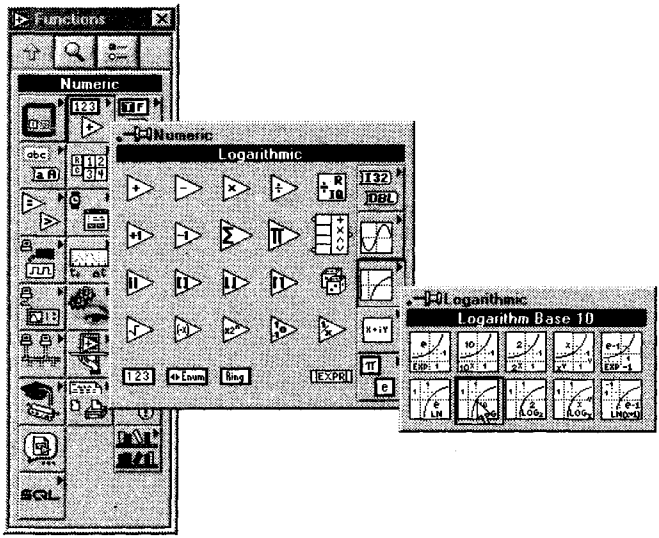


Рис. 3.34

Следует отметить, что некоторые палитры имеют подпалитры, содержащие большое количество объектов; они отмечены маленьким треугольником в верхнем правом углу иконки. Мы поговорим о такого типа подпалитрах в следующей главе.

Палитры **Элементы управления** и **Функции** можно сделать видимыми, просто вызвав контекстное меню в любом пустом месте лицевой панели или блок-диаграммы. Вызов контекстного меню производится щелчком правой кнопкой мыши в Windows, Sun, HP и щелчком кнопкой мыши с удерживанием клавиши <command> в Mac.

3.5.2. Закрепление палитры

Если вы часто пользуетесь подпалитрой, вам, вероятно, захочется закрепить ее, чтобы она не исчезала. Это можно сделать, щелкнув мышью по изображению *кнопки* (thumbtack), расположенной в верхнем левом углу палитры. Эта операция возможна тогда, когда вы в иерархическом порядке проходите по палитрам, нажимая правую кнопку мыши. Теперь имеется отдельное окно, которое можно расположить в любом месте, а затем закрыть по окончании работы с ним. Допустимо оставить открытыми любое количество подпалитр.

3.5.3. Настраиваемые палитры

Если организация палитр **Элементы управления** и **Функции**, принятая по умолчанию в LabVIEW, вас не устраивает, то можете настроить их по собственному желанию. Войдите в меню редактора палитры с помощью щелчка мыши по иконке **Опции** (Palette options). Теперь создавайте собственную палитру путем добавления новых подпалитр, сокрытия объектов или перемещения их с одной палитры на другую. Например, если вы создаете ВП, используя тригонометрические функции, поместите его в существующую подпалитру **Тригонометрические** (Trigonometric) для более легкого доступа. В процессе редактирования палитры поместите наиболее часто употребляемые функции на верхний уровень для более быстрого доступа к ним и в то же время расположите ненужные функции в нижней части подпалитры. Вы также можете решить, показывать иконки и текст в палитре отдельно или вместе (см. главу 4). Вы также можете использовать встроенные наборы «Сбор данных (Data Acquisition)» или «Проверка и измерение (Test and Measurements)», если эти конфигурации являются для вас более удобными.

3.5.4. Палитра инструментов

Инструментом является специальный рабочий режим курсора мыши. Вы используете инструменты для специфического редактирования и управления функциями подобно тому, как использовали бы их в стандартной программе.

Как в случае с палитрами **Элементы управления** и **Функции**, окно палитры **Инструменты** может быть перенесено в любое место или закрыто. Для того чтобы выбрать инструмент, щелкните мышью по соответствующей кнопке палитры **Инструменты**, и курсор мыши соответственно изменится. Если вы не уверены в правильности выбора инструмента, удерживайте курсор на кнопке до появления подсказки.

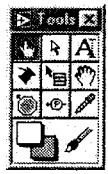


Рис. 3.35

Инструмент **Управление** (Operating) позволяет изменить значения элементов управления и отображения лицевой панели. Вы можете оперировать кнопками, переключателями и другими элементами с помощью этого инструмента – отсюда его название. Это единственный инструмент на лицевой панели, доступный во время работы программы.

Инструмент **Перемещение** (Positioning) служит для выбора, перемещения и изменения размера объектов.

Инструмент **Ввод текста** (Labelling) используется для создания и редактирования текстовых ярлыков.

Инструмент **Соединение** (Wiring, «катушка») применяется для соединения объектов на блок-диаграмме. Он также используется для подключения элементов управления и индикаторов лицевой панели к терминалам соединительной панели ВП.


Color Tool

Инструмент **Цвет** (Color) служит для раскрашивания объектов панелей, переднего и заднего планов. Вы можете установить цвета переднего и заднего планов, щелкнув мышью по соответствующей цветовой области в палитре **Инструменты**. Если вы подведете к объекту инструмент **Цвет**, то сможете выбрать любой оттенок из появившейся цветовой палитры.


Pop-up Tool

Инструмент **Вызов** (Pop-up) открывает меню объекта, если вы щелкнете мышью на этом объекте. Вы можете использовать его для доступа в контекстное меню вместо стандартного способа (правая кнопка мыши в Windows и UNIX и <command> в MacOS).


Scroll Tool

Инструмент **Быстрая прокрутка** (Scroll) дает возможность просмотреть данные в активизированном окне.


Breakpoint Tool

Инструмент **Контрольная точка** (Breakpoint) устанавливает точки разрыва на диаграмме ВП, чтобы помочь отладить код. С его помощью можно на время остановить выполнение программы, посмотреть, что происходит, и изменить в случае необходимости значения входных данных.


Probe Tool

Инструмент **Установка отладочных индикаторов** (Probe, пробник) используется для создания зондов на проводниках, соединяющих элементы блок-диаграммы. Таким образом, вы можете наблюдать проходящую по ним информацию во время работы программы.


Color Copy

Инструмент **Копирование цвета** (Color Copy, «пипетка») служит для копирования цвета из существующего объекта, а затем с помощью инструмента **Цвет** – для переноса этого цвета на другие объекты. Данный метод очень полезен при необходимости скопировать точный оттенок цвета, который вы не помните.

Для смены инструментов необязательно щелкать мышью по нужному, достаточно несколько раз нажать клавишу <Tab>. Или использовать клавишу пробела для переключения между инструментами управления и перемещения, когда активна лицевая панель. Когда же активна блок-диаграмма, нажатие клавиши пробела будет переключать инструменты перемещения и соединения. Клавиши пробела и табуляции циклически переключают наиболее часто используемые инструменты. Попробуйте – и вы увидите, как они сэкономят время!

Перейти в палитру **Инструменты** можно с помощью щелчка правой кнопки мыши (с удерживанием клавиши <shift> в Windows и <command>+<shift> в MacOS).

3.6. Инструментальная панель

Инструментальная панель, или *линейка*, расположенная в верхней части окон LabVIEW, содержит кнопки, предназначенные для управления выполнением

программы, а также опции управления текстом и команды для выравнивания и распределения объектов. В процессе работы вы увидите, что на панели инструментов в окне диаграмм содержится несколько новых функций, а также что при запуске программы несколько связанных с редактированием опций панели инструментов исчезают. Если вы не уверены в том, что делает та или иная кнопка, задержите на ней курсор до появления подсказки, описывающей ее функцию.

Рис. 3.36


Run Button


Run Broken


Run Active

Кнопка **Запуск** (Run) в форме стрелки запускает программу, если вы щелкнете по ней мышью. Она изменяет вид во время работы программы. Если ВП не может быть скомпилирован, то стрелка кнопки запуска становится поврежденной.


Continuous Run Button

Кнопка **Непрерывный запуск** (Continuous Run) заставляет программу непрерывно выполняться, пока вы не нажмете кнопку **Стоп** (Stop). Это похоже на работу оператора GO TO, поэтому пользоваться им надо осторожно.


Abort Button

Кнопка **Прервать** (Abort) легко узнается, так как очень похожа на маленький стоп-сигнал, она активизируется во время начала работы программы; в противном случае она делается серой. Для остановки работающей программы щелкните мышью на этой кнопке.

Использование кнопки **Прервать** похоже на выдергивание кабеля питания компьютера. Программа будет остановлена немедленно, а данные вычислений могут быть потеряны. Лучше запрограммировать более безопасный механизм остановки программ, как мы продемонстрируем позже.


Pause Button

Кнопка **Пауза** (Pause) временно останавливает процесс работы программы, и вы можете использовать одношаговые операции отладки программы, такие как **войти в**, **перешагнуть**, **выйти**. Нажмите кнопку еще раз для того, чтобы вновь запустить программу.


Step Into


Step Over


Step Out

Кнопки одношаговых операций, **Шаг внутрь** (Step Into), **Шаг через** (Step Over), **Шаг из** (Step Out) заставляют ВП делать один шаг при их нажатии, что удобно при отладке программы. О том, как использовать эти кнопки, мы поговорим более подробно в главе 5.


Execution Highlight

Кнопка **Подсветка выполнения** (Execution Highlightning) заставляет ВП подсвечивать поток данных, проходящий через блок-диаграмму. Когда подсветка включена, можно увидеть промежуточные величины данных на блок-диаграмме, которые не проявляются при других условиях.

Кнопка **Предупреждение** (Warning) появляется в случае, если вы сконфигурировали виртуальный прибор для показа предупреждений или у вас есть предупреждение. Вы можете просмотреть предупреждения, щелкнув мышью по кнопке. Предупреждение не является ошибкой, оно лишь говорит о том, что, возможно, вы делаете то, что не планировали (например, у вас есть элемент управления на лицевой панели, не подключенный ни к одному элементу).

Вы можете поменять шрифт, размер, стиль и цвет текста LabVIEW с помощью циклического меню **Шрифты** (Font ring) на панели инструментов.

LabVIEW имеет автоматический механизм выравнивания для рационального расположения иконок и оптимального использования свободного пространства. Выберите с помощью инструмента перемещения объекты, которые нужно выровнять, затем перейдите к циклическому меню **Выравнивание** (Alignment ring) на панели инструментов и выберите способ выравнивания этих объектов (вровень с верхним краем, вровень с левым краем, вертикально и т.д.). Если вы хотите добиться равномерного распределения объектов, используйте циклическое меню **Распределение** (Distribution ring).

Точно так же LabVIEW дает возможность сгруппировать объекты и рассматривать их как один элемент управления для графического редактирования, а также установить глубину расположения объектов, чтобы определить, какие объекты должны быть на переднем плане, а какие – на заднем. Вы можете это сделать с помощью циклического меню **Переупорядочивание** (Reorder ring).

Рис. 3.37

*Рис. 3.38. Меню **Выравнивание** и **Распределение***

Рис. 3.39

3.6.1. Режим выполнения и режим редактирования программы

ВП открывается в *режиме редактирования*, так что вы можете внести в него изменения. Когда вы запускаете ВП, программа автоматически переходит в *режим выполнения*, и вы лишаетесь возможности редактирования. Когда ВП находится в режиме выполнения, на лицевой панели действует только инструмент управления. По завершении работы программы ваш ВП возвращается в режим редактирования (если только вы вручную не переключили его в режим выполнения перед запуском – тогда он останется в режиме выполнения).

Вы можете переключиться в режим выполнения путем выбора опции **Перейти в режим выполнения** (Change to Run Mode) в меню управления и вновь переключиться в режим редактирования путем выбора пункта **Перейти в режим редактирования** (Change to Edit Mode). Вам нет нужды заботиться о режимах выполнения и редактирования. Но если вы случайно обнаружили, что у вас остался лишь

один инструмент управления, и не можете сделать каких-либо изменений, то вам, по крайней мере, станет ясно, почему это произошло.

Если вы предпочитаете запускать ВП сразу в режиме выполнения (возможно для того, чтобы случайный пользователь не мог внести изменения), выберите подменю **Опции** из меню **Управление**. Перейдите на вкладку **Разное** и укажите **Открывать ВП в режиме выполнения**.

3.7. Контекстное меню

Наряду с выпадающим меню мы рассмотрим другой тип меню LabVIEW – контекстное. Возможно, вам придется пользоваться им чаще, чем другими видами меню. Для вызова контекстного меню объекта установите курсор на объекте, затем щелкните правой кнопкой мыши (в Windows и UNIX) или нажмите кнопку <command> и щелкните мышью (в MacOS). Вы также можете щелкнуть по объекту инструментом вызова. На экране появится контекстное меню.

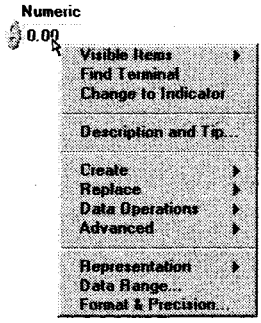


Рис. 3.40

Практически каждый объект LabVIEW имеет контекстное меню опций и команд. Опции, находящиеся в контекстном меню, зависят от типа объекта и будут различными при различных режимах функционирования ВП – редактировании или выполнении. Например, числовые элементы управления имеют другое меню по сравнению с графическим индикатором. Если вы вызовете контекстное меню в пустом пространстве на лицевой панели или блок-диаграмме, то получите палитры **Элементы управления** или **Функции** соответственно.

Как вызвать контекстное меню:

- Windows и UNIX: правой кнопкой мыши;
- Mac: нажатием кнопки <command> и щелчком мышью;
- все платформы: щелчком по объекту инструментом вызова контекстного меню.

Контекстное меню используется в LabVIEW повсеместно. Оно содержит многие опции настройки объектов. Так что помните: если надо что-то сделать, попробуйте контекстное меню.

Если текущий инструмент – **Цвет**, то при вызове контекстного меню вы увидите цветовую палитру.

3.7.1. Особенности контекстного меню

Большинство объектов контекстного меню распадается на подменю, которые называются иерархическими и которые обозначаются стрелкой «вправо» (рис. 3.41). Иногда иерархические меню имеют набор взаимоисключающих опций. Выбранная текущая опция обозначается галочкой при работе с текстом или окружена прямоугольником для графики.

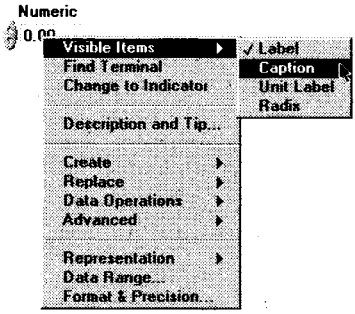


Рис. 3.41

Некоторые объекты меню имеют диалоговые окна, содержащие опции, которые вы можете настраивать. Объекты меню, ведущие к диалоговым окнам, показаны в виде **(...)**.

Объекты меню без стрелок или троеточий являются командами, которые выполняются сразу после выбора. В названии команды обычно написано, что будет выполнено при ее выборе. Например, **Заменить на индикатор** (Change to Indicator). После выбора некоторые команды заменяются в меню на противоположные. Например, после того как вы выбрали **Заменить на индикатор**, эта опция меню превращается в **Заменить на элемент управления** (Change to Control).

Иногда различные части объекта имеют различные контекстные меню. Например, если вы вызовете контекстное меню для ярлыка объекта, то увидите только одну опцию – **Подогнать размеры под текст**. А вызывая меню в любой другой точке объекта, вы увидите полный набор опций. Так что, если вы вызовете контекстное меню и не увидите того, что хотели, попытайтесь вызвать его в другой точке объекта.

3.7.2. Описание особенностей контекстного меню

Контекстные меню дают возможность детально настроить многие характеристики объекта. Следующие опции имеют место во многих контекстных меню, и мы полагаем, что они достаточно важны для того, чтобы их отдельно описать.

Видимые объекты

Многие объекты имеют меню **Видимые элементы** (Visible Items), с помощью которых вы можете показать или скрыть определенные элементы оформления, такие как ярлыки, заголовки, полосы прокрутки или соединительные терминалы. Если вы выберете **Видимые элементы**, то получите другое меню, в котором перечисляются все элементы,

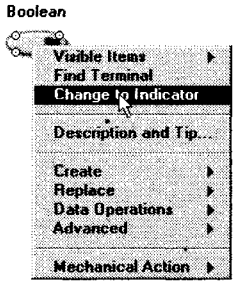


Рис. 3.42. Окно контекстной помощи

которые могут быть показаны (этот перечень меняется в зависимости от объекта). Если элемент имеет рядом галочку, то он является видимым, если же нет – элемент скрыт. Для того чтобы переключить статус элемента, щелкните кнопкой мыши по соответствующей опции.

Опции Найти терминал и Найти управляющий элемент/индикатор

Если вы выберете пункт **Найти терминал** (Find Terminal) из контекстного меню элемента лицевой панели, то LabVIEW отыщет и выделит соответствующий ему терминал на блок-диаграмме. Если вы выберете **Найти управляющий элемент/индикатор** (Find Control/Indicator) из контекстного меню терминала блок-диаграммы, то LabVIEW покажет соответствующий ему объект на лицевой панели.

Опции Заменить на управляющий элемент и Заменить на индикатор

Выбрав опцию **Заменить на управляющий элемент** (Change to Control), вы замените существующий элемент управления (объект ввода данных) на элемент отображения (объект вывода данных) – и наоборот, если вы выберете **Заменить на управляющий элемент** (Change to Indicator). Если объектом является элемент управления, то его контекстное меню содержит опцию **Заменить на индикатор**. Если это элемент отображения, то контекстное меню содержит опцию **Заменить на управляющий элемент**.

Поскольку опция **Заменить на управляющий элемент/индикатор** находится в контекстном меню, то легко случайно ошибиться в выборе, не понимая последствий этого действия. Так как управляющие элементы и индикаторы абсолютно не взаимозаменяемы на блок-диаграмме, вы можете получить непредвиденный результат.

Терминалы управляющих элементов на блок-диаграмме имеют границу толще, чем терминалы индикаторов. Всегда обращайте внимание, точно ли объект является управляющим элементом или индикатором, во избежание конфуза.

Описание и подсказка

Выбор опции **Описание и подсказка** (Description and Tip) позволяет описать объект и сделать подсказку к нему. Описание будет появляться в окне контекстной помощи для данного элемента управления, а подсказка проявится тогда, когда вы остановите курсор мыши на этом элементе.

Опция Создать

Опция **Создать** (Create) облегчает создание узла данных, локальной переменной или ссылки на данный объект (подробнее об этом рассказывается в главе 12).

Опция Заменить

Опция **Заменить** (Replace) является очень важной. Она позволяет входить в палитры **Элементы управления** или **Функции** (в зависимости от того, с чем вы работаете:

с лицевой панелью или блок-диаграммой) и заменять выбранный объект другим, по желанию. Там, где это возможно, проводники останутся неповрежденными.

Опция Операции с данными

Подменю **Операции с данными** (Data Operation) имеет несколько опций, позволяющих манипулировать данными, находящимися в элементах управления или индикаторах:

- **Установить в значение по умолчанию** (Reinitialize to Default) возвращает объект к его значению по умолчанию, тогда как **Сделать текущую величину значением по умолчанию** (Make Current Value Default) устанавливает текущие данные в качестве значения по умолчанию;
- используйте **Удалить данные** (Cut Data), **Копировать данные** (Copy Data) и **Вставить данные** (Paste Data) для извлечения или ввода данных в элементы управления или индикаторы;
- **Соединение DataSocket** (DataSocket Connection) выводит диалоговое окно для конфигурации этого элемента с целью подключения к DataSocket URL. Более подробно об этом вы узнаете в главе 14.

Опция Дополнительно

Элементы опции **Дополнительно** (Advanced) дают доступ к некоторым редко используемым параметрам, которые употребляются для тонкой настройки поведения элементов управления и индикаторов:

- **Управление клавишами** (Key Navigation) применяется для создания комбинации клавиш клавиатуры с целью вызова объекта лицевой панели. Когда вы вводите эту комбинацию в процессе работы ВП, LabVIEW действует так, как будто вы щелкнули мышью по объекту, и курсор становится активным в поле данного объекта;
- **Синхронное отображение** (Syncronous Display) – элемент, который заставляет LabVIEW обновить изображение элемента управления или индикатора при вводе/выводе новой информации. Эта опция интенсивно использует ресурсы системы, поэтому пользоваться ею нужно только в крайнем случае;
- **Настройка** (Customize) выводит Редактор элементов управления (Control Editor) для настройки графического представления элемента управления. Подробнее об этом в главе 15;
- **Скрыть элемент управления/индикатор** (Hide Control/Indicator). Вы можете использовать эту опцию тогда, когда хотите, чтобы объект не был виден на лицевой панели, но присутствовал на блок-диаграмме. Если потребуется вновь показать объект на лицевой панели, то вы должны выбрать **Показать элемент управления/индикатор** (Show Control/Indicator) в контекстном меню терминала блок-диаграммы;

- **Разрешить состояние** (Enable State) дает возможность установить состояние элемента управления как **разрешить** (enabled), **запретить** (disabled), **запретить и скрыть** (disabled & grayed). Эта опция удобна тогда, когда вам нужно показать элемент управления или отображения на лицевой панели, но вы не хотите, чтобы кто-то другой пользовался им.

Существуют и другие виды опций контекстного меню, которые являются специфичными для различных элементов управления (числовых, логических и др.), но мы поговорим о них позднее.

Не старайтесь сразу запомнить все эти особенности: во время создания прибора вы будете пользоваться ими и лучше поймете их работу.

Те же самые объекты будут иметь различные контекстные меню в режимах выполнения и редактирования. Если вы не можете найти определенную опцию, значит, либо ее вообще нет для данного объекта, либо ваш прибор находится не в том режиме функционирования, либо вызывать меню следует в другой точке объекта.

3.8. Справка

3.8.1. Окно контекстной помощи

Окно контекстной помощи (Context Help Window) LabVIEW предлагает необходимую информацию по функциям, константам, подприборам, элементам управления и индикаторам. Для того чтобы задействовать окно, выберите опцию **Показать окно контекстной помощи** (Show Context Help) из меню **Справка** или нажмите клавишную комбинацию быстрого вызова: <control>+<H> в Windows, <command>+<H> в MAC, <meta>+<H> в Sun и <alt>+<H> в HP-UX и Linux. Если на вашей клавиатуре имеется клавиша <help>, то можете нажать ее. Допустимо изменить размер окна помощи и переместить его в любое место на экране.

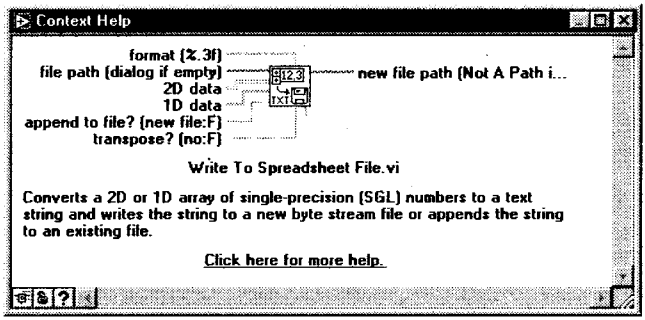


Рис. 3.43

Когда вы удерживаете курсор на функции, узле подприбора (ВПП) или иконке ВП (включая иконку уже открытого ВП в верхнем правом углу окна), то в окне контекстной помощи появляется иконка для этой функции или ВПП с проводниками соответствующего типа данных, присоединенными к каждому входу. *Проводники входных данных направлены влево, а проводники выходных данных – вправо.* Если ВП имеет описание данных, эта информация также отображается.


Lock Button

Для некоторых подпрограмм или функций окно контекстной помощи показывает имена обязательных входных данных жирным шрифтом, а значения по умолчанию заключает в скобки. В некоторых случаях может быть использовано значение по умолчанию, и вам не потребуется подавать никаких входных данных. Вы можете блокировать окно помощи, так что его содержание не будет меняться во время перемещения мыши, путем выбора опции **Блокировать окно контекстной помощи** (Lock Context Help) в меню **Справка** или нажатия кнопки **Блокировать** (Lock) в окне помощи.

Если вы поместите инструмент соединения на каком-либо вводе функции или ВПП, то в окне контекстной помощи будет мигать точка на соответствующем вводе, показывая, какое подключение вы осуществляете.

Для ВП и функций с большим количеством входных и выходных данных окно контекстной помощи может быть переполнено, и LabVIEW предложит выбор между простым и подробным просмотром информации. Вы можете использовать простой просмотр для важных соединений, не обращая внимания на менее важные.


Simple-Detailed Help

Переключение между типами просмотра осуществляется путем нажатия кнопки **Простой/детальный просмотр** (Simple/Detailed Diagram Help) в нижнем левом углу окна помощи. В режиме простого просмотра все необходимые соединения даются жирным шрифтом, рекомендуемые – простым шрифтом, а необязательные не показываются вовсе. Чтобы проинформировать вас о наличии дополнительных соединений, которые не показаны на дисплее и которые можно увидеть при подробном просмотре, в местах входа и выхода данных появляются концы проводников.

При подробном просмотре необходимые соединения показаны жирным шрифтом, рекомендуемые – простым, а необязательные появляются в виде бледно-серого шрифта.

Если нет необходимости подключать входные данные, то рядом с именем входа в скобках появляется значение по умолчанию. Если функция может оперировать многочисленными типами данных, окно контекстной помощи показывает наиболее часто используемый тип.

Опция Online Help


Online Help

Окно контекстной помощи в LabVIEW обеспечивает быструю ссылку на функции, виртуальные приборы, элементы управления и отображения. Однако иногда вам бы хотелось взглянуть на более подробное,

индексированное описание для получения информации по использованию ВП или какой-либо функции. В LabVIEW есть расширенная гипертекстовая справка. Для этого вам нужно выбрать пункт **Содержание и указатель** (Contents and Index) из меню **Справка** или нажать кнопку в окне контекстной помощи.

Вы можете ввести ключевое слово для поиска, просмотра расширенного словарного указателя или сделать выбор по разнообразным тематикам, а также установить свои ссылки на документы гипертекстовой справки, о чем мы поговорим в главе 15.

В настоящее время не все виртуальные приборы LabVIEW имеют документы гипертекстовой справки; в этом случае раздел **Online Help** и кнопка в окне контекстной помощи будут недоступны.

3.9. Несколько слов о виртуальных подприборах

Если вы хотите воспользоваться всеми преимуществами LabVIEW, то должны понять и использовать иерархическую природу ВП. *Подприбор* – это просто отдельный ВП, применяемый другим ВП. Новый ВП можно использовать в качестве ВПП на блок-диаграмме высокоуровневого ВП, снабдив иконкой и определив его соединительную панель. ВПП LabVIEW является аналогом подпрограммы в языке C или другом подобном языке программирования. Поскольку не имеется никаких ограничений по количеству подпрограмм в языке C, также не существует и каких-либо ограничений по количеству ВПП, применяемых в ВП LabVIEW (если позволяет память).

Если на блок-диаграмме имеется большое количество иконок, допустимо сгруппировать их в один ВПП для упрощения диаграммы. Вы также можете использовать один ВПП для выполнения функции, общей для нескольких различных ВП высокого уровня. Этот модульный подход облегчает процесс отладки ВП, его понимания и изменения. Позднее мы подробнее поговорим о создании ВПП, поскольку эта деятельность является очень важной частью программирования в LabVIEW.

3.10. Упражнение 3.2: основные элементы лицевой панели и блок-диаграммы

В этом подразделе мы проделаем несколько простых упражнений, чтобы освоиться со средой программирования LabVIEW. Попытайтесь самостоятельно сделать следующие основные операции. Если возникнут проблемы, просмотрите главу снова.

1. Откройте новый ВП и переключитесь с лицевой панели на блок-диаграмму.

Используйте сочетания клавиш, показанные в выпадающем меню.

2. Измените размер окон так, чтобы одновременно видеть лицевую панель и блок-диаграмму.

Для этого используйте стандартные команды вашей операционной системы, либо функцию **Tile**.

3. Создайте числовой элемент управления, строковый элемент управления и логический индикатор на передней панели путем выбора их из палитры **Элементы управления**.

   Чтобы создать цифровой элемент управления, щелкните мышью по кнопке **Числовые** в палитре **Элементы управления** и выберите **Числовой элемент управления** (Digital Control) из появившейся подпалитры.

   Щелкните мышью по лицевой панели в том месте, где должен появиться выбранный элемент. И вот он здесь! А теперь точно таким же образом создайте элемент управления строковыми данными и логический индикатор. Обратите внимание на то, как во время создания объекта на лицевой панели LabVIEW создает соответствующие терминалы на блок-диаграмме. Также отметьте, что числовые терминалы с плавающей запятой имеют оранжевый цвет (целые числа будут синего цвета), строковые данные – розовый цвет, а логические – зеленый. Эта цветовая гамма облегчает распознавание типа данных.

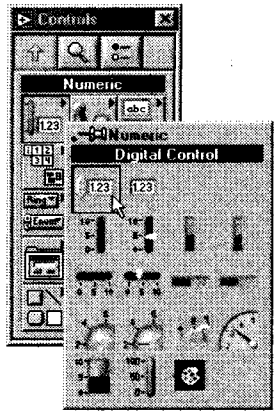


Рис. 3.44

4. Теперь вызовите контекстное меню числового элемента управления (путем нажатия правой кнопки мыши в платформах Windows и UNIX или нажатием <command> в Mac) и выберите опцию **Изменить на индикатор**. Отметьте, как изменяется внешний вид элемента лицевой панели (маленькие стрелки исчезают) и как изменяется терминал на блок-диаграмме (граница индикатора намного тоньше). Переключайте режим работы объекта между управлением и индикацией до тех пор, пока не увидите разницу – как на передней панели, так и на блок-диаграмме. Обратите внимание, что элементы лицевой панели некоторых

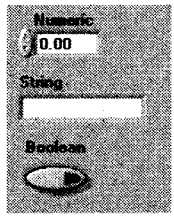


Рис. 3.45

объектов управления и отображения (например, булевых) выглядят одинаково, но их терминалы на блок-диаграмме всегда будут разными.


Positioning Tool

5. Выберите инструмент перемещения («стрелка») из палитры **Инструменты**, затем выделите им любой объект на лицевой панели. Нажмите клавишу <delete> для его удаления. Удалите все объекты на лицевой панели, чтобы получить пустую лицевую панель и пустую блок-диаграмму.


Enter Button

6. Поместите другой числовой элемент управления из подпалитры **Числовые** палитры **Элементы управления** на лицевую панель. После этого не делайте щелчка мышью, и вы увидите маленькое выделенное окно над элементом управления. Напечатайте `Number 1`, и этот текст появится в окне. Щелкните мышью по кнопке **Ввод** панели инструментов, чтобы ввести текст. Вы только что создали ярлык. А теперь создайте: другой числовой элемент управления с именем `Number 2`, числовой элемент отображения с именем `N1+N2` и числовой элемент отображения с именем `N1-N2`.


Operating Tool

С помощью инструмента управления щелкайте по стрелке элемента Number 1, пока его значение не будет равным `4.00`. Задайте Number 2 значение `3.00`.

7. Вернитесь к блок-диаграмме. Поместите функцию **Сложение** (Add) из подпалитры **Числовые** палитры **Функции** на блок-диаграмму (подобно созданию объектов на лицевой панели). Повторите операцию и поместите функцию **Вычитание** (Substract) на диаграммную панель.
8. Вызовите контекстное меню функции **Сложение** и выберите опцию **Видимые элементы ⇒ Терминалы** (ранее эта опция не была помечена, свидетельствуя о том, что терминалы в данный момент невидимы). Как только вы их увидите, посмотрите, как располагаются входы и выходы; затем вновь воспроизведите стандартную иконку путем выбора **Видимые элементы ⇒ Терминалы** на этот раз опция появляется с меткой, указывая на то, что входы видимы).
9. Выведите контекстное окно помощи, используя клавиатуру либо команду **Показать окно контекстной помощи** из меню **Справка**. Установите курсор на функции **Сложение**. Окно контекстной помощи дает исчерпывающую информацию об использовании данной функции и модели соединения. Теперь переведите курсор на функцию **Вычитание** и проследите за изменениями в окне помощи.


Wiring Tool

10. Вы можете использовать инструмент перемещения для перестановки некоторых терминалов, как это показано на рис. 3.47. Затем с помощью инструмента соединения подключите терминалы. Вначале выберите нужный инструмент из палитры **Инструменты**. Затем для того, чтобы нарисовать линию соединения, щелкните мышью один раз по терминалу DBL и один раз по соответствующему терминалу функции **Сложение**. На экране появится сплошная линия оранжевого цвета. Если вы сделали ошибку и на экране появилась пунктирная линия черного цвета,

выберите фрагмент проводника с помощью инструмента перемещения и нажмите кнопку <delete>, затем повторите операцию. Щелкните мышью один раз и затем отпустите ее, начав процесс соединения, затем щелкайте мышью каждый раз, когда хотите добавить новый сегмент (который будет образовывать с предыдущим прямой угол), и, наконец, щелкните мышью на конечной точке соединения.

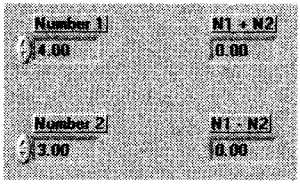


Рис. 3.46

Отметьте, что когда вы перемещаете инструмент соединения над функциями **Сложение** и **Вычитание**, концы проводников появляются в месте расположения входов. Кроме того, при прохождении курсора над терминалом его имя появляется во всплывающей подсказке. Точно так же, как и при обучении печатанию, процесс соединения элементов является довольно сложным, пока вы не освоите его в достаточной мере.

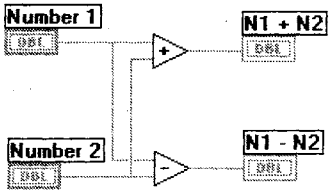


Рис. 3.47

11. Переключитесь на лицевую панель и вызовите контекстное меню иконки ВП (маленькое окно в верхнем правом углу). Из меню выберите **Показать соединительную панель**. Посмотрите на появившуюся соединительную панель. Если вам не удается вызвать контекстное меню, то попытайтесь вызвать его для иконки на блок-диаграмме.

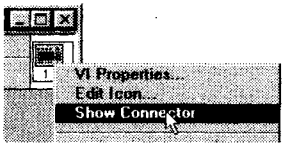


Рис. 3.48

Снова щелкните на соединительной панели и просмотрите ее меню, чтобы увидеть, какие опции имеются в вашем распоряжении. Панель определяет входные и выходные параметры ВП, с тем чтобы вы могли использовать его в качестве ВПП и подавать/снимать с него данные. Допустимо выбрать различные модели для соединительной панели в зависимости от количества параметров, которые вы хотите передать. Вернитесь вновь к иконке с помощью опции **Показать иконку**. Помните, что иконка является художественным отображением вашей программы. Когда вы используете ВП в качестве ВПП, вы присоединяетесь к этой иконке на блок-диаграмме ВП высокого уровня так же, как и в случае присоединения к функции **Сложение**.


Run Button

12. Запустите ВП, щелкнув мышью по кнопке **Запуск**. Индикатор N1+N2 покажет значение `7.00`, а индикатор N1–N2 покажет значение `1.00`. Вы можете изменить входные значения и снова запустить ВП.

13. Сохраните ВП путем выбора опции **Сохранить** из меню **Файл.** Назовите его **Add.vi** и поместите в вашей директории MYWORK или в библиотеке виртуальных приборов.

3.11. Итоги

Среда программирования LabVIEW имеет три основные части: *лицевую панель, блок-диаграмму и соединительную панель.* Лицевая панель является пользовательским интерфейсом программы – вы можете ввести данные через *элементы управления* и получить выходные данные на *индикаторах.* Когда вы помещаете объект на лицевую панель посредством палитры **Элементы управления**, соответствующий терминал появляется на блок-диаграмме, делая данные лицевой панели доступными для использования в программе. Проводники переносят данные между *узлами*, которые являются исполнительными элементами программы LabVIEW. Узел выполняется лишь тогда, когда присутствуют все необходимые для него входные данные, что называется принципом обработки *потока данных.*

Программа должна иметь *иконку и соединительную панель.* Когда вы используете ВП в качестве виртуального подприбора, то его иконка используется на блок-диаграмме другого ВП. Его соединительная панель, обычно скрытая под иконкой, определяет входные и выходные параметры ВПП.

LabVIEW имеет два типа меню: выпадающее и всплывающее (контекстное). Выпадающие меню расположены в обычном месте – в верхней части окна или экрана, тогда как доступ в контекстное меню может быть получен путем щелчка правой кнопкой мыши (в Windows и UNIX) или нажатия <command> (в Mac) на объекте или использования инструмента **Вызов**. Выпадающее меню содержит более универсальные команды, тогда как команды контекстного меню действуют лишь на определенный объект.

Палитра **Инструменты** дает возможность доступа к особым операционным режимам курсора мыши. Вы можете пользоваться этими инструментами для редактирования и выполнения функций, как в стандартных программах. Элементы управления и индикаторы лицевой панели находятся в палитре **Элементы управления**, а постоянные величины, функции и структуры блок-диаграммы – в палитре **Функции**. Эти палитры часто имеют объекты, располагающиеся во вкладках – *подпалитрах.* Так что поиск нужного объекта может занять у вас некоторое время.

Окно контекстной помощи дает ценную информацию о функциях и способах их применения. Это окно доступно через меню **Справка**. LabVIEW также обеспечивает дополнительную гипертекстовую справку, которую можно получить через меню **Справка** или путем нажатия кнопки **Online Help** в окне контекстной помощи.

Вы можете легко превратить любой ВП в ВПП путем создания его иконки и соединительной панели. Использование полностью самостоятельных ВПП имеет много преимуществ: они облегчают отладку, позволяют вызывать одни и те же функции без копирования кода и являются альтернативой огромным запутанным блок-диаграммам.

Обзор

В этой главе мы изучим основные принципы программирования в LabVIEW. Вы научитесь использовать различные типы данных и создавать, изменять, соединять и запускать собственные ВП. Вы узнаете некоторые полезные упрощенные способы для быстрого создания ВП. И, пожалуйста, запомните все эти основные концепции перед тем, как продолжить изучение, поскольку они являются важной составляющей частью всех разработок в LabVIEW.

ЗАДАЧИ

- Овладеть техникой редактирования в LabVIEW
- Изучить различные типы элементов управления и индикаторов и их специальные опции
- Научиться основам создания ВП, то есть соединению терминалов на блок-диаграмме и редактированию
- Создать и запустить простейшие ВП

ОСНОВНЫЕ ТЕРМИНЫ

- Предпочтения
- Числовые (Numeric)
- Строки (String)
- Логические (Boolean)
- Путь (Path)
- Кольцевой список (Ring control)
- Формат и точность (Format and Precision)
- Числовое представление (Numeric Representation)
- Ярлык (Label)
- Заголовок (Caption)

ОСНОВЫ ПРОГРАММИРОВАНИЯ В LabVIEW

4

4.1. Создание виртуальных приборов – теперь ваша очередь!

Мы рассмотрели несколько основных элементов среды программирования LabVIEW, а теперь покажем, как создавать собственные виртуальные приборы. Поскольку люди лучше запоминают действия, когда они сами их *выполняют*, вы можете создавать программу шаг за шагом параллельно с чтением данного пособия.

4.1.1. Размещение объектов на лицевой панели

Создание программы обычно начинается с размещения элементов управления и индикаторов на лицевой панели. Это необходимо для того, чтобы ввести ваши входные данные и получить выходные данные виртуального прибора. Вы уже проделывали это ранее, в ходе выполнения упражнений, поэтому здесь мы лишь напоминаем такую методику (и это будет хорошим началом для данного интерактивного раздела). Если провести курсором по палитре **Элементы управления** (Controls), то в верхней части палитры можно увидеть имена подпалитр. Щелкните правой кнопкой мыши (щелчок при нажатой кнопке <command> в MacOS) и удерживайте ее указатель на одной из кнопок для доступа к соответствующей подпалитре. Имена объектов появятся в верхней части подпалитры при прохождении по ним курсора. Выберите нужный объект в подпалитре, отпустив кнопку мыши.

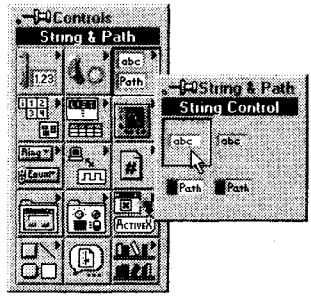


Рис. 4.1

Теперь щелкните мышью по лицевой панели в том месте, где вы хотите разместить объект, – и вот он здесь!

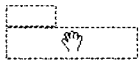


Рис. 4.2

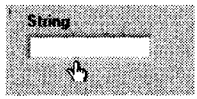


Рис. 4.3

Увидеть палитру элементов управления можно также, вызвав контекстное меню в любом пустом месте лицевой панели.

А теперь создайте новый ВП и поместите числовой элемент управления на лицевой панели. Помните, что при размещении объекта на лицевой панели соответствующий терминал появляется на блок-диаграмме. Возможно, для вас будет полезно выбрать пункт **Разместить слева и справа** (Tile Left and Right) в меню **Окно**, чтобы одновременно видеть лицевую панель и окно блок-диаграммы.

4.1.2. Маркировка объектов

Ярлык (Label) – блок текста, называющий компонент лицевой панели и блок-диаграммы. При создании объекта он появляется в окне лицевой панели с именем по умолчанию (например, «Numeric», «String» и т.п.). Если вы хотите переименовать ярлык, то введите текст с клавиатуры. После изменения текста ярлыка осуществите любое из действий, указанных ниже, для завершения процесса ввода:

- нажмите клавишу <enter> на цифровой клавиатуре;
- щелкните мышью по кнопке **Ввод** на панели инструментов;
- щелкните мышью по лицевой панели или блок-диаграмме за пределами ярлыка;
- нажмите клавиши <shift>+<enter> в Windows и HP или <shift>+<return> в Mac и Sun.

Ярлык появляется на объекте лицевой панели и соответствующем терминале блок-диаграммы.

LabVIEW имеет два типа ярлыков: собственные и свободные. Собственные ярлыки принадлежат отдельному объекту и перемещаются вместе с ним; они обозначают лишь этот объект. Когда вы создаете на лицевой панели элемент управления или индикатор, то его сопровождает незаполненный собственный ярлык, ожидающий ввода данных. Объект на лицевой панели и соответствующий терминал на блок-диаграмме имеют один и тот же собственный ярлык. Свободный ярлык не связан с каким-либо отдельным объектом и может быть по желанию создан или удален.

Разрешается выбрать опцию **Видимые элементы ⇒ Ярлык** (Label) из соответствующего контекстного меню объекта для создания или изменения ярлыка,

который в данный момент не является видимым. Вы можете спрятать собственные ярлыки, но не вправе скопировать или удалить без их «хозяев». Ярлыки структур и функций по умолчанию спрятаны. Возможно, вы захотите так отредактировать ярлыки, чтобы они отображали функциональное назначение, которое несут эти объекты в вашем ВП. Также допустимо сделать видимыми ярлыки ВПП (фактически, они являются их именами), но не отредактировать их.

Теперь определите ярлык для только что созданного числового элемента управления.

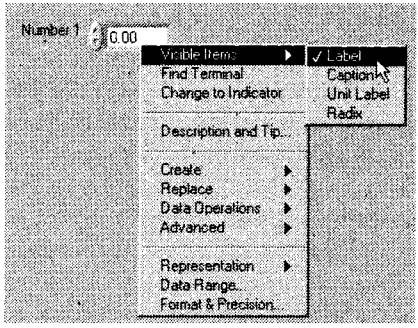


Рис. 4.4

Заголовки

Кроме ярлыков объекты лицевой панели могут иметь *заголовок* (caption). Заголовок – это тот же ярлык, содержащий текст, который описывает элемент управления или индикатор. Чтобы создать заголовок элемента управления или отображения, вызовите его контекстное меню и выберите **Видимые элементы ⇒ Заголовок** (Caption).

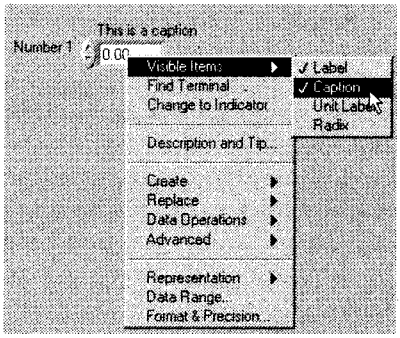


Рис. 4.5

Зачем вообще нужны заголовок и ярлык? В большинстве случаев они не требуются. В некоторых ситуациях более сложного программирования (мы поговорим об этом позднее) необходимо использовать ярлык элемента управления, чтобы сослаться на него; также может понадобится отдельный заголовок. С помощью заголовков вы можете сократить содержание ярлыков («Температура»), а заголовок сделать более длинным («Он показывает текущую температуру в градусах Цельсия»). Заголовки можно рассматривать как удобный способ взаимосвязи комментария с объектом лицевой панели.

Создание свободных ярлыков

Свободные ярлыки не привязаны к объекту. Вы можете их создавать, перемещать и располагать на лицевой панели по вашему усмотрению. Применяйте их для наименования панелей и блок-диаграмм. Для создания свободных ярлыков и редактирования любого видимого текста используйте инструмент ввода текста.


Labeling tool

Для того чтобы создать свободный ярлык, выберите инструмент ввода текста в палитре **Инструменты** и щелкните мышью в любом свободном месте. Слева появляется маленький прямоугольник с текстовым курсором, готовый принять входные данные в виде печатного текста. Наберите текст и введите его одним из четырех вышеописанных методов. Если вы не впечатаете текст в ярлык, то ярлык исчезнет, как только вы сделаете щелчок мышью. Создайте свободный ярлык на лицевой панели и назовите его как хотите.

4.1.3. Изменение шрифта, его стиля, размера и цвета

Вы можете изменить параметры текста в LabVIEW, используя опции в циклическом выпадающем меню **Шрифт** (Font) на панели инструментов. Выберите объекты с помощью инструмента перемещения или выделите текст инструментами ввода текста или управления. Затем назначьте новые параметры в меню **Шрифт**. Изменения произойдут со всеми выбранными или выделенными объектами. Если не выбран ни один объект, будут изменены шрифты по умолчанию, что в будущем повлияет на отображение текста.

Рис. 4.6

Измените ярлык в примере таким образом, чтобы использовался шрифт 18 pt. Если вы выберете подменю **Настройка шрифтов** (Font Dialog), то появится диалоговое окно. С его помощью вы можете изменить многие параметры.

Для разных частей интерфейса LabVIEW применяет различные шрифты: System, Application и Dialog. Эти шрифты всегда используются по умолчанию LabVIEW, и внесение в них изменений влияет на все элементы управления.

- шрифт Application – шрифт, по умолчанию используемый для палитр **Элементы управления**, **Функции** и для текста в новых элементах управления;
- шрифт System задействуется для всех меню;
- LabVIEW применяет шрифт Dialog для текста в диалоговом окне.

4.1.4. Размещение объектов на блок-диаграмме

Интерфейс пользователя не представляет собой ничего особенного, если он не связан с функционирующим ВП. Реальный ВП вы создаете, когда размещаете функции, ВПП и структуры на блок-диаграмме. Для этого войдите в палитру **Функции** точно так же, как входили в палитру **Элементы управления.** Затем выберите нужный объект в подпалитре и щелкните мышью по блок-диаграмме для его размещения.

Перенесите функцию **Сложение** из подпалитры **Числовые** палитры **Функции** на блок-диаграмму.

4.1.5. Методы редактирования

Как только вы разместили объекты на панелях, у вас может возникнуть желание передвинуть их в другое место, скопировать, удалить и т.п. Ниже рассказывается, как это сделать.

Выделение объектов

Прежде чем переместить объект, вы должны его выделить. Чтобы выделить объект, щелкните кнопкой мыши, когда инструмент перемещения находится над ним. Как только объект выделен, LabVIEW окружает его движущимся пунктирным контуром (рис. 4.7).

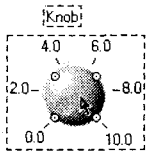


Рис. 4.7. Выделение пунктирной линией

Чтобы выбрать больше, чем один объект, щелкните мышью при нажатой клавише <shift> по каждому дополнительному объекту.

Другим способом выбора одного или более объектов является их охват прямоугольником выбора. Чтобы это осуществить, щелкните в открытой области инструментом перемещения и передвигайте его в диагональном направлении до тех пор, пока выбираемые объекты не окажутся в пределах прямоугольника выбора. Когда вы отпустите кнопку мыши, прямоугольник исчезнет, а каждый выбранный объект будет окружен пунктирным контуром. Этот контур иногда называют «бегущими муравьями» по вполне очевидным причинам. Как только нужные объекты выбраны, вы можете их перемещать с места на место, копировать или удалять.

Нельзя одновременно выбрать объект лицевой панели и объект на блок-диаграмме. Однако вы сможете выбрать большое количество объектов на одной и той же панели – лицевой или блок-диаграмме.

Щелкнув мышью на невыбранном объекте или открытой области, вы отменяете выделение всего, выбранного ранее. Операция <shift>+щелчок мышью по объекту выделяет или снимает выделение объекта без какого-либо влияния на другие выделенные объекты.

А сейчас выделите числовой элемент управления, который вы создали ранее.

Перемещение объектов

Вы можете перемещать объект с одного места на другое путем его выделения и перетаскивания. Если удерживать клавишу <shift> в нажатом положении и при этом перемещать объект, то LabVIEW ограничит движение в горизонтальном или вертикальном направлениях (в зависимости от первоначального направления движения объекта). Также вы можете перемещать выбранные объекты на определенные небольшие расстояния путем нажатия на соответствующую клавишу со стрелкой. А если при этом одновременно удерживать клавишу <shift>, то объект будет передвигаться на бóльшие расстояния.

Если вы передумаете перемещать объект, когда уже начали операцию, то переведите курсор за пределы открытых окон и пунктирный контур исчезнет. Отпустите кнопку мыши, и объект займет прежнее место.

Переместите ваш числовой элемент управления на другую сторону экрана.

Копирование объектов

После того как вы выделили объект LabVIEW, вы можете его скопировать. Из меню **Правка** выберите опцию **Копировать** (Copy), щелкните мышью там, где вы хотите разместить новый объект, и укажите опцию **Вставить** (Paste). Вы также можете клонировать объект с помощью инструмента перемещения. Для этой цели нажмите <ctrl> и щелкните мышью по объекту, если вы используете Windows, выполните <option>+щелчок в MacOS, <meta>+щелчок, если вы пользуетéсь Sun, и <alt>+щелчок в Linux.

Затем переместите курсор, удерживая кнопку мыши. При этом вы перенесете новый объект в виде пунктирного контура, тогда как оригинал останется на месте. Вы также можете скопировать объекты на лицевой панели и блок-диаграмме из одного ВП в другой. Например, если вы выбираете код на одной блок-диаграмме и переносите его в другую, то соответствующие элементы соединений также передвинутся и все необходимые объекты лицевой панели будут созданы.

Нельзя клонировать терминалы элементов управления или индикаторов на блок-диаграмме – вы должны клонировать непосредственно объекты лицевой панели.

Скопируйте числовой элемент управления, используя оба метода. На лицевой панели появятся три элемента с именами Number 1, Number 2 и Number 3 и соответствующие терминалы на блок-диаграмме. Отметьте автоматическое изменение ярлыков, осуществляемое LabVIEW. Чтобы определить их принадлежность щелкните по любому элементу управления или отображения, а затем выберите **Найти терминал** или **Найти элемент управления**. LabVIEW отыщет и выделит соответствующий терминал.

Удаление объектов

Для того чтобы удалить объект, выделите его, а затем выберите **Удалить** (Clear) в меню **Правка** или нажмите клавишу <delete>.

По умолчанию удаление элементов управления и индикаторов возможно только с лицевой панели. Если вы попытаетесь удалить их на блок-диаграмме, то операция будет проигнорирована.

Хотя разрешается удалять бóльшую часть объектов, вам не удастся удалить компоненты элементов управления или индикаторов, такие как ярлыки и числовые дисплеи. Однако вы можете спрятать эти компоненты путем выбора соответствующей опции в разделе **Видимые элементы** контекстного меню.

Удалите один из ваших числовых элементов управления.

Изменение размера объектов

Вы можете изменить размеры большинства объектов. Когда вы проводите инструментом перемещения над объектом, размеры которого надо изменить, в углах объекта появляются риски (рис. 4.8).

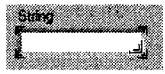


Рис. 4.8

Когда вы проводите инструментом перемещения над риской, курсор мыши трансформируется в уголок. Щелкните мышью и перемещайте курсор до тех пор, пока пунктирная линия не очертит нужный вам размер.

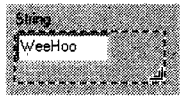


Рис. 4.9

Чтобы прервать эту операцию, продолжайте перемещать угол рамки за пределы окошка до тех пор, пока пунктирная рамка не исчезнет. Затем отпустите кнопку мыши. Объект сохранит свой первоначальный размер.

Если вы будете удерживать клавишу <shift> во время изменения размеров, то объект изменит свои размеры в горизонтальном или вертикальном направлениях или, сохраняя пропорции, в обоих направлениях – в зависимости от объекта и первоначального направления перемещения.

Измените размеры одного из ваших числовых элементов управления.

Размеры некоторых элементов управления и индикаторов разрешается изменять только в определенном направлении. Например, числовой элемент управления можно растянуть только горизонтально (однако, если вы выберете больший шрифт для цифр, то элемент станет больше и в вертикальном направлении).

Перемещение, группировка и блокировка объектов

Объекты могут находиться в верхней части и зачастую скрывать другие объекты – либо потому, что вы их там поместили, либо случайно. LabVIEW имеет несколько команд в меню **Правка**, которые перемещают их относительно друг друга. Эти команды полезны для поиска «потерянных» объектов в ваших ВП. Если вы видите объект, окруженный тенью, то он может находиться над другим объектом. Так, на рис. 4.10 управление строками находится не внутри цикла, а над ним.

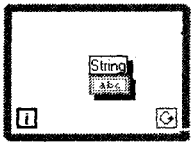


Рис. 4.10

Опция **Сдвинуть на передний план** (Move to Front) перемещает выбранный объект на передний план группы объектов.

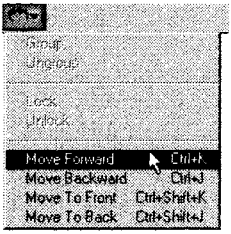


Рис. 4.11

Опция **Сдвинуть вперед** (Move Forward) перемещает выбранный объект на одну позицию вперед в группе объектов.

Опции **Сдвинуть на задний план** (Move to Back) и **Сдвинуть назад** (Move Backward) действуют так же, как опции **Сдвинуть на передний план** и **Сдвинуть вперед**, только они перемещают объекты вглубь группы объектов, а не наружу.

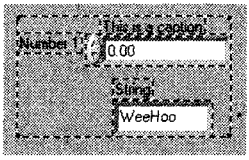


Рис. 4.12. Два элемента управления сгруппированы

На лицевой панели вы можете сгруппировать вместе два или более объектов. Это осуществляется путем выделения объектов, которые нужно группировать, и выбором опции **Группировка** (Group) из меню **Переупорядочивание** (Reorder). Группирование объектов делает их поведение таким, как если бы это был один объект, во время их перемещения, копирования и изменения размеров.

Опция **Снять группировку** (Ungroup) разбивает группу объектов на индивидуальные объекты.

Опция **Блокировать** (Lock) фиксирует размер объекта и его положение, и вы не сможете изменить его размер, переместить его в другое место или стереть. Необходимость в этом возникает тогда, когда вы редактируете лицевую панель с множеством объектов и не хотите случайно изменить определенные элементы управления.

Окрашивание объектов

Допустимо изменить цвет большинства объектов LabVIEW, включая элементы управления и индикаторы, задний план лицевой панели, ярлыки и некоторые элементы блок-диаграммы. Не все элементы могут менять цвет. Например, терминалы объектов лицевой панели на блок-диаграмме и проводники имеют цветовое кодирование типов данных, проходящих по ним, и поэтому вам не удастся изменить их цвет.

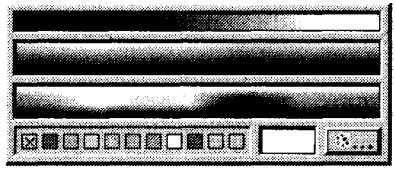


Рис. 4.13

Для того чтобы изменить цвет объекта или окна заднего плана, щелкните по нему правой кнопкой мыши в режиме инструмента раскрашивания. Появится цветовая палитра (рис. 4.13).

Не нажимая кнопок мыши, проведите курсором по палитре. Вы окрасите объект или задний план в тот цвет, над которым проходит курсор. Это дает возможность предварительно посмотреть объект в новом цвете. Если нажать и отпустить кнопку мыши на каком-то цвете, объект сохранит выбранный цвет. Для отмены данной операции выведите курсор из цветовой палитры перед тем, как отпустить кнопку мыши. Выбирая кнопку ... в палитре цветов, вы вызовете диалоговое меню выбора цвета.

Окрасьте один из ваших элементов управления, используя приведенную выше методику. Затем раскрасьте другой элемент управления с помощью палитры **Инструменты**.

Подбор сочетания цветов

Color Copy

Иногда бывает трудно подобрать оттенок цвета, который вы использовали ранее. Вы можете скопировать цвет объекта и перенести его на другой объект без помощи цветовой палитры. Применяйте инструмент копирования цвета («пипетка») в палитре **Инструменты** для установки активных цветов: щелкните этим инструментом на объекте того цвета, который вы хотите выбрать; затем переключитесь на инструмент раскрашивания и окрасьте другие объекты.

Вы можете активизировать инструмент копирования цвета путем нажатия клавиши <ctrl> и щелчка мышью в Windows, <option>+щелчок в Mac, <meta>+щелчок в Sun и <alt>+щелчок в Linux инструментом раскрашивания («кисточка») по объекту, цвет которого вы хотите скопировать. Затем отпустите клавишу и щелкните инструментом раскрашивания по другому объекту; объект окрасится в выбранный вами цвет.

Прозрачность

Если вы выберете кнопку с буквой «T» на цветовой палитре (вверху справа) и окрасите объект, то LabVIEW сделает объект невидимым (прозрачным). Вы можете использовать эту особенность для наслаивания объектов друг на друга. Например, можно поместить невидимые элементы управления в верхней части индикаторов или создать числовые элементы управления без стандартного трехмерного вида. Прозрачность влияет лишь на внешний вид объекта. Объект нормально реагирует на операции, производимые мышью или с клавиатуры.

Выравнивание объектов и их распределение

Иногда нужно, чтобы ВП выглядел изящно, и потребуется выровнять и расположить объекты на дисплее. Функции, встроенные в LabVIEW, делают этот процесс

весьма простым. Для выравнивания объектов вдоль линии выделите их с помощью инструмента перемещения, затем перейдите в меню **Выравнивание** (Align), расположенное на панели инструментов рядом с меню **Шрифт** и выберите способ их выравнивания. Меню **Распределение** (Distribute) работает таким же образом для пространственного выравнивания объектов.

*Рис. 4.14. Циклическое меню **Выравнивание***

*Рис. 4.15. Циклическое меню **Распределение***

Будьте осторожны, пользуясь названными функциями, поскольку вы можете завершить операции так, что все объекты окажутся друг на друге и вам не будет ясно, почему это произошло. Например, если три кнопки поставлены в ряд и вы выравниваете их по левому краю, пользуясь опцией **Левые края** (Left Edges), то все левые края кнопок будут совмещены, и кнопки окажутся друг на друге. Если такое произошло, выберите пункт **Отменить** (Undo) из меню **Правка** или используйте инструмент перемещения для их последовательного перетаскивания.

Упражнение 4.1: практика редактирования

В данном разделе вы на практике познакомитесь с некоторыми методами редактирования, которые только что изучили. Помните, что палитру **Элементы управления** видно только тогда, когда активна лицевая панель, а палитру **Функции** можно видеть лишь при активном окне блок-диаграммы.

1. Откройте пример **Editing Exercise.vi**, находящийся в библиотеке CH4 . LLB директории EVERYONE. Лицевая панель виртуального прибора **Editing Exercise** содержит несколько объектов LabVIEW. Вашей целью является изменение лицевой панели изображенного ВП.
2. Прежде всего следует переместить числовой элемент управления. Выберите инструмент перемещения в палитре **Инструменты**. Щелкните мышью по числовому элементу управления и переместите его в другое место. Обратите внимание, что ярлык сопровождает элемент управления – элемент управления *владеет* этим ярлыком. Щелкните мышью по пустому месту, чтобы снять выделение с элемента управления, затем сделайте щелчок по ярлыку и переместите его в другое место. Вы увидите, что элемент управления его не сопровождает. Собственный ярлык может быть установлен в любом месте относительно элемента управления, но при перемещении элемента управления ярлык будет двигаться вместе с ним.
3. Переместите три ползунковых переключателя как одну группу. Щелкните инструментом перемещения в свободной области рядом с переключателями и, удерживая кнопку мыши нажатой, перемещайте до тех пор, пока все три переключателя не окажутся в пределах маркерного прямоугольника. Нажмите на кнопку мыши и переместите переключатели в другое место.
4. Удалите строковый элемент управления, выделив его с помощью инструмента перемещения, а затем нажав клавишу <delete> или выбрав опцию **Удалить** в меню **Правка**.

5. Скопируйте свободный ярлык. Удерживайте клавишу <ctrl> в Windows, <option> в Mac, <meta> в Sun или <alt> в Linux, затем щелкните мышью по свободному ярлыку и переместите копию в другое место.

6. Измените тип шрифта свободного ярлыка. Выделите текст, используя инструмент ввода текста. Дважды щелкните мышью по тексту или щелкните мышью и переместите курсор по тексту, чтобы его выделить. Измените выделенный текст с помощью опций меню **Шрифт**. Затем сделайте невидимым основание ярлыка, вызвав контекстное меню инструментом раскрашивания и выбрав кнопку **T** (Прозрачность) из цветовой палитры. Помните, что для вызова контекстного меню нужно пользоваться правой кнопкой мыши в Windows, Sun и Linux и <command>+щелчок в Macintosh. Или же вы можете задействовать инструмент вызова контекстного меню из палитры **Инструменты** и просто щелкнуть по объекту в этом режиме.

7. Теперь снова воспользуйтесь меню **Шрифт** для изменения типа шрифта, размеров и цвета текста по оси Y графика осциллограммы.

8. Создайте собственный ярлык для числового индикатора. Щелкните по нему правой кнопкой мыши в Windows, Sun и Linux или щелкните мышью, удерживая нажатой клавишу <command>, в Mac; затем выберите **Видимые элементы ⇒ Ярлык** в контекстном меню. Напечатайте `Digital Indicator` внутри окна. Для ввода текста нажмите клавишу <enter> на клавиатуре, щелкните по кнопке ввода на панели инструментов или щелкните кнопкой мыши за пределами ярлыка.

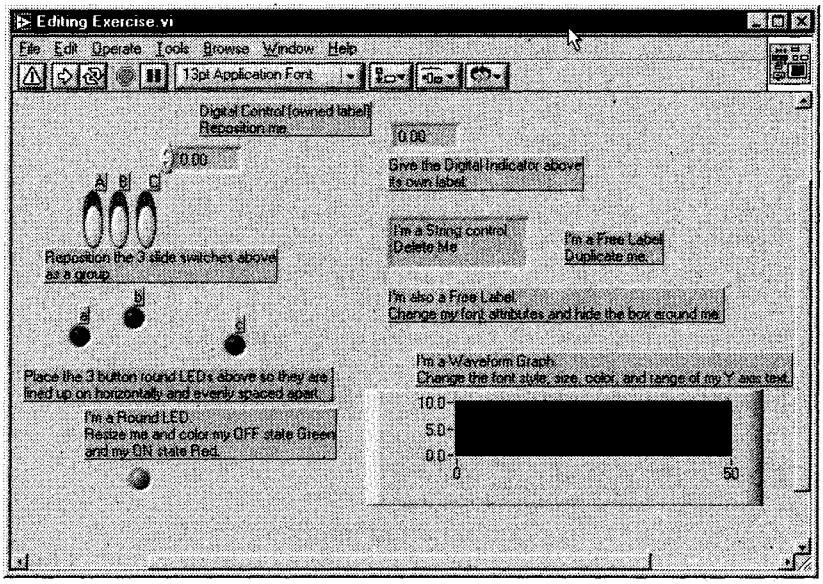


Рис. 4.16


Positioning Tool

9. Измените размер круглого светодиода. Подведите инструмент перемещения к углу светодиода. Стрелка указателя станет курсором для изменения размеров. Нажмите левую кнопку мыши и передвигайте курсор вовне светодиода для его увеличения. Если вы хотите, чтобы соотношение горизонтального и вертикального размеров оставалось постоянным, то удерживайте нажатой клавишу <shift> во время изменения размеров.
10. Измените цвет светодиода. Используя инструмент раскрашивания, вызовите контекстное меню. Выберите цвет из цветовой палитры. Когда вы нажмете кнопку мыши, объект окрасится в выбранный цвет. Щелкните инструментом управления на светодиоде для изменения его значения на «on», а затем окрасьте это состояние.


Color Tool

11. Расположите три светодиодных индикатора таким образом, чтобы они находились на одной горизонтальной линии и были равномерно распределены в пространстве. Используя инструмент перемещения, щелкните на свободной области рядом со светодиодами и охватите их прямоугольной областью выделения. Выровняйте их горизонтально с помощью опции **Вертикально центры** (Vertical Centers) из циклического меню **Выравнивание**, а затем рассредоточьте равномерно в пространстве посредством опции **Горизонтально центры** (Horisontal Centers) из меню **Распределение**.


Vertical Centers Axis


Horizontal Centers Axis

12. Панель теперь будет выглядеть так, как изображено на рис. 4.17.

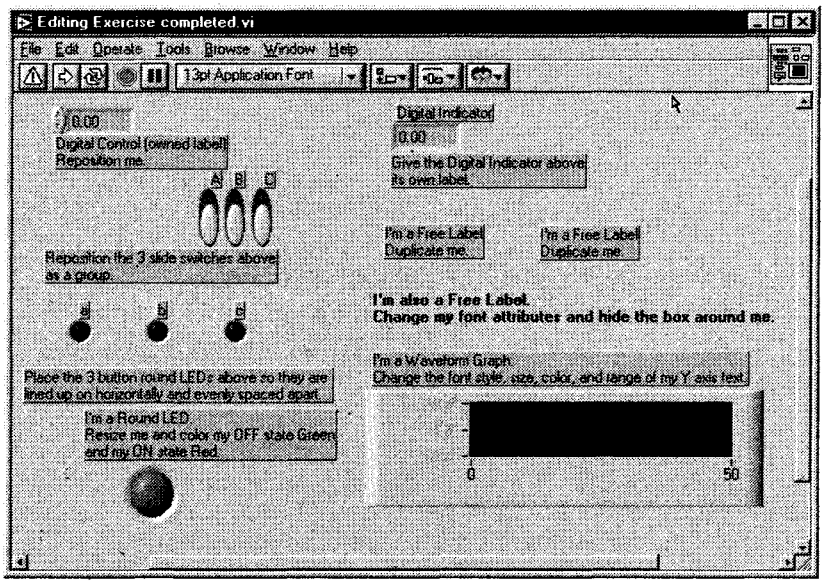


Рис. 4.17

13. Закройте ВП, выбрав опцию **Закрыть** в меню **Файл**. Не сохраняйте никаких изменений. Похлопайте себя по плечу – вы освоили методы редактирования LabVIEW!

4.2. Основные элементы управления и индикаторы

А теперь мы поговорим об элементах палитры **Элементы управления**. LabVIEW имеет четыре типа простых элементов управления и индикаторов: *числовой* (numeric), *логический* (Boolean), *строковый* (string) и редко используемый тип *путь* (path). Вы также встретите несколько более сложных типов данных, таких как массивы, группы, таблицы, диаграммы и графики, о которых мы расскажем позднее.

Палитру **Элементы управления** можно увидеть только тогда, когда активна лицевая панель. Поэтому будьте внимательны и не ищите возможности пользоваться палитрой **Элементы управления**, когда вы работаете с диаграммной панелью.

Когда вам необходимо ввести числовые или текстовые величины в любой элемент управления или индикатор, воспользуйтесь инструментами управления или ввода текста. Новый или измененный текст регистрируется тогда, когда вы нажмете клавишу <enter> на цифровой клавиатуре, или щелкнете по кнопке **Ввод** на панели инструментов, или щелкнете мышью за пределами объекта.

Нажатие на клавишу <enter> на буквенно-цифровой клавиатуре вызовет перевод строки, но не регистрацию вашего изменения (если, конечно, вы не сконфигурировали систему по-другому). Для ввода текста в LabVIEW необходимо использовать клавишу <enter> именно на цифровой клавиатуре. Если вы все же хотите задействовать буквенно-цифровую клавиатуру, то нажимайте для ввода текста <shift>+<enter>.

4.2.1. Числовые элементы управления и индикаторы

Числовые элементы управления дают возможность ввести числовые данные в ВП; числовые индикаторы демонстрируют числовые значения. LabVIEW имеет много типов числовых объектов: круглые ручки управления, ползунковые переключатели, резервуары, термометры и, естественно, простой числовой дисплей. Чтобы пользоваться этими элементами, выберите их из подпалитры **Числовые** палитры **Элементы управления**. Каждый из числовых элементов может быть и элементом управления, и элементом отображения, хотя по умолчанию для каждого объекта принят определенный вид. Например, термометр (thermometer) по умолчанию – индикатор, поскольку в будущем вы наверняка станете пользоваться только этим режимом его работы. И наоборот, кнопка появляется на лицевой панели в качестве

элемента управления, так как кнопки обычно являются инструментами ввода данных.

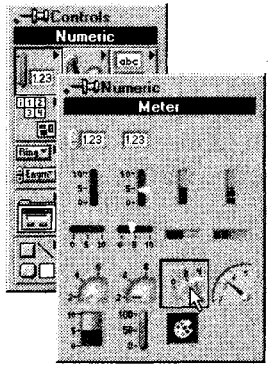


Рис. 4.18

Представление данных

Внешний вид терминалов числовых элементов управления и индикаторов на блок-диаграмме зависит от *представления* (representation) данных. Различные типы данных предусматривают различные методы их хранения, что помогает использовать память более эффективно, так как разные представления числовых данных могут задействовать различное количество байтов памяти. Кроме этого, представление определяет, рассматривать ли данные как числа *со знаком* (допустимы отрицательные величины) и *без знака* (возможны только нулевые или положительные значения). Терминалы блок-диаграммы имеют синий цвет для целых чисел и оранжевый для чисел с плавающей запятой (целые числа не имеют цифр справа от точки). Терминалы содержат несколько букв, описывающих тип данных, например «DBL» для чисел с плавающей запятой с удвоенной точностью.

Имеющиеся в LabVIEW представления числовых данных показаны в табл. 4.1, вместе с их размерами в байтах и рисунками терминалов числовых элементов управления.

Таблица 4.1

Представление	Аббревиатура	Терминал	Размер (байтов)
byte	I8		1
unsigned Byte	U8		1
word	I16		2
unsigned word	U16		2
long	I32		4
unsigned long	U32		4
single precision	SGL		4
double precision	DBL		8
extended precision	EXT		10[a]/12[б]/16[в]
complex single	CSG		8
complex double	CDB		16
complex extended	CXT		20[a]/24[б]/32[в]

[a]Windows

[б]MacOS

[в]UNIX

Вы можете изменить тип представления числовых постоянных, элементов управления и индикаторов, вызвав их контекстное меню и выбрав опцию **Представление** (Representation). Помните, что вызов контекстного меню осуществляется щелчком по объекту правой кнопкой мыши в Windows и UNIX или сочетанием <command>+щелчок в MacOS. После этого вы можете сделать выбор из подпалитры, показанной на рис. 4.19.

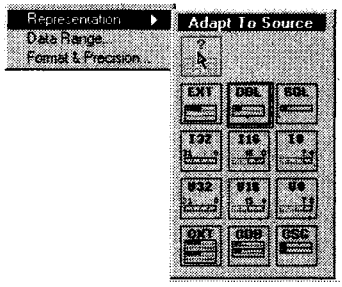


Рис. 4.19

Если вас беспокоит малый объем памяти компьютера, то вы можете пользоваться представлениями с минимальными требованиями к памяти, которые будут содержать ваши данные без потери информации. Это особенно актуально, когда вы работаете с большими структурами, такими как массивы данных. Функция **Приспособить к источнику** (Adapt To Source) автоматически присваивает представление исходных данных индикатору. Кроме этого LabVIEW содержит функции, преобразующие один тип данных в другой. Подробнее об этом рассказывается в главах 9 и 12.

Формат и точность

LabVIEW позволяет выбрать формат представления данных числовыми индикаторами – для показа числовых значений либо для показа времени и даты. Если индикаторы настроены для показа числовых значений, то вы можете сделать выбор между различными системами счисления – с плавающей запятой, научной, инженерной или относительным временем в секундах; также разрешается выбрать *точность* (precision) отображения, которая означает число цифр справа от десятичной запятой (от 0 до 20). Указанная точность влияет лишь на отображение значения, а внутренняя точность остается зависимой от представления данных.

Вы можете установить формат и точность путем выбора опции **Формат и точность** (Format & Precision) из контекстного меню объекта. Появится диалоговое окно (рис. 4.20). Если вам нужно показать время и дату, то выберите пункт **Время и дата** (Time & Date) из меню **Формат** (Format) – диалоговое окно изменится соответствующим образом (рис. 4.21).

Иногда трудно определить точные значения из графических элементов управления и отображения – графиков и термометров. Используя опцию контекстного меню **Видимые элементы ⇒ Цифровой дисплей** (Digital Display), создайте рядом с объектом цифровое окно, которое будет отображать точное числовое значение. Это цифровое отображение является частью самого объекта и не имеет терминала на блок-диаграмме.

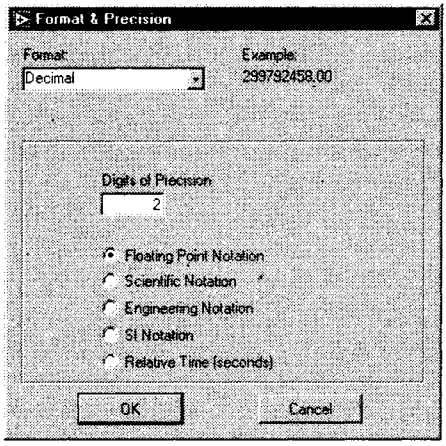


Рис 4.20

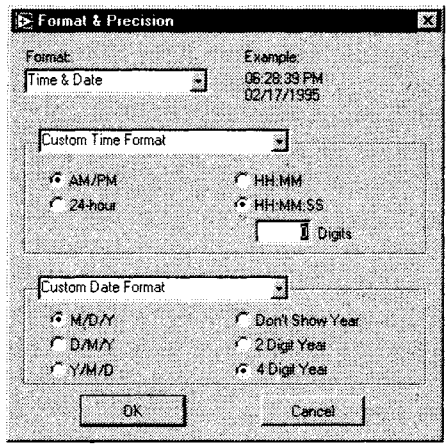


Рис. 4.21

Управление числовым диапазоном

LabVIEW дает возможность настраивать определенный действующий диапазон числовых значений и приращений данных. Например, вы вводите данные в диапазоне от 0 до 100 с приращением 2. Вы можете установить нужный диапазон, вызывая контекстное меню соответствующего объекта и выбирая опцию **Диапазон данных** (Data Range).

В появившемся диалоговом окне (рис. 4.22) вы можете оставить представление по умолчанию (сохранив в окне выбора метку) или изменить представление числа, максимальное и минимальное значения вводимой величины, установить желаемое приращение и изменить значение по умолчанию для этого объекта, а также определить сценарий действий, если значения окажутся за пределами диапазона.

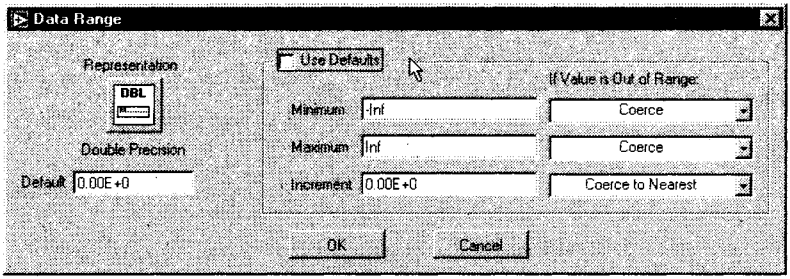


Рис. 4.22

- если выбрать функцию **Игнорировать** (Ignor) для величин, выходящих за пределы диапазона, то LabVIEW не изменит и не отметит их. Щелкая мышью по стрелкам увеличения или уменьшения значения элемента

управления, вы будете менять это значение на величину установленного вами приращения вплоть до максимального (или минимального) значения. Однако вы все же сможете впечатать значения, выходящие за пределы;

- если вы выберете опцию **Ограничить** (Coerce) для ваших данных, то LabVIEW установит все значения меньшие минимальной величины, в минимум, а все значения, превышающие максимальную величину, в максимум. Величины, не кратные приращению, будут округлены.

Кольцевые списки

Кольцевые списки (rings) являются специальными числовыми объектами, которые ставят в соответствие 16-битовым целым беззнаковым числам строки, рисунки или и то и другое. Вы можете найти их в подпалитре **Кольцевые списки** (Ring & Enum) палитры **Элементы управления**. Они особенно полезны для выбора взаимоисключающих опций, таких как режимы выполнения, функции вычисления и т.п.

Создавая кольцевой список, вы вводите текст или вставляете в него картинку, которые становятся связанными с определенным номером (0 для первого текстового сообщения, 1 для следующего и т.д.). Вы можете увидеть этот номер (рис. 4.23), выбрав **Видимые элементы** ⇒ **Цифровой дисплей** из контекстного меню списка.

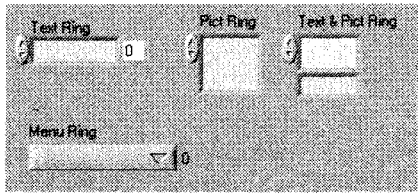


Рис. 4.23

Новый список содержит один объект со значением 0 и пустое изображение. Если вы хотите добавить другой номер и соответствующее сообщение, то выберите **Добавить элемент после** (Add Item After) или **Добавить элемент до** (Add Item Before) в контекстном меню, в результате появится незаполненное окно. Вы можете впечатать текст с помощью инструмента ввода текста или импортировать картинку.

Если щелкнуть по списку инструментом управления, появится список всех возможных сообщений и картинок. Списки необходимы для выбора пользователем какой-либо опции, которая бы соответствовала числовому значению на блок-диаграмме. Попытайтесь создать список на лицевой панели; затем выведите числовой дисплей и добавьте несколько новых опций.

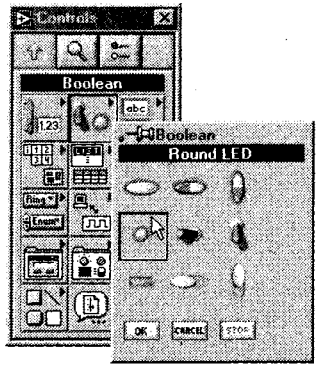


Рис. 4.24

4.2.2. Логические элементы

Логические элементы (Booleans) названы по имени Джорджа Буля, английского логика и математика, чьи работы легли в основу булевой алгебры. Под логическими значениями мы будем понимать положения «on» или «off». Логические значения могут иметь одно из двух состояний: ИСТИНА (True) или ЛОЖЬ (False).

LabVIEW предлагает множество переключателей, светодиодов и кнопок для логических элементов управления и индикаторов, которые находятся в подпалитре **Логические** палитры **Элементы управления**. Вы можете изменить состояния логических элементов щелчком мыши в режиме инструмента управления. Как и числовые элементы, каждый логический элемент может быть элементом управления и индикатором, но имеет тип по умолчанию, основывающийся на его возможном использовании (например, переключатели являются элементами управления, а светодиоды – элементами отображения).

Логические терминалы окрашены на блок-диаграмме в зеленый цвет и содержат буквы «TF».

Терминалы элементов управления имеют жирные границы, в то время как индикаторы – тонкие. Очень важно различать эти два случая, поскольку они функционально не эквивалентны (управляющий элемент – источник данных, а индикатор – приемник данных; они не могут заменять друг друга).

Маркированные кнопки

В LabVIEW существуют три кнопки с текстовыми сообщениями, встроенными в них: **OK**, **Cancel** и **Stop**.

Не только названные кнопки, но все логические структуры также имеют опцию **Видимые элементы** ⇒ **Текст** (Boolean Text), которая показывает «on» или «off» в зависимости от их состояний. Такой текст мало информативен для пользователя. Каждая маркированная кнопка может содержать два текстовых сообщения: одно для состояния ИСТИНА и одно для состояния ЛОЖЬ. Когда вы нажмете на кнопку, в состоянии ИСТИНА появится надпись «on», а в состоянии ЛОЖЬ – «off». Эти сообщения разрешается изменить с помощью инструмента ввода текста.

Механическое действие

Логические элементы управления имеют удобную опцию в контекстном меню, называемую **Механическое действие** (Mechanical Action). Она дает возможность определить, каким образом будет срабатывать кнопка логического элемента управления при ее нажатии (либо она сработает при нажатии, либо при отпускании, либо срабатывает в течение некоторого времени, необходимого для считывания значения, а затем возвращается в свое первоначальное состояние) – см. главу 8.

Создание логического элемента с картинками

Вы можете использовать собственный стиль для логических элементов путем введения картинок для состояний ИСТИНА и ЛОЖЬ. Более подробно об этом рассказано в главе 15.

4.2.3. Строковые данные

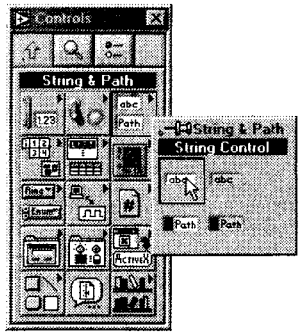


Рис. 4.25

Строковые элементы управления и индикаторы (strings) демонстрируют текстовые данные. Строки зачастую содержат данные в формате ASCII, который представляет собой стандартный способ хранения алфавитно-цифровых символов. Строковые терминалы и проводники, по которым проходят строковые данные, окрашены в розовый цвет. Терминалы содержат буквы «abc». Вы можете найти строки в подпалитре **Строки и пути** (String & Path) палитры **Элементы управления**.

Хотя строковые элементы управления и индикаторы могут содержать числовые символы, они не включают числовых данных. Вы не можете совершать какие-либо числовые операции над строковыми данными. Если вам нужно использовать числовую информацию, содержащуюся в строке (для совершения, например, арифметических действий), преобразуйте ее в числовой формат с помощью необходимых функций (см. главу 9).

Строковые элементы управления и индикаторы достаточно просты в использовании. Более подробно о них говорится в главе 9.

4.2.4. Пути к размещению файлов

Вы можете использовать элементы управления и индикаторы путей (paths) для индикации размещения файлов, папок или директорий. Если функция, которая должна была возвратить путь, не достигла цели, то она выдаст сообщение «Not A Path» в индикаторе пути. Пути являются отдельными, независимыми от платформы типами данных, особенно пути к размещению файлов. Их терминалы и проводники на блок-диаграмме окрашены в голубовато-зеленый цвет. Путь определяется именем логического диска, затем директорией или названием папки и, наконец, самим именем файла. При работе на компьютерах в среде Windows директории и имена файлов отделены обратным слэшем (\); в Mac папки и имена файлов отделены двоеточием (:), в UNIX файлы и директории разделяет прямой слэш (/).

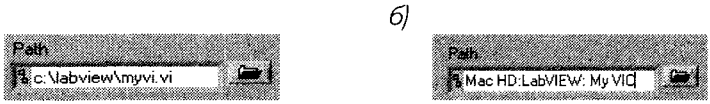


Рис. 4.26. Путь: а – в Windows; б – в MacOS

4.2.5. Улучшение внешнего вида

Ради удовольствия вы можете использовать специальную подпалитру **Оформление** (Decorations) палитры **Элементы управления** для улучшения внешнего вида лицевой панели ВП. Эти украшения выполняют только эстетическую функцию: они являются единственными объектами палитры **Элементы управления**, которые не имеют соответствующих терминалов на блок-диаграмме.

4.2.6. Создание элементов управления и индикаторов

Чтобы сделать процесс программирования более увлекательным, LabVIEW предлагает создать собственные элементы управления и индикаторы. Если в LabVIEW не имеется нужных элементов, создайте их сами! Подробнее об этом рассказывается в главе 16.

4.2.7. Кратко об основных элементах управления и индикаторах

Чтобы быть уверенным, что вы хорошо знаете типы данных, мы еще раз напомним четыре типа простых элементов управления и отображения:

- *числовые* содержат стандартные числовые величины;
- *логические* могут иметь одно из двух состояний: «on» или «off» (ИСТИНА или ЛОЖЬ, 1 или 0);
- *строки* содержат текстовые данные. Хотя они могут включать числа (0–9), вы должны превратить строковые данные в числовые, прежде чем осуществлять над ними арифметические операции;
- *пути* обеспечивают вас независимым от платформы типом данных, что особенно касается путей к расположению файлов.

4.3. Подключение

Рационально организованная лицевая панель ВП, наполненная элементами управления и индикаторами, не принесет никакой пользы, пока вы не соедините все элементы на блок-диаграмме, чтобы программа могла функционировать. Следующие подразделы описывают методы соединения элементов, которые необходимо знать.

Пользуйтесь инструментом соединения («катушка») для соединения терминалов. Острием курсора или инструмента является конец развернутого сегмента «катушки».

Чтобы соединить один терминал с другим, щелкните инструментом соединения по первому терминалу, переместите инструмент на второй терминал и сделайте щелчок. Не имеет значения, по какому терминалу вы сначала щелкнете. Как только острие курсора расположится над терминалом, область ввода начнет мерцать. Щелчок кнопкой мыши соединяет проводник с этим терминалом.

Как только вы осуществили первое подключение к одному из терминалов, LabVIEW рисует проводник при перемещении курсора по блок-диаграмме, как будто леска сматывается с катушки. При этом вам не нужно удерживать кнопку мыши нажатой.

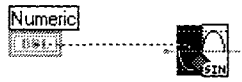


Рис. 4.27

Чтобы начать соединение от уже существующего проводника, осуществите действия, описанные выше, начиная или заканчивая на уже существующем проводнике. При расположении «катушки» над проводником он начинает мерцать.

4.3.1. Автоматическое соединение

Другим способом соединения элементов является использование функции автоматического соединения. При выборе какой-либо функции в палитре **Элементы управления** вы можете заметить, что LabVIEW рисует временные проводники (они выглядят как «усы», выходящие из функции) для указания мест возможного соединения. Если вы пронесете элемент управления рядом с терминалом или другим объектом, который имеет рабочий вход (или выход), то увидите, что LabVIEW соединяет эти объекты. Отпускание кнопки мыши в этот момент «закрепит» процесс соединения.

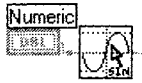


Рис. 4.28

Для того чтобы автоматическое соединение работало, необходимо пронести объект очень близко к другому. Если автоматическое соединение кажется вам неловким или в вашем случае не подходит, не беспокойтесь – просто используйте соединение вручную.

Вы можете соединить терминал за пределами структуры с терминалом, находящимся в структуре, используя базовые операции соединения (см. главу 6). LabVIEW создает туннель в структуре в том месте, где проводник пересекается с границей структуры. Рис. 4.29 иллюстрирует, как этот туннель выглядит во время рисования проводника; на рис. 4.30 показан туннель после завершения операции.

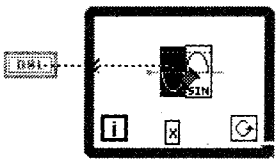


Рис. 4.29

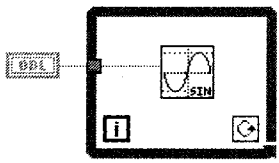


Рис. 4.30

4.3.2. Соединение сложных объектов

Когда вы присоединяете элементы к встроенному узлу или ВПП, обратите внимание на соединительные «усы» и строки, содержащие советы, которые появляются при приближении инструмента соединения к иконке. «Усы» проводников, показанные вокруг иконки виртуального прибора на рис. 4.31, демонстрируют тип данных в виде стиля, толщины и цвета проводников, необходимых для ввода. Точки на концах «усов» показывают входные данные, тогда как «усы» выходных данных точек не имеют. «Усы» рисуются в предлагаемых направлениях их использования, что удобно при работе с функцией автоматического соединения, которую мы только что описали.

Рис. 4.31

Вы также можете обратиться к окну контекстной помощи, на котором выделяется каждый ввод/вывод. Когда вы проводите инструментом соединения над вводом, то соответствующий ввод иконки, изображенной в окне помощи, начинает мигать, чтобы вы удостоверились, что совершаете правильное подключение. Допустимо воспользоваться окном контекстной помощи для определения необходимых, рекомендуемых или необязательных соединений.

4.3.3. Поврежденные проводники

При совершении ошибки во время соединения элементов появляется поврежденный (broken) проводник в виде черной пунктирной линии вместо обычно окрашенного проводника. До тех пор пока все ошибки не будут устранены, кнопка запуска программы остается поврежденной и программа не выполняется. Вы можете устранить поврежденный проводник, выделив и удалив его. Лучшим способом является одновременное удаление неисправных проводников путем выбора опции **Удалить поврежденные проводники** (Remove Broken Wires) в меню **Правка** или с использованием клавиш <ctrl>+<B> в Windows или <command>+<B> в MacOS.

Иногда неисправные проводники могут быть не более чем фрагментами, скрытыми под другими объектами, или настолько малыми, что вы можете их не увидеть. В этом случае просто выберите опцию **Удалить поврежденные проводники**.

Если вам неизвестно, по какой причине проводник оказался поврежденным, то щелкните мышью по неисправной кнопке запуска программы или вызовите контекстное меню неисправного проводника и выберите опцию **Список ошибок** (List Errors). Появится диалоговое окно с описанием проблемы.

4.3.4. Советы по соединению элементов

Следующие подсказки могут немного облегчить процесс соединения элементов на блок-диаграмме:

- без щелчка мыши вы можете изменить направление соединения на 90° («изогнуть» его) только один раз;
- щелкните мышью для поворота проводника и изменения его направления;
- меняйте направление выхода проводника из терминала нажатием клавиши пробела;
- для начала или завершения процесса соединения элементов дважды щелкните инструментом соединения по открытой области;
- при пересечении проводников появляется небольшой зазор в первом нарисованном проводнике, как будто он находится под вторым проводником (рис. 4.32). Для устранения этого эффекта вы можете выбрать **Инструменты** ⇒ **Опции** (Options) и войти в меню **Блок-диаграмма** (Block Diagram), затем отметить окошко **Показывать точки в местах пересечения проводников** (Show dots at wire junctions);
- щелкните правой кнопкой мыши для удаления проводника во время процесса соединения или используйте сочетание <command>+щелчок в Mac;
- воспользуйтесь окном контекстной помощи для получения большей информации об объекте и облегчения процесса соединения.

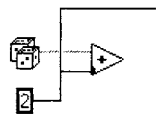


Рис. 4.32

Создайте числовые элементы управления на лицевой панели и подключите их к входам функции **Сложение**. Пока не подключайте выход.

4.3.5. Удлинение проводников

Вы можете перемещать соединенные объекты по одному или группами, перетаскивая их на новое место с помощью инструмента перемещения. Проводники, соединяющие выбранные объекты, растягиваются автоматически. Если вы сделаете копии выбранных объектов или переместите их из одной блок-диаграммы в область этой же диаграммы или на другую блок-диаграмму (например, из блок-диаграммы в область структуры, такой как «Цикл по условию»), то LabVIEW не перенесет соединительные проводники, если вы их также не выделите.

При растягивании проводников иногда образуются обрезки или свободные концы. Вы можете их удалить командой **Удалить поврежденные проводники** из меню **Правка** перед выполнением программы.

Теперь переместите функцию **Сложение** инструментом перемещения и проследите, как изменяются соответствующие проводники.

4.3.6. Выделение и удаление проводников

Сегмент проводника представляет собой одиночный кусок провода, расположенный горизонтально или вертикально. Точка, в которой соединяются три или четыре сегмента, называется *контактом*. *Изгиб* в проводнике является местом, где соединяются два сегмента. Ответвление проводников содержит все сегменты проводников от узла до узла, от терминала до узла, от терминала до терминала, если между ними нет контакта. Выделение сегмента осуществляется одним щелчком мыши по проводнику в режиме инструмента перемещения. Ответвление выделяется двойным щелчком. Тройной щелчок выделяет весь проводник. Нажмите клавишу <delete> или <backspace> для удаления выделенного сегмента проводника.

Выделите и удалите один из ваших проводников, затем восстановите соединение.

4.3.7. Перемещение проводников

Легко переместить один или более сегментов путем выделения и перетаскивания их инструментом перемещения. Используя клавиши-стрелки, вы можете перемещать выделенный сегмент на один пиксел, что удобно для точного расположения проводника. Для приспособления к изменениям LabVIEW растягивает соседние невыбранные сегменты. Разрешается выбрать и одновременно перетащить множество сегментов проводника, включая оборванные сегменты. Если вы перемещаете туннель структуры, то LabVIEW обычно сохраняет соединения между терминалом и присоединенным узлом. Передвиньте сегмент проводника с помощью инструмента перемещения, а затем с помощью клавиш-стрелок.

4.3.8. Соединение с объектами, находящимися за пределами экрана

Если блок-диаграмма слишком велика, чтобы уместиться на экране, вы можете с помощью панелей прокрутки переместить за пределы экрана любые объекты. Перемещение «катушки» во время процесса соединения чуть-чуть за пределы окна блок-диаграммы приводит к автоматической прокрутке окна блок-диаграммы. Также вы можете щелкнуть инструментом перемещения по пустому месту и перетаскивать его за пределы блок-диаграммы, создавая, таким образом, больше свободного пространства.

4.3.9. Автоматическое добавление констант, элементов управления и индикаторов

Вместо того чтобы создавать константу, элемент управления или индикатор путем выбора их в палитре, а затем соединения их вручную с терминалом, вы можете щелкнуть на входе терминала и выбрать опции **Создать** ⇒ **Константа** (Create ⇒ Constant),

Создать ⇒ **Элемент управления** (Creat ⇒ Control) или **Создать** ⇒ **Индикатор** (Creat ⇒ Indicator) для автоматического создания объекта с соответствующим типом данных для этого терминала. Новый объект подключается автоматически. Помните о такой возможности при создании программ – она очень полезна!

Создайте элемент отображения для демонстрации результатов операции **Сложение**, вызвав контекстное меню этой функции и выбрав **Создать** ⇒ **Индикатор**. LabVIEW создаст терминал индикатора на блок-диаграмме, соединенного с выходом функции **Сложение**, а также соответствующий элемент отображения на лицевой панели.

4.4. Запуск виртуального прибора


Run Button

Вы можете запустить ВП, выбрав команду **Запуск** (Run) из меню **Управление** или щелкнув мышью по кнопке **Запуск** (Run). Во время выполнения программы кнопка **Запуск** меняет свой вид.


Run Active

Если кнопка **Запуск** окрашена в черный цвет и выглядит так, как будто она «движется», то в этот момент программа выполняется на самом высоком уровне.


Run Button (sub VI)

Если кнопка **Запуск** имеет маленькую стрелку внутри большой стрелки, это означает, что программа выполняется в качестве ВПП, вызванного другим ВП.


Continuous Run Button

Если вы хотите, чтобы программа выполнялась непрерывно, нажмите кнопку **Непрерывный запуск** (Continuous Run), но будьте осторожны: это не тот прием программирования, который следует принять на вооружение. Вы можете случайно загнать программу в бесконечный цикл и будете вынуждены перезагрузить компьютер, чтобы выйти из этого состояния. Если вы все-таки оказались в таком положении, попытайтесь сделать следующее: нажмите клавишу, запускающую команду **Прервать** (Abort): <ctrl>+<.> в Windows, <command>+<.> в Mac, <meta>+<.> в Suns и <alt>+<.> в Linux.


Abort Button

Нажмите кнопку **Прервать**, чтобы прекратить выполнение высокоуровневого ВП. Если ВП используется более чем одним работающим высокоуровневым ВП, то кнопка становится серой. Нажатие на кнопку **Прервать** вызывает немедленное прекращение выполнения программы и не является хорошей практикой программирования, поскольку полученные данные могут оказаться недействительными. Вы должны предусмотреть возможность программной остановки работы ВП, которая аккуратно свернет выполнение программы. Как это сделать, вы узнаете позднее.


Pause Button

Кнопка **Пауза** (Pause) временно останавливает выполнение программы, а затем возобновляет выполнение, если вы вновь ее нажмете.

Вы можете запустить множество программ в одно и то же время. После начала работы ВП переключитесь в окно лицевой панели или блок-диаграммы следующего ВП и начинайте его выполнение, как описано выше. Обратите внимание, что если вы выполняете ВПП как высокоуровневый ВП, то все виртуальные приборы,

которые его вызывают как ВПП, останавливают свою работу до завершения его выполнения. Вы не можете одновременно заставить работать ВПП в качестве высокоуровневого ВП и подприбора.

4.4.1. Упражнение 4.2: создание термометра

А теперь мы намерены собрать воедино ВП, который мог бы действительно что-то сделать. Данная программа будет считывать значение напряжения из канала на многофункциональной плате ввода/вывода, если у вас есть такая плата, или имитировать эту плату, если у вас ее нет, и демонстрировать полученное значение на термометре лицевой панели. Вы должны подключить нулевой канал платы к датчику температуры или похожему источнику напряжения (желательно в диапазоне от 0 до 1 В). Если у вас есть плата, то прочитайте инструкцию к ней. Более подробно об этом рассказывается в главах 10 и 11.

Не забудьте сохранить ваш пример, поскольку позже вы им воспользуетесь. Если вы все же его не сохранили, то в случае необходимости обратитесь к нашей версии **Thermometer.vi** в директории `EVERYONE\CH4.LLB`.

1. Откройте новую лицевую панель.
2. Поместите на нее термометр, выбрав его в подпалитре **Числовые** палитры **Элементы управления**. При появлении термометра на лицевой панели измените его ярлык и назовите `Температура`. Измените масштаб термометра, щелкнув инструментом управления по метке «10.0» и вводя число `100` инструментом ввода текста.

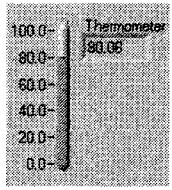


Рис. 4.33

3. Постройте одну из изображенных на рис. 4.34 и 4.35 блок-диаграмм. Чтобы одновременно видеть лицевую панель и блок-диаграмму, выберите функцию **Разместить слева и справа** в меню **Окно**. Постройте блок-диаграмму А, если вы используете плату ввода/вывода (DAQ), или блок-диаграмму В, если вы вынуждены имитировать получение данных. *Вам не следует создавать обе, если вы действительно не захотите в этом попрактиковаться.* Функцию **АЦП Выборка из канала** (AI Sample Channel.vi), используемую на этой блок-диаграмме, вы можете найти в палитре **Функции** ⇒

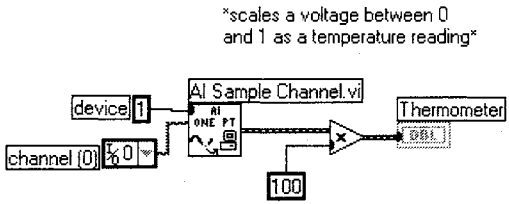


Рис. 4.34. Диаграмма А с использованием платы ввода/вывода

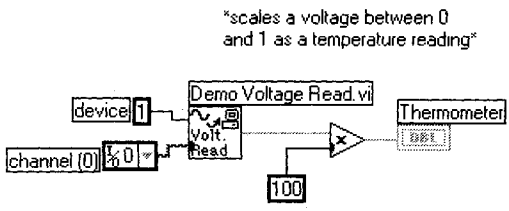


Рис. 4.35. Диаграмма В с имитацией получения данных

Сбор данных (Data Acquisition) ⇒ **Аналоговый ввод** (Analog Input). Если у вас нет платы, используйте ВПП **Demo Voltage Read**, показанный на блок-диаграмме и расположенный в палитре **Функции** ⇒ **Учебник** (Tutorial). Не забудьте воспользоваться окном контекстной помощи для правильного подсоединения к входам. Посмотрите, где мигает соответствующий вход в окне контекстной помощи, когда вы помещаете инструмент «катушка» над входом/выходом функции: это позволит проверить, правильно ли вы соединяете терминалы.

Вы можете создать константу, вызвав контекстное меню соответствующего входа ВПП **Demo Voltage Read** или **АЦП Выборка из канала** и выбрав **Создать** ⇒ **Константа**. Убедитесь, что вы щелкнули по вводу, к которому хотите подсоединиться, – в противном случае вы можете создать константу для ошибочно выбранного ввода.

Другие компоненты блок-диаграммы описываются ниже.

Для определения канала аналогового ввода данных, который вы хотите использовать (это может быть и ненулевой канал), потребуется константа канала ввода/вывода устройства сбора данных (DAQ Channel I/O). Это иконка пурпурного цвета, расположенная в подпалитре **Сбор данных** палитры **Функции**. При работе с ВПП **Demo Voltage Read** это значение игнорируется, но мы оставляем его для имитации реального получения данных. *Несмотря на то что константа канала ввода/вывода содержит числовые символы, следует использовать константу типа данных канала ввода/вывода, а не числовой тип данных, поскольку это приведет к невозможности соединения.*

При помещении константы канала ввода/вывода на блок-диаграмму она не содержит никаких данных. Вы можете выбрать значение с помощью инструмента управления (если предварительно определили имя канала в **Measurement & Automation Explorer**) либо впечатать данные, используя инструмент ввода текста.

Если вы не сконфигурировали плату с помощью **Measurement & Automation Explorer** или используете моделирование, вам необходимо сначала вызвать контекстное меню константы канала ввода/вывода и выбрать опцию **Разрешить неопределенные имена** (Allow Undefined Names).

Цифровая константа, расположенная в подпалитре **Числовые** палитры **Функции**, определяет номер устройства, присвоенный вашей плате, который может и не быть равным 1. При использовании ВПП **Demo Voltage Read** это значение игнорируется.

При размещении числовой константы на блок-диаграмме она будет содержать выделенное значение «0». Вы можете тотчас ввести новое число (если перед этим не сделали щелчок мышью в каком-нибудь месте).

При работе в среде Windows номер назначает **Measurement & Automation Explorer** (возможно, это будет 1, если вы не используете несколько плат). На компьютерах под управление UNIX номер устройства назначается через конфигурационный файл. В системе Mac номером устройства является номер слота, в который вставлена плата; его вы можете увидеть, открыв контрольную панель **NI-DAQ**. Более подробная информация будет дана в главе 10.

Блок-диаграмма A предусматривает, что датчик температуры присоединен к каналу 0 вашей платы ввода/вывода (другой источник напряжения тоже приветствуется, но он может не обеспечить должного значения «температуры», если его напряжение не будет лежать в окрестности 0,8 В).

Эта цифровая константа просто переводит электрическое напряжение в «действующую» температуру. Если напряжение не находится в пределах между 0 и 1,0, то вы можете изменить значение этой константы, чтобы сделать выходную «температуру» более подходящей.

4. Используйте окно контекстной помощи, находящееся в меню **Справка**, для отображения схемы соединения функциональных терминалов. Обратите особое внимание на цветовое кодирование во избежание неисправных соединений. Помните, что типы числовых данных окрашены в голубой или оранжевый цвета, строковые данные имеют розовый цвет, а логические значения окрашены в зеленый.

5. Запустите программу, щелкнув по кнопке **Запуск**. Вы увидите, как термометр показывает напряжение, поступающее с платы или с имитационной подпрограммы. Если вам не удается заставить программу работать, прочтите главу 5, в которой объясняется методика отладки. Повторите операцию. Если не работает плата ввода/вывода, не волнуйтесь и попытайтесь использовать ВП с моделирующей подпрограммой (подсказка: используйте функцию **Заменить**, чтобы ввести ВПП **Demo Voltage Read.vi**). Мы лишь хотим дать хороший пример программирования в среде LabVIEW, а не познакомить вас с набором проблем при получении данных.

6. Сохраните ВП в директории MYWORK путем выбора опции **Сохранить** в меню **Файл**. Назовите его **Thermometer.vi**. Этот ВП будет в дальнейшем использоваться в качестве ВПП.

Клавишные комбинации быстрого вызова

Многим опциям LabVIEW назначены комбинации клавиш для их быстрого вызова. Например, чтобы создать новое окно лицевой панели, вы можете выбрать опцию **Создать** (New) в меню **Файл** либо нажать эквивалентный набор клавиш: <ctrl>+<N> (для Windows) или <command>+<N> (для MacOS).

В основном вспомогательные клавиши <ctrl> в Windows или <command> в MacOS эквивалентны клавишам <meta> в Sun и <alt> в Linux и HP-UX.

Примеры

Просмотрите примеры, которые находятся в директории EXAMPLES LabVIEW. Вы можете использовать эти программы в том виде, как они есть, или изменить их по вашему желанию. Просмотрите примеры с помощью опции **Примеры** (Examples) в меню **Справка**.

4.5. Полезные подсказки

4.5.1. Смена инструментов

Если LabVIEW находится в режиме редактирования, то нажатие на клавишу <tab> переключает друг за другом все инструменты, доступные для данной панели. Если активна лицевая панель, то LabVIEW циклически переходит от инструмента управления к инструменту перемещения, затем к инструменту ввода текста и потом к инструменту раскрашивания. Если активной является блок-диаграмма, то LabVIEW проходит через инструменты аналогичным образом, заменяя инструмент раскрашивания инструментом соединения.

Вы также можете нажимать клавишу пробела, чтобы переключаться между инструментами управления и перемещения на лицевой панели или инструментами соединения и перемещения на блок-диаграмме.

4.5.2. Изменение направления соединяющего проводника

Нажатие клавиши пробела во время соединения элементов изменяет направление, которое имеет проводник, выходящий из терминала. Таким образом, если вы случайно перемещаетесь в горизонтальном направлении от начальной точки (терминала), но хотите, чтобы проводник вначале перемещался вертикально, то нажатие клавиши пробел изменит начальное направление движения из горизонтального в вертикальное.

4.5.3. Отмена операции соединения

Чтобы удалить проводник в процессе соединения, щелкните правой кнопкой мыши (в Windows и UNIX) или выведите курсор мыши за пределы экрана и щелкните там.

4.5.4. Удаление последней точки изменения направления проводника

Щелчок мышью во время соединения элементов изменяет направление проводника. Комбинация <shift>+щелчок (Windows) или <command>+щелчок (MacOS) удаляет последнюю точку изменения направления проводника. При еще одном нажатии <shift>+щелчок удаляется предпоследняя точка и т.д. Если этой точкой является терминал, то команда <shift >+щелчок или <command>+щелчок удаляет весь проводник.

4.5.5. Вставка объекта в существующие соединения

Вы можете вставить объект, например арифметическую или логическую функцию, в существующую схему соединения без разрыва проводника. Щелкните мышью по проводнику в том месте, где вы хотите вставить объект, и выберите функцию **Вставить** (Insert), затем укажите объект, который нужно вставить, в появившейся палитре **Функции**.

4.5.6. Точное перемещение объекта

Вы можете перемещать выбранные объекты на очень малые расстояния с помощью клавиш управления курсором. Удерживайте клавиши в нажатом положении для повторения операции. Для того чтобы переместить объекты на большие расстояния, удерживайте клавишу <shift>, одновременно нажимая на клавишу-стрелку.

4.5.7. Быстрое приращение значений числовых элементов управления

Если вы нажмете на клавишу <shift> и щелкнете по кнопкам увеличения или уменьшения значения числового элемента управления, то их изменение произойдет очень быстро. Значение шага увеличивается на порядок с каждой итерацией: например, на 1, затем на 10, затем на 100 и т.д. При достижении границы диапазона шаг начинает уменьшаться также порядками и достигает нормального значения при приближении к пределу.

4.5.8. Введение разделов в кольцевые списки

Для того чтобы быстро ввести объект в кольцевой список, нажмите <shift>+<enter> или <shift>+<return> после того, как вы напечатали его имя и поместили курсор для введения следующего объекта.

4.5.9. Копирование объекта

Для того чтобы скопировать объекты, выделите их и, удерживая в нажатом положении клавишу <ctrl> (Windows) или <option> (MacOS), перетащите копию в новое место. Оригиналы останутся там, где они были. Разрешается также копировать объекты в окно другого ВП.

4.5.10. Перемещение объекта только в одном направлении

Удерживая клавишу <shift> нажатой во время перемещения или копирования объектов LabVIEW, вы ограничите их перемещение в вертикальном или горизонтальном направлениях в зависимости от первоначального движения мыши.

4.5.11. Сочетание цветов

Чтобы скопировать цвет объекта, щелкните мышью в режиме инструмента копирования цвета. Затем окрасьте другие объекты, щелкнув по ним инструментом раскрашивания.

4.5.12. Замена объектов

Вы легко можете заменить объект лицевой панели или блок-диаграммы, щелкнув по нему мышью и выбрав функцию **Заменить**. Появится палитра **Элементы управления** или **Функции** (в зависимости от окна), где предлагается выбрать новый объект или функцию. Новая функция заменит старую, и все проводники останутся неповрежденными.

4.5.13. Создание дополнительного рабочего пространства

Для увеличения рабочего пространства окна лицевой панели или блок-диаграммы с помощью инструмента перемещения начните выделение области, лежащей за пределами границы окна. На лицевой панели или блок-диаграмме появится линия прокрутки, вы увидите прямоугольник, отмеченный пунктирной линией, который и определяет ваше новое пространство.

4.5.14. Создание собственных палитр

Если вы часто пользуетесь определенным объектом лицевой панели или блок-диаграммы, воспользуйтесь преимуществом «прилипающих» палитр, которые все время остаются открытыми. Для этого оставьте выбранную палитру открытой и щелкните левой кнопкой мыши на кнопке-«булавке» палитры. В LabVIEW также есть опция добавления ВП или созданных пользователем элементов управления к стандартным палитрам, которые служат для быстрого доступа. Чтобы создать палитру, выполните следующие операции:

1. Откройте палитру, которую хотите переделать.
2. Щелкните мышью по кнопке **Опции** (верхний ряд, третья кнопка в каждой палитре).

3. Выберите функцию **Редактировать палитры** (Edit Palettes) из диалогового окна.
4. После выполнения этих операций сохраните новую палитру, дав ей имя. Позднее вы сможете переключаться между созданными палитрами, палитрами по умолчанию и любыми другими, выбирая пункт **Опции** в палитре.

4.5.15. Настройка индивидуальных параметров пользователя

LabVIEW предлагает много настроек, находящихся в подменю **Опции** (Options) меню **Инструменты**. Вы можете выбрать типы настроек в выпадающем меню, расположенном в верхней части диалогового окна **Опции**.

Выберите **Опции** из меню **Инструменты** и посмотрите различные настройки, имеющиеся в вашем распоряжении. Если вы захотите узнать больше о настройках, то прочитайте другие учебные пособия по LabVIEW или воспользуйтесь справкой.

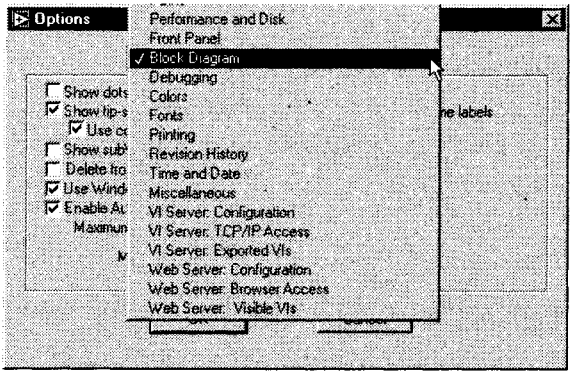


Рис. 4.36

4.6. Итоги

LabVIEW содержит специальные инструменты и методики редактирования, соответствующие его графической природе. Инструмент управления изменяет значение объекта. Инструмент перемещения выделяет, удаляет и перемещает объекты. Инструмент соединения создает проводники, которые соединяют объекты на блок-диаграмме. Инструмент ввода текста создает и изменяет собственные и свободные ярлыки. Собственные ярлыки принадлежат определенному объекту и не могут быть удалены или передвинуты независимо, тогда как свободные ярлыки не имеют таких ограничений.

В LabVIEW есть четыре типа элементов управления и индикаторов: *числовые*, *логические*, *строковые* и *пути*. Каждый имеет отдельный тип данных и особые опции контекстного меню. Терминалы элементов управления и индикаторов, а также проводники на блок-диаграмме обладают цветовыми кодами в соответствии с типом данных. Терминалы и проводники, работающие с числами с плавающей запятой, окрашены в оранжевый цвет, с целыми числами – в синий, с логическими значениями – в зеленый, со строковыми данными – в розовый и с путями данных – в голубовато-зеленый.

Объекты лицевой панели или блок-диаграммы находятся в палитрах **Элементы управления** или **Функции** соответственно. Вы можете получить доступ в эти палитры, щелкнув правой кнопкой мыши по свободному месту лицевой панели или блок-диаграммы.

Для запуска ВП щелкните мышью по кнопке **Запуск** или выберите функцию **Запуск** (Run) в меню **Управление.** Если кнопка **Запуск** не работает, значит, в вашем ВП что-то не в порядке. Прочитайте следующую главу, чтобы узнать хорошие методики отладки программ.

4.7. Дополнительные упражнения

4.7.1. Упражнение 4.3: сравнение чисел

Создайте ВП, который бы сравнивал два входных числа. Если они равны, то зажигается светодиод на лицевой панели. Назовите его **Comparison Practice.vi**.

4.7.2. Упражнение 4.4: простейший калькулятор

Создайте ВП, который мог бы складывать, вычитать, умножать и делить два входных числа и показывать результаты на лицевой панели. Для начала используйте лицевую панель, изображенную на рис. 4.39. Назовите этот ВП **Very Simple Calculator.vi**.

Если вы не смогли выполнить эти упражнения, решения можно найти в папке `EVERYONE\CH4.LLB` на компакт-диске.

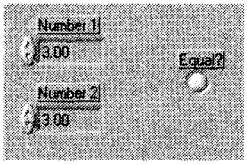


Рис. 4.37

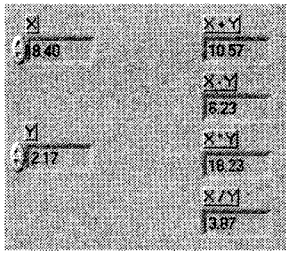


Рис. 4.38

Обзор

В этой главе вы более углубленно изучите основы программирования в среде LabVIEW. Мы поговорим об операциях сохранения и считывания, специальных файлах библиотек, особенностях отладки программ, возможностях использования подприборов, процедурах документирования.

ЗАДАЧИ

- Научиться загружать и сохранять ВП (и делать это правильно)
- Научиться использовать технику отладки программ в LabVIEW
- Создать свой первый виртуальный подприбор и понять, как его применять
- Документировать достижения, чтобы ими могли пользоваться другие

ОСНОВНЫЕ ТЕРМИНЫ

- Библиотека ВП
- Неисправный ВП
- Режим пошагового выполнения
- Узел данных (Node)
- Подсветка выполнения
- Пробник (Probe)
- Точка останова (Breakpoint)
- Виртуальный подприбор (SubVI)
- Редактор иконки
- Рекомендуемые, обязательные и необязательные входные данные

5 И ВНОВЬ ОБ ОСНОВАХ ПРОГРАММИРОВАНИЯ В LabVIEW

5.1. Загрузка и сохранение виртуальных приборов

Естественно, что во время совершенствования навыков работы с LabVIEW вам придется загружать и сохранять виртуальные приборы. LabVIEW имеет много средств, которые помогут в сохранении файлов. В этом разделе речь пойдет о том, как эти средства заставить работать на вас.

Вы можете загрузить ВП, выбрав функцию **Открыть** (Open) из меню **Файл** (File) и затем выбрав ВП из появившегося диалогового окна. Во время загрузки ВП вы увидите окно состояния, которое описывает этот прибор и позволяет отменить процесс загрузки. Разрешается загрузить определенный ВП и одновременно запустить LabVIEW, дважды щелкнув мышью по иконке прибора или, при работе в среде Windows или Macintosh, путем перемещения иконки прибора и совмещения ее с иконкой LabVIEW.

Сохраните ВП, выбрав опцию **Сохранить** (Save) – или подобную опцию – в меню **Файл**. LabVIEW выводит диалоговое окно, предлагая выбрать место, где вы хотите сохранить программу. Если вы сохраняете ВП как индивидуальные файлы, то их имена должны соответствовать ограничениям наименования файлов, имеющимся в операционной системе (например, MacOS 9.x ограничивает имена файлов 31 символом). Некоторые проекты LabVIEW требуют большого количества ВП (сотни), которые могут занять очень много места. Чтобы избежать этих ограничений, сохраните виртуальные инструменты в сжатом виде в специальном файле LabVIEW, называемом *библиотекой виртуальных приборов*. Подробнее об этом читайте в данной главе.

Имейте в виду, что LabVIEW обращается к ВП по имени. Поэтому в оперативной памяти не может быть одновременно двух одноименных ВП: во время поиска LabVIEW загрузит первый попавшийся ВП с этим именем, который может оказаться не тем, который вы хотите.

Обратите внимание, что звездочка отмечает заголовки тех виртуальных приборов, которые вы модифицировали, но пока не сохранили эти изменения. Мы уверены, что вы знаете о необходимости постоянного сохранения вашей работы, мы лишь напоминаем вам о важности этой операции во время работы на компьютере – ведь заранее не знаешь, когда ударит молния.

Никогда не сохраняйте виртуальные приборы в директории `vi.lib`. Эта директория обновляется при установке новой версии LabVIEW, так что если вы сохраните файлы там, то можете потерять свои наработки.

5.1.1. Опции сохранения

Вы можете сохранить ВП, используя одну из четырех опций в меню **Файл**.

Выберите опцию **Сохранить** (Save) для сохранения нового ВП, затем определите для него имя и местоположение на диске или используйте эту опцию для сохранения изменений в существующем ВП в заранее определенном месте.

Рис. 5.1

Укажите опцию **Сохранить как** (Save as) для переименования ВП в памяти и сохранения копии на диске под новым именем. Если вы введете новое имя ВП, то LabVIEW не изменит оригинальную версию прибора, сохраненного на диске. Кроме того, все ВП, находящиеся в этот момент в памяти и вызывающие старый ВП, теперь будут вызывать новый. Если вы не измените имя ВП в диалоговом окне, то LabVIEW попросит уточнить, хотите ли вы переписать первоначальный файл.

Опция **Сохранить с опциями** (Save with Options) выводит диалоговое окно, в котором вы можете выбрать, сохранить ли ВП для его дальнейшего использования в качестве самостоятельной программы, прибора в составе библиотеки

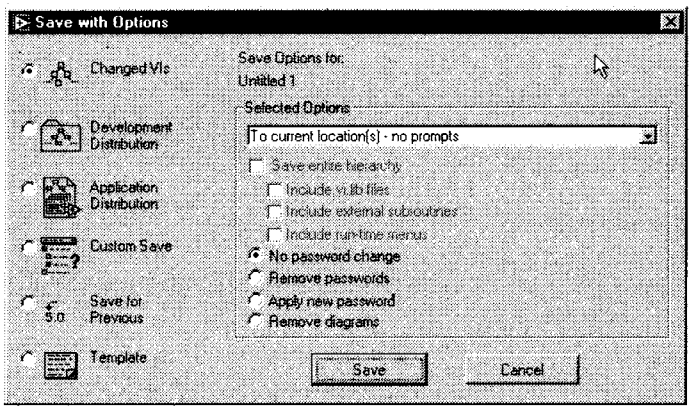


Рис. 5.2

(development distribution), части приложения (application distribution), сохранить всю иерархию ВП или сохранить как ВП, совместимый с предшествующей версией LabVIEW. Вы также можете сохранить ВП без блок-диаграмм или защитить последние введением пароля, но вначале убедитесь, что имеется лишняя копия с блок-диаграммой и/или вы помните пароль в случае необходимости его модификации. Для сохранения определенного ВП в каком-либо новом месте во избежание многочисленных подсказок используйте опцию **В новое местоположение – с одной подсказкой** (To new location – single prompt).

Если вы сохранили ВП без блок-диаграммы, вы не сможете его редактировать. Всегда делайте копию оригинального прибора перед тем, как сохранить его без блок-диаграммы

5.1.2. Возврат в прежнее состояние

Вы можете воспользоваться опцией **Возврат** (Revert) из меню **Файл** для возврата к последней сохраненной версии ВП, с которым в данный момент работаете. Появляется диалоговое окно для подтверждения отмены всех изменений в ВП.

5.1.3. Диалоговые окна сохранения и загрузки

LabVIEW поддерживает формат файлового диалогового окна, который используется вашей системой. Во время открытия или сохранения виртуального прибора появляется файловое диалоговое окно системы. Если щелкнуть мышью по библиотеке виртуальных приборов (специальная структура хранения файлов LabVIEW), то LabVIEW заменит системное диалоговое окно собственным файловым диалоговым окном для того, чтобы вы могли выбрать файлы внутри библиотеки.

Поскольку для выбора и сохранения данных в библиотеке ВП в некоторой степени неудобно пользоваться системным диалоговым окном, вы можете так настроить опции LabVIEW, чтобы работать со стандартным диалоговым окном LabVIEW, особенно если вам нужны в основном библиотеки ВП. Выберите пункт **Опции** в меню **Инструменты**, а затем перейдите в меню **Различное** (Miscellaneous) и снимите галочку **Использовать системное файловое диалоговое окно** (Use native file dialogs).

Диалоговое окно MacOS по сохранению файлов не может работать с библиотеками ВП. Для того чтобы сохранить информацию в библиотеке ВП, щелкните мышью по кнопке **Использовать LLBs** (Use LLBs) или откройте стандартное диалоговое окно LabVIEW вместо имеющегося.

5.1.4. Меню просмотра типа файлов

В нижней части диалогового окна сохранения или загрузки файлов есть выпадающее меню, которое дает возможность увидеть все файлы (**View All**), увидеть лишь файлы ВП и элементов управления (**VIs & Controls**), ВП, шаблонов

(**Templates**) или элементов управления (**Controls**), либо выделять файлы с помощью установленного вами специального шаблона (**Custom Pattern**).

В диалоговых окнах некоторых операционных систем нет опции **Специальный шаблон**

Если вы выберете опцию **Шаблон**, откроется другое окно, в котором вы можете точно определить расширения файлов. В диалоговом окне отобразятся лишь файлы, соответствующие разрешенным расширениям. Обратите внимание, что звездочка появляется внутри окна автоматически, она подразумевает любой символ (или символы).

5.2. Библиотеки виртуальных приборов

Библиотеки виртуальных приборов являются особыми файлами LabVIEW, которые в среде LabVIEW имеют такие же возможности по загрузке, сохранению и открытию, как директории и папки. Вы можете сгруппировать несколько ВП и сохранить их как библиотеку. Библиотеки ВП имеют и преимущества, и недостатки. Например, они могут содержать лишь сжатые версии ВП, но не файлы данных или какие-либо другие. Кроме того, операционная система рассматривает библиотеки ВП как единые файлы, и доступ к их содержимому открывается только из LabVIEW.

Причины использования библиотек LabVIEW:

- можно использовать до 255 символов для наименования файлов;
- переместить библиотеку ВП на другие платформы значительно легче, чем переместить множество отдельных ВП;
- допустимо немного уменьшить объем проекта на диске, поскольку библиотеки ВП сжаты для уменьшения используемого дискового пространства.

Причины сохранения ВП в виде отдельных файлов:

- можно использовать файловую систему для управления отдельными ВП (например, копирование, перемещение, переименование, дублирование) без помощи LabVIEW;
- разрешается использовать поддиректории;
- сохранение ВП и элементов управления в отдельных файлах надежнее, чем если бы вы сохранили весь проект в одном файле;
- допустимо обращаться к встроенным в полную версию LabVIEW элементам управления исходными кодами либо воспользоваться программой разработки исходных кодов другой фирмы.

Обратите внимание на то, что многие ВП, поставляемые вместе с LabVIEW, хранятся в библиотеках ВП, расположенных в соответствующих местах на всех

платформах. Сохраняйте ваши примеры в директории MYWORK, чтобы иметь более легкий доступ к отдельным файлам.

5.2.1. Как пользоваться библиотеками ВП

Создать библиотеку ВП из диалогового окна **Сохранить** или **Сохранить как** можно, щелкнув мышью по кнопке **Новая библиотека ВП** (New VI Library) в Windows или по кнопке **Новый** (New) в MacOS. Если вы работаете в системе Mac, сконфигурированной для использования собственных диалоговых окон, щелкните по кнопке **Use LLBs** в диалоговом окне сохранения и затем выберите **Новый** из появившегося окна.

Введите имя новой библиотеки в появившемся диалоговом окне, показанном на рис. 5.3, и добавьте расширение .llb. Затем щелкните мышью по кнопке **VI library**, чтобы создать библиотеку. Если вы не введете расширение .llb, то LabVIEW добавит его автоматически.

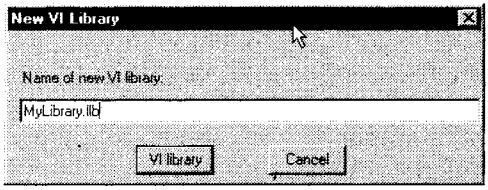


Рис. 5.3

Обычно библиотеку создают в процессе сохранения ВП, поэтому после ее создания появляется диалоговое окно, где предлагается назвать ВП и сохранить его в новой библиотеке.

Как только вы создали библиотеку, вы можете сохранить в ней ВП и получить к ним доступ через LabVIEW как к папкам и директориям, но вы не увидите отдельные ВП из операционной системы. Помните, что в MacOS, которая использует собственные диалоговые окна, нужно выбрать **Use LLB** из диалогового окна **Сохранить** для обеспечения доступа к приборам.

5.2.2. Диалоговое окно редактирования библиотеки ВП

Поскольку вы не можете редактировать содержимое библиотеки ВП инструментами операционной системы, используйте диалоговое окно **Редактировать библиотеку ВП** (Edit VI Library). Это окно, находящееся в меню **Инструменты** (рис. 5.4), демонстрирует список файлов в библиотеке ВП. Двигаясь по списку, можно обнаружить в нижней части диалогового окна даты создания и модификации файла, на котором остановился курсор.

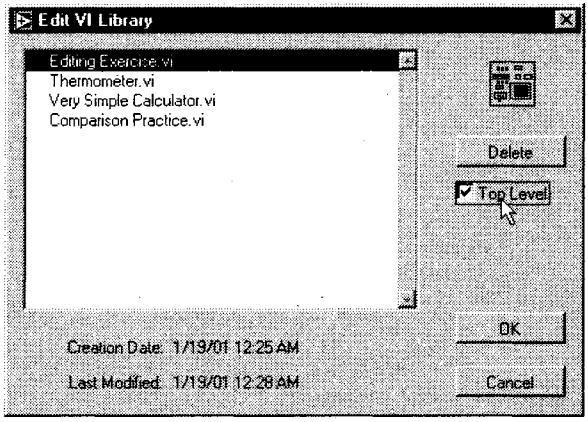


Рис. 5.4

Если вы отметите ВП в качестве прибора верхнего уровня (**Top Level**), то он будет загружаться автоматически при открытии библиотеки ВП. Вы можете иметь несколько помеченных таким образом ВП в любой библиотеке. Имена этих ВП появляются отдельно в верхней части диалогового окна загрузки, что позволяет определить, какие ВП являются главными, а какие подпрприборами.

5.2.3. Менеджер библиотеки ВП

Вы можете применять **Менеджер библиотеки ВП** (VI Library Manager), расположенный в меню **Инструменты**, для упрощения операций копирования, переименования и удаления файлов внутри библиотек ВП. Допустимо воспользоваться этим инструментом для создания новых библиотек и директорий, а также для преобразования библиотек ВП в директории и наоборот. Создание новых библиотек и директорий, преобразование библиотек ВП в директорию и наоборот являются важными операциями в случае манипулирования библиотеками с помощью инструментов управления исходными кодами.

5.3. Методика отладки программ

Вы когда-нибудь создавали программу без ошибок? Наверное, нет. LabVIEW имеет много встроенных инструментов отладки, которые помогут вам разработать собственный ВП. В этом разделе описывается, каким образом использовать эти инструменты с максимальной отдачей.

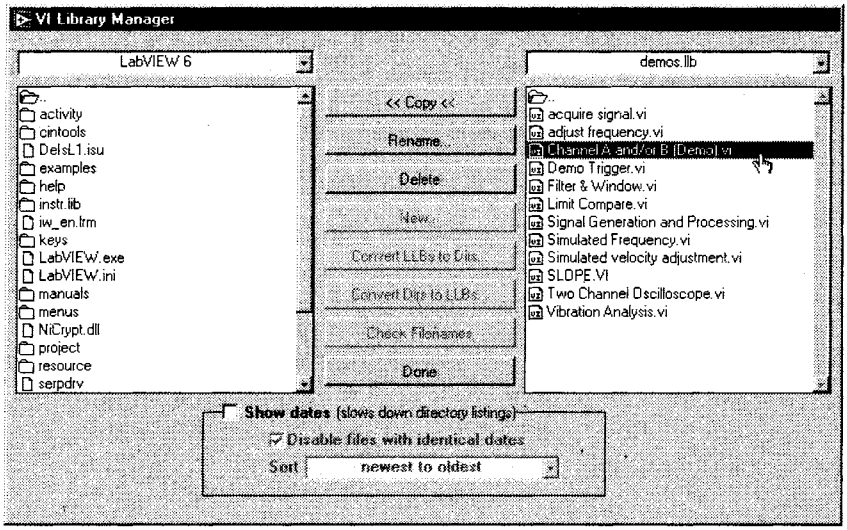


Рис. 5.5

5.3.1. Отладка неисправного ВП


Run Broken

Неработающий ВП нельзя запустить. Кнопка **Запуск** имеет вид сломанной стрелки, указывающей на то, что в ВП имеются неполадки. Такое состояние является совершенно нормальным в процессе создания или редактирования ВП, пока вы не закончите соединение всех терминалов на блок-диаграмме. Иногда потребуется использовать опцию **Удалить неисправные проводники** (Remove Broken Wires), находящуюся в меню **Правка**, для удаления неподключенных проводников, но будьте осторожны – не удалите по ошибке нужные проводники.

Чтобы выяснить причину неработоспособности ВП, щелкните мышью по неисправной кнопке **Запуск** или выберите опцию **Показать список ошибок** (Show Error List) в меню **Окно**. Появится информационное окно **Список ошибок** (Error List), в котором отображаются все ошибки, содержащиеся в данном ВП. При желании вы можете посмотреть список ошибок, относящихся к другому открытому ВП, путем использования выпадающего меню в верхней части окна. Чтобы получить больше информации об определенной ошибке, щелкните по ней мышью. Для определения местонахождения ошибки в ВП сделайте двойной щелчок по ошибке в списке либо выделите ее и нажмите кнопку **Найти** (Find). LabVIEW выведет соответствующее окно на экран и выделит объект, вызвавший данную ошибку.

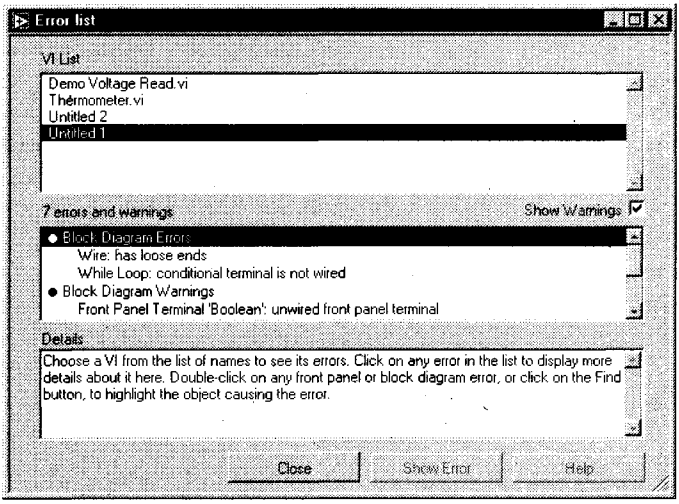


Рис 5.6

5.3.2. Предупреждения


Warning

Если вам потребуется дополнительная помощь при отладке программы, то можете использовать опцию **Показать предупреждения** (Show Warning) в окне **Список ошибок**, сделав отметку в соответствующем поле. Наличие предупреждения не является недопустимым и не приводит к неработоспособности программы. Предупреждение не влияет на LabVIEW, так же как и терминал элемента управления, который не подключен ни к одному элементу. Если вы поставили метку **Показать предупреждения** и у вас есть какие-нибудь предупреждения, то на линейке инструментов появится кнопка **Предупреждение** (Warning). Чтобы увидеть окно со списком предупреждений (в окне **Список ошибок**), щелкните по этой кнопке. В окне будут объясняться причины предупреждений.

Также вы можете сконфигурировать опции LabVIEW, чтобы показывать предупреждения по умолчанию. Перейдите в меню **Отладка** (Debbuging) в диалоговом окне **Опции** (выбрав **Инструменты** ⇒ **Опции**) и поставьте галочку в окне **По умолчанию показывать предупреждения в окне ошибок** (Show warnings in error box by default).

5.3.3. Наиболее распространенные ошибки

Одни ошибки делаются намного чаще, чем другие, поэтому мы решили объединить их в список, чтобы вы в дальнейшем смогли избежать их повторения. Если кнопка запуска вашего ВП в нерабочем состоянии, то, возможно, устранение одной из перечисленных ниже неисправностей поможет решить проблему.

- функциональный ввод, требующий данных, не подключен. Во время работы ВП при тестировании разных алгоритмов ни один объект на блок-диаграмме не должен оставаться неподключенным;
- на блок-диаграмме имеется неисправный проводник, который вышел из строя из-за несовпадения типа данных либо один конец которого не подключен, так как скрыт под чем-нибудь или настолько мал, что его трудно увидеть. Команда **Удалить неисправные проводники** в меню **Файл** удаляет неисправные проводники, но отыскать несоответствия между типами данных вы должны сами;
- неисправен ВПП, или вы отредактировали его соединительную панель после помещения иконки на блок-диаграмму. Используйте опции контекстного меню **Заменить** или **Выполнить повторную компоновку ВПП** (Relink to SubVI), чтобы вновь подключится к ВПП;
- у вас ошибка из-за объекта, который является поврежденным, невидимым или переделанным с помощью узла свойств (подробнее об этом в главе 12);
- вы случайно соединили два элемента управления или подключили два элемента управления к одному и тому же элементу отображения. В окне **Список ошибок** появляется надпись: «Signal: has multiple sources» (Сигнал имеет много источников). Обычно такая ошибка исправляется заменой одного из элементов управления на элемент отображения.

5.3.4. Пошаговое выполнение ВП


Pause Button

Во время отладки программы иногда нужно выполнять блок-диаграмму узел за узлом. *Узлы* данных включают в себя подприборы, функции, структуры, узлы взаимодействия с программным кодом, формульные узлы и узлы свойств. Чтобы начать пошаговое выполнение, следует запустить ВП щелчком мыши по одной из кнопок пошагового выполнения (вместо кнопки **Запуск**), затем временно остановить ВП, введя точку останова или щелкнув мышью по кнопке **Пауза**. Для возобновления выполнения программы вновь нажмите кнопку **Пауза**.

Вы можете использовать подсветку выполнения программы (см. ниже) во время пошаговых действий с виртуальным прибором и, таким образом, визуально проследить за данными во время их перехода из одного узла в другой.

В режиме *пошагового выполнения* нажмите любую из трех кнопок, находящихся в активном состоянии, для перехода к следующему шагу. Каждая кнопка определяет способ выполнения следующего шага.


Step Into

Нажмите кнопку **Шаг внутрь** (Step Into), после этого произойдет выполнение первого шага ВПП или структуры, а затем – пауза. Можно также нажимать клавишу «стрелка вниз» одновременно с клавишей <ctrl> в Windows или <command> в Mac.


Step Over

Нажмите кнопку **Шаг через** (Step Over). После этого произойдет выполнение ВПП или структуры (последовательности, цикла и т.п.),

а затем – пауза. Также удобна клавиша «стрелка вправо» одновременно с клавишей <ctrl> в Windows или <command> в Mac.


Step Out

Нажмите кнопку **Шаг из** (Step Out) для окончания выполнения текущей блок-диаграммы, структуры или ВПП. После этого действия будет пауза. Либо можете использовать клавишу «стрелка вверх» одновременно с клавишей <ctrl> в Windows или <command> в Mac.

5.3.5. Подсветка при выполнении программы


Execution Highlighting

Иногда удобно визуально отследить, где находятся ваши данные и что с ними происходит. В LabVIEW вы можете наблюдать анимацию выполнения блок-диаграммы ВП. Для включения этого режима нажмите кнопку **Подсветка выполнения** (Execution Highlightning) на линейке инструментов.

Движение данных из одного узла в другой отмечается перемещающимися вдоль проводников кружочками. Вы заметите, что подсвечивание значительно снижает производительность ВП. Еще раз щелкните мышью по кнопке **Подсветка выполнения** для возобновления нормального выполнения программы.

Если выбрать опцию **Автопробник** (Auto probe) в меню **Отладка** диалогового окна **Опции**, то во время работы в режиме подсветки значения узлов данных будут показываться автоматически.

Обычно подсветкой пользуются в режиме пошаговых действий для того, чтобы увидеть, как данные проходят через узлы. Если эти режимы применяются вместе, то рельефные изображения выполнения на иконках подпрограмм будут показывать, какие ВП находятся в работе, а какие ждут своей очереди.

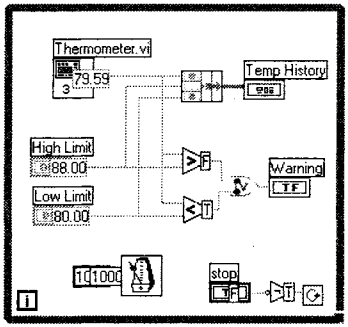


Рис. 5.7

5.3.6. Инструмент установки отладочных индикаторов (пробник)

Используйте **Пробник** (probe) для проверки промежуточных значений в ВП, который при выполнении выдает сомнительные или неожиданные результаты. Например, представьте, что у вас есть диаграмма с рядом операций, каждая из

которых может явиться причиной выдачи неправильных данных. Чтобы установить это, надо создать элемент отображения для демонстрации промежуточных результатов в проводнике либо, оставив ВП в режиме выполнения, воспользоваться пробником. Чтобы задействовать пробник, выберите инструмент установки отладочных индикаторов в палитре **Инструменты** и щелкните его курсором по проводнику, либо щелкните правой кнопкой мыши по проводнику и выберите опцию **Пробник** (Probe). Если ВП не находится в режиме выполнения, то дисплей пробника (плавающее окно) вначале оказывается пустым. При запуске ВП дисплей пробника показывает значение, проходящее через соответствующий проводник.

Вы можете пользоваться пробником в режиме подсветки и в пошаговом режиме, чтобы сделать просмотр значений более удобным. Каждый пробник и относящийся к нему проводник автоматически нумеруются LabVIEW для облегчения их отслеживания. Номер пробника может быть невидимым, если имя проверяемого объекта длиннее, чем само окно пробника. Если вы не знаете, какой пробник относится к какому проводнику, щелкните правой кнопкой мыши по пробнику или проводнику и выберите либо **Найти пробник** (Find Probe) либо **Найти проводник** (Find wire), чтобы выделить соответствующий объект.

С помощью пробника вы не сможете изменить данные.

В процессе установки отладочных индикаторов вы также можете использовать любой элемент отображения путем выбора опции **Произвольный пробник** (Custom Probe) в контекстном меню проводника, а затем указать необходимый элемент отображения, при помощи которого в дальнейшем будет осуществляться проверка. Например, для этих целей применяется развертка осциллограммы, чтобы показать изменение переменной в цикле, поскольку график демонстрирует как старые, так и текущие величины. LabVIEW не позволяет выбрать элемент отображения другого типа данных, кроме типа самого проводника.

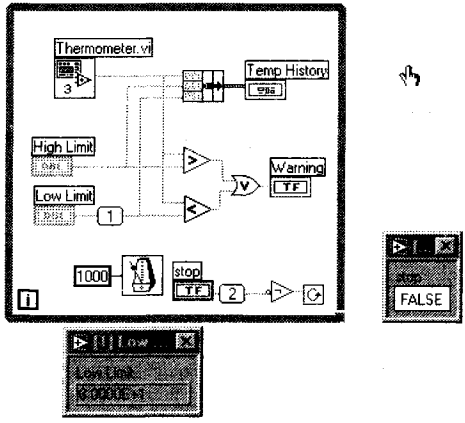


Рис 5.8

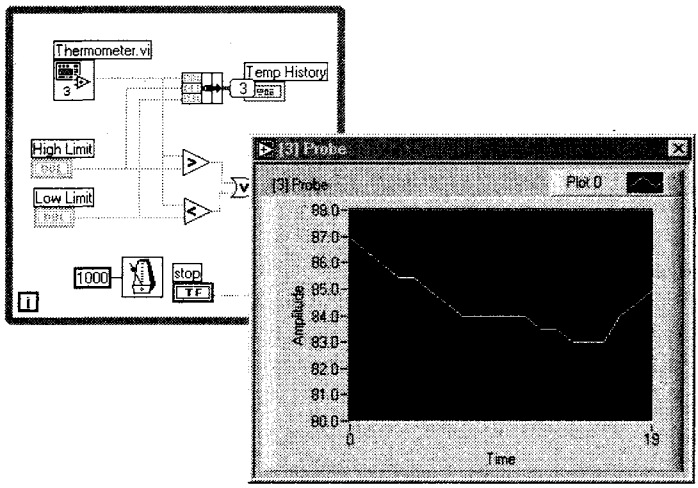


Рис. 5.9

5.3.7. Использование точек останова выполнения программы

Не поддавайтесь панике – *точки останова* (breakpoints) не повреждают ВП, они лишь останавливают его выполнение для проведения его отладки. Паузы нужны тогда, когда вы хотите проверить данные, поступающие в ВП, узел или проводник во время выполнения программы. Когда выполнение диаграммы достигает точки останова, она активизирует кнопку **Пауза**. После этого можно провести пошаговое выполнение программы, прозондировать данные в проводниках, изменить величины объектов на лицевой панели или просто продолжить выполнение программы путем нажатия кнопки **Пауза** или кнопки **Запуск**.


Breakpoint Tool

Для того чтобы установить точку останова, щелкните по объекту блок-диаграммы инструментом ввода контрольной точки из палитры **Инструменты**. Чтобы ликвидировать точку, снова щелкните по объекту. Появление курсора означает установление или ликвидацию остановки.

В зависимости от расположения точки останова ведут себя по-разному:


Clear Breakpoint

- если точка установлена на *блок-диаграмме*, то вокруг диаграммы появляется кайма красного цвета и пауза возникает при завершении выполнения блок-диаграммы;


Set Breakpoint

- если точка установлена в *узле данных*, то он окружается каймой красного цвета и пауза возникает перед выполнением узла данных;
- если точка установлена на *проводнике*, то на нем появляется кружок красного цвета, а любой пробник для этого проводника будет окружен красной каймой. Пауза возникает после прохождения данных через проводник.

Когда выполнение ВП затормозится из-за точки останова, окно блок-диаграммы выйдет на передний план, и объект, вызывающий остановку, будет мигать.

Точки останова сохраняются вместе с ВП, но активизируются только в процессе выполнения.

5.3.8. Временное прекращение выполнения программы

Вы можете устанавливать или устранять точки останова с помощью опции **Останавливать при вызове** (Suspend when Called), находящейся в меню **Выполнение** (Execution) опции **Свойства ВП** (VI Properties), доступ в которую вы можете получить из контекстного меню иконки ВП на лицевой панели. Опция **Останавливать при вызове** останавливает выполнение ВП при всех вызовах, которые используют этот ВП. Если ВПП вызывается из двух мест на блок-диаграмме, то точки останова сработают при обоих вызовах этого ВПП.

Если вы хотите, чтобы точка останова приостановила выполнение программы только при определенном вызове ВПП, то установите точку с помощью опции **Настройка узла ВПП** (Sub VI Node Setup). Чтобы найти эту опцию, вызовите контекстное меню ВПП (на блок-диаграмме вызывающего ВП). Более подробно об опциях настройки ВП вы узнаете в главе 13.

5.3.9. Упражнение 5.1: отладка программы

В данном упражнении мы будем искать и устранять неисправности в неработающем ВП. Затем вы попрактикуетесь в применении других инструментов отладки программ, включая подсветку во время выполнения, режим пошаговых действий и использование пробника.

1. Откройте ВП **Debug Exercise.vi**, находящийся в директории `EVERYONE\CH5.LLB`.
2. Переключитесь на блок-диаграмму. Обратите внимание, что стрелка запуска программы находится в нерабочем состоянии. Вы должны выяснить, почему, и исправить положение, так чтобы ВП мог заработать.

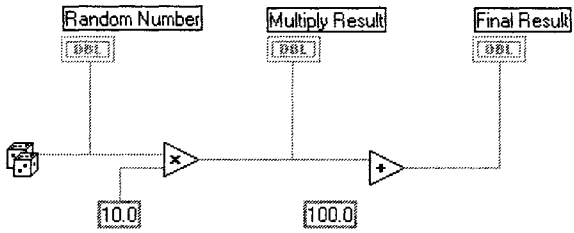


Рис. 5.10

3. Щелкните кнопкой мыши по сломанной стрелке. Появляется окно **Список ошибок** (рис. 5.11), демонстрирующий ошибки в ВП.
4. Щелкните мышью по надписи«Add: contains unwired or bad terminal». После этого в окне **Список ошибок** вы увидите более подробное описание ошибки. Теперь дважды щелкните мышью по ошибке или по кнопке **Найти**. LabVIEW выделит виновную функцию по блок-диаграмме, что облегчит вам поиск ошибки.
5. Создайте отсутствующий проводник данных. После этого кнопка **Запуск** будет работоспособной. Если этого не произойдет, используйте опцию **Удалить неисправные проводники**.

Если вы не можете найти потерянный проводник, подумайте, сколько входов должно быть у функции **Сложение** (Add).

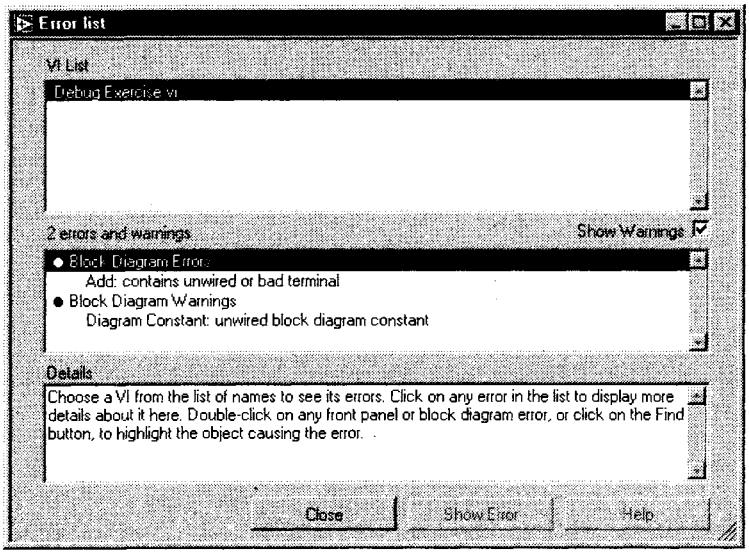


Рис. 5.11

6. Переключитесь вновь на лицевую панель и запустите ВП несколько раз.

Execution Highlighting

7. Расположите лицевую панель и блок-диаграмму (используя команду **Расположить** (Tile) в меню **Окно**) таким образом, чтобы вы могли их видеть одновременно. Задействуйте подсветку выполнения и запустите ВП в пошаговом режиме путем нажатия соответствующих кнопок на линейке инструментов блок-диаграммы.

Step Into

8. Щелкайте мышью по кнопке **Шаг через** всякий раз, когда хотите, чтобы выполнялся узел данных (или щелкните мышью по кнопке **Шаг из**, чтобы закончить выполнение содержимого блок-диаграммы). Обратите

внимание на появление данных на лицевой панели по мере пошагового выполнения программы. Сначала ВП генерирует случайное число, а затем умножает его на 10,0. И наконец, ВП добавляет к результату умножения 100.0. Обратите внимание, как обновляется каждый из элементов отображения на лицевой панели при поступлении новых данных на терминалы блок-диаграммы, что является прекрасным примером программирования потока данных. Помните, что вы можете выйти из пошагового режима и завершить работу ВП нажатием кнопки **Пауза**. Всплывающие подсказки, характеризующие назначение одношаговых кнопок, изменяют свое содержание с целью более точного описания операций, которые они произведут при щелчке по ним мышью.

9. А теперь задействуйте пробник, щелкнув правой кнопкой мыши по любому сегменту проводника данных и выбрав опцию **Пробник**.
10. Еще раз выполните программу по шагам и отметьте, как пробник демонстрирует данные, проходящие через соответствующий проводник.

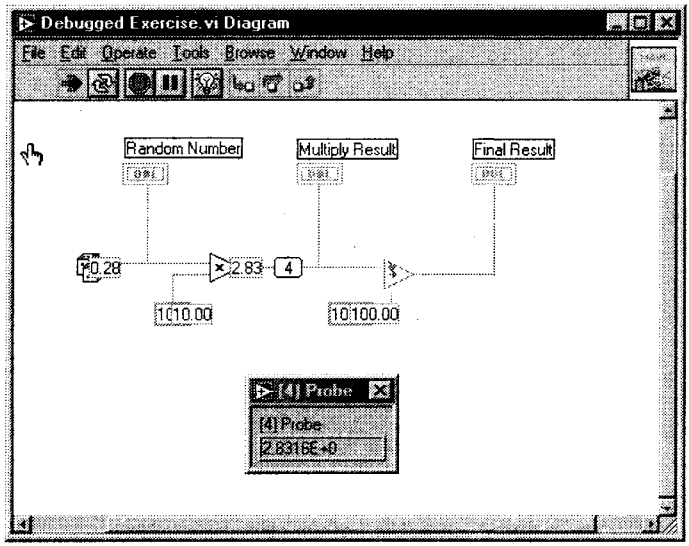


Рис. 5.12

11. Выключите подсветку выполнения программы, щелкнув по кнопке подсветки.
12. Сохраните ваш ВП в директории MYWORK, выбрав опцию **Сохранить как** из меню **Файл**. Назовите его **Debbuged Exercise.vi**. Если вы рисковый человек или считаете, что библиотеки ВП лучше подойдут для ваших целей, то попытайтесь создать библиотеку ВП и сохранить в ней вашу работу.
13. Закройте ВП, выбрав опцию **Закрыть** в меню **Файл**.

5.4. Создание подприборов

Большие возможности и удобство использования LabVIEW обусловлены модульным принципом его программ. Создавая и применяя подприборы (ВПП), вы можете строить ВП модуль за модулем. Подприбор представляет собой ВП, используемый в другом ВП (или вызванный другим ВП). Узел данных ВПП (состоящий из иконки и соединительной панели) аналогичен вызову подпрограммы в обычных языках программирования. Блок-диаграмма может содержать несколько одинаковых узлов данных ВПП, которые вызывают один и тот же ВПП несколько раз.

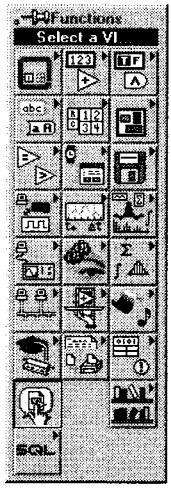


Рис. 5.13

Допустимо использовать любой виртуальный инструмент как ВПП на блок-диаграмме другого ВП при условии, что созданы иконка и соединительная панель этого инструмента. Разместите существующие ВП на блок-диаграмме, чтобы пользоваться ими как ВПП с помощью кнопки **Выбрать ВП** (Select a VI) в меню палитры **Функции**. Выбор этой опции вызывает диалоговое окно менеджера файлов, в котором можно указать любой ВП. После этого его иконка появится на блок-диаграмме.

ВП не может вызывать сам себя непосредственно в качестве ВПП. Если вам действительно это необходимо, используйте косвенный вызов прибора – **Ссылку на ВП**, о котором мы будем говорить в главе 12.

5.4.1. Создание виртуального подприбора на основе ВП

Прежде чем использовать заданный ВП в качестве ВПП, вы должны определить, каким образом он будет получать данные из вызывающего ВП и передавать их обратно. Для этого необходимо назначить элементы управления и отображения данного прибора терминалам его соединительной панели. Вы также должны создать иконку для представления этого ВП на блок-диаграмме другого.

Создание иконки

Каждый ВПП должен иметь иконку для представления его на блок-диаграмме вызывающего ВП. Иконка – это графический символ ВПП. Вы можете создать иконку, выбрав опцию **Редактировать иконку** (Edit Icon) из контекстного меню иконки в верхнем правом углу лицевой панели. Вы должны находиться в режиме редактирования, чтобы войти в это меню.

Открыть *Редактор иконки* (Icon Editor) можно, дважды щелкнув мышью по иконке. Появляется окно Редактора, как показано на рис. 5.15.

Инструменты для создания иконки представлены в табл. 5.1.

Таблица 5.1

	Карандаш	Рисует и стирает элементы изображения
	Линия	Рисует прямые линии. Нажмите <shift> для ограничения рисования линий только в горизонтальном, вертикальном и диагональном направлениях
	Пипетка	Копирует цвет переднего плана с элемента в иконке. Используйте клавишу <shift> для выбора «пипеткой» цвета заднего плана
	Наполненное ведро	Заполняет выделенную область цветом переднего плана
	Прямоугольник	Рисует прямоугольник в цвете заднего плана. Дважды щелкните мышью по этому инструменту для окаймления иконки цветом переднего плана. Используйте клавишу <shift>, чтобы трансформировать прямоугольник в квадрат
	Наполненный прямоугольник	Рисует прямоугольник, окаймленный цветом переднего плана и наполненный цветом заднего плана. Дважды щелкните мышью для окаймления иконки цветом переднего плана и наполните ее цветом заднего плана
	Выбор	Выбирает область иконки для перемещения, копирования или других изменений
A	Текст	Вводит текст в иконку. Дважды щелкните на этом инструменте для изменения атрибутов шрифта
	Передний план/задний план	Показывает текущий цвет переднего и заднего планов. Щелкните мышью на каждом для получения палитры, из которой вы можете выбирать новые цвета

Кнопки в правой части редактирующего экрана выполняют следующие функции:

- **Undo**: отменяет последнюю операцию редактирования;
- **OK**: сохраняет ваш рисунок как иконку ВП и возвращает его в окно лицевой панели;
- **Cancel**: возвращает в окно лицевой панели без сохранения изменений.

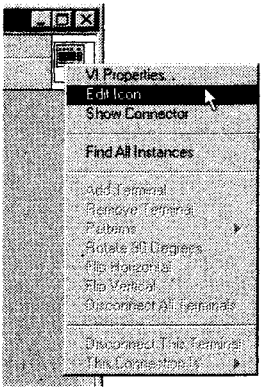


Рис. 5.14

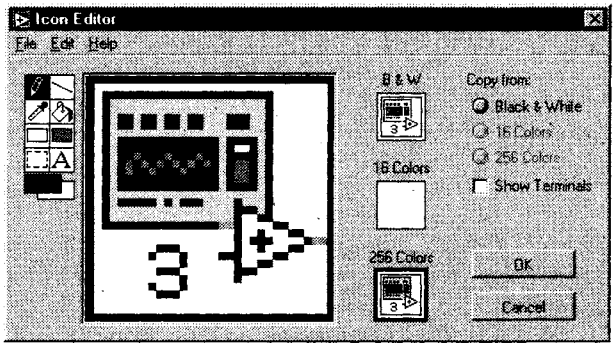


Рис. 5.15

Хотя в настоящее время эта операция практически не употребляется, вы можете создать отдельную иконку для дисплеев, работающих в монохроматическом, 16- или 256-цветном режиме (эта особенность является наследием более ранних версий LabVIEW, когда приходилось пользоваться монохроматическими или 16-цветными мониторами). Вы можете создать и сохранить каждую версию иконки по отдельности, а также трансформировать цветную иконку в черно-белую (или наоборот), используя кнопки **Скопировать из** (Copy from). ВП всегда должен иметь, по крайней мере, черно-белую иконку, так как цветные иконки не появляются в меню палитры и не могут демонстрироваться на черно-белых мониторах. Если черно-белой иконки нет, LabVIEW покажет пустую иконку.

Назначение функций терминалам соединительной панели

Прежде чем использовать ВП в качестве ВПП, вам необходимо задать функции терминалам соединительной панели точно так же, как вы определяете параметры для подпрограммы в обычном языке программирования. Соединительная панель – это механизм LabVIEW для передачи данных в ВПП и извлечения их из него. Соединительная панель ВП связывает элементы управления и отображения с входными и выходными вводами. Чтобы настроить соединительную панель прибора, щелкните правой кнопкой мыши по иконке прибора и выберите опцию **Показать соединительную панель** (Show Connector), а если вы хотите снова увидеть иконку, щелкните правой кнопкой мыши по соединительной панели и выберите пункт **Показать иконку** (Show Icon). LabVIEW формирует соединительную панель, базируясь на числе элементов управления и отображения лицевой панели прибора. Если вы хотите выбрать другую соединительную панель, используйте меню **Шаблоны** (Patterns) из контекстного меню соединительной панели. Вы также можете вращать и переворачивать соединительную панель, если она неудобно расположена в пространстве, – для этого служат команды в контекстном меню панели.

Для назначения терминала элементу управления или отображения произведите следующие операции:

1. Щелкните мышью по терминалу соединительной панели. Курсор автоматически превратится в инструмент соединения и окрасится в черный цвет, как это показано на рис. 5.16.
2. Щелкните мышью по элементу управления или индикатору, который должен быть назначен выбранному терминалу. Этот элемент окаймляется движущейся пунктирной линией (рис. 5.17).
3. Щелкните мышью по открытой области лицевой панели. Пунктирная линия исчезнет, а терминал окрасится в цвет, соответствующий типу назначенного элемента (рис. 5.18), что означает правильное подключение элемента управления или отображения к терминалу.

Если терминал имеет белый или черный цвет, то подключение является неправильным. Повторите в случае необходимости алгоритм подключения, описанный выше. Соединительная панель вашего ВП может включать до 28 терминалов.

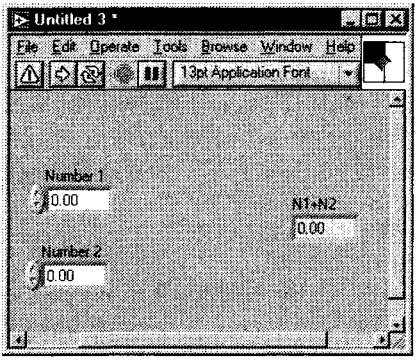


Рис. 5.16

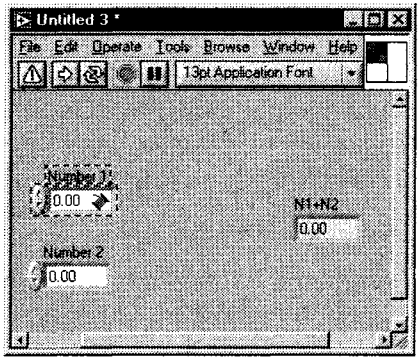


Рис. 5.17

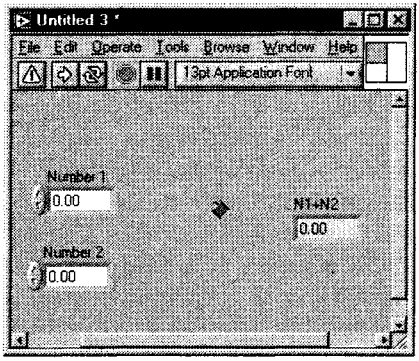


Рис. 5.18

Вы можете поменять порядок первых двух этапов.

В случае ошибки воспользуйтесь кнопкой **Отключить** (Disconnect) из контекстного меню соединительной панели, чтобы отключить определенный терминал, или кнопкой **Отключить все** (Disconnect All), чтобы отключить все терминалы.

5.4.2. Создание ВПП из блок-диаграммы

Иногда вам даже не приходит в голову, что проще было бы использовать ВПП для определенного участка кода. К счастью, вы можете создать ВПП путем преобразования части кода в ВП. Задействуйте инструмент перемещения для выделения части блок-диаграммы, которую следует заменить ВПП, затем выберите опцию **Создать ВПП** (Create SubVI) из меню **Правка** и понаблюдайте, как LabVIEW замещает эту часть ВПП с правильными подключениями и иконкой. После этого вы можете дважды щелкнуть по новому ВПП, чтобы увидеть его лицевую панель, отредактировать иконку, посмотреть на его соединительную панель и сохранить под новым именем. Используйте опцию **Создать ВПП** с осторожностью, поскольку можете получить неожиданные результаты (см. главу 13).

5.4.3. Окно помощи ВПП: рекомендуемые, обязательные и необязательные входные данные

Если вы вызовете окно помощи ВПП, находящегося на блок-диаграмме, то в нем появится его описание и схема подключения. Ярлыки входных вводов отобразятся слева, а выходных – справа. Чтобы увидеть параметры и описание действующего ВП, вы также можете вызвать контекстное окно помощи, наведя курсор мыши на его иконку. Как редактировать описание вы узнаете в следующем разделе.

Функции, встроенные в LabVIEW, автоматически определяют, подключили ли вы необходимые входные данные. В противном случае ВП работать не будет до тех пор, пока вы этого не сделаете. Вы можете так сконфигурировать ВПП, чтобы они, аналогично функциям, имели обязательные, рекомендуемые и необязательные входные данные. Если вы не подаете входные данные на вводы, которые являются *обязательными*, то не сможете запустить ВП в качестве подпрограммы. Если входные (выходные) данные не поступают (снимаются) на вводы, которые являются *рекомендуемыми*, вы сможете запустить ВП, но в окне **Список ошибок** появится предупреждение (если этот режим включен), информирующее о том, что эти вводы не подключены. Если входные данные являются *необязательными*, то не имеется никаких ограничений и подключение часто рассматривается как успешное.

Чтобы определить, является ли подключение обязательным, рекомендуемым или необязательным, щелкните правой кнопкой мыши по терминалу подключения соединительной панели и используйте выпадающее меню **Этот ввод** (This Connector Is).

Отметка рядом с опциями **Обязательный** (Required), **Рекомендуемый** (Recommended) или **Необязательный** (Optional) определит его действующее состояние.

В окне контекстной помощи обязательные соединения показаны жирным шрифтом, рекомендуемые имеют вид простого текста, а необязательные окрашены в серый цвет. Если окно помощи настроено на простой обзор, то необязательные соединения будут скрыты.

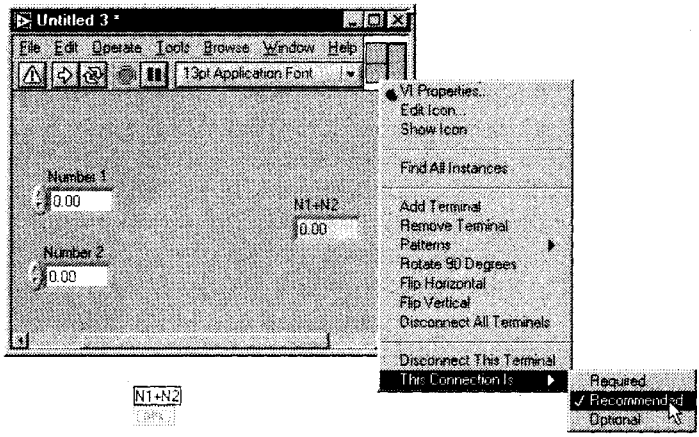


Рис. 5.19

5.5. Документирование работы

Документирование ВП является важным процессом. Оно необходимо, чтобы другие люди понимали назначение и функционирование ВП и чтобы вы помнили, что и зачем делали. В этом подразделе описываются несколько способов документирования работы в LabVIEW.

5.5.1. Создание описаний и подсказок для отдельных объектов

Если вы хотите ввести описание объекта LabVIEW, такого как элемент управления/отображения или функция, то выберите опцию **Описание и подсказка** (Description and Tip) из контекстного меню объекта. Введите описание в появившееся диалоговое окно **Описание** (Description), показанное на рис. 5.20, и щелкните по **ОК** для его сохранения. Вы также можете ввести подсказку в окно **Подсказка** (Tip).

LabVIEW показывает текст описания в контекстном окне помощи, когда вы проводите курсором по элементу управления или отображения на лицевой панели. Подсказка отображается как всплывающая надпись при остановке курсора на

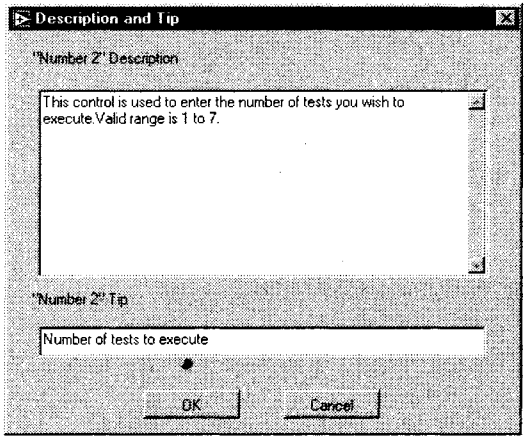


Рис. 5.20

объекте лицевой панели в режиме работы ВП вне зависимости от того, открыто окно помощи или нет.

Лучшим способом организации быстрой помощи при работе с вашими ВП является ввод подсказок и описаний для всех их элементов управления и отображения, а также для функций прибора.

5.5.2. Документирование ВП с помощью опции Свойства ВП

LabVIEW предоставляет легкий способ документирования всего ВП. При выборе в меню **Файл** опции **Свойства ВП** ⇒ **Документирование** (Documentation) появляется диалоговое окно документации действующего ВП.

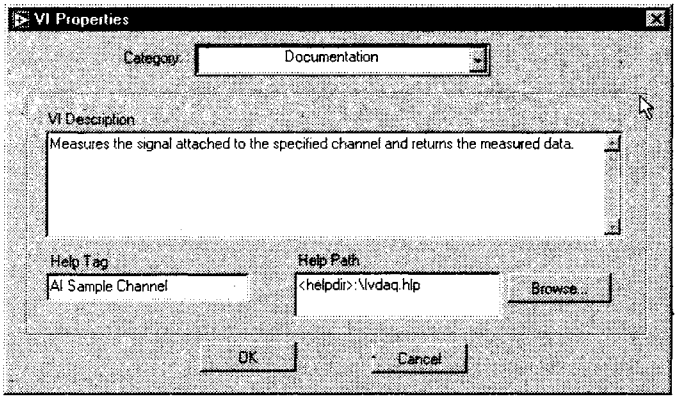


Рис. 5.21

Вы можете использовать диалоговое окно **Свойства ВП** для осуществления следующих действий:

- введение в поле **Категория – Документирование** описания виртуального прибора. Область описания имеет полосу прокрутки, так что вы можете отредактировать или просмотреть очень длинные характеристики. Если вы используете ВП в качестве ВПП, то контекстное окно помощи покажет это описание, когда курсор находится над иконкой ВПП на блок-диаграмме. По желанию вы можете создать ярлык для окна помощи (Help tag) и маршрут к внешнему файлу помощи;
- просмотр в категории **Общие** (General) списка изменений, сделанных в виртуальном приборе с момента его последнего сохранения, путем нажатия кнопки **История изменений** (Revision History);
- просмотр в категории **Общие** пути ВП (то есть места, где он хранится);
- проверка в категории **Использование памяти** (Memory Usage), какой объем памяти занимает ВП. Часть окна, описывающая использование памяти, показывает объем оперативной памяти системы, занимаемый ВП. Цифра указывает лишь на объем памяти, занятой самим ВП, а не его ВПП;
- многие другие интересные вещи, описание которых мы оставляем на дополнительные главы.

5.6. Немного о распечатке виртуальных приборов

LabVIEW предлагает три типа распечатки, которые вы можете использовать для сохранения копии вашей работы на бумаге:

- опция **Печать текущего окна** (Print Window) в меню **Файл** служит для распечатки содержимого действующего окна;
- опция **Печать** (Print) в меню **Файл** позволяет сделать полную распечатку ВП, включая информацию о лицевой панели, блок-диаграмме, подприборах, элементах управления, истории ВП и т.п. Различные настройки печати, в которых позволяют определить необходимый формат. Допустимо не только распечатывать копии на бумаге, но также печатать в файлы форматов HTML и RTF. Более подробно об этом рассказывается в главе 15;
- программируемая печать LabVIEW осуществляет распечатки лицевых панелей ВП под контролем вашего приложения. Выберите опцию **Распечатать по завершении** (Print at Complition) в меню **Действие** (Operate) для использования программируемой печати. LabVIEW распечатает содержание лицевой панели после завершения выполнения ВП. Если виртуальный прибор представляет собой ВПП, то прежде чем вернуться к вызывающей программе, LabVIEW выполнит печать ВПП по завершении его выполнения (см. главу 15).

5.7. Упражнение 5.2: создание ВПП – практикуясь, вы совершенствуетесь

Теперь вновь вернемся к компьютеру. Нужно превратить ВП **Thermometer**, который вы создали в предыдущей главе, в ВПП для использования его на блок-диаграмме другого ВП.

1. Откройте **Thermometer.vi,** который вы создали в упражнении 4.2. Если он находится в директории `MYWORK` (или в библиотеке), то вы его легко найдете. Если вы не можете его найти, воспользуйтесь прибором **Thermometer.vi**, находящимся в библиотеке `EVERYONE\CH4.LLB`.
2. Создайте иконку для ВП. Щелкните правой кнопкой мыши в поле иконки на лицевой панели и выберите опцию меню **Редактировать иконку**, чтобы открыть Редактор иконки. Используйте инструменты, описанные в разделе 5.4.1, для создания иконки; затем щелкните по кнопке **ОК** для возврата к виртуальному прибору. Ваша иконка появится в правом верхнем углу прибора, как это показано на рис. 5.22.

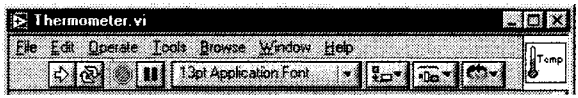


Рис. 5.22

3. Создайте соединительную панель, щелкнув правой кнопкой мыши в поле иконки и выбрав опцию **Показать соединительную панель**. Поскольку есть только один элемент отображения на лицевой панели, то и поле будет иметь только один терминал, который появится в виде окна белого цвета (рис. 5.23).

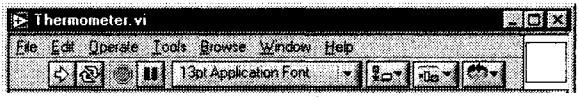


Рис. 5.23

4. Назначьте терминал элементу отображения термометра. Используя инструмент соединения (возможно, курсор автоматически примет его форму), щелкните по терминалу соединительной панели. Терминал окрасится в черный цвет. После этого щелкните по индикатору термометра. Индикатор будет обведен движущейся пунктирной линией. Сделайте щелчок в открытой области на лицевой панели. Пунктирная линия исчезнет, выбранный терминал панели перекрасится из черного цвета

в серый – значит, ему назначен элемент отображения. Щелкните правой кнопкой мыши и выберите опцию **Показать иконку** для возврата к иконке.

5. Задокументируйте элемент отображения температуры, выбрав опцию **Описание и подсказка** в контекстном меню. Введите описание и подсказку, как показано на рис. 5.24, и щелкните по **ОК** при завершении.
6. Документируйте ваш прибор **Thermometer.vi**, выбрав из меню **Файл** опцию **Свойства ВП ⇒ Документирование** и впечатав описание его функции (рис. 5.25). Щелкните по **ОК**, чтобы вернуться к виртуальному прибору.

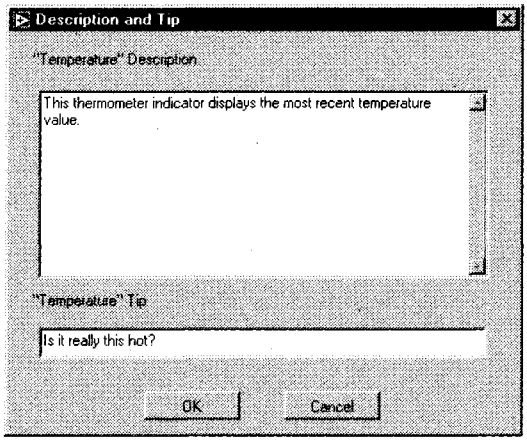


Рис. 5.24

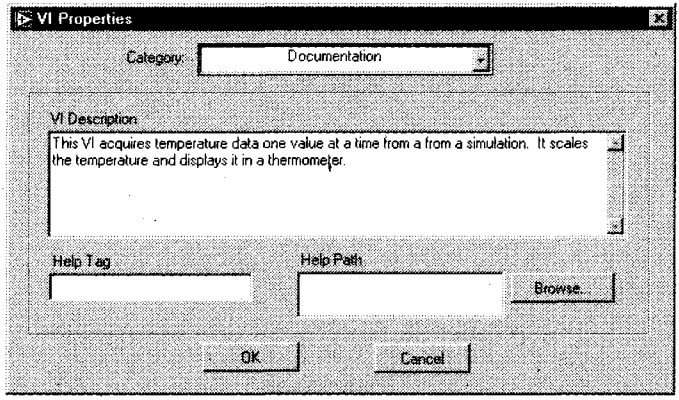


Рис. 5.25

7. Теперь выведите контекстное окно помощи, выбрав опцию **Показать окно контекстной помощи** (Show Context Help) в меню **Справка**. Задержав курсор над полем иконки, вы увидите в окне помощи описание и схему подключения виртуального прибора. Если элемент отображения температуры не имеет ярлыка на лицевой панели, то он также не будет иметь его в окне помощи.
8. Если у вас есть подключенный к компьютеру принтер, то выберите опцию **Печать текущего окна** (Print Window) в меню **Файл** для распечатки активного окна. Вы можете по желанию распечатать лицевую панель или блок-диаграмму.
9. Сохраните изменения, выбрав опцию **Сохранить** в меню **Файл.** Прекрасная работа! Поскольку в следующей главе вы будете применять этот ВП в качестве ВПП, сохраните его в директории `MYWORK`, чтобы облегчить поиск.
10. Шутки ради используйте инструмент перемещения для выбора части блок-диаграммы, как это показано на рис. 5.26. Затем выберите опцию **Создать ВПП** в меню **Правка** для автоматического преобразования этой части в подприбор. Обратите внимание, что элемент отображения температуры остается частью виртуального прибора, вызывающего ВПП. Сделайте двойной щелчок на новом ВПП, чтобы увидеть его лицевую панель.

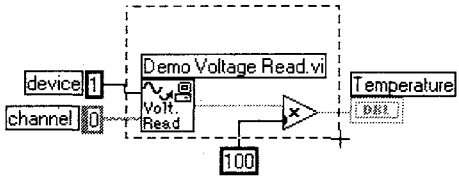


Рис. 5.26

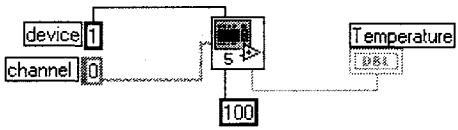


Рис. 5.27

11. Закройте новый ВПП и **Thermometer.vi**. На этот раз не сохраняйте никаких изменений.

Часто люди создают иконку ВП и забывают о его соединительной панели. Если вы не назначите терминалам соединительной панели элементы лицевой панели, то не сможете подключить входные и выходные каналы к вашему прибору, когда попытаетесь использовать его в качестве ВПП. При этом будет трудно определить причину невозможности такого подключения.

5.8. Итоги

LabVIEW предлагает несколько способов сохранения ВП. Вы можете сохранить их в библиотеках ВП, которые являются специальными файлами LabVIEW. Операционная система рассматривает библиотеки ВП как единые файлы; лишь LabVIEW может осуществить доступ к отдельным ВП внутри библиотеки. Прежде чем выбрать наиболее подходящий способ сохранения вашей работы, взвесьте все «за» и «против» использования библиотек ВП. Однако независимо от вашего выбора сохранять ее нужно как можно чаще.

Если ВП неработоспособны сразу же после создания, воспользуйтесь полезными методами отладки программ LabVIEW. Вы можете осуществить *пошаговое* выполнение блок-диаграммы от одного узла данных к другому, *анимировать* диаграмму, используя *подсвечивание выполнения* программы, и приостанавливать выполнение ВПП во время их вызова для просмотра входных и выходных значений, устанавливая *точку останова* при выполнении. Вы также можете использовать *пробники* для отображения текущего значения величины, передаваемой по проводнику. Каждая из этих функций позволяет ближе подойти к решению проблемы.

Подприборы LabVIEW являются эквивалентами обычных подпрограмм. Все ВПП должны иметь иконку и соединительную панель. Вы можете создавать ВПП из существующих виртуальных приборов или из выбранных частей блок-диаграмм. Чтобы избежать ошибок в подключении ВПП, точно определите их входы как *обязательные, рекомендуемые* и *необязательные* и используйте окно контекстной помощи. Разместить существующий ВПП на блок-диаграмме виртуального прибора можно при помощи опции **Выбрать ВП** из палитры **Функции** и указания нужного ВПП в диалоговом окне. ВПП представляют собой одну из наиболее важных особенностей LabVIEW. Допустимо совершенствовать и отлаживать низкоуровневые подприборы, а затем вызывать их из высокоуровневых ВП.

Как всегда при программировании, необходимо документировать вашу работу в LabVIEW. Документировать весь ВП удобно, введя описание в диалоговом окне **Документирование** из вкладки **Свойства ВП** меню **Файл**. Это описание появляется в контекстном окне помощи, когда вы проводите курсором над иконкой ВП. Вы можете документировать отдельные объекты лицевой панели и функции

блок-диаграммы, выбрав опцию **Описание и подсказка** в контекстном меню объекта, а затем введя текст в появившееся диалоговое окно. Описание объектов лицевой панели также появится в окне помощи, если вы проведете над ними курсором.

LabVIEW предлагает несколько вариантов распечатки ВП: можно распечатать активное окно либо точно определить, какие части ВП следует распечатать (лицевую панель, блок-диаграмму или информацию о ВПП), либо запрограммировать в ВП процесс распечатки.

Поздравляем! Вы только что изучили основные операции LabVIEW. Теперь вы готовы познакомиться со структурами и функциями LabVIEW и узнать, как с их помощью создавать очень функциональные программы.

5.9. Дополнительные упражнения

Ниже приведено несколько заданий. Если их решение окажется сложным для вас, посмотрите ответы в библиотеке `EVERYONE\CH5.LLB`.

5.9.1. Упражнение 5.3: определите среднее значение

Создайте ВПП, который усреднял бы три входных числа и выдавал соответствующий результат. Помните о необходимости создания иконки и соединительной панели для подприбора. Сохраните ВП как **Find the Average.vi**.

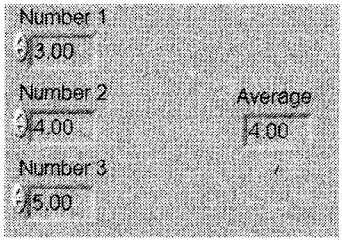


Рис. 5 28

5.9.2. Упражнение 5.4: деление на нуль (кто говорит, что вы не можете?)

Создайте ВП, который бы генерировал случайное число между 0 и 10, поделите его с помощью заданного входного числа и выведите результат на лицевую панель. Если заданное число равно 0, то ВП зажигает светодиод, фиксируя ошибку «деление на нуль». Сохраните ВП как **Divide by Zero.vi**.

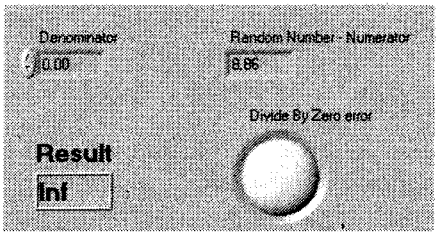


Рис. 5.29

Используйте функцию **Равны?** (Equal?), находящуюся в подпалитре **Сравнение** (Comparison) палитры **Функции**.

Обзор

Структуры – важный тип узла данных – управляют выполнением ВП аналогично операторам обычных языков программирования. В этой главе вы познакомитесь с четырьмя основными структурами LabVIEW: циклами по условию и фиксированным числом итераций, структурами варианта и последовательности. Также вы научитесь вычислению длинных формул при помощи узла Формула, вызову всплывающего окна, содержащего сообщение, и нескольким основным приемам управления длительностью выполнения программ. Если хотите, можете посмотреть несколько примеров ВП, созданных на основе структур и расположенных в библиотеке LabVIEW `EXAMPLES\GENERAL\STRUCTS.LLB`.

ЗАДАЧИ

- Научиться использовать цикл по условию и цикл с фиксированным числом итераций и понять различие между ними
- Осознать необходимость использования сдвиговых регистров в графическом программировании
- Понять различные типы структур варианта – числовую, строковую и логическую
- Научиться управлять порядком выполнения ВП при помощи структуры последовательности
- Использовать узел Формула для введения длинных математических выражений
- Создать всплывающие диалоговые окна, которые говорили бы то, что вы им задали
- Научиться использованию некоторых функций управления временем выполнения программы

ОСНОВНЫЕ ТЕРМИНЫ

- Цикл с фиксированным числом итераций (For Loop)
- Цикл по условию (While Loop)
- Терминал счетчика итераций (Iteration terminal)
- Терминал условия выхода из цикла (Conditional terminal)
- Терминал числа итераций (Count terminal)
- Точка входа/выхода в структуру (Tunnel)
- Точка приведения типов (Coercion dot)
- Сдвиговый регистр (Shift register)
- Структура варианта (Case Structure)
- Терминал селектора структуры варианта (Selector terminal)
- Диалоговое окно (Dialog box)
- Структура последовательности (Sequence Structure)
- Узел Формула (Formula Node)

6 УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ ПРОГРАММЫ С ПОМОЩЬЮ СТРУКТУР

6.1. Два типа структур циклов

Если вы уже программировали на каком-либо языке, то вам, возможно, приходилось использовать повторение выполнения части кода. LabVIEW предлагает две структуры циклов для облегчения реализации этого процесса: *цикл с фиксированным числом итераций* (For Loop) и *цикл по условию* (While Loop) для управления повторяющимися операциями в виртуальном приборе. Цикл с фиксированным числом итераций выполняется определенное количество раз, а цикл по условию выполняется до тех пор, пока определенное условие больше не будет являться истинным. Вы можете найти оба цикла в подпалитре **Структуры** (Structures) палитры **Функции**.

6.1.1. Цикл с фиксированным числом итераций

Цикл с фиксированным числом итераций (For Loop) выполняет код внутри его границ (поддиаграмму) некоторое число *итераций* (count). Это число равно величине, введенной в *терминал числа итераций* (count terminal). Число *отсчетов* вы можете установить, подавая определенное значение извне цикла на терминал числа итераций. Если вы подключите к этому терминалу значение 0, цикл не будет выполняться.

Терминал счетчика итераций (iteration terminal) содержит текущее число завершенных итераций цикла; 0 – во время первой итерации, 1 – во время второй и т.д. до N–1, где N – количество выполнений цикла, которое вы задали.

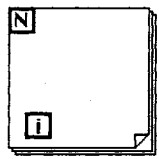


Рис. 6.1. Цикл с фиксированным числом итераций с терминалом числа итераций и терминалом счетчика итераций

Цикл с фиксированным числом итераций эквивалентен следующему псевдокоду:

```
for i = 0 to N-1
Execute subdiagram
```

6.1.2. Цикл по условию

Цикл по условию (While Loop) выполняет код внутри его границ до тех пор, пока логическое значение (Boolean value), подключенное к *терминалу условия выхода из цикла* (conditional terminal) не перейдет в состояние ЛОЖЬ (False). LabVIEW проверяет терминал условия выхода *по окончании* каждой итерации. Если значение соответствует ИСТИНА (True), то выполняется следующая итерация. По умолчанию терминал условия выхода находится в состоянии ЛОЖЬ. Если вы оставите его неподключенным, цикл выполняться не будет (хотя в предыдущих версиях LabVIEW цикл выполнялся бы только один раз).

Терминал счетчика итераций (iteration terminal) цикла по условию ведет себя точно так же, как и в случае с циклом с фиксированным числом итераций.

Цикл по условию эквивалентен следующему псевдокоду:

```
Do
Execute subdiagram
While condition is TRUE
```

Вы можете изменить состояние, которое проверяет терминал условия выхода из цикла. Если раньше цикл выполнялся, пока на вход терминала поступало значение ИСТИНА (while true), теперь цикл остановится, если на вход поступит значение ИСТИНА (unless it's true). Чтобы этого добиться, щелкните правой кнопкой мыши по терминалу условия и выберите опцию **Остановить, если Истина** (Stop if True). Цикл по условию будет выглядеть, как показано на рис. 6.3.

Здесь цикл эквивалентен следующему псевдокоду:

```
Do
Execute subdiagram
While condition is NOT TRUE
```

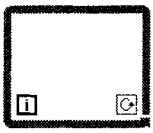


Рис. 6.2. Цикл по условию с терминалом условия выхода из цикла и терминалом счетчика итераций

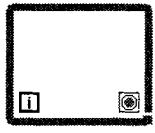


Рис. 6.3. Цикл по условию с терминалом условия выхода из него в режиме ***Остановить, если Истина***

6.1.3. Размещение объектов внутри структур

For Loop Cursor

While Loop Cursor

Когда вы впервые выбираете структуру в подпалитре **Структуры** палитры **Функции**, курсор примет форму указанной структуры в миниатюре – например, цикла с фиксированным числом итераций или цикла по условию. Щелкните мышью там, где вы хотите поместить угол структуры, и перемещайте курсор для определения ее границ. При отпускании кнопки мыши появляется структура, содержащая все объекты, которые находятся в ее пределах.

Когда вы создали структуру на блок-диаграмме, вы можете разместить в ней другие объекты либо путем их внесения, либо размещения внутри при выборе из палитры **Функции**. Если вы *вносите* что-то в структуру, граница структуры становится выделенной на время перемещения объекта внутрь. Если вы *извлекаете* объект из структуры, то граница *диаграммной панели* (или граница внешней структуры) становится выделенной на время вынесения объекта из структуры.

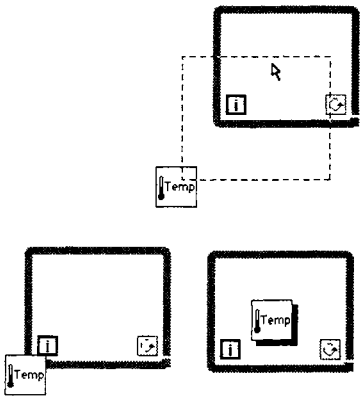


Рис. 6.4

Вы можете изменить размеры существующей структуры путем захвата и перемещения ее угла инструментом перемещения.

Если во время перемещения структура накладывается на другой объект, перекрываемый объект отобразится над ее краем. Если перемещаемая структура полностью закрывает другой объект, появится густая тень, предупреждающая вас о том, что объект находится *над* или *под* структурой, а не *внутри* ее. Обе эти ситуации показаны на рис. 6.4.

Терминалы внутри циклов

Данные поступают в цикл и выходят из него через маленькое окно на границе цикла, называемое *точкой входа/выхода в структуру* (tunnel). Поскольку LabVIEW работает в соответствии с принципом потока данных, то прежде чем цикл начнет выполняться, входные точки должны передать в него свои данные. *Выходные точки цикла выводят данные лишь после завершения всех итераций.*

Также в соответствии с принципом потока данных, *для того чтобы терминал обновлял свои значения на каждой итерации цикла, вы должны поместить его внутрь цикла.* Например, цикл по условию, показанный на рис. 6.5а, проверяет состояние логического элемента управления на каждой итерации. Если с элемента считывается значение ЛОЖЬ, то цикл завершает работу.

Если вы поместите терминал логического элемента управления за пределами цикла по условию, как это показано на рис. 6.5б, то вы создадите цикл либо с бесконечным числом итераций, либо с однократным выполнением – в зависимости от начального значения логического элемента. В соответствии с принципом потока данных LabVIEW считывает логическое значение прежде, чем оно поступает в цикл, а не в процессе выполнения цикла или после его завершения.

Рис. 6.5

Точно так же числовой индикатор (Digital Indicator) в цикле на рис. 6.6 будет обновляться во время каждой итерации. Числовой индикатор на рис. 6.7 обновится только один раз после завершения цикла. Он будет содержать случайное число, появившееся при последней итерации.

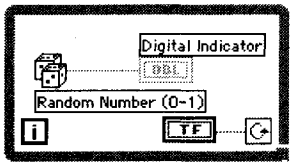


Рис. 6.6

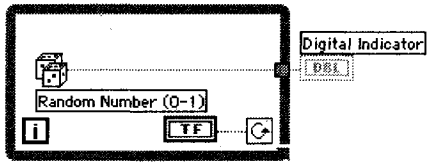


Рис. 6.7

Если вы хотите удалить цикл, не удаляя его содержимого, щелкните правой кнопкой мыши по его границе и выберите опцию **Удалить цикл по условию** (Remove While Loop) или **Удалить цикл с фиксированным числом итераций** (Remove For Loop) соответственно. Если вы просто выделите цикл инструментом перемещения и удалите его, то все объекты внутри будут также удалены.

Вы можете создать массивы данных, используя цикл, и сохранить их на его границах с помощью функции LabVIEW *автоматическая индексация* (auto-indexing). Мы поговорим о массивах данных и индексировании в следующей главе.

Помните, что во время первой итерации обоих типов циклов значение счетчика числа итераций равно нулю. Если вы хотите показать, сколько повторений цикла произошло, прибавьте единицу к значению счетчика.

6.1.4. Упражнение 6.1: счет с помощью циклов

В этом упражнении вы создадите цикл с фиксированным числом итераций, который отобразит значение счетчика на развертке осциллограммы лицевой панели. Выберите число итераций (опция **Number of Iterations**), и цикл начнет отсчет от нуля до этого числа минус 1 (поскольку отсчет начинается с нуля). Затем создайте цикл по условию, который будет считать до тех пор, пока вы не остановите его переключателем логических значений (Boolean switch). Вы также увидите эффект, который возникает при помещении элементов управления и отображения за пределами цикла по условию.

1. Создайте новую лицевую панель, выбрав опцию **Новый ВП** (New VI) в меню **Файл** или щелкнув мышью по кнопке **Новый ВП** в первоначальном диалоговом окне LabVIEW.
2. Постройте лицевую панель и блок-диаграмму, показанную на рис. 6.8 и 6.9. Цикл с фиксированным числом итераций располагается в подпалитре **Структуры** палитры **Функции**. Команда **Разместить слева и справа** (Tile Left and Right) из меню **Окно** позволит вам увидеть одновременно лицевую и диаграммную панели.

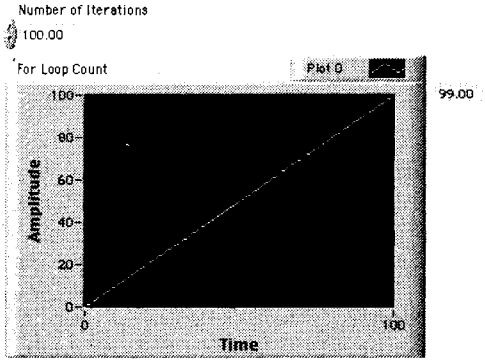


Рис. 6.8

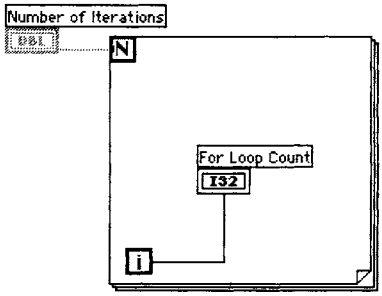


Рис. 6.9

Поместите развертку осциллограммы (**Waveform Chart**) из подпалитры **Графики** (Graph) палитры **Элементы управления** (Controls) на лицевую панель ВП. Назовите ее **Счетчик числа итераций**. Более подробно мы поговорим о диаграммах и графиках в главе 8. Используйте цифровой элемент управления из подпалитры **Числовые** (Numeric) для управления значением **Число итераций**.

3. Щелкните правой кнопкой мыши по развертке осциллограммы и выберите опцию **Автомасштабирование по оси Y** (AutoScale Y) в меню **Шкала Y** (Y Scale), чтобы график автоматически подстраивал масштаб для отображения возрастающего значения счетчика цикла. Снова щелкните правой кнопкой мыши по развертке осциллограммы и выберите **Видимые элементы ⇒ Цифровой дисплей** (Visible Items ⇒ Digital Display). Теперь введите значение в элемент управления **Число итераций** и запустите ВП. Обратите внимание, что цифровой элемент отображения считает от 0 до N–1 (где N – введенное вами число). Каждый раз, когда цикл выполняется, на графике по оси Y отображается значение счетчика цикла относительно времени по оси X. В этом случае каждый отсчет времени индицирует выполнение одной итерации цикла.
4. Обратите внимание на наличие маленькой серой точки в месте соединения терминала числа итераций и проводника числа итераций. Она называется точкой приведения типов, и мы поговорим о ней позднее. Щелкните правой кнопкой мыши по элементу управления числа итераций и выберите опцию **I32 Long** в меню **Представление** (Representation) для ее устранения.
5. При желании вы можете сохранить виртуальный прибор, но мы не будем больше им пользоваться. Откройте новое окно и создайте цикл по условию.
6. Постройте ВП, изображенный на рис. 6.11. Помните, что логические элементы появляются на лицевой панели в состоянии ЛОЖЬ.
7. Operating Tool Переведите переключатель в положение ИСТИНА, щелкнув по нему инструментом управления («палец»), и запустите ВП. Для того чтобы остановить выполнение, переведите переключатель в положение

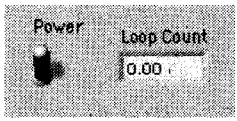


Рис. 6.10

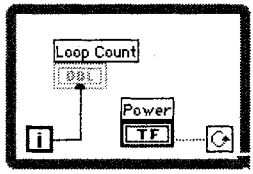


Рис. 6.11

ЛОЖЬ. Значение элемента отображения **Счетчик цикла** будет обновляться во время каждой итерации.

8. Когда переключатель находится в положении ЛОЖЬ, снова запустите ВП. Отметьте, что цикл по условию выполняется только один раз. Помните, что цикл проверяет состояние терминала условия выхода в конце повторения.
9. Теперь перейдите к диаграммной панели и переместите элемент отображения счетчика числа итераций за пределы цикла, как показано на рис. 6.12. Вам придется вновь соединять элемент отображения, а точка выхода создается автоматически во время выведения проводника из цикла.

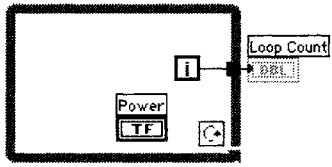


Рис. 6.12

10. Убедитесь, что переключатель находится в положении ИСТИНА, и запустите ВП. Обратите внимание, что значение элемента отображения обновляется лишь после того, как цикл закончился. Он содержит конечное значение счетчика итераций, которое появляется после завершения цикла. Более подробно о выводе данных из цикла вы узнаете в главе 7. *Пока же не пытайтесь вывести скалярные данные из цикла с фиксированным числом итераций, как вы это делали в цикле по условию, иначе у вас появятся неисправные проводники и вы не поймете почему.* Это легко сделать, но вначале вам следует познакомиться с автоматическим индексированием.
11. Сохраните виртуальный прибор. Поместите его в директорию MYWORK и назовите **Loop count.vi**.


Abort Button

12. Теперь, чтобы показать, чего делать *не* следует, выведите логический переключатель из цикла (но оставьте подключенным). Убедитесь, что переключатель находится в положении ИСТИНА, и запустите виртуальный прибор, затем остановите его выполнение. Он не останавливается? Как только LabVIEW входит в цикл, он не будет проверять значение элементов управления за пределами цикла (точно так же, как он не обновлял элемент отображения счетчика итераций, пока цикл не завершился). Нажмите кнопку **Прервать** (Abort) на инструментальной панели для остановки выполнения. Если переключатель при запуске цикла находился в положении ЛОЖЬ, то цикл выполнится только один раз. Закройте ВП без сохранения изменений.

Точка приведения типов

Помните маленькую серую точку в месте соединения терминала числа итераций и проводника **Число итераций** в последнем упражнении? Это *точка приведения типов* (coercion dot). Она названа так потому, что LabVIEW приводит (преобразует) одно числовое представление к другому. Если соединить два терминала различных числовых представлений, то LabVIEW преобразует одно из них в другое, которое имеет другой терминал. В предыдущем упражнении терминал числа итераций имел представление 32-битовых целых чисел, тогда как элемент управления числом итераций представляет собой (по умолчанию) число удвоенной точности с плавающей запятой. В этом случае LabVIEW преобразует число удвоенной точности с плавающей запятой в длинное целое число. Выполняя эту операцию, LabVIEW создает в памяти новую копию числа уже в соответствующем представлении. Такая копия занимает определенный объем памяти. Несмотря на то что дополнительные затраты памяти для скалярных чисел пренебрежимо малы, этот объем быстро увеличивается при использовании массивов данных. Попытайтесь свести к минимуму появление точки приведения типов в больших массивах данных путем изменения представления ваших элементов управления и отображения, чтобы оно точно совпадало с представлением проходящих по ним данных.

Когда ВП преобразует числа с плавающей точкой в целые, то он округляет их до ближайшего целого числа. Число с десятичным значением «.5» округляется до ближайшего четного целого числа.

Проще всего создать элемент управления для терминала числа итераций с правильным типом данных и представлением, вызвав контекстное меню терминала и выбрав опцию **Создать константу** (Create Constant) для создания постоянного числа на блок диаграмме или **Создать элемент управления** (Create Control) для создания элемента управления на лицевой панели.

6.2. Сдвиговые регистры

Сдвиговые регистры (shift regisers), применяемые в цикле по условию и в цикле с фиксированным числом итераций, являются особым типом переменной, используемой для передачи величин из одной итерации цикла в следующую. Они уникальны и необходимы в LabVIEW – графической среде программирования. Сдвиговый регистр создается нажатием правой кнопки мыши на левой или правой границе цикла и выбором опции **Добавить сдвиговый регистр** (Add Shift Register) в контекстном меню.

Сдвиговый регистр состоит из пары терминалов, расположенных друг напротив друга на вертикальных сторонах границы цикла. В правом терминале хранятся данные, полученные при завершении итерации цикла. Эти данные «сдвигаются» в конце итерации и появляются в левом терминале в начале следующей итерации (рис. 6.13). Сдвиговый

регистр может содержать любой тип данных – числовой, логический, строковый, массива и т.п. Сдвиговый регистр автоматически подстраивается к типу данных первого объекта, к которому вы его подсоедините. После создания регистра он окрашен в черный цвет, но затем присваивает себе цвет типа данных, к источнику которых он подсоединен.

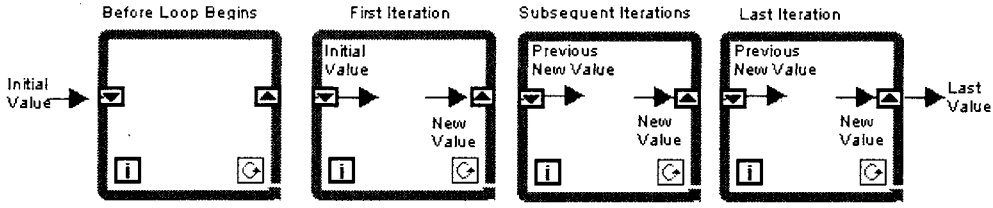


Рис. 6.13. Сдвиговые регистры

Вы можете сконфигурировать сдвиговый регистр для запоминания значений, полученных во время нескольких предыдущих итераций, как показано на рис. 6.14. Это весьма полезная функция для усреднения величин данных, полученных при различных итерациях. Чтобы обеспечить доступ к данным от предыдущих итераций, создайте дополнительные терминалы, щелкнув правой кнопкой мыши по *левому* терминалу регистра и выбрав опцию **Добавить элемент** (Add Element) в контекстном меню.

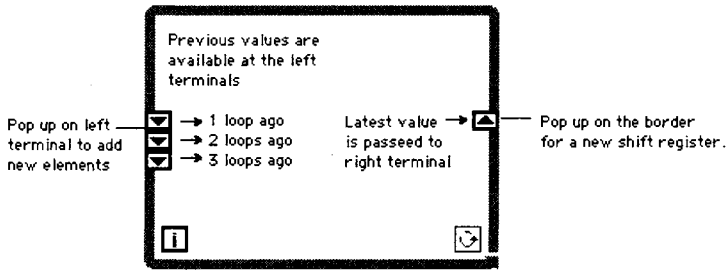


Рис. 6.14

Допустимо создать большое количество различных сдвиговых регистров, сохраняющих много различных переменных за одну итерацию. Для этого просто щелкните правой кнопкой мыши по границе цикла и добавляйте регистры до тех пор, пока не получите желаемое количество пар. Левый терминал всегда будет параллельным правому – если вы передвинете один, то второй тоже переместится. Таким образом, если есть много регистров в цикле и вы не можете определить, какие из них являются параллельными, выберите любой – его пара выделится автоматически, или передвиньте один из них немного в сторону и его пара тоже переместится.

Не делайте распространенной ошибки – не путайте большого числа переменных, находящихся в памяти многочисленных сдвиговых регистров, с единственной переменной, полученной в результате многочисленных предыдущих итераций и находящейся в памяти одного регистра. На рис. 6.15 показано это различие.

Если вы все-таки немного запутались, не беспокойтесь. Понятие сдвиговых регистров является достаточно новой концепцией, отличной от других, с которыми вы могли столкнуться в традиционных языках программирования. Выполнение следующего упражнения позволит вам познакомиться с ними более детально.

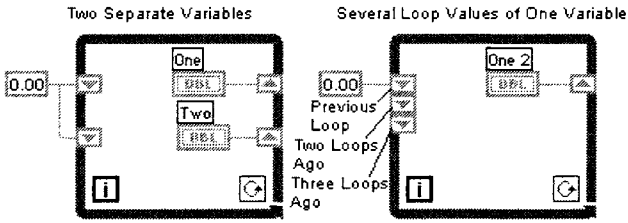


Рис. 6.15. Две различные переменные (слева) и несколько значений одной переменной (справа)

Внимательно следите за правильностью подключения проводника к регистру, иначе вы можете создать нежелательную точку входа/выхода в цикл.

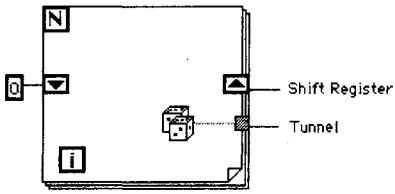


Рис. 6.16

6.2.1. Упражнение 6.2: использование сдвигового регистра

Чтобы изучить работу сдвиговых регистров, понаблюдайте за их использованием для доступа к величинам, полученным в результате предыдущих итераций цикла. В этом ВП вам понадобятся значения счетчика из предыдущих итераций.

1. Откройте прибор **Shift Register Example.vi,** расположенный в директории `EVERYONE\CH6.LLB`.
   На лицевой панели имеется четыре числовых индикатора. Индикатор **Текущий отсчет** будет показывать текущее значение счетчика цикла (он соединен с терминалом счетчика итераций). Индикатор **Предыдущий**

отсчет будет показывать предыдущее значение счетчика цикла. Элемент отображения **Отсчет две итерации назад** будет показывать значение счетчика две итерации назад и т.д.

2. Откройте окно блок-диаграммы, выбрав опцию **Показать диаграммную панель** (Show Diagram) в меню **Окно**.
   Нулевое значение, соединенное с терминалами левого регистра, записывает ноль в эти регистры в начале выполнения программы. На следующей итерации старое значение счетчика итераций (**Текущий отсчет**) переместится в левый верхний терминал и станет предыдущим отсчетом. Затем значение предыдущего отсчета перейдет в терминал **Отсчет две итерации назад** и т.д.

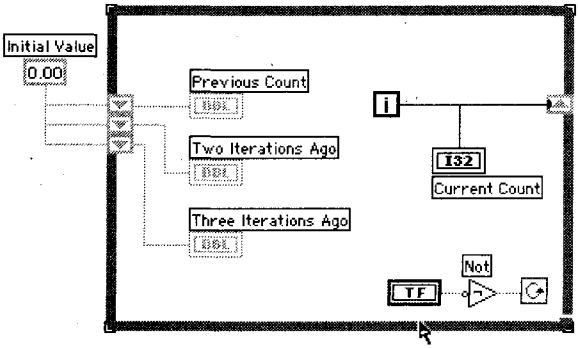


Рис 6 18

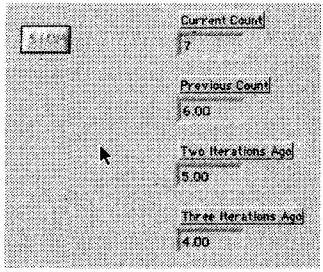


Рис 6 17


Execution Highlighting

3. После просмотра диаграммной панели сделайте одновременно видимыми лицевую панель и диаграмму, выбрав опцию **Разместить слева и справа** (Tile Left and Right) в меню **Окна**.


Step Into

4. Включите подсветку выполнения, щелкнув мышью по кнопке **Подсветка хода выполнения** (Execution Highlightning).
5. Запустите ВП и внимательно проследите за движением кружков. Если они двигаются слишком быстро, остановите выполнение ВП и щелкните мышью по кнопке **Шаг внутрь**, чтобы перевести ВП в пошаговый режим. Снова щелкните мышью по кнопке выполнения пошаговых операций ВП. Проследите, как изменяются значения индикаторов на лицевой панели.
   Обратите внимание, что на каждой итерации цикла по условию ВП «проталкивает» предыдущие значения через левые терминалы сдвигового регистра, используя алгоритм последовательного обслуживания FIFO («первым поступил – первым выводится»). Каждое повторение цикла увеличивает значение счетчика итераций **Текущий отсчет**, соединенного с терминалом правого сдвигового регистра. Это значение переходит в левый терминал, **Предыдущий отсчет**, в начале следующего повторения цикла. Остальные величины регистра в левом терминале

сдвигаются вниз. В этом примере ВП сохраняет только три последние величины. Для сохранения большего числа величин добавьте больше элементов в правый терминал сдвигового регистра, щелкнув правой кнопкой мыши по этому терминалу с последующим выбором опции **Добавить элемент**.

Остановите выполнение ВП, нажав кнопку **Стоп** на лицевой панели. Если вы находитесь в пошаговом режиме, то удерживайте кнопку пошагового выполнения нажатой до тех пор, пока выполнение не завершится.

6. Закройте ВП. Не сохраняйте никаких изменений. Вы хорошо поработали!

6.2.2. Зачем нужны сдвиговые регистры

Посмотрите на пример, изображенный на рис. 6.19. В цикле (A) вы создаете возрастающую сумму значений счетчика итераций. Каждый раз новая сумма сохраняется в сдвиговом регистре. В конце цикла общая сумма поступает на индикатор. В цикле (B) нет сдвигового регистра, поэтому вы не сможете сохранить данные между итерациями. Вместо этого вы всякий раз добавляете нуль к текущему значению «i», и лишь последнее значение 9 выйдет из цикла.

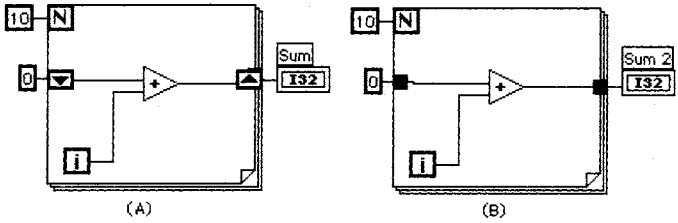


Рис. 6.19

А как быть в случае, когда нужно усреднить значения, полученные в последовательных итерациях цикла? Например, вам понадобится измерять температуру со скоростью один раз в секунду, а затем усреднить эти значения за час. С учетом графической структуры LabVIEW вы не сможете перенести значение, полученное за одну итерацию цикла, в следующую итерацию без использования сдвигового регистра.

6.2.3. Инициализация сдвиговых регистров

Чтобы избежать непредвиденной и, возможно, неправильной работы циклов, *всегда следует инициализировать сдвиговые регистры*, пока не будет серьезной причины этого не делать. Чтобы инициализировать, то есть записать регистр определенным значением, соедините проводником это значение и левый терминал сдвигового регистра извне цикла, как это показано на рис. 6.20. Если вы его не инициализируете, то во время первоначального запуска программы значение будет равно значению по умолчанию для типа данных регистра. Во время последующих

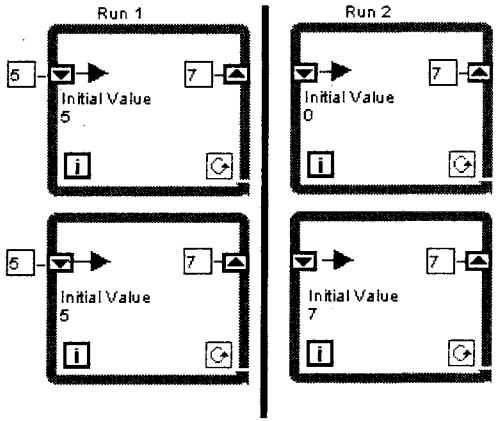


Рис. 6.20

запусков прибора регистр будет содержать данные, оставшиеся от предыдущих запусков программы.

Например, если в регистре сдвига находятся логические данные, то во время первоначального запуска значение будет ЛОЖЬ. Если же в регистре сдвига имеются числовые данные, то первоначальным значением будет нуль. Во время второго запуска ВП неинициализированный регистр будет содержать значения, оставшиеся от первоначального запуска. Изучите показанное на рис. 6.20, чтобы понять необходимость инициализации сдвиговых регистров. Два цикла в левой колонке показывают, что происходит при повторном запуске программы, содержащей инициализированный сдвиговый регистр. Правая колонка демонстрирует, что происходит, когда вы запускаете программу, содержащую неинициализированный регистр два раза. Обратите внимание на начальные величины сдвиговых регистров в двух нижних циклах.

LabVIEW никогда не удаляет величины, сохраненные в сдвиговом регистре, до тех пор, пока вы не закроете ВП и он не очистит область занимаемой памяти. Другими словами, если вы запустите ВП, содержащий неинициализированные сдвиговые регистры, то начальные значения сдвиговых регистров будут совпадать со значениями, оставшимися от предыдущего запуска ВП. Вряд ли вам нужно такое развитие событий, да и проблемы, возникающие при этом, трудно отследить.

6.3. Структуры варианта

Достаточно пока о циклах – давайте перейдем к другой, не менее мощной структуре. *Структура варианта* (Case Structure) является методом выполнения текста, содержащего условие, то есть аналогом оператора

импликации (if-then-else). Вы можете найти эту структуру в подпалитре **Структуры** палитры **Функции**. Структура варианта, показанная на рис. 6.21, имеет две или более поддиаграммы или варианта. Лишь одна из них выполняется в зависимости от логического, числового или строкового значения, которое вы подаете на *терминал селектора структуры варианта.*

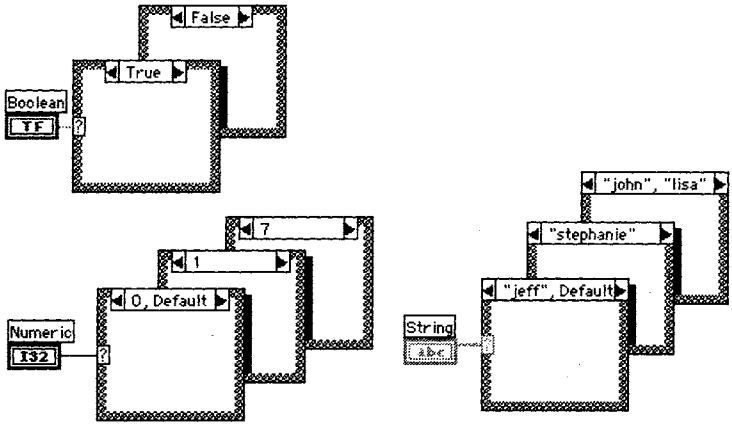


Рис. 6.21

Если к терминалу селектора структуры варианта подключено логическое значение, то структура будет иметь два варианта: ЛОЖЬ и ИСТИНА.

Если к терминалу селектора подключены числовые или строковые данные, то структура может иметь почти неограниченное количество вариантов, начиная с нулевого. Первоначально существует лишь два варианта, но вы легко увеличите их количество. Допустимо использовать несколько значений для одного варианта, отделяя их запятыми, как показано на рис. 6.21. Кроме того, вы всегда можете выбрать вариант **По умолчанию** (Default), который будет выполняться, если величина, подаваемая на терминал селектора структуры, не соответствует никакому другому варианту. Это весьма удобно в том случае, если вы не хотите думать о каком-либо возможном варианте, но хотите использовать общий (подходящий для любых случаев) вариант.

При размещении структуры варианта на лицевой панели в первый раз она будет представлена в логической форме. Для того чтобы использовать в структуре числовые значения, необходимо подать числовой тип данных на терминал селектора.


Decrement Arrow


Increment Arrow

Структуры варианта могут иметь многочисленные поддиаграммы, но *единовременно вы сможете увидеть только одну,* как в колоде карт (совсем не то, что было показано на рис. 6.21, где мы схитрили и показали несколько рисунков). Щелкнув мышью по левой или правой стрелке селектора в верхней части структуры, вы увидите соответственно предыдущие или последующие поддиаграммы. Вы также можете сделать щелчок мышью по селектору в верхней части структуры, чтобы вызвать

выпадающее меню, показывающее все варианты, а затем выбрать нужный. Другим способом переключения вариантов является щелчок правой кнопкой мыши по границе структуры и выбор опции **Показать вариант** (Show Case).

Если вы подадите на терминал селектора число с плавающей точкой, LabVIEW округлит это число до ближайшего целого. LabVIEW принудительно приводит отрицательные числа к нулю и уменьшает любое значение, которое превышает наибольший номер варианта, для приравнивания его к наибольшему номеру.

Вы можете поместить терминал селектора структуры варианта в любом месте вдоль левой границы. Этот терминал всегда должен быть подключенным. Когда вы сделаете это, селектор автоматически присвоит себе тип подводимых данных. Если вы измените тип данных, подаваемых на терминал селектора, с числовых на логические, то варианты 0 и 1 изменятся на ЛОЖЬ и ИСТИНА. Если же имеются другие варианты (от 2 до n), то LabVIEW не сбросит их – вдруг изменение в типе данных было случайным. Тем не менее вы должны удалить лишние варианты, прежде чем структура начнет выполняться.

Для строковых типов данных, подаваемых на терминал селектора, нужно точно определить величины, что достигается помещением строковых данных в кавычки. Единственным исключением является слово `Default`, которое в кавычки никогда не заключается.

6.3.1. Подключение терминалов ввода/вывода

Данные во всех входных терминалах (точках ввода и терминале селектора) структуры варианта доступны для всех вариантов. При работе с вариантами не обязательно использовать входные данные или выводить данные из структуры, но *если в одном варианте данные выводятся, то все варианты должны выдавать данные.* При выводе данных наружу из одного варианта структуры во всех вариантах появится незакрашенная выходная точка в том же самом месте. Стрелка запуска ВП будет «сломанной» до тех пор, пока вы не подадите в эту точку данные из каждого варианта. Тогда точка окрасится в черный цвет (а затем в цвет передаваемых данных), а стрелка запуска будет в рабочем состоянии (при условии, что вы не сделали других ошибок). Убедитесь, что вы подключили проводник *непосредственно* к существующей выходной точке, иначе вы можете случайно создать большое их количество.

Вы спросите, почему надо подключать выходные данные структуры для каждого варианта? Да потому, что структура варианта должна передавать значение в следующий узел данных вне зависимости от того, какой вариант выполняется. LabVIEW заставляет вас выбирать нужное значение, а не делает это сам, что полезно с точки зрения получения хороших навыков программирования.

6.3.2. Добавление вариантов

Если вы щелкнете правой кнопкой мыши по границе структуры варианта, то появившееся меню предложит опции **Создать вариант после** (Add Case After)

и **Создать вариант перед** (Add Case Before) текущим вариантом. Вы также можете скопировать текущий вариант, выбрав опцию **Скопировать вариант** (Duplicate Case). Удалить текущий вариант (и все, что в нем находится) легко с помощью опции **Удалить вариант** (Delete This Case).

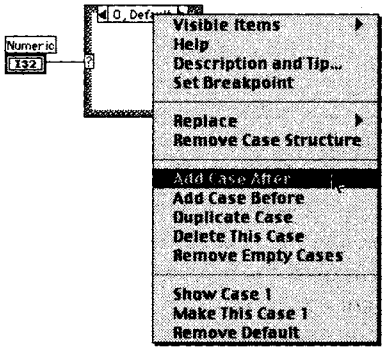


Рис. 6.22

6.3.3. Диалоговые окна

Отвлечемся ненадолго от структур, чтобы поговорить о диалоговых окнах. Функции **Однокнопочный диалог** (One Button Dialogue) и **Двухкнопочный диалог** (Two Button Dialogue), изображенные на рис. 6.23 и 6.24, вызывают *диалоговое окно*, содержащее введенную вами информацию. Вы можете найти эти функции в подпалитре **Время и диалоги** (Time & Dialogue) палитры **Функции**. Диалоговое окно функции **Однокнопочный диалог** будет оставаться открытым до тех пор, пока вы не нажмете кнопку **ОК**, а окно функции **Двухкнопочный диалог** будет открытым, пока вы не щелкнете по кнопке **ОК** или **Cancel**. Разрешается переименовать эти кнопки, подав на соответствующие вводы функций строковые данные «имя кнопки» (button name). Эти диалоговые окна являются *модальными*: вы не сможете активизировать другое окно LabVIEW, пока они открыты. Диалоговые окна весьма полезны для запрашивания или сообщения данных оператору вашего ВП.

Рис. 6.23. Однокнопочный диалог

Рис. 6.24. Двухкнопочный диалог

6.3.4. Упражнение 6.3: извлечение квадратного корня

Это упражнение позволит вам приобрести некоторый навык работы со структурами варианта и диалоговыми окнами. Создайте ВП, который вычисляет квадратный корень из положительного входного числа. Если входное число является отрицательным, то ВП вызовет диалоговое окно и возвратит ошибку.

1. Откройте новую панель.
2. Создайте лицевую панель, изображенную на рис. 6.25.
   С помощью числового элемента управления будет вводиться **Число**. Индикатор **Величина квадратного корня** покажет квадратный корень из числа.
3. Откройте окно блок-диаграммы. Создайте код, изображенный на рис. 6.26 и 6.27.

4. Поместите структуру варианта (подпалитра **Структуры**) в окно блок-диаграммы. Как и в случае с циклом с фиксированным числом итераций и циклом по условию, щелкните курсором выбранной структуры и, перемещая его, определите нужные границы.

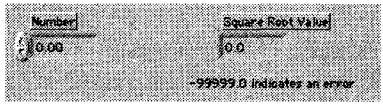


Рис. 6.25

Функция **Больше или равно?** (Greater or Equal?) возвращает логическое значение, поэтому структуру варианта нужно оставить в форме по умолчанию, то есть в логической.

Помните, что варианты отображаются только по одному. Для изменения вариантов щелкните кнопкой мыши по стрелкам селектора структуры варианта. Обратите внимание, что рис. 6.26 и 6.27 показывают оба варианта структуры. Таким образом, вы знаете, что создавать. *Не создавайте две различные структуры в этом упражнении!*

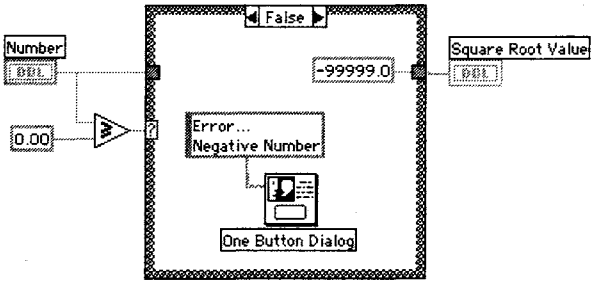


Рис. 6.26. Вариант ЛОЖЬ

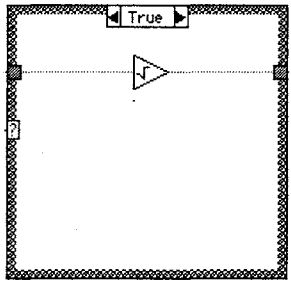


Рис. 6.27. Вариант ИСТИНА

5. Выберите другие объекты диаграммы и соедините их так, как показано на рис. 6.26 и 6.27. Используйте окно контекстной помощи для отображения вводов и выводов различных функций.

Функция **Больше или равно?** из подпалитры **Сравнение** (Comparison) определяет, является вводимое число большим или равным нулю.

Функция **Квадратный корень** (Square Root) из подпалитры **Числовые** (Numeric) вычисляет квадратный корень вводимого числа.

Функция **Числовая константа** (Numeric Constant) из подпалитры **Числовые** используется для вывода числа –99999.0 в случае ошибки, а константа 0 определяет, является ли вводимое число отрицательным.

Функция **Однокнопочный диалог** (меню **Время и диалоги**) в данном упражнении вызывает диалоговое окно, которое содержит информацию «Ошибка ... Отрицательное число».

Функция **Строковая константа** (String Constant) из подпалитры **Строка** (String) используется для ввода текста в диалоговое окно. Текст можно добавить с помощью инструментов управления или ввода текста.

В данном упражнении ВП будет выполнять вариант ИСТИНА либо вариант ЛОЖЬ структуры варианта. Если вводимое число больше или равно нулю, то ВП будет выполнять вариант ИСТИНА, который вычисляет квадратный корень от этого числа. Если число меньше нуля, то исполнение варианта ЛОЖЬ приводит к выходному числу –99999.0 и открытию диалогового окна с сообщением «Ошибка... Отрицательное число».

Помните о том, что вы обязательно должны определить значение, подаваемое в выходную точку структуры, для обоих вариантов. Поэтому мы и побеспокоились подать число –99999.0 в случае ошибки. Если вы создадите выходную точку для одного варианта, эта же точка возникнет и для других вариантов. Не соединенная ни с чем выходная точка выглядит как пустой квадратик. Позаботьтесь соединить с каким-либо источником данных все несоединенные точки. Причем начинайте соединение всегда с точки, иначе вы случайно создадите новую выходную точку.

6. Вернитесь к лицевой панели и запустите ВП. Попробуйте ввести одно число больше нуля, а другое – меньше нуля.
7. Сохраните и закройте ВП. Назовите его **Square Root.vi** и поместите в директории MYWORK или библиотеке виртуальных приборов.

Логика виртуального прибора, вычисляющего квадратный корень, такова:

```
If (Число >= 0) then
  Величина квадратного корня = SQRT (Число)
Else
  Величина квадратного корня = -99999.0
  Display Message «Ошибка ... Отрицательное число»
End If
```

6.3.5. Функция выбора

В простых случаях применения логики `if-then-else` иногда более удобно пользоваться функцией LabVIEW **Выбор** (Select), которая работает так же, как и структура варианта.

Функция **Выбор**, находящаяся в подпалитре **Сравнение** палитры **Функции**, возвращает значение **t**, если входное значение ИСТИНА, и значение **f**, если на

Рис. 6.28. Функция выбора

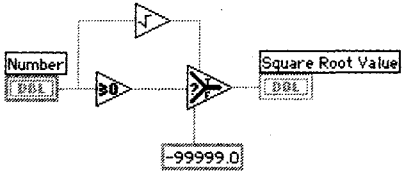


Рис. 6.29

вход подается ЛОЖЬ. С помощью этой функции можно сделать то же самое, что и в случае структуры варианта в последнем упражнении – за исключением вызова диалогового окна.

6.4. Структуры последовательности

Определение порядка выполнения программы путем организации ее элементов в определенную последовательность называется *управлением потоком данных*. В обычных языках программирования, таких как Basic или C, всегда присутствует управление потоком, так как операторы выполняются в том порядке, в каком они написаны в программе. Для осуществления управления потоком при обработке данных в LabVIEW используется *структура последовательности* (Sequence Structure). Структура последовательности выполняет кадр 0, за которым следует кадр 1, затем кадр 2 и т.д., пока не выполнится последний кадр. И лишь после этого данные покидают структуру.

Структура последовательности, показанная на рис. 6.30, весьма похожа на кадр кинопленки. Ее можно найти в подпалитре **Структуры** палитры **Функции**. Так же как и в структуре варианта, единовременно здесь отображается только один кадр. Чтобы увидеть другие кадры, нужно нажимать на стрелки селектора в верхней части структуры. Также можно щелкнуть кнопкой мыши по селектору, чтобы посмотреть существующие кадры, а затем выбрать нужный, либо щелкнуть правой кнопкой мыши по границе структуры и выбрать опцию **Показать кадр** (Show Frame). Когда вы в первый раз помещаете структуру последовательности на блок-диаграмму, она будет иметь только один кадр; следовательно, нет ни стрелок, ни номеров в верхней части структуры, чтобы показать, какой кадр следует. Создайте новые кадры, щелкнув правой кнопкой мыши по границе структуры и выбрав опцию **Создать кадр после** (Add Frame After) или **Создать кадр перед** (Add Frame Before) текущим кадром.

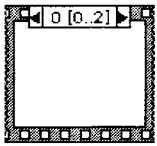


Рис. 6.30. Структура Последовательности

Структуру последовательности используют для управления порядком выполнения узлов данных, которые не зависят друг от друга. В рамках каждого кадра, как и в остальной части блок-диаграммы, зависимость данных определяет порядок выполнения узлов данных. С другим способом управления порядком выполнения, называемым *искусственной зависимостью данных*, вы познакомитесь в главе 16.

Терминалы входных и выходных данных этой структуры могут иметь только один источник данных – в отличие от структуры варианта, выходные терминалы которого должны иметь отдельный источник данных для каждого варианта. Выходных данные могут быть получены из любого кадра, однако данные выходят из структуры только тогда, когда она полностью завершает свое выполнение, а не каждый ее кадр. Данные входных терминалов доступны для всех кадров.

6.4.1. Терминалы локальной переменной

Чтобы передать данные из одного кадра в любой последующий, вы должны воспользоваться так называемым *терминалом локальной переменной* (sequence local). Для создания терминала локальной переменной выберите опцию **Создать локальную переменную** (Add Sequence Local) из контекстного меню *границы структуры*. Эта опция будет недоступна, если вы щелкнете правой кнопкой мыши слишком близко от другого терминала локальной переменной или над окном дисплея поддиаграммы. Вы можете переместить терминал локальной переменной в любое свободное место на границе структуры. Используйте команду **Удалить** (Remove) из контекстного меню терминала локальной переменной для удаления терминала либо выделите, а затем удалите его.

Когда терминал локальной переменной впервые появляется на блок-диаграмме, он представляет собой маленький желтый прямоугольник. На рис. 6.31–6.34 показан терминал локальной переменной в различных формах. Когда вы подключаете источник данных к этому терминалу, в нем появляется стрелка, направленная наружу, говорящая о том, что этот терминал содержит источник данных. Терминалы в последующих кадрах содержат стрелку, направленную внутрь, – значит, этот терминал является источником данных для этих кадров. В кадрах, предшествующих кадру источника данных, вы не сможете использовать терминал локальной переменной (хотя бы потому, что на него еще не было подано какое-либо значение), и он появляется в виде заштрихованного прямоугольника.

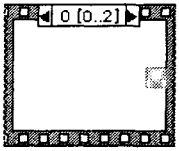


Рис. 6.31. Локальную переменную нельзя использовать

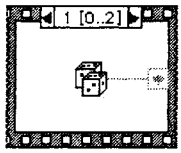


Рис. 6.32. Терминал локальной переменной – приемник данных

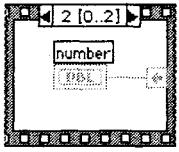


Рис. 6.33. Терминал локальной переменной – источник данных

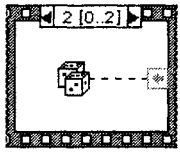


Рис. 6.34. Локальная переменная – источник данных; вы не можете подсоединить к ее терминалу входное значение

6.4.2. Регулирование и хронометраж времени выполнения ВП

Иногда полезно отслеживать время работы вашего ВП и управлять им. С этими задачами прекрасно справляются функции **Задержка (мс)**, **Счетчик времени (мс)** и **Задержка до следующего кратного интервала мс**, расположенные в подпалитре **Время и диалоги** палитры **Функции**.

Функция **Задержка (мс)** – Wait (ms) – заставляет ВП ждать определенное число миллисекунд, прежде чем продолжить выполнение (рис. 6.35).

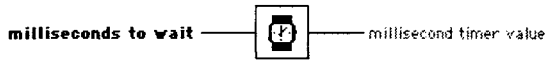


*Рис. 6.35. Функция **Задержка (мс)***

Функция **Задержка до следующего кратного интервала мс** (Wait Until Next ms Multiple) заставляет LabVIEW ожидать, пока показания внутренних часов не сравняются или не превысят кратного количества миллисекунд, поданных на вход функции, прежде чем возобновить выполнение ВП (рис. 6.36). Эта функция заставляет циклы выполняться через определенные интервалы времени и позволяет синхронизировать работу. Две названные функции похожи, но не идентичны. Например, задержка выполнения с помощью функции **Задержка до следующего кратного интервала мс**, возможно, будет меньше заданного числа миллисекунд при первой итерации, поскольку зависит от значения часов во время упорядочивания (то есть от того, сколько времени потребуется для перехода к следующей итерации и возобновления работы ВП). Кроме того, если цикл все еще выполняется, а часы уже прошли кратный миллисекундный интервал, то ВП будет ждать до тех пор, пока часы достигнут следующего кратного миллисекундного интервала. Таким образом, ВП может выйти из синхронизации и замедлиться. Убедитесь, что вы все учли при использовании этих функций.

*Рис. 6.36. Функция **Задержка до следующего кратного интервала мс***

Функция **Счетчик времени (мс)** – Tick Count (ms) – возвращает значение внутренних часов операционной системы в миллисекундах (рис. 6.37). Она в основном используется для подсчета прошедшего времени. Имейте в виду, что внутренние часы не всегда имеют большую разрешающую способность: один отсчет часов может составлять до 55 мс в Windows 95/98, 10 мс в Windows 2000/NT, 17 мс в Linux, Solaris и MacOS.

*Рис. 6.37. Функция **Счетчик времени (мс)***

6.4.3. Упражнение 6.4: числа совпадения

Теперь у вас есть возможность поработать со структурой последовательности и одной из временных функций. Создайте ВП, вычисляющий время, занимаемое процессом совпадения заданного числа с числом, сгенерированным счетчиком случайных чисел. Запомните этот алгоритм – возможно, у вас возникнет необходимость рассчитать время выполнения ВП LabVIEW.

1. Откройте новую лицевую панель.
2. Создайте лицевую панель, как показано на рис. 6.38.
3. Уменьшите точность элементов управления и отображения **Число для совпадения**, **Текущее число** и **Число итераций** до нуля, выбрав опцию **Формат и точность** (Format & Precision) в их контекстных меню. Введите 0 для опции **Разряды точности** (Digits of Precision), так чтобы справа от десятичной запятой не появилось ни одной цифры.

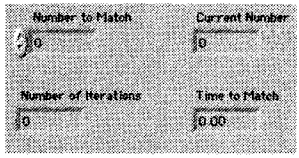


Рис. 6.38

4. Откройте окно диаграммы и постройте диаграмму, как показано на рис. 6.39–6.41.
5. Разместите структуру последовательности (палитра **Структуры**) в окне диаграммы. Это делается таким же образом, как и для цикла с фиксированным числом итераций и цикла по условию – щелкните курсором в форме структуры и задайте желаемые границы.

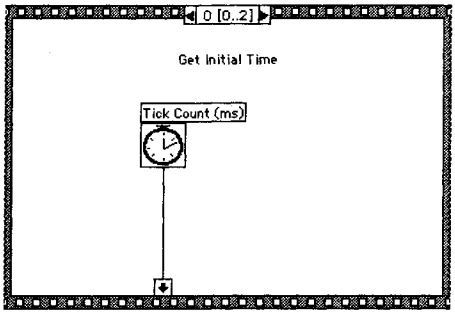


Рис. 6.39

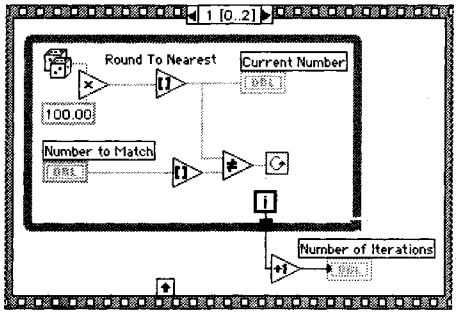


Рис. 6.40

Вам нужно создать три отдельных кадра структуры последовательности. Для создания нового кадра щелкните правой кнопкой мыши по границе кадра и выберите опцию **Создать кадр после** в контекстном меню.

6. Создайте терминал локальной переменной, щелкнув правой кнопкой мыши по нижней границе нулевого кадра и выбрав пункт **Создать локальную переменную** в контекстном меню. Терминал локальной

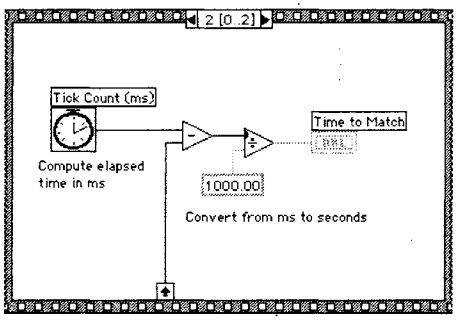


Рис. 6.41

переменной отобразится в виде пустого квадрата. Стрелка внутри квадрата появится автоматически, когда вы подключитесь к локальной переменной.

7. Достройте диаграмму. Ниже будут описаны некоторые новые функции. В процессе подключения используйте окно контекстной помощи для отображения вводов и выводов функций.

Функция **Счетчик времени (мс)** находится в палитре **Время и диалоги**; она возвращает показания внутренних часов.

Функция **Случайное число (0-1)** – Random Number (0-1) – в палитре **Числовые** возвращает случайное число в диапазоне от 0 до 1.

Функция **Умножить** (Multiply) в палитре **Числовые** умножает случайное число на 100, так что в результате возвращается случайное число в диапазоне от 0.0 до 100.0.

Функция **Округлить до ближайшего** (Round to Nearest) в палитре **Сравнение** округляет случайное число до ближайшего целого числа.

Функция **Не равно?** (Not Equal?) в палитре **Сравнение** сравнивает случайное число с числом, введенным на лицевой панели, и возвращает значение ИСТИНА, если числа не равны; в противном случае функция возвращает значение ЛОЖЬ.

Функция **Инкремент** (Increment) в палитре **Числовые** добавляет единицу к значению счетчика цикла для отображения величины **Число итераций**, чтобы компенсировать индексацию, начинающуюся с нуля.

При выполнении нулевого кадра функция **Счетчик времени (мс)** возвращает показания внутренних часов в миллисекундах. Это значение подается на терминал локальной переменной, поэтому оно будет доступным в последующих кадрах. В кадре 1 ВП выполняет цикл по условию до тех пор, пока введенное число не совпадет с числом, возвращенным функцией **Случайное число (0-1)**. В кадре 2 функция **Счетчик времени (мс)** возвращает новый отсчет времени в миллисекундах. Виртуальный прибор вычитает старый отсчет времени (полученный в нулевом кадре и переданный через терминал локальной переменной) из нового для

вычисления прошедшего времени, а затем делит на 1000 для перевода единиц измерения из миллисекунд в секунды.
Существует более эффективный способ выполнения этого упражнения, основанный на использовании двухкадровой структуры последовательности. Однако нам бы хотелось, чтобы вы пользовались терминалами локальной переменной.

8. Включите подсветку выполнения, которая замедляет работу ВП, для того, чтобы увидеть текущее генерируемое число на лицевой панели.
9. Введите число в элемент управления **Число для совпадения** и запустите ВП. Для ускорения выполнения выключите подсветку.
10. Используйте команду **Сохранить** для сохранения ВП в директории MYWORK или библиотеке виртуальных приборов с именем **Time to Match.vi,** затем закройте его.

6.5. Узел Формула

Теперь, когда вы познакомились с четырьмя основными структурами управления потоками данных LabVIEW, мы представим структуру, которая не влияет на поток данных программы. Узел Формула является окном с изменяемыми размерами для ввода алгебраических формул непосредственно в блок-диаграмму. Эта особенность особенно полезна, когда для вычисления необходимо использовать длинную формулу. Например, рассмотрим простое уравнение $y = x^2 + x + 1$. Если вы напишете код с использованием арифметических функций LabVIEW на блок-диаграмме для вычисления даже такой простой формулы, он будет достаточно сложным для понимания по сравнению с текстовыми уравнениями.

Вы можете ввести то же выражение в узел Формула, как это показано на рис. 6.43[1].

С помощью узла Формула допустимо непосредственно ввести формулу или формулы в окно вместо создания сложных подразделов блок-диаграммы. Входные и выходные терминалы узла Формула можно создать, щелкнув правой кнопкой мыши по границе узла и выбрав опцию **Добавить ввод** (Add Input) или **Добавить вывод** (Add Output) в контекстном меню. Имя переменной чувствительно

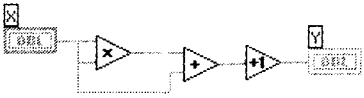


Рис. 6.42

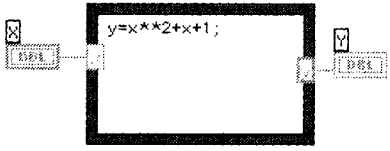


Рис. 6.43

[1] В ранних версиях LabVIEW оператор возведения в степень для узла Формула был символом ^. В LabVIEW 6.0 и более поздних версиях символ ^ означает Исключающее ИЛИ, а символ ** – возведение в степень

к регистру букв. *Каждая строка в узле Формула должна заканчиваться точкой с запятой* (;).

Найти функцию **Узел Формула** можно в подпалитре **Структуры** палитры **Функции**.

Информацию об операциях и функциях, которые могут применяться внутри узла Формула, вы найдете в окне контекстной помощи (рис. 6.44).

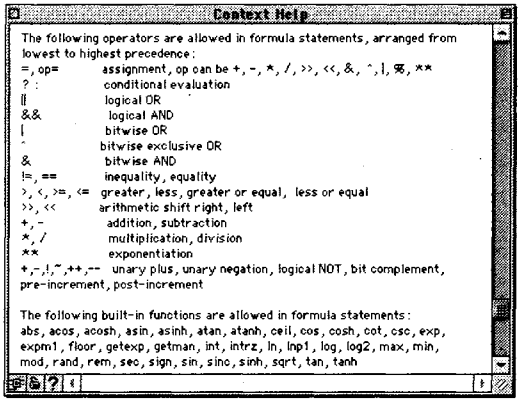


Рис. 6.44

Следующий пример показывает выбор данных по условию внутри узла Формула. Посмотрите на фрагмент кода, похожий на упражнение 6.3, в котором вычисляется квадратный корень величины **x**, если **x** положительное число. При этом выдается результат **y**. Если **x** отрицательное число, то код выдает значение **y**, равное –99.

```
if (x>=0) then
  y= sqrt(x)
else
  y=-99
end if
```

Вы можете ввести фрагмент кода, используя узел Формула, как это показано на рис. 6.45.

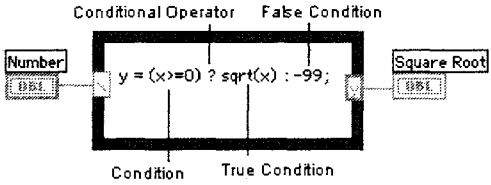


Рис. 6.45

6.5.1. Упражнение 6.5: упражнение с узлом Формула

Вы можете создать ВП, который использует узел Формула для решения уравнения `y=sin(x)` и выдает результат в виде графика.

1. Откройте новую лицевую панель. Выберите **График осциллограммы** (Waveform Graph) в подпалитре **Графики** палитры **Элементы управления**. Назовите его **График**. Более подробно о графиках вы узнаете в главе 8, но это упражнение может показаться скучным без картинок, поэтому мы решили несколько забежать вперед.

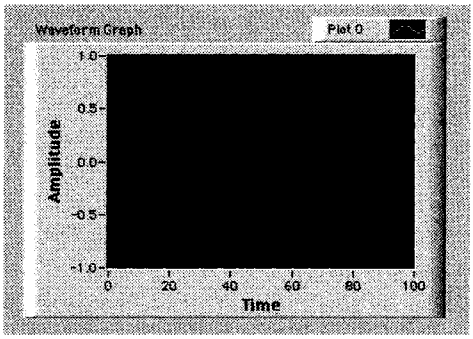


Рис. 6.46

2. Постройте блок-диаграмму, изображенную на рис. 6.47.
   С помощью узла Формула вы можете непосредственно ввести математические формулы. Создайте терминал ввода, щелкнув правой кнопкой мыши по границе и выбрав функцию **Добавить ввод** в контекстном меню. Затем создайте терминал вывода, выбрав функцию **Добавить вывод** в контекстном меню.
   Когда терминалы ввода и вывода созданы, дайте им имена переменных, точно соответствующие именам переменных, которые используются

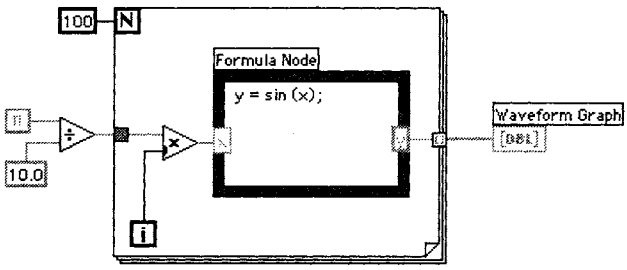


Рис. 6.47

в формуле. Помните, что имена чувствительны к регистру букв, а каждая строка заканчивается точкой с запятой (;).

Pi Constant

Постоянная *pi* расположена в палитре **Функции ⇒ Числовые ⇒ Дополнительные числовые константы** (Additional Numeric Constants).

Во время каждой итерации ВП умножает значение счетчика итераций на *pi/10*. Результат умножения подается в узел Формула, где от этого числа вычисляется синус. Затем ВП сохраняет результат в массиве на границе цикла с фиксированным числом итераций. (Более подробно о массивах данных вы узнаете в главе 7. Там вы увидите, почему по умолчанию из цикла с фиксированным числом итераций выводится массив данных, а из цикла по условию поступают скалярные данные.) После того как цикл с фиксированным числом итераций закончит выполнение, виртуальный прибор отобразит данные массива на графике.

3. Вернитесь к лицевой панели и запустите ВП. Допустимо использовать готовую функцию **Синус** (Sine) в палитре **Функции ⇒ Числовые ⇒ Тригонометрические** (Trigonometric) для выполнения тех же операций, что и узел Формула в этом упражнении, но LabVIEW не имеет встроенных функций для каждой необходимой формулы, поэтому вам следует попрактиковаться.
4. Сохраните ВП в директории MYWORK или в библиотеке виртуальных приборов под именем **Formula Node Exercise.vi.** Закройте ВП.

Логика выполнения ВП такова:

```
for i = 0 to 99
  x = i * (PI/10)
  y = sin(x)
  array[i] = y
next i
Graph (array)
```

6.6. Итоги

LabVIEW имеет две структуры для повторения выполнения поддиаграммы: *цикл по условию* и *цикл с фиксированным числом итераций*. Обе структуры представляют собой окна с изменяемыми размерами. Чтобы поддиаграмму заставить повторяться, необходимо поместить ее внутри границы цикла. Цикл по условию выполняется до тех пор, пока значение, подаваемое на терминал условия выхода, остается ИСТИНА. Цикл с фиксированным числом итераций выполняется определенное количество раз.

Сдвиговые регистры, применяемые в цикле с фиксированным числом итераций и в цикле по условию, перемещают значения, полученные после выполнения одной итерации цикла, в начало другой. Вы можете сконфигурировать сдвиговые регистры для использования значений, полученных за многие предыдущие

итерации. Для каждой итерации, которую вы хотите вспомнить, нужно добавить новый элемент в левый терминал сдвигового регистра. Также разрешается использовать в цикле несколько сдвиговых регистров для хранения нескольких переменных.

В LabVIEW есть две структуры для управления потоком данных: *структура варианта* и *структура последовательности*. Единовременно можно увидеть лишь один вариант или кадр этих структур. Кадры (варианты) перебирают, используя маленькие стрелки в верхней части структуры либо щелкая инструментом управления в окне в верхней части структуры.

Структура варианта служит для разветвления на различные поддиаграммы в зависимости от входных данных, поступающих на терминал селектора структуры. Функционально она совпадает с оператором `if-then-else` в обычных языках программирования. Просто поместите поддиаграммы, которые нужно выполнить, внутрь границы каждого варианта структуры и подключите элемент управления (источник данных) к терминалу селектора структуры варианта. Структуры варианта могут быть логическими (два возможных варианта), числовыми или строковыми (вплоть до $2^{15}-1$ вариантов) – LabVIEW автоматически определяет тип данных, который вы подключаете к терминалу селектора.

Иногда принципы обработки потока данных заставляют программу выполняться не в том порядке, в каком вам бы хотелось. Структура последовательности дает возможность установить нужный порядок выполнения функций на блок-диаграмме. Часть диаграммы, выполняемая в первую очередь, располагается в первом кадре (кадр 0) структуры последовательности; поддиаграмма, выполняемая во вторую очередь, расположена во втором кадре и т.д.

Терминалы *локальной переменной* используются для передачи данных между кадрами структуры последовательности. Данные, поступившие с терминала локальной переменной, доступны лишь в тех кадрах, которые следуют за кадром, где вы создаете терминал локальной переменной.

С помощью *узла Формула* вы можете непосредственно ввести формулы в блок-диаграмму, что весьма полезно для решения сложных функциональных уравнений. Помните, что имена переменных чувствительны к регистру букв и что каждая строка в узле должна оканчиваться точкой с запятой (;).

В подпалитре **Время и диалоги** находятся функции вызова диалогового окна и управления временными параметрами ВП. Функции **Однокнопочный диалог** и **Двухкнопочный диалог** вызывают диалоговое окно, которое содержит введенную вами информацию. Функция **Задержка (мс)** делает паузу в выполнении ВП на определенное количество миллисекунд. Функция **Задержка до следующего кратного интервала мс** может заставить итерации циклов выполняться в течение определенного количества миллисекунд, делая паузу до тех пор, пока внутренние часы не сравняются или не превысят значения (в миллисекундах), поданного на вход функции. Эти две функции ожидания аналогичны, но не идентичны. Функция **Счетчик времени (мс)** возвращает показания внутренних часов операционной системы.

6.7. Дополнительные упражнения

6.7.1. Упражнение 6.6: уравнения

Создайте ВП, который применяет узел Формула для решения следующих уравнений:

$y_1 = x_3 + x_2 + 5$
$y_2 = (m*x) + b$

Используйте только один узел Формула для обоих уравнений. (Не забудьте поставить точку с запятой после каждого уравнения в узле.) Назовите этот ВП **Equations.vi**.

6.7.2. Упражнение 6.7: калькулятор

Создайте ВП в виде калькулятора. Лицевая панель должна иметь цифровые элементы управления для ввода двух чисел и числовой индикатор для показа результата операции (сложение, умножение или деление) над этими числами. Используйте для данной операции ползунковый элемент управления. Назовите ВП **Calculator.vi**.

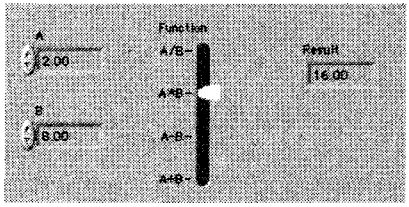


Рис. 6.48

Чтобы задать функцию (сложение, вычитание, умножение или деление), вам понадобятся текстовые ярлыки ползункового элемента управления, которые находятся в его контекстном меню (опция **Текстовые ярлыки** – Text Labels). Если вы не видите этой опции, проверьте, точно ли вы вызываете контекстное меню ползунка, а не его масштабных меток. Ползунковый элемент управления с текстовыми ярлыками ведет себя аналогично элементам кольцевого списка (text ring controls). При первоначальном выборе опции **Текстовые ярлыки** ползунок будет иметь два возможных положения – max и min. С помощью инструмента ввода текста эти надписи можно поменять. Для добавления возможных положений вызовите контекстное меню текстового дисплея, находящегося рядом с ползунком, выберите опции **Добавить элемент после** (Add Item After) или **Добавить элемент до** (Add Item Before) и введите текст ярлыка.

6.7.3. Упражнение 6.8: комбинация цикла с фиксированным числом итераций с циклом по условию

Используя лишь цикл по условию, создайте комбинацию цикла с фиксированным числом итераций и цикла по условию, которая останавливается по достижении числа N (установленного элементом управления на лицевой панели) или при щелчке по кнопке **Стоп**. Назовите этот ВП **Combo For/While Loop.vi**.

Не забывайте, что цикл по условию выполняется до тех пор, пока на вход терминала условия выхода из цикла поступает логическое значение ИСТИНА. Возможно, вы захотите использовать функцию **И** (And), находящуюся в подпалитре **Логические** (Boolean) палитры **Функции**. Также помните, что, пока цикл выполняется, ни один индикатор или значение элемента управления, находящиеся вне цикла, не будут обновляться. Поэтому, чтобы ваш прибор функционировал правильно, всегда располагайте кнопку останова внутри цикла.

6.7.4. Упражнение 6.9: диалоговое окно

Создайте ВП, который может считать значение переключателя лицевой панели и вывести сообщение, включен или выключен переключатель, в диалоговом окне. Назовите ВП **VI Dialogue Display.vi**. Откажитесь от применения кнопки непрерывного запуска программы, в противном случае у вас будет бесконечно выполняющийся цикл. Если вы оказались в таком положении, остановите выполнение ВП, используя клавиши <control>+<.> в Windows, <command>+<.> в Macintosh, <meta>+<.> в Sun и <alt>+<.> в Linux.

Обзор

В этой главе вы изучите два более сложных составных типа данных – массивы и кластеры. Они позволяют очень гибко манипулировать данными и сохранять информацию. Вы познакомитесь с многочисленными применениями массивов и кластеров, а также научитесь использовать встроенные функции LabVIEW для управления и обработки таких типов данных.

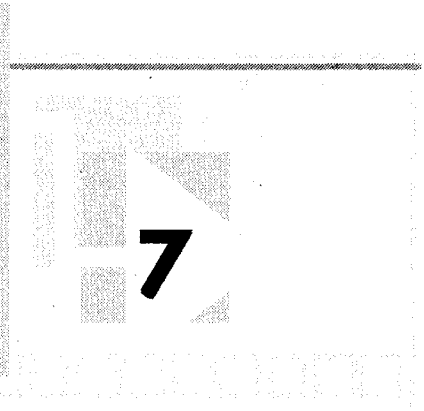
ЗАДАЧИ

- Изучить встроенные функции работы с массивами
- Понять концепцию полиморфизма
- Научиться использовать кластеры, а также, разделять и объединять их
- Понять, чем кластеры отличаются от массивов

ОСНОВНЫЕ ТЕРМИНЫ

- Массив
- Автоиндексация
- Полиморфизм
- Кластер
- Объединение в кластер
- Разделение кластера

СОСТАВНЫЕ ДАННЫЕ LabVIEW: МАССИВЫ И КЛАСТЕРЫ



7.1. Что такое массивы

До настоящего времени мы имели дело лишь со скалярными числами (скалярная величина представляет собой тип данных, которые содержат единственное значение), но сейчас пришло время поговорить о более сложных вещах. *Массив* (array) LabVIEW представляет собой набор элементов данных одного типа, так же как и в традиционных языках программирования. Массив может иметь одну или несколько размерностей, то есть быть одномерным или многомерным, и включать до 2^{31} элементов на одну размерность (естественно, в зависимости от объема памяти). Элементом массива может быть любой тип данных за исключением массива, таблицы или графика.

Доступ к элементам массива осуществляется посредством их индексов. *Индекс* (*index*) каждого элемента находится в диапазоне от 0 до N–1, где N – полное количество элементов в массиве. Одномерный массив (1D), показанный в табл. 7.1, иллюстрирует эту структуру.

Таблица 7.1

Индекс	0	1	2	3	4	5	6	7	8	9
10-элементный массив	12	32	82	8.0	4.8	5.1	6.0	1.0	2.5	1.7

Обратите внимание, что первый элемент имеет индекс 0, второй – 1 и т.д. Позже вы увидите, что осциллограммы (и другие типы данных) часто хранятся в виде массивов и каждая точка осциллограммы содержит элемент массива. Массивы также используются для хранения данных, сгенерированных в циклах, где на каждой итерации цикла создается один элемент массива.

7.2. Создание элементов управления и отображения массивов

Чтобы создать элементы управления и индикаторы для сложных типов данных, таких как массивы и кластеры, необходимо последовательно выполнить два действия. Вначале следует решить, будет ли ваш массив состоять из элементов управления или индикаторов, а затем объединить *шаблон массива* (array shell) с *объектом данных*, который может быть числовым, логическим, маршрутным или строковым (а также кластерным, но об этом позже). Шаблон массива находится в подпалитре **Массив и кластер** (Array&Cluster) палитры **Элементы управления**.

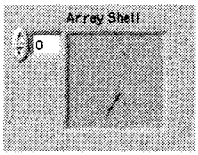


Рис. 7.1

Для создания массива переместите объект данных в окно отображения элементов. Можете непосредственно поместить туда объект, щелкнув правой кнопкой мыши внутри окна во время первоначального выбора объекта данных из палитры **Элементы управления**. Окно отображения элемента изменится в размерах, показывая, что произошло согласование типа данных, (рис. 7.2), но остается серым до тех пор, пока вы не введете в него данные. Обратите внимание, что все элементы массива должны быть либо элементами управления, либо индикаторами, но не их комбинацией.

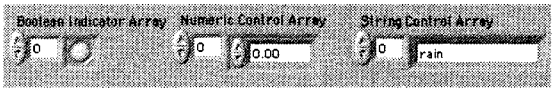
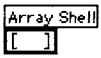


Рис. 7.2

Когда вы помещаете шаблон массива на лицевую панель, то его терминал на блок-диаграмме будет черного цвета, что характерно для неопределенного типа данных. В терминале показаны скобки, (рис. 7.3а), которые являются способом отображения структуры массива в LabVIEW. Когда вы зададите массиву тип данных (поместив элемент управления или отображения в окно отображения элемента), терминал блок-диаграммы массива присвоит себе его цвет и надпись (хотя скобки сохраняются), как это показано на рис. 7.3б. Заметьте: проводники массива толще, чем проводники, переносящие скалярные величины.

Вы можете ввести данные в массив, как только зададите их тип. Используйте инструменты ввода текста или управления для ввода данных или, если ваши

а)



б)

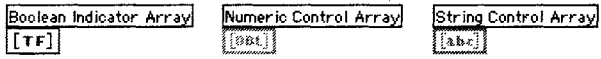


Рис. 7.3

данные имеют числовой характер, щелкните по стрелкам в окне отображения элементов, чтобы увеличить или уменьшить их значение.

Если нужно изменить размеры объекта в окне отображения, обратитесь к инструменту перемещения и убедитесь, что он превращается в стандартные скобки, когда вы помещаете его в углу окна (возможно, вам придется поместить его внутрь окна, а затем немного подвигать для получения соответствующих скобок). Если вы хотите одновременно показать большее количество элементов массива, передвигайте инструмент перемещения около угла окна отображения до тех пор, пока не появится курсор в виде сетки, а затем вытяните угол в горизонтальном или вертикальном направлениях (данные не изменятся из-за формы массива). В этом случае отобразится большее количество элементов. Самый близкий к индикатору индекса элемент всегда соответствует номеру, показанному на индикаторе.

Создать массив констант на блок-диаграмме можно так же, как при создании числовых, логических или строковых констант. Выбор опции **Постоянный массив** (Array Constant) в подпалитре **Массив** (Array) палитры **Функции** создает шаблон массива, куда вы просто помещаете подходящий тип данных (в виде константы). Эта возможность полезна при инициализации сдвиговых регистров или при передаче типа данных в функции работы с файлами или Internet (об этом мы будем говорить позднее).

Если вы захотите очистить элемент управления (отображения, константы данных) массивом, щелкните правой кнопкой мыши по индикатору индекса (Но не по самому элементу) и выберите опцию **Операции с данными ⇒ Очистить массив** (Data Operations ⇒ Empty Array).

7.3. Использование автоматического индексирования

Цикл с фиксированным числом итераций и цикл по условию могут автоматически индексировать и аккумулировать массивы на их границах путем добавления одного нового элемента в каждом повторении цикла. Эта способность называется *автоиндексированием* (auto-indexing). Следует помнить одну важную вещь: *по умолчанию автоиндексирование включено в цикле с фиксированным числом итераций и выключено в цикле по условию.* На рис. 7.4 показано, как цикл с фиксированным

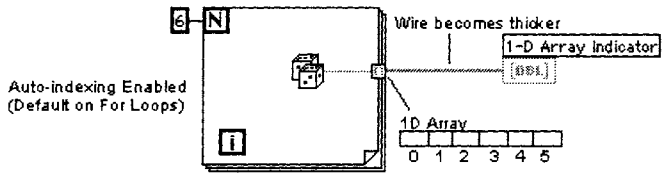


Рис. 7.4

числом итераций автоматически создает массив в его границах. Каждая итерация создает следующий элемент массива. После завершения цикла массив выходит из него и поступает на элемент отображения. Данные в массиве недоступны до тех пор, пока цикл не завершится. Обратите внимание, что проводник становится толще после превращения в проводник данных массива на границе цикла.

Если необходимо вывести скалярную величину из цикла с фиксированным числом итераций без создания массива, отключите автоиндексирование путем вызова контекстного меню в точке ввода/вывода (квадратик с символом [ ]) и выбора опции **Отключить индексирование** (Disable Indexing).

Поскольку по умолчанию автоиндексирование в цикле по условию не включено, то, если вы хотите вывести данные из этой структуры в виде массива, нужно вызвать контекстное меню в точке ввода/вывода и выбрать опцию **Включить индексирование** (Enable Indexing).

На рис. 7.6 автоиндексирование отключено, и лишь последняя величина, возвращенная функцией **Случайное число (0-1)**, выводится из цикла. Обратите внимание, что проводник сохраняет свои размеры после выхода из цикла. Всегда смотрите на размер проводника, поскольку автоиндексирование является общим источником проблем среди начинающих: они часто создают массивы, не желая этого, или не создают их, когда они нужны, а затем удивляются, почему проводник оказывается неисправным.

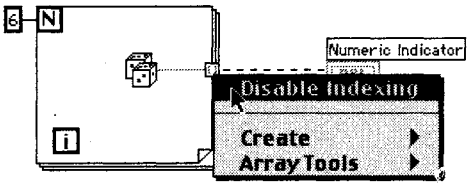


Рис. 7.5

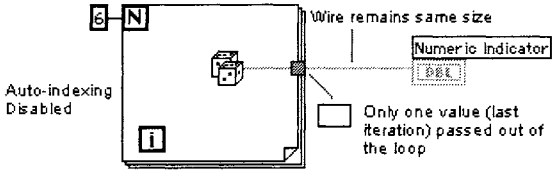


Рис. 7.6

Автоиндексирование применяется также в случаях, когда вы вводите массивы в циклы. Если индексирование включено, как это показано в цикле (A) на рис. 7.7, то цикл будет переходить к следующему индексу массива при каждой итерации (обратите внимание на утончение проводника во время его вхождения в цикл). Если же индексирование отключено, как показано в случае (B), то массив целиком передается в цикл.

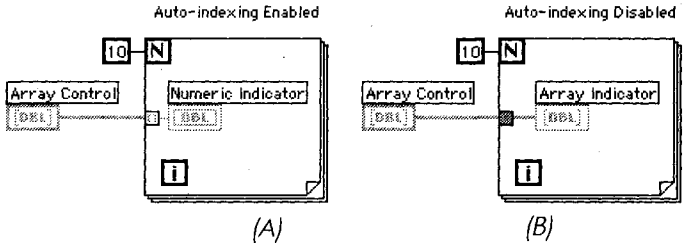


Рис. 7.7

Поскольку цикл с фиксированным числом итераций часто используется для обработки массивов, в LabVIEW по умолчанию включено автоиндексирование для этой структуры. Однако для цикла по условию автоиндексирование по умолчанию выключено. Если вы хотите его включить, вызовите контекстное меню точки ввода/вывода массива в цикл и выберите опцию **Включить индексирование**. Всегда проверяйте состояние индексации – в нем источник многих ошибок.

Использование автоиндексирования для установки количества повторений циклов с фиксированным числом итераций

Когда вы разрешаете автоиндексирование для массива, входящего в цикл с фиксированным числом итераций, LabVIEW автоматически устанавливает число повторений равным размеру массива, устраняя таким образом необходимость подключения определенного значения к терминалу числа итераций. Если вы зададите взаимоисключающие значения, например задав число и введя в действие автоиндексирование (или разрешив автоиндексировать два массива с разными размерами), то LabVIEW установит значение терминала количества итераций в наименьший из вариантов. На рис. 7.8 показано, что размер массива, а не число, подключенное к терминалу, определяет количество итераций цикла, так как размер массива является наименьшим из двух.

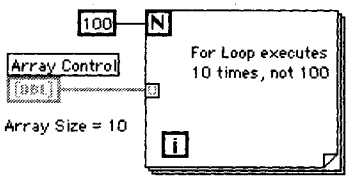


Рис. 7.8

7.4. Двумерные массивы

Двумерный (2D) массив хранит элементы в структуре вида решетки. Для определения местонахождения элемента необходимы два индекса: индекс по вертикали и индекс по горизонтали, каждый из которых начинается с нуля, как и все элементы с индексами в LabVIEW. На рис. 7.9 показан массив с четырьмя строками и шестью столбцами, в котором хранится 6×4 элементов.

Grid Cursor

Вы можете увеличить размерность элементов управления/отображения массивов путем вызова контекстного меню элемента управления/отображения *индекса массива* (но не элемента управления/отображения значениями массива) и выбора опции **Добавить размерность** (Add Dimension). На рис. 7.10 показан двумерный массив цифровых элементов управления. Обратите внимание, что теперь есть два индекса для отображения каждого элемента. Вы можете использовать инструмент перемещения в виде сетки для расширения окна, отображающего значения массива, в двух измерениях, что позволит одновременно увидеть большее количество элементов.

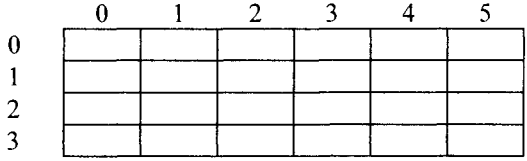


Рис. 7.9. Массив с шестью столбцами и четырьмя строками – 24 элемента

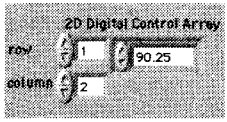


Рис. 7.10

Удалить ненужные размерности можно выбором опции **Удалить размерность** (Remove Dimension) из контекстного меню элемента управления/отображения индекса.

Осциллограммы, полученные из разных каналов многофункциональной платы ввода/вывода, допустимо сохранить в двумерном массиве, в каждом столбце которого будут храниться данные одного канала.

Создание двумерных массивов

Если вы не хотите вводить данные с лицевой панели, то для создания двумерного массива можно использовать два цикла с фиксированным числом итераций, один внутри другого. Внутренний цикл создает строку, а внешний складывает эти строки для заполнения столбцов массива. На рис. 7.11 показаны два цикла с фиксированным числом итераций, которые создают двумерный массив случайных чисел с применением автоиндексирования.

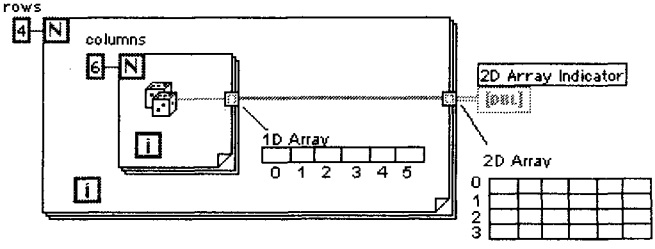


Рис. 7.11

7.5. Упражнение 7.1: создание массивов с помощью автоиндексирования

Теперь у вас есть возможность лучше познакомиться с массивами и автоиндексированием посредством работы с ними. В этом упражнении вы увидите ВП для создания массивов данных, в котором используется автоиндексирование как в цикле с фиксированным числом итераций, так и в цикле по условию.

1. Откройте пример **Building Arrays.vi**, расположенный в директории EVERYONE\CH7.LLB. Этот прибор создает два массива и отображает их на лицевой панели, используя при этом цикл с фиксированным числом итераций для создания двумерного массива и цикл по условию для создания одномерного массива. Цикл с фиксированным числом итераций выполняется определенное количество раз, для остановки цикла по условию нужно нажать кнопку **Стоп** (или он остановится сам после 101 повторения).
2. Взгляните на лицевую панель, затем перейдите к блок-диаграмме. Обратите внимание на то, как циклы с фиксированным числом итераций создают строки и столбцы двумерного массива на своих границах, используя автоиндексирование. Заметьте также, как автоиндексированные проводники утолщаются по мере выхода за границы цикла.

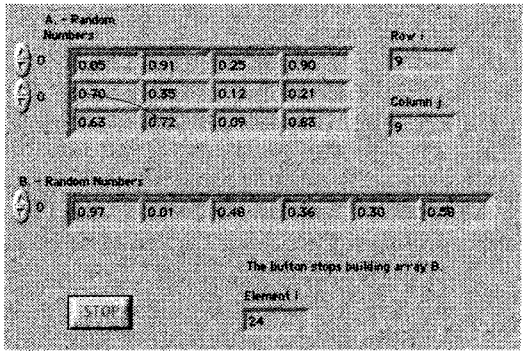


Рис. 7.12

3. Прежде чем получить данные из цикла по условию, щелкните правой кнопкой мыши по точке ввода/вывода, содержащей случайное число, и выберите опцию **Включить индексирование**. Чтобы увидеть, как это работает, щелкните правой кнопкой мыши по точке ввода/вывода и выберите опцию **Отключить индексирование**. Проводник, выходящий из цикла, будет разорван. Снова щелкните правой кнопкой мыши по точке ввода/вывода и выберите опцию **Включить индексирование** для восстановления проводника.

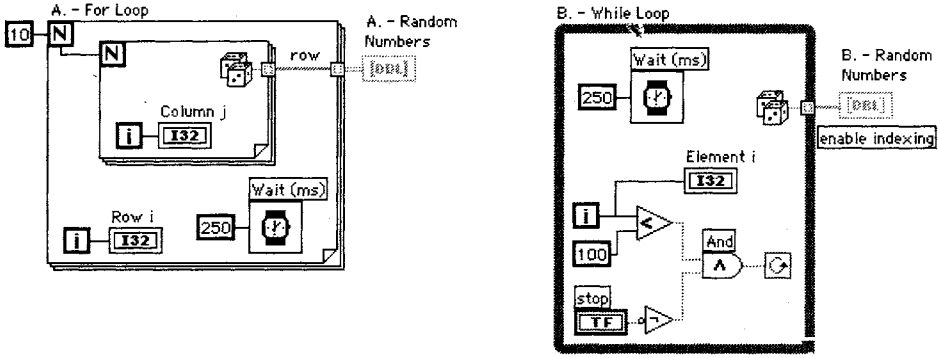


Рис. 7.13

В этом цикле используется логический алгоритм, гарантирующий остановку цикла, если пользователь не нажал кнопку **Стоп**, после определенного количества итераций (101 повторение). Если пользователь не нажимает кнопку **Стоп**, а цикл выполнился менее 101 раза, то он будет продолжать выполняться. Если любое из этих условий изменяется, то цикл останавливается.

Почему цикл должен выполняться 101 раз, а не 100? Вспомните, что цикл по условию проверяет терминал условия выхода в конце каждого повторения. В конце 100-го повторения i = 99 (поскольку начинается с нуля), а не 100, и цикл продолжает выполняться. В конце 101-го повторения значение счетчика становится равным 100, и цикл останавливается (при условии, что он не прерван нажатием кнопки **Стоп**).

4. Запустите ВП. Не забудьте нажать кнопку **Стоп** для остановки цикла по условию, поскольку элемент отображения на лицевой панели не обновится до тех пор, пока не будет создан весь массив (значения элементов управления и отображения за пределами цикла не считываются и не обновляются во время его выполнения).
5. Закройте ВП без сохранения каких-либо изменений.

7.6. Функции работы с массивами

LabVIEW имеет много функций манипулирования массивами (подпалитра **Массив** палитры **Функции**). Напоминаем, что индексы массивов начинают отсчитываться с нуля – первый элемент имеет индекс 0, второй – индекс 1 и т.д. В данном разделе рассматриваются некоторые общие функции, но вы можете обратится к подпалитре **Массив**, чтобы увидеть другие встроенные функции:

- функция **Инициализация массива** (Initialize Array) создает и заполняет все элементы n-мерного массива значением по вашему выбору (рис. 7.14). Вы можете изменить размерность массива растягиванием его границ инструментом перемещения («стрелка») для создания дополнительных

вводов **числа элементов размерности** (dimension size). Эта функция полезна для выделения памяти определенного размера или для инициализации сдвиговых регистров данными из массива;
Рис. 7.15 показывает, как создать с помощью функции **Инициализация массива** десятиэлементный одномерный массив, в котором каждый элемент содержит значение 0;

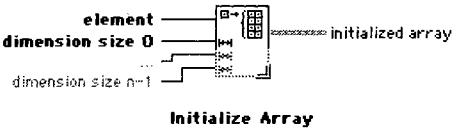


Рис. 7.14. Функция ***Инициализация массива***

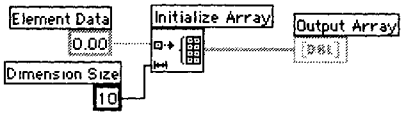


Рис. 7.15

- функция **Число элементов массива** (Array Size) возвращает число элементов входного массива (рис. 7.16). Если входной массив является n-мерным, то функция возвращает одномерный n-элементный массив, в котором каждый элемент содержит число элементов одной из размерностей массива (рис. 7.17);

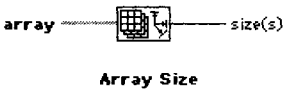


Рис. 7.16. Функция ***Число элементов массива***

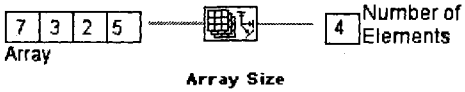


Рис. 7.17

- в зависимости от конфигурации функция **Создать массив** (Build Array) комбинирует или объединяет два массива или добавляет в массив дополнительные элементы (рис. 7.18). Будучи впервые помещена на блок-диаграмму, функция выглядит как иконка, изображенная слева. Вы можете растянуть границу этой функции для увеличения количества вводов. Функция **Создать массив** имеет два типа ввода: для *массива* и для *элемента*. Таким образом, с ее помощью легко создавать массив одновременно из массива данных и из скалярных величин.
Например, функция **Создать массив**, изображенная на рис. 7.19, объединяет два массива и одну скалярную величину в новый массив.

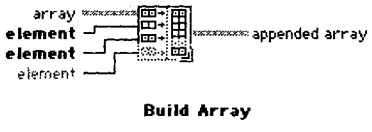


Рис. 7.18. Функция ***Создать массив***

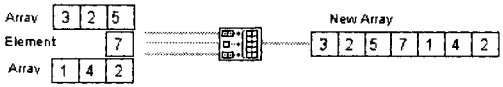


Рис. 7.19

Обращайте особое внимание на вводы функции **Создать массив**. Ввод для массива изображается в виде двух квадратиков с точками внутри, в то время как ввод скалярной величины показан пустым квадратиком. Хотя LabVIEW адаптирует ввод в зависимости от типа данных, которые вы подаете (массив или скаляр), эти типы данных не являются взаимозаменяемыми. Поэтому, не удивляйтесь, если у вас неожиданно возникнут неисправные проводники.

В зависимости от того, какой тип данных подключается, ввод этой функции автоматически адаптируется к входному элементу, будь это скаляр или массив.

С помощью функции **Создать массив** можно создавать или добавлять элементы в многомерные массивы. Элемент, добавляемый в многомерный массив, должен быть на одну размерность меньше массива (например, допустимо добавить 1D-массив в 2D-массив). Вы можете построить двумерный массив, используя функцию **Создать массив** и подсоединяя одномерные массивы в качестве элементов (каждый одномерный массив становится строкой двумерного массива). Если же необходимо объединить несколько одномерных массивов вместо создания одного двумерного, в контекстном меню функции **Создать массив** выберите опцию **Объединить вводы** (Concatenate Inputs);

- функция **Подмножество массива** (Array Subset) возвращает часть массива, который содержит количество элементов **длина** (length), начиная с **индекса** (index) – рис. 7.20. Не забывайте, что индексом третьего элемента является 2, так как индексирование начинается с нуля (рис. 7.21);

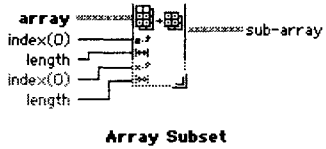


*Рис. 7.20. Функция **Подмножество массива***

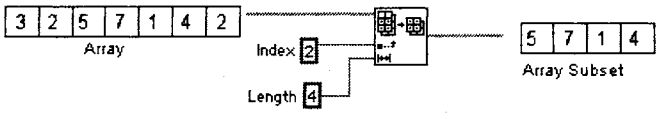


Рис. 7.21

- функция **Выборка из массива** (Index Array) осуществляет доступ к любому элементу массива. На рис. 7.22 показано, как с помощью этой функции

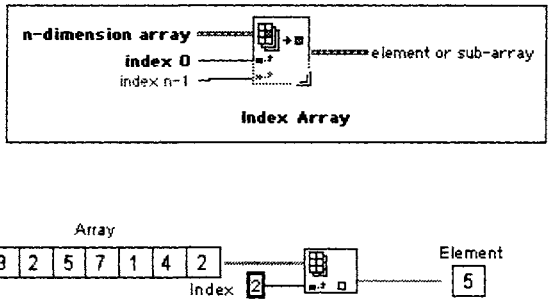


Рис. 7.22

осуществляется доступ к третьему элементу массива. В данном случае функция **Выборка из массива** извлекает скалярный элемент из массива. Чтобы извлечь отдельный скалярный элемент, соедините индекс строки нужного элемента с верхним входом, а индекс столбца – с нижним входом функции.

Вы также можете использовать эту функцию для «вырезания» из массива строки, столбца или скалярного элемента. Чтобы это сделать, оставьте один из вводов функции неподключенным. При вырезании строки двумерного массива соедините значение индекса выбранной строки с первым индексным вводом функции. Чтобы отделить столбец двумерного массива, оставьте ввод первого индекса неподключенным и соедините значение индекса выбранного столбца со вторым индексным вводом.

Обратите внимание, что, когда вы оставляете ввод неподключенным, символ соответствующего индекса изображается в виде пустого прямоугольника. На рис. 7.23 показан процесс вырезания столбца и строки из двумерного массива.

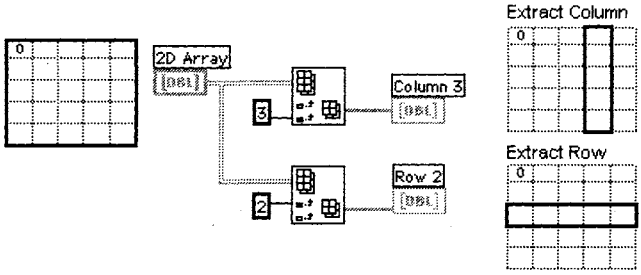


Рис. 7.23

7.7. Упражнение 7.2: работа с массивами

В этом упражнении вы создадите ВП, объединяющий два массива, а затем вырезающий элемент из середины нового объединенного массива.

1. Откройте ВП **Array Exercise.vi**, который расположен в директории EVERYONE\CH7.LLB. Лицевая панель этого прибора содержит два входных массива (каждый показывает три элемента), два цифровых элемента управления и выходной массив (показывающий восемь элементов). Для создания нового массива ВП объединяет два массива и значения элементов управления в такой последовательности:

   Начальный массив + элемент 1 + элемент 2 + завершающий массив.

   Лицевая панель уже создана. Закончите построение блок-диаграммы. Элементы управления и отображения массивами будут иметь серый цвет, пока вы или программа не введете в них какие-либо данные (рис. 7.24).
2. Постройте блок-диаграмму, как показано на рис. 7.25. Используйте окно контекстной помощи для нахождения соответствующих терминалов функций.

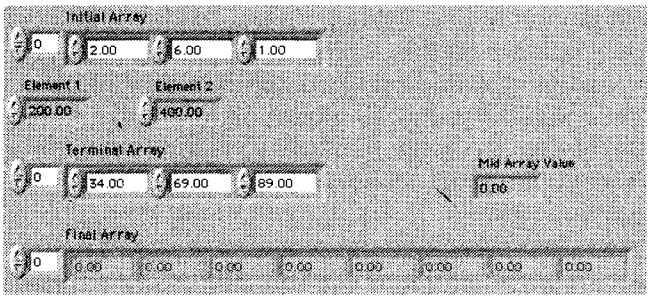


Рис. 7.24

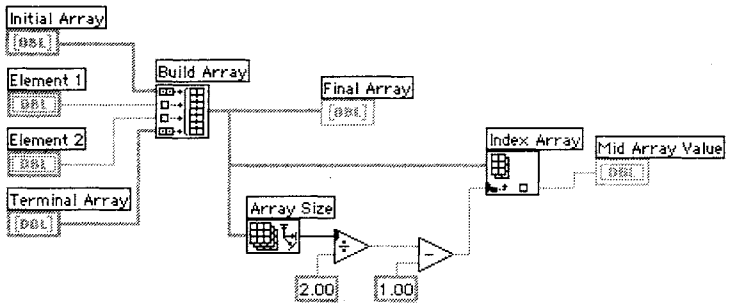


Рис. 7.25


Build Arrayfunctio

Функция **Создать массив** (палитра **Массив**) в этом упражнении соединяет входные данные для создания нового массива в такой последовательности:

Начальный массив + элемент 1 + элемент 2 + завершающий массив.


Build Array

Функция, помещенная на блок-диаграмму, выглядит, как иконка слева. Установите инструмент перемещения в нижнем правом углу функции и измените ее размеры так, чтобы она включала четыре ввода.


Array Input


Element Input

Вводы автоматически определят, какой тип данных вы подключаете: массив или скаляр. Символы слева отображают массив или скаляр.


Array Size

Функция **Число элементов массива** (палитра **Массив**) возвращает число элементов в объединенном массиве.


Index Array

Функция **Выборка из массива** возвращает элемент из середины массива.

LabVIEW создает массив с помощью функции **Создать массив**. Затем вычисляется индекс среднего элемента массива, равный половине длины массива минус единица (отсчет начинается с нуля). Поскольку массив имеет четное количество элементов, то серединным будет один из двух средних элементов.

3. Вернитесь к лицевой панели и запустите ВП. Попробуйте сделать несколько различных комбинаций входных данных.
4. Сохраните ВП в директории `MYWORK`, после этого закройте его.

Вы также можете взглянуть на программы в директории `EXAMPLES/GENERAL/ARRAYS.LLB`, чтобы узнать, что еще можно делать с массивами данных

7.8. Полиморфизм

Другим полезным качеством LabVIEW является *полиморфизм* (polymorphism) его арифметических функций: **Сложения** (Add), **Умножения** (Multiply), **Деления** (Divide) и т.д. Полиморфизм – это длинное слово для названия простого принципа: входные данные функций могут иметь различные размерности и представления. Например, используя одну и ту же функцию, вы можете сложить скалярную величину и массив или сложить два массива. На рис. 7.26 показаны некоторые полиморфические комбинации функции **Сложения**.

Результатом первой комбинации является скалярная величина. Во второй комбинации скалярная величина добавляется к каждому элементу массива. В третьей комбинации каждый элемент массива добавляется к соответствующему элементу другого массива. Во всех случаях используется та же самая функция **Сложение**, но она выполняет различные виды действий.

Как показано на рис. 7.27, на каждом повторении цикла с фиксированным числом итераций вырабатывается одно случайное число (в диапазоне от 0 до 1),

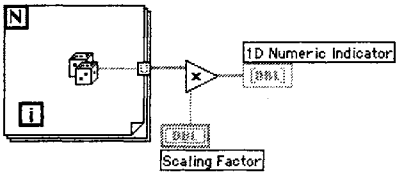


Рис. 7.27

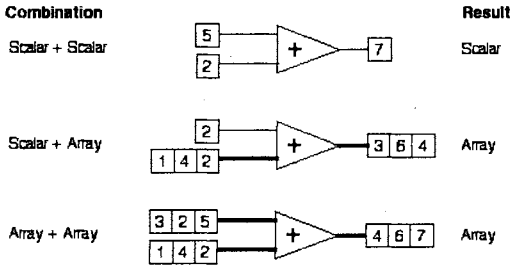


Рис. 7.26

которое сохраняется в массиве на границе цикла. После того как цикл завершится, функция **Умножение** умножает каждый элемент массива на заданный вами масштабный коэффициент. Элемент отображения массива на лицевой панели покажет масштабированный массив.

На рис. 7.28 представлены некоторые возможные полиморфические комбинации функции **Сложение**. Подробнее о кластерах вы узнаете в следующем разделе.

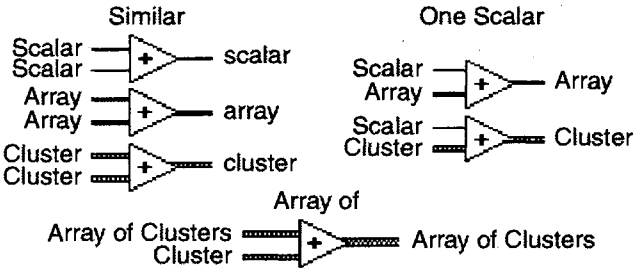


Рис. 7.28

Если вы выполняете арифметические действия с массивами с разным количеством элементов, то результирующий массив будет иметь размерность наименьшего из двух. Другими словами, LabVIEW выполняет действия над соответствующими элементами двух массивов до тех пор, пока элементы одного из них не кончатся. Оставшиеся элементы более длинного массива игнорируются.

7.9. Упражнение 7.3: полиморфизм на примере массивов

В этом упражнении вы создадите ВП, который демонстрирует полиморфизм на примере массивов.

Grid Cursor

1. Откройте новую панель и создайте ВП, показанный на рис. 7.29. Вначале создайте два массива, прежде всего выбрав шаблон массива в подпалитре **Массив и кластер** палитры **Функции**. Затем поместите числовой элемент отображения в окно шаблона. Для того чтобы увидеть несколько элементов массива, вы должны передвинуть угол окна отображения элементов с помощью сеточного курсора инструмента перемещения. Вы можете сделать видимыми одновременно большое количество элементов одномерного массива, вытягивая границу окна в горизонтальном или в вертикальном направлении.

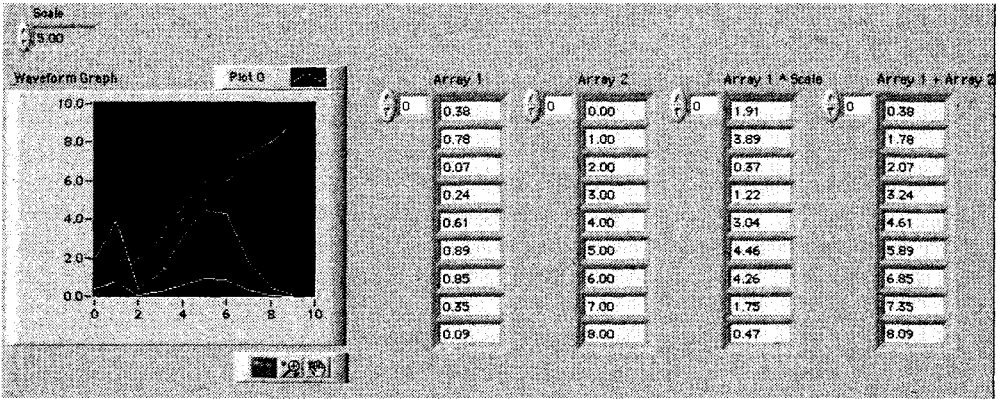


Рис. 7.29

Все четыре массива в этом упражнении содержат элементы отображения. Присвойте им особые ярлыки, чтобы в дальнейшем их не перепутать. Если вы все-таки забудете, какой объект лицевой панели соответствует какому терминалу блок-диаграммы, то просто щелкните правой кнопкой мыши по любому из них и выберите опцию **Найти терминал** (Find Terminal) или **Найти индикатор** (Find Indicator), и LabVIEW выделит соответствующий объект.

2. После того как вы создали массивы, выберите функцию **График осциллограммы** в подпалитре **График** (Graph) палитры **Элементы управления**. Подробнее о графиках вы узнаете в следующей главе.
3. Не забудьте создать элемент управления масштабом.
4. Постройте блок-диаграмму, как показано на рис. 7.30. Будьте внимательны: соединение элементов может быть довольно запутанным.
   В цикле с фиксированным числом итераций автоиндексирование задействовано по умолчанию, поэтому массивы будут создаваться автоматически.
5. Функции **Сложение**, **Умножение** и **Случайное число (0-1)** находятся в палитре **Числовые**.
6. Используйте функцию **Создать массив** из палитры **Массив**. Увеличьте ее с помощью инструмента перемещения таким образом, чтобы она

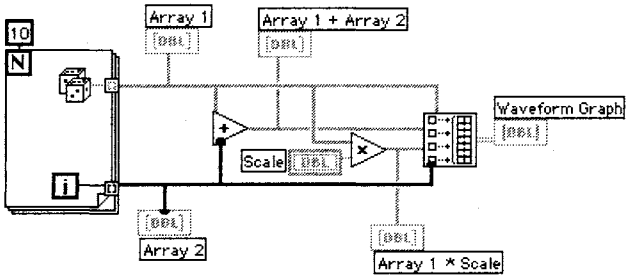


Рис. 7.30

имела четыре ввода. По умолчанию вводы функции не объединяются. На выходе функции **Создать массив** будет двумерный массив. Каждый входной массив становится строкой, так что выходной двумерный массив состоит из четырех строк и десяти столбцов.

7. Запустите ВП. На графике каждый элемент массива располагается напротив своего индекса одновременно для всех четырех массивов: **данные массива1**, **данные массива2**, **данные массива1×масштаб** и **массив1+массив2**.

   Эти результаты демонстрируют несколько видов использования полиморфизма в LabVIEW. Например, **массив1** и **массив2** могут быть входными осциллограммами, которые нужно масштабировать.
8. Сохраните ВП как **Polymorphism Example.vi** и разместите его в директории `MYWORK` или в библиотеке ВП. Закройте виртуальный прибор.

7.10. Составная арифметика

Имея дело с арифметикой, необходимо упомянуть функцию **Составная арифметика** (Compound Arithmetic). Эта функция дает возможность оперировать более чем двумя числами одновременно. Функция **Составная арифметика** устраняет необходимость использования множества терминалов **Сложение**, **Умножение**, **И**, **ИЛИ** и **Исключающее ИЛИ** при осуществлении действий с одной из этих функций с несколькими числами одновременно (**И**, **ИЛИ** и **Исключающее ИЛИ** являются логическими (булевыми) арифметическими действиями).

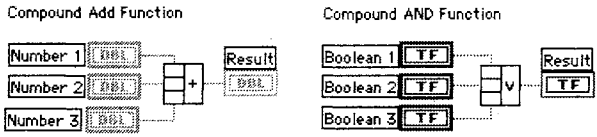


*Рис. 7.31. Составная функция **Сложение** (слева); составная функция **И** (справа)*

Функцию **Составная арифметика** можно найти в подпалитрах **Числовые** или **Логические** палитры **Функции**. Как и при использовании многих других функций, допустимо увеличить ее размер и создать больше вводов. Эта функция имеет только одну форму. Чтобы изменить вид действия (из набора **Сложение**, **Умножение**, **И**, **ИЛИ** и **Исключающее ИЛИ**), щелкните правой кнопкой мыши по выводу функции и выберите опцию **Сменить режим** (Change Modes). Вы также можете щелкнуть инструментом управления («палец») по виду действия для его модификации.

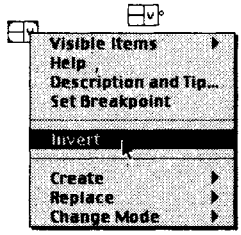


Рис. 7.32

Для изменения знака числовых вводов и вывода или логических значений (от ЛОЖЬ до ИСТИНА или наоборот) выберите опцию **Инвертировать** (Invert) в контекстном меню. Маленький кружок на вводе или выводе символизирует инвертированное значение.

Несколько слов о логической арифметике

Логические арифметические действия – **И** (And), **ИЛИ** (Or), **НЕ** (Not), **Исключающее ИЛИ** (Exclusive Or), **Исключающее ИЛИ–НЕ** (Not Exclusive Or), **И–НЕ** (Not And) и **ИЛИ–НЕ** (Not Or) – могут оказаться весьма полезными. Примеры с использованием логической арифметики встречаются во многих разделах книги. В данном разделе мы остановимся лишь на некоторых основных функциях. Если вы не помните, что делает та или иная функция, обратитесь к окну помощи.

Функция **НЕ** является наиболее легкой для описания, поскольку она просто инвертирует входное значение. Если значение входа ИСТИНА, то функция **НЕ** преобразует его на выходе в значение ЛОЖЬ. Если значение входа ЛОЖЬ, то **НЕ** изменяет его на выходе в ИСТИНА.

Функция **И** создает на своем выходе значение ИСТИНА только в том случае, если все вводы имеют значения ИСТИНА.

Функция **ИЛИ** создает на выходе значение ИСТИНА, если хотя бы один вход имеет значение ИСТИНА.

Функции **И** и **ИЛИ** имеют следующие выходные значения в зависимости от входных данных:

ЛОЖЬ И ЛОЖЬ = ЛОЖЬ
ИСТИНА И ЛОЖЬ = ЛОЖЬ
ЛОЖЬ И ИСТИНА = ЛОЖЬ
ИСТИНА И ИСТИНА = ИСТИНА

ЛОЖЬ ИЛИ ЛОЖЬ = ЛОЖЬ
ИСТИНА ИЛИ ЛОЖЬ = ИСТИНА
ЛОЖЬ ИЛИ ИСТИНА = ИСТИНА
ИСТИНА ИЛИ ИСТИНА = ИСТИНА

7.11. Все о кластерах

Теперь, когда вы достаточно хорошо познакомились с массивами данных, вы легко поймете, что такое кластеры. Как и массив, *кластер* (cluster) является структурой, группирующей данные. Однако в отличие от массива кластер может группировать

данные различных типов (числовые, логические и т.д.). Это понятие аналогично *struct* в языке программирования C или объектам данных, определенным как *элементы класса*, в C++ или Java. Кластер может быть мысленно представлен в виде связки проводов, как в телефонном кабеле. Каждый провод в кабеле представляет элемент кластера. Поскольку кластер имеет только один «провод» на блок-диаграмме (несмотря на то, что по нему проходит множество данных разных типов), кластеры уменьшают нагромождение проводников и количество терминалов подключения, необходимых для подпрограмм (рис. 7.33). В дальнейшем вы обнаружите, что данные в виде кластеров часто появляются во время их вывода на графики и диаграммы.

Доступ к элементам кластера можно получить путем их полного *разделения* (unbundling) или разделения по индексу элемента. Метод разделения зависит от выбранной вами функции и имеет свою область применения. Разделение элементов кластера можно представить как расщепление разноцветных проводов в телефонном кабеле. В отличие от массивов, которые могут динамически изменять размер, кластеры имеют фиксированный размер или фиксированное количество проводов (рис. 7.34).

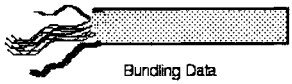


Рис. 7.33. Объединение данных в кластер

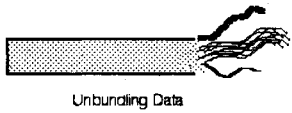


Рис. 7.34. Разделение кластера

Вы можете соединить терминалы кластеров только в том случае, если они имеют одинаковый тип; другими словами, оба кластера должны иметь одинаковое количество элементов и соответствующие элементы должны совпадать как по типу данных, так и по их порядку. Принцип полиморфизма может быть также применен и к кластерам при условии совпадения типа данных.

Кластеры часто встречаются при обработке ошибок. На рис. 7.35 показаны кластеры ошибок, **Error In.ctl** и **Error Out.ctl**, которые используются LabVIEW для передачи сведений об ошибках среди множества виртуальных приборов на блок-диаграмме (например, большинство ВП сбора данных и ВПП ввода/вывода в файл/из файла имеют встроенные кластеры обработки ошибок). Эти кластеры ошибок применяются настолько часто, что их специально выделили для более легкого доступа к ним – соответствующие ВП есть в подпалитре **Массив и кластер** палитры **Элементы управления**.

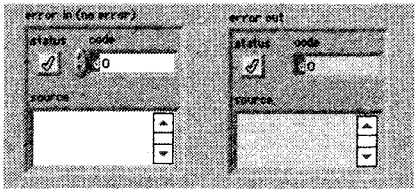


Рис. 7.35

7.12. Создание элементов управления и отображения для кластеров

Создайте кластер, поместив шаблон **Кластер** (Cluster) из подпалитры **Массив и Кластер** на лицевую панель. Вы теперь можете разместить любой объект лицевой панели внутри кластера. Как в случае с массивами, допустимо поместить объекты внутрь кластера напрямую при извлечении их из палитры **Элементы управления** либо перетащить существующий объект и поместить его в кластер. *Объектами внутри кластера могут быть только элементы управления либо только индикаторы.* Нельзя поместить элементы управления и индикаторы в одном кластере, так как сам кластер должен быть или первого, или второго типа. Кластер становится элементом управления или индикаторы в зависимости от типа первого внесенного в него объекта. В случае необходимости можно изменить размеры кластера с помощью инструмента перемещения. На рис. 7.36 изображен кластер с четырьмя элементами управления.

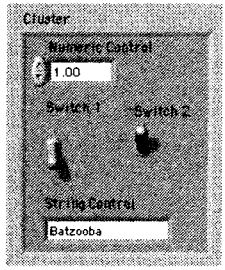


Рис. 7.36

Создать кластер из констант на блок-диаграмме вы можете в два этапа аналогично массивам.

Если вы хотите, чтобы визуальные размеры кластера соответствовали объектам внутри него, щелкните правой кнопкой мыши по его границе (не внутри кластера) и выберите опцию **Автоматическое установление размера** (Autosizing).

7.13. Упорядочивание элементов кластера

Элементы кластера имеют логический порядок независимо от их местоположения в рамках шаблона. Первый объект, помещенный в кластер, становится нулевым элементом, второй – первым и т.д. При удалении одного элемента порядок автоматически изменяется. *Если вы хотите соединить один кластер с другим, то должны четко отслеживать порядок элементов вашего кластера, поскольку порядок и тип данных должны быть идентичными.* Кроме этого, при необходимости сразу разделить весь кластер нужно знать, какое значение соответствует какому выходу в функции **Разделить** (Unbundle) – подробнее об этом рассказывается в разделе 7.17.

Изменить порядок в пределах кластера можно, вызвав контекстное меню на его границе и выбрав опцию **Изменить порядок элементов кластера** (Reoder Controls in Cluster). На инструментальной панели появляется новый набор кнопок, а внешний вид кластера изменяется (рис. 7.37).

Белые окна элементов показывают их текущие порядковые номера, черные окна – новые. Щелчок по элементу присваивает ему номер, отображенный на инструментальной панели. Вы можете ввести новый номер в это поле, прежде чем щелкнуть по объекту.

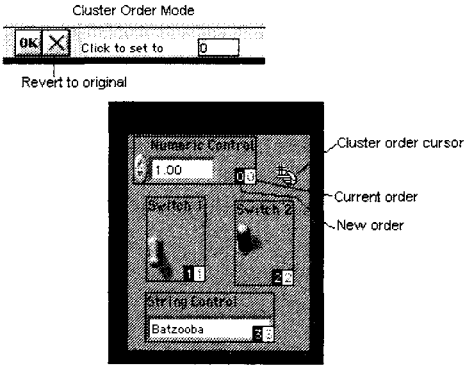


Рис. 7.37


Revert


ok

Если вам не нравятся сделанные изменения, вернитесь к прежнему порядку, щелкнув по кнопке **Revert**. Если же вы создали желаемый порядок, то можете сохранить его и вернуться к лицевой панели, щелкнув мышью по кнопке **OK**. При этом вы выйдете из режима редактирования порядка элементов кластера.

7.14. Использование кластеров для подачи и получения данных в/из ВПП

Соединительная панель ВП включает максимум 28 вводов/выводов. Возможно, вы не захотите подавать информацию на все 28 входов при вызове подприбора, так как соединение элементов может оказаться весьма затруднительным делом, и здесь легко допустить ошибку. При объединении ряда элементов управления или отображения в кластер допустимо применять один ввод для передачи или извлечения информации в/из ВПП. Вы можете использовать кластеры, чтобы обойти предел в 28 вводов/выводов, или довольствоваться меньшим количеством входов (что делает их более крупными и легкими для подключения).

7.15. Объединение данных


Bundle

Функция **Объединить** (Bundle) из палитры **Кластер** объединяет отдельные компоненты в кластер или позволяет заменить элементы в существующем кластере (рис. 7.38). Когда вы помещаете функцию на блок-диаграмму, она имеет вид иконки слева. Вы можете увеличить количество входов, вытягивая угол функции инструментом перемещения. При подключении к каждому вводу в его пустом поле появляется символ, указывающий на тип подключаемых данных. Порядком элементов результирующего кластера будет порядок входов в функцию **Объединить**.

Если вы хотите лишь создать кластер, нет необходимости подключать какой-либо проводник к центральному входу кластера (с названием **кластер** – cluster) функции **Объединить**. Но этот вход следует подключать, если вы заменяете элемент кластера.

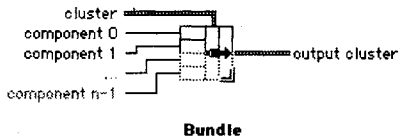


Рис. 7.38. Функция ***Объединить***

Будьте осторожны при работе с описанной функцией. Если вы добавите один элемент в кластер без предварительной подгонки размеров функции **Объединить** на блок-диаграмме, программа не будет работать.

7.16. Замена элемента кластера

Если вы хотите *заменить* какой-либо элемент в кластере, то вначале измените размер функции **Объединить** таким образом, чтобы она содержала столько же входных терминалов, сколько элементов содержит кластер (в противном случае вы получите разорванный проводник). Затем соедините кластер со средним терминалом функции **Объединить** (на входах функции **Объединить** появятся символы типов данных элементов внутри кластера). После этого подключите новые значения к соответствующим входам функции. Нужно подключиться только к тем терминалам, элементы которых в кластере подлежат изменению.

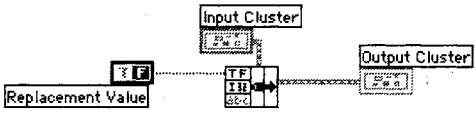


Рис. 7.39

7.17. Разделение кластеров

Функция **Разделить** (Unbundle) из палитры **Кластер** разделяет кластер на компоненты. *Выходные компоненты расположены сверху вниз в том же порядке, что и в кластере.* Если компоненты принадлежат к одному типу, то порядок элементов в кластере является единственным способом их различения. Когда вы помещаете функцию на блок-диаграмму, она имеет вид иконки, изображенной слева. Увеличить число выходов можно вытягиванием угла функции инструментом перемещения. Вам необходимо изменить размер функции **Разделить**, чтобы она содержала столько же выходов, сколько имеется элементов во входном кластере, иначе появится разорванный проводник. Когда вы подключаете кластер к функции **Разделить** с правильными размерами, то в ранее пустых выходных терминалах появятся символы типов данных кластера.

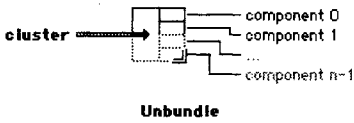


Рис. 7.40. Функция ***Разделить***

Знать порядок элементов в кластере необходимо для работы с его данными при помощи функций **Объединить** и **Разделить**. Например, если в кластере содержатся два булевых элемента управления, то очень легко ошибиться и выбрать вместо первого выключателя второй, поскольку с помощью функции **Разделить** элементы кластера разделяются по порядку, а не по имени. При этом все проводники в вашем приборе будут в рабочем состоянии, а результат окажется неправильным.

Будьте осторожны, пользуясь этой функцией. Если вы добавите один элемент в кластер без предварительной подгонки размеров функции **Разделить** на блок-диаграмме, то программа не будет работать!

В LabVIEW есть способ объединения и разделения кластеров по имени элементов. Но об этом немного позже.

7.18. Упражнение 7.4: работа с кластером

В этом упражнении вы создадите ВП для обучения работе с кластерами. Вы создадите кластер, разделите его на элементы, затем вновь соедините элементы и отобразите значения в новом кластере.

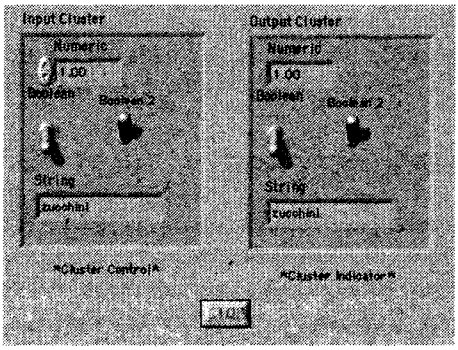


Рис. 7.41

1. Откройте новую панель и поместите в нее шаблон **Кластер** (палитра **Массив и кластер**). Назовите его **Входной кластер**. Увеличьте шаблон (убедитесь, что вы захватили границу кластера, – в противном случае ничего не произойдет).
2. Поместите внутрь шаблона **Входной кластер** цифровой элемент управления, два логических переключателя и элемент управления строками.
3. Теперь создайте шаблон **Выходной кластер** путем копирования входного. Щелкните правой кнопкой мыши по объекту в новом кластере (или на границе кластера) и выберите опцию **Заменить на индикатор** (Change to Indicator). Смените также ярлык кластера. Эта методика

позволяет получить правильный порядок в кластере и является достаточно эффективной.

Вы можете создать выходной кластер точно таким же способом, как и входной, используя элементы отображения вместо элементов управления (внимательно следите за порядком помещения элементов в кластер – он должен совпадать с порядком элементов входного кластера).

4. Убедитесь, что элементы входного и выходного кластеров имеют одинаковый порядок. Это можно сделать, вызвав контекстное меню на границе каждого кластера и выбрав опцию **Изменить порядок элементов кластера**. Если порядки элементов кластеров разные, то сделайте один из них аналогичным другому.
5. Поместите на лицевой панели прямоугольную кнопку **Стоп** (Stop Button) из палитры **Логические**. Обратите внимание, что по умолчанию состояние этой кнопки – ЛОЖЬ. Не меняйте этого состояния.

6. Постройте блок-диаграмму, как показано на рис. 7.42. Отметьте тот факт, что, хотя каждый кластер содержит четыре объекта, на блок-диаграмме вы увидите лишь по одному терминалу на кластер. Не забудьте изменить функцию терминала условия выхода из цикла по условию так, чтобы она была **Остановить, если Истина** (это можно сделать вызовом контекстного меню терминала условия выхода).

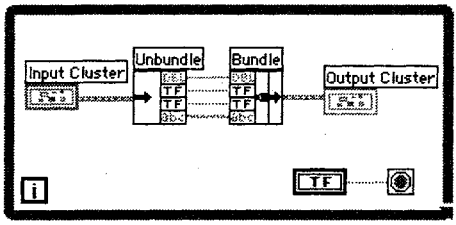


Рис. 7.42

Функция **Разделить** разделяет кластер. После этого вы получаете доступ к отдельным элементам кластера. Измените ее размер таким образом, чтобы она имела четыре выхода. Как только вы подключите к функции **Разделить** входной кластер, в ее терминалах появятся символы типов данных.

Функция **Объединить** воссоздает кластер. Измените ее размер, чтобы она имела четыре входа.

Доступ к функциям **Объединить** и **Разделить** можно получить, вызвав контекстное меню на терминале кластера, к которому вы хотите подключиться, и выбрав соответствующую функцию в меню **Кластерные инструменты** (Cluster Tools). Эта функция будет содержать правильное количество вводов и выводов.

7. Вернитесь к лицевой панели и запустите ВП. Введите различные значения элементов кластера управления и проследите, как кластер отображения

повторяет эти значения. Нажмите кнопку **Стоп**, чтобы остановить выполнение.

Вы, наверное, заметили, что было бы проще соединить выход входного кластера с входом выходного кластера, и ВП сделал бы то же самое, но мы хотели, чтобы вы научились работать с функциями **Объединить** и **Разделить**.

8. Закройте и сохраните ВП как **Cluster Exercise.vi** в директории `MYWORK` или в библиотеке виртуальных приборов.

7.19. Объединение и разделение по имени

Часто не требуется разделять весь кластер – нужен лишь один или два элемента. Для этого можно воспользоваться функциями **Объединить по имени** (Bundle by Name) и **Разделить по имени** (Unbundle by Name).

Функция **Объединить по имени** (рис. 7.43) находится в палитре **Кластер**. Она обращается к элементам по имени, а не по порядку (как это делает функция **Объединить**). Здесь вы можете получить доступ только к необходимым вам элементам. В то же время функция **Объединить по имени** не способна создавать новые кластеры; она вправе лишь заменить один элемент в существующем кластере. В отличие от функции **Объединить**, среднему входному терминалу функции **Объединить по имени** всегда необходимо подключение, чтобы сообщить функции, в каком кластере нужно заменить элемент.

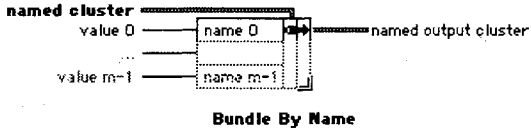


*Рис. 7.43. Функция **Объединить по имени***

Функция **Разделить по имени** (рис. 7.44), которая также находится в палитре **Кластер**, возвращает элементы, имена которых вы задали. Вам не нужно думать о порядке элементов кластера или корректировать размер функции **Разделить**.

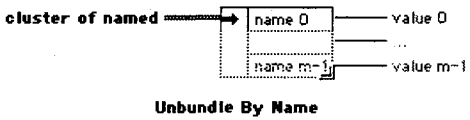


*Рис. 7.44. Функция **Разделить по имени***

Когда вы используете функции разделения или объединения по имени, каждый элемент кластера должен иметь ярлык. Иначе вы не получите доступа к нужному элементу: LabVIEW же не знает, какой элемент вы хотите выбрать!

Например, если вам захочется заменить значение **Boolean2** в последнем упражнении, можете воспользоваться функцией **Объединить по имени**, не беспокоясь о порядке элементов кластера или размере функции.

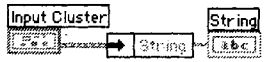


Рис. 7.46

Точно так же при необходимости получить доступ к значению **String** следует воспользоваться функцией **Разделить по имени**.

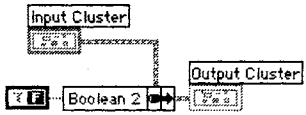


Рис. 7.45

Как только вы подсоедините входной кластер к функции **Объединить по имени** или **Разделить по имени**, имя первого элемента кластера появится в окне ввода или вывода функции. Чтобы получить доступ к другому элементу, щелкните инструментом управления («палец») или ввода текста (A) по вводу или выводу. Вы увидите перечень имен всех элементов кластера, имеющих ярлыки. Выберите нужный элемент из списка, и его имя появится в терминале. Вы также можете получить доступ к этому перечню, щелкнув правой кнопкой мыши по имени и указав функцию **Выбрать объект** (Select Item).

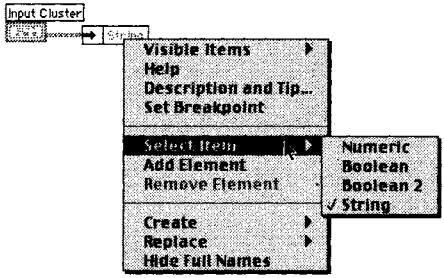


Рис. 7.47

Разрешается изменить размеры обеих функций, чтобы они отображали столько элементов, сколько вам необходимо. После этого можете выбрать каждый компонент для индивидуального доступа. Не нужно беспокоиться о том, что программа перестанет работать при изменении элементов кластера: функции объединения и разделения по имени не теряют работоспособности, пока вы не удалите элемент, на который они ссылаются.

7.20. Упражнение 7.5: еще раз о кластерах

В этом упражнении вы создадите ВП, который проверяет, является значение числового элемента управления **Число1** во входном кластере большим или равным нулю. Если оно меньше 0, то ВП вычислит абсолютные величины всех элементов управления. Если значение **Число1** больше или равно 0, то ВП оставит значения всех элементов управления без изменения. Независимо от значения **Число1** виртуальный прибор умножает все величины на 0,5 и отображает результаты в выходном кластере, демонстрируя использование полиморфизма при работе с кластерами.

1. Откройте новую лицевую панель и поместите на нее шаблон **Кластер** (палитра **Массив и кластер**). Назовите его **Входной кластер**.
2. Создайте элементы управления **Число1**, **Число2** и **Ползун** из палитры **Числовые**, сразу помещая их внутри шаблона кластера. Создавать их

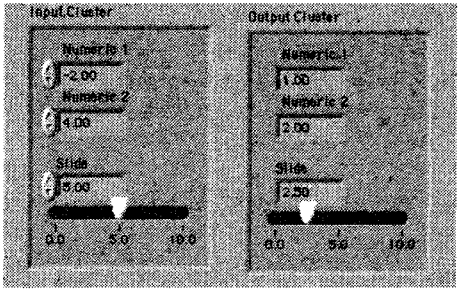


Рис. 7.48

нужно в определенном порядке (поскольку вам придется соединять входной кластер с выходным) и сразу присваивать им имена.

3. Создайте таким же образом шаблон **Выходной кластер**, используя элементы отображения (все элементы должны быть введены в кластер в том же порядке). Вы можете создать этот кластер с помощью копирования входного кластера с последующим изменением его ярлыка.
4. Постройте блок-диаграмму, как показано на рис. 7.49. В ней должны быть задействованы оба варианта ЛОЖЬ и ИСТИНА структуры варианта.

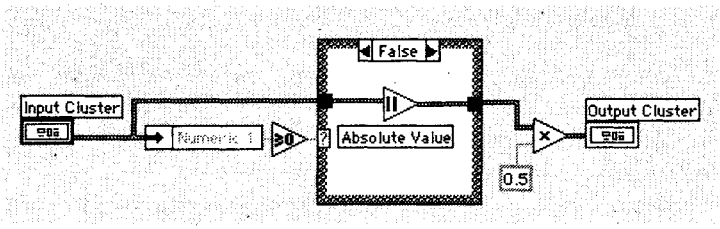


Рис. 7.49

Функция **Разделить по имени** выделяет значение элемента управления **Число1** из входного кластера, и теперь вы можете сравнить его с нулем. Если **Число1** не появилось на выходе функции, щелкните инструментом управления по этому полю и выберите его из списка элементов кластера.

Функция **Больше или равно 0?** (Greater Or Equal to 0?) возвращает значение ИСТИНА, если входное значение больше или равно нулю.

Функция **Абсолютная величина** (Absolute Value) возвращает входное значение, если оно больше или равно 0, или значение, противоположное входному, если оно меньше 0. В этом упражнении функция берет абсолютную величину значений элементов всего кластера.

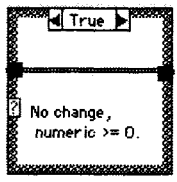


Рис. 7.50

5. Запустите ВП. Попробуйте ввести положительное и отрицательное значение элемента **Число1**. Обратите внимание на использование полиморфизма

для одновременного умножения всех значений в кластере на 0,5 и нахождения абсолютной величины всего кластера.

6. Сохраните ВП как **Cluster Comparison.vi** в директории MYWORK или в библиотеке виртуальных приборов.

7.21. Взаимозаменяемые массивы и кластеры

Иногда удобно поменять массивы на кластеры и наоборот. Этот трюк может оказаться весьма полезным, поскольку LabVIEW включает в себя намного больше функций, работающих с массивами, чем с кластерами. Например, имеется кластер кнопок на лицевой панели, и вы хотите поменять порядок кнопок на противоположный. Функция **Обращение индексов** одномерного массива (Reverse1D Array) выполнила бы эту операцию, но она работает только с массивами. Не беспокойтесь! Вы можете использовать функцию **Кластер в массив** (Cluster to Array) для преобразования кластера в массив. Обратная операция осуществляется с помощью функции **Массив в кластер** (Array to Cluster) – рис. 7.51.

*Рис. 7.51. Функции **Массив в кластер** и **Кластер в массив***

Функция **Кластер в массив** конвертирует кластер с количеством элементов N одного типа данных в массив с количеством элементов N того же типа данных. Индекс массива соответствует порядковому номеру в кластере (то есть нулевой элемент кластера становится значением массива с индексом 0). Нельзя применять эту функцию в кластере, содержащем массивы в качестве элементов, так как LabVIEW не позволяет создавать массивы, состоящие из массивов. Также обратите внимание, что при использовании этой функции все элементы в кластере должны быть одного типа.

Функция **Массив в кластер** преобразует одномерный массив с числом элементов N в кластер с числом элементов N того же типа данных. Для включения этой функции вы должны щелкнуть правой кнопкой мыши по терминалу **Массив в кластер** и выбрать опцию **Размер кластера** (Cluster Size) для установления размера выходного кластера, поскольку кластеры, в отличие от массивов, не устанавливают свой размер автоматически. Размер кластера по умолчанию равен 9. Если ваш массив имеет меньшее количество элементов, чем это определено размером кластера, LabVIEW автоматически создаст дополнительные элементы кластера со значениями по умолчанию для типа данных кластера. Однако, если количество элементов входного массива больше величины, установленной в окне размера кластера, то проводник блок-диаграммы, идущий к выходному кластеру, будет разорванным, пока вы не отрегулируете его размер.

Обе функции удобны в использовании, особенно когда нужно показать элементы кластера на лицевой панели, но при этом необходимо управлять индексами

элементов на блок-диаграмме. Названные функции находятся в подпалитрах **Массив и кластер** палитры **Функции**.

7.22. Итоги

Массив представляет собой набор упорядоченных элементов данных одного типа. В массивах LabVIEW могут быть данные любого типа, за исключением диаграммы, графика или другого массива. Массив создается следующим образом: вначале поместите шаблон массива в окне, затем добавьте в шаблон выбранные элементы управления или отображения.

LabVIEW предлагает большое количество функций для управления массивами, например **Создать массив** и **Выборка из массива**, находящиеся в подпалитре **Массив** палитры **Функции**. В большинстве случаев вам придется использовать эти функции при работе с одномерными массивами; однако указанные функции достаточно гибки и могут применяться при работе с многомерными массивами (иногда сначала потребуется изменить их размер).

Как цикл с фиксированным числом итераций, так и цикл по условию могут аккумулировать массивы на своих границах, при помощи *автоиндексирования*, что очень полезно при создании и обработке массивов данных. Помните, что по умолчанию LabVIEW включает автоиндексирование в циклах с фиксированным числом итераций и отключает в циклах по условию.

Полиморфизм означает способность функции подстраиваться к входным данным различной размерности. Мы говорили о возможностях полиморфизма для арифметических действий, однако и многие другие функции являются полиморфными.

Кластеры также группируют данные, но в отличие от массивов они могут принимать данные различных типов. Вы можете создать их на лицевой панели следующим образом: вначале поместите шаблон кластера на лицевую панель, затем добавьте в шаблон выбранные элементы управления или отображения. Имейте в виду, что объектами внутри кластера должны быть либо элементы управления, либо элементы отображения, но не их комбинация.

Кластеры уменьшают количество проводников или терминалов, ассоциированных с ВП. Например, если на лицевой панели ВП много элементов управления и отображения, которые должны быть впоследствии подключены к терминалам, то намного легче сгруппировать их в кластер и использовать только один ввод.

Функция **Разделить** разделяет кластер на компоненты. Функция **Разделить по имени** работает так же, как и функция **Разделить**, но осуществляет доступ к элементам по их ярлыку. Вы можете получить доступ к любому количеству элементов, используя функцию **Разделить по имени**, а ко всему кластеру – используя функцию **Разделить** (следовательно, вам следует побеспокоиться о количестве вводов и порядке элементов кластера).

Функция **Объединить** соединяет отдельные компоненты в кластер или заменяет элемент в кластере. Функция **Объединить по имени** не может объединять кластеры,

но способна заменять отдельные элементы в кластере без необходимости доступа ко всем элементам. Кроме того, применяя функцию **Объединить по имени**, вам не нужно беспокоиться о порядке элементов кластера или о правильности размера функции **Объединить**. Убедитесь лишь, что при использовании функций **Объединить по имени** и **Разделить по имени** все элементы кластера имеют ярлыки.

7.23. Дополнительные упражнения

7.23.1. Упражнение 7.6: изменение порядка

Создайте ВП, который бы изменял порядок массива, содержащего 100 случайных чисел. Например, элемент 0 становится элементом 99, элемент 1 – элементом 98 и т.д. Назовите виртуальный прибор **Reverse Random Array.vi**.

7.23.2. Упражнение 7.7: извлечение подмассива

Создайте ВП, который генерировал бы массив, содержащий 100 случайных чисел, и показывал часть массива, например от индекса 10 до индекса 50. Назовите виртуальный прибор **Subset Random Array.vi**.

Используйте функцию **Подмножество массива** (палитра **Массив**) для выделения подмножества массива.

7.23.3. Упражнение 7.8: игра в кости

Создайте ВП, имитирующий вращение игральной кости (возможные значения 1–6), и проследите за количеством раз, когда игральная кость показывает то или иное значение. Входными данными, которые можно задавать, являются количество вращений кости, а выходные данные включают в себя (для каждого возможного значения) количество раз, когда игральная кость показывает это значение. Назовите виртуальный прибор **Die Roller.vi**.

Для отслеживания значений от нескольких итераций цикла вам потребуются сдвиговые регистры.

7.23.4. Упражнение 7.9: умножение элементов массива

Создайте ВП, который использовал бы одномерный входной массив, затем перемножал пары элементов (начиная с 0 и 1) и выдавал на выходе результирующий массив. Например, в результате выполнения программы для входного массива с элементами 1, 23, 10, 5, 7, 11 будет получен выходной массив с элементами 23, 50, 77. Назовите виртуальный прибор **Array Pair Multiplier.vi**.

Обзор

Графики и развертки осциллограмм LabVIEW позволяют отобразить данные в графической форме. Развертки строят осциллограммы согласованно, добавляя к старым данным новые по мере их поступления, поэтому вы можете наблюдать текущее значение в контексте предыдущих. Графики строят уже сгенерированные массивы данных, как правило, не сохраняя предыдущих. В этой главе вы изучите графики и развертки осциллограмм, различные методы их использования и некоторые особенности. Также вы познакомитесь с графиками и развертками интенсивности, трехмерными графиками и графиками цифровых осциллограмм. Наконец, вы узнаете о типе данных «осциллограмма», который очень полезен для представления временных данных в LabVIEW.

ЗАДАЧИ

- Понять, как использовать графики и развертки
- Уметь различать три режима развертки: панорамный, временная развертка и временная развертка с маркером
- Изучить механическое действие логических выключателей
- Понять функциональное различие разверток и графиков осциллограмм
- Научиться улучшать внешний вид разверток и графиков, изменяя их масштаб и используя палитру управления, панели редактирования и курсоров
- Познакомиться с графиками и развертками интенсивности, трехмерными графиками, необходимыми для построения трехмерных данных
- Узнать о графиках цифровых осциллограмм, служащих для отображения цифровых данных
- Изучить тип данных «осциллограмма»: из каких компонентов он состоит и как его использовать

ОСНОВНЫЕ ТЕРМИНЫ

- График
- Развертка осциллограммы (Waveform chart)
- Панель редактирования графика (Plot legend)
- Панель редактирования шкалы (Scale legend)
- Палитра элементов управления графиком (Graph palette)
- Курсор
- График осциллограммы (Waveform graph)
- Двухкоординатный график (XY Graph)
- Графики и развертки интенсивности
- Трехмерные графики
- Графики цифровых осциллограмм
- Тип данных «осциллограмма»
- t_0, dt, Y

СРЕДСТВА ВИЗУАЛЬНОГО ОТОБРАЖЕНИЯ LabVIEW: РАЗВЕРТКИ И ГРАФИКИ ОСЦИЛЛОГРАММ

8

8.1. Развертки осциллограмм

Кривая на графике является графическим отображением зависимости величины Y от величины X. Часто величина Y представляет значения данных, тогда как величина X представляет время. **Развертка осциллограммы** (Waveform chart), расположенная в подпалитре **График** палитры **Элементы управления**, является особым числовым элементом отображения, который может показать в графическом виде одну или несколько кривых. Наиболее часто развертки осциллограммы используются внутри циклов. В них сохраняются и отображаются на постоянно обновляющемся дисплее данные, которые были получены ранее, а также новые данные по мере их поступления. В развертке осциллограммы величина Y представляет собой новые данные, а X – время (чаще всего каждое значение Y создается во время

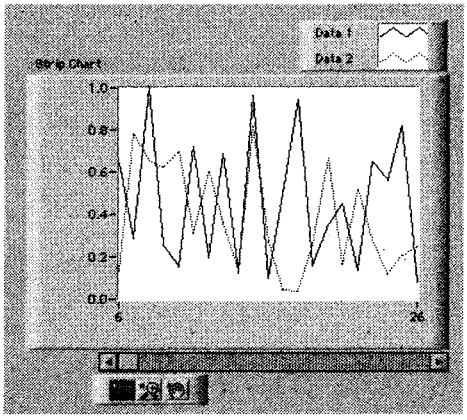


Рис. 8.1

итерации цикла; таким образом, значение X представляет собой время выполнения одного цикла). LabVIEW имеет только один вид развертки осциллограммы, но у развертки есть три различных режима обновления для показа интерактивных данных. На рис. 8.1 представлен пример многолучевой развертки осциллограммы.

8.1.1. Режимы обновления развертки осциллограммы

Развертка осциллограммы имеет три режима обновления: *панорамная развертка* (strip chart mode), *временная развертка* (scope chart mode) и *временная развертка с маркером* (sweep chart mode), как показано на рис. 8.2. Режим обновления можно сменить, щелкнув правой кнопкой мыши по развертке осциллограммы и выбрав одну из опций в меню **Дополнительно ⇒ Режим обновления** (Advanced ⇒ Update Mode). Если вы хотите сменить режим во время работы ВП (когда содержание контекстного меню немного изменяется), выберите опцию **Режим обновления** в контекстном меню осциллограммы.

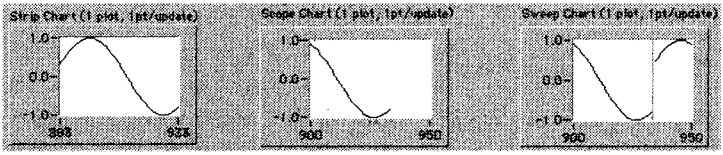


Рис. 8.2

При работе развертки в панорамном режиме у осциллограммы появляется прокручивающееся изображение подобно бумажной ленточной диаграмме. Осциллограммы в режимах временной развертки и временной развертки с маркером имеют изображения с обратным ходом, как в осциллографе. В режиме временной развертки кривая, достигнув правой границы, стирается и вновь начинает прорисовываться с левой границы. Режим временной развертки с маркером работает аналогичным образом, но изображение не исчезает при достижении правой границы: начало поступления новых данных отмечает движущаяся вертикальная линия – маркер. Она перемещается через изображение по мере поступления новых данных. Различия между режимами легче понять, когда вы увидите их в действии. В следующем упражнении вы немного поэкспериментируете с ними.

Построение осциллограмм в режимах временной развертки и временной развертки с маркером происходит значительно быстрее, чем в панорамном режиме, поскольку при перерисовке графика загрузка ресурсов компьютера меньше.

8.1.2. Однолучевая развертка осциллограммы

Наиболее простым способом использования развертки осциллограммы является подключение скалярной величины к терминалу развертки на блок-диаграмме (рис. 8.3). На каждой итерации цикла рисуется еще одна точка на развертке осциллограммы.

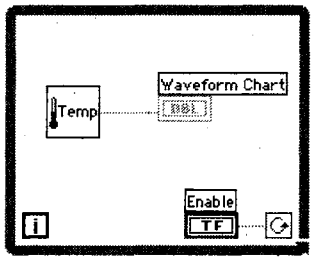


Рис. 8.3

8.1.3. Создание многолучевой развертки осциллограммы

Развертки осциллограмм могут содержать более чем один луч. Однако, поскольку вы не можете соединить несколько источников на блок-диаграмме с одним терминалом графика, то вначале следует объединить данные, используя функцию **Объединить**. На рис. 8.4 функция **Объединить** объединяет выходы трех различных ВП, измеряющих температуру, в кластер с последующим отображением значений на развертке осциллограммы. Обратите внимание на изменение внешнего вида терминала развертки осциллограммы во время соединения его с функцией **Объединить**. Для того чтобы построить больше графиков, просто увеличьте количество входных терминалов функции **Объединить** путем изменения ее размера инструментом перемещения («стрелка»).

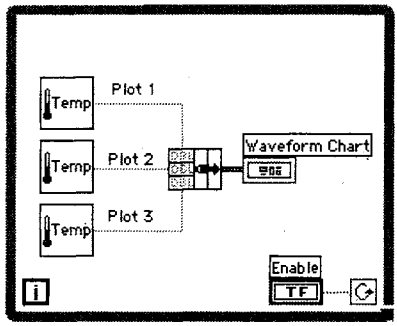


Рис. 8.4

Для ознакомления с разверткой осциллограммы, ее режимоми работы и типами входных данных откройте и запустите ВП **Charts.vi** из библиотеки CH8.LLB директории EVERYONE.

8.1.4. Цифровой дисплей развертки осциллограммы

Как и у многих других числовых элементов отображения, у развертки осциллограммы есть опция управления цифровым дисплеем (вызовите контекстное меню развертки и выберите опцию **Видимые элементы ⇒ Цифровой дисплей**). Цифровой дисплей (digital display) показывает только что поступившие значения, изображаемые на развертке.

8.1.5. Полоса прокрутки

Развертки также имеют полосу прокрутки (scroll bar), которую можно сделать видимой или спрятать. Полоса прокрутки используется для демонстрации устаревших данных, изображение которых уже вышло за пределы графического индикатора.

8.1.6. Очистка содержимого графического индикатора

Иногда необходимо удалить все ранее полученные данные с графического индикатора развертки осциллограммы. Для этого выберите опцию **Операции с данными ⇒ Очистить развертку** (Data Operations ⇒ Clear Chart) в контекстном меню графика, когда прибор находится в режиме редактирования. (Помните, что, если ВП не запущен, то обычный режим работы программы – это режим редактирования. В таком случае для переключения режимов выберите опцию **Перейти в режим запуск/редактирование** (Change to Run/Edit Mode) в меню **Управление** (Operate). Если ВП находится в режиме выполнения, то опция **Очистить развертку** находится в контекстном меню, а не на вкладке **Операции с данными**.

8.1.7. Отдельные и совмещенные кривые графиков

Если у вас развертка осциллограммы с множеством кривых, допустимо выбрать показывать все кривые относительно общей оси Y (совмещенный режим) или для каждого графика использовать собственную ось Y (режим отдельных графиков). Для выбора типа отображения служат опции **Набор графиков** (Stack Plots) или **Совмещенные графики** (Overlay Plots) в контекстном меню графика. Рис. 8.5 иллюстрирует различие между отдельными и совмещенными графиками.

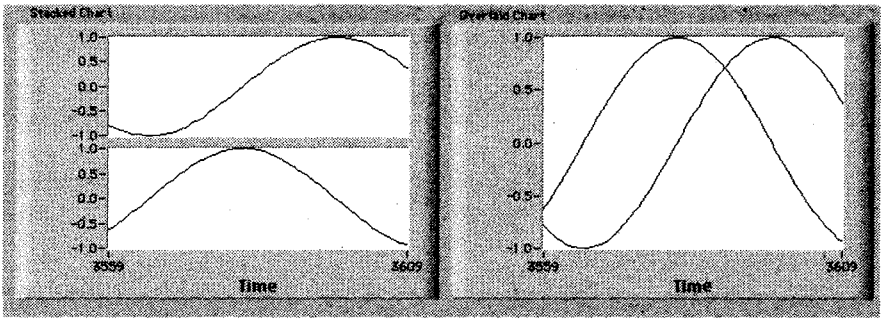


Рис. 8.5

8.1.8. Несколько шкал X и Y

Если вы хотите отобразить несколько кривых на одном индикаторе в совмещенном режиме, требуется сделать график, который имел бы различные шкалы для каждого луча. Если диапазон изменения координаты Y одного луча от 0,0 до 1,0, а другого от –1000 до +1000, то будет затруднительно увидеть их оба без отдельной шкалы для каждого. Вы можете создать отдельные шкалы для оси X и для Y, щелкнув правой кнопкой мыши по соответствующей оси и выбрав опцию **Шкала Y/X ⇒ Двойная шкала** (Y/X Scale Duplicate Scale). Рис. 8.6 иллюстрирует две шкалы Y для графика с несколькими кривыми.

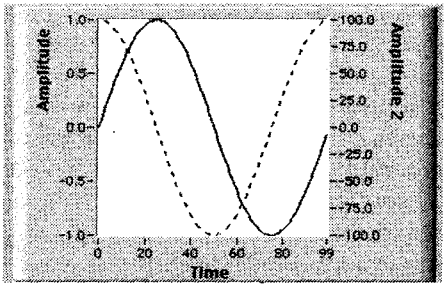


Рис. 8.6

8.1.9. Длина графика

По умолчанию развертка осциллограммы сохраняет до 1024 точек. Если нужно сохранить большее или меньшее количество данных, выберите опцию **Длина истории графика** (Chart History Length) в контекстном меню развертки и установите новое значение величиной до 100000 точек. Изменение размера буфера не влияет на количество данных, изображаемых на графическом индикаторе. Для визуализации большего или меньшего количества данных на индикаторе просто измените его размер. Однако увеличение размера буфера увеличивает количество данных, которые вы можете посмотреть с помощью прокрутки.

8.2. Упражнение 8.1: слежение за температурой

В этом примере вы рассмотрите создание виртуального прибора для измерения температуры и визуализации результатов на развертке осциллограммы. В приборе будет использоваться ВПП **Thermometer**, который вы создали на одном из предыдущих занятий.

1. Откройте новую лицевую панель. Вам придется вновь создать панель, изображенную на рис. 8.7.
2. Поместите в окне лицевой панели вертикальный переключатель (палитра **Логические**). Назовите его **Включить**. Этот переключатель будет использоваться для остановки сбора данных температуры.

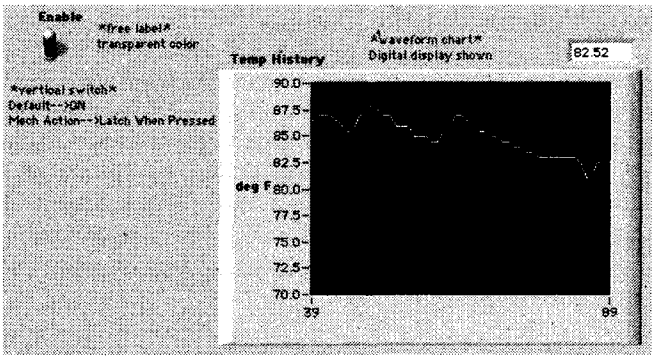


Рис. 8.7

3. Поместите развертку осциллограммы (палитра **График**) в окне панели. Назовите развертку осциллограммы **Динамика изменения температуры**. Развертка осциллограммы будет показывать значения температуры в реальном времени.
4. Развертка осциллограммы имеет цифровой дисплей, который отображает последнее измеренное значение. Щелкните правой кнопкой мыши по развертке осциллограммы и выберите опцию **Видимые элементы** ⇒ **Цифровой дисплей** в контекстном меню.
5. Поскольку сенсорное устройство измеряет комнатную температуру (в единицах шкалы Фаренгейта), вам следует изменить масштаб развертки осциллограммы, чтобы увидеть величину температуры (в противном случае ее значения окажутся за пределами графика). Используя инструмент ввода текста (A), дважды щелкните мышью по отметке «10.0» вертикальной шкалы осциллограммы и напечатайте `90`. После этого щелкните мышью за пределами области текста. Щелчок вводит значение. Можно также нажать кнопку **Ввод** на панели инструментов, чтобы ввести изменения в масштаб. Аналогичным образом измените значение «0.0» на `70`.
6. Откройте окно блок-диаграммы и постройте диаграмму, как показано на рис. 8.8.

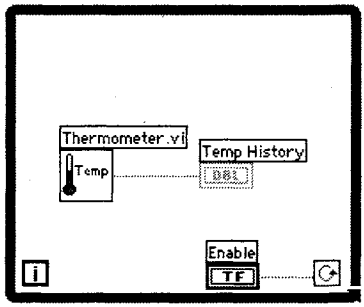


Рис. 8.8

7. Поместите цикл по условию (палитра **Структуры** – Structures) на блок-диаграмму и установите необходимый размер цикла.
8. Поместите два терминала внутрь цикла по условию, если они оказались вне цикла.

Thermometer.vi

9. Импортируйте прибор **Thermometer**.
   Виртуальный прибор **Thermometer.vi** возвращает измеренное значение температуры с соответствующего датчика (или имитирует – в зависимости от вашей установки). Этот прибор у вас уже создан в упражнениях главы 4 и модифицирован в главе 5. Загрузите его с помощью кнопки **Выбрать ВП** (Select A VI) палитры **Функции**. Скорее всего, он находится в директории `MYWORK`. Если же у вас его нет, пользуйтесь ВП **Thermometer.vi** из библиотеки `CH5.LLB` или **Digital Thermometer.vi**, расположенным в подпалитре **Учебник** (Tutorial) палитры **Функции**.
10. Соедините элементы на блок-диаграмме, как показано на рис. 8.8.
11. Вернитесь к лицевой панели и включите вертикальный переключатель, щелкнув по нему инструментом управления. Запустите ВП.
    Помните, что цикл по условию является неопределенной структурой. Поддиаграмма в пределах его границ будет выполняться до тех пор, пока на терминал условия выхода поступает значение ИСТИНА. В этом примере ВПП **Thermometer.vi** будет возвращать новое значение измеренной температуры и показывать его на развертке осциллограммы до тех пор, пока переключатель находится в положении «включить» (ИСТИНА).
12. Для прекращения работы прибора щелкните мышью по вертикальному переключателю, подав таким образом логическое значение ЛОЖЬ на терминал условия выхода из цикла, и цикл завершит выполнение.
13. Развертка осциллограммы имеет буфер, в котором содержится некоторое количество точек, исчезнувших из поля зрения на дисплее. Вы можете снова увидеть эти данные, вызвав контекстное меню развертки осциллограммы и выбрав опцию **Видимые Элементы ⇒ Полоса прокрутки**. Для регулировки размера полосы прокрутки и ее местоположения используйте инструмент перемещения.
    Для прокручивания развертки осциллограммы щелкните по любой стрелке в полосе прокрутки.
    Для очистки буфера дисплея и установки в исходное состояние развертки осциллограммы вызовите ее контекстное меню и выберите опцию **Операции с данными ⇒ Очистить развертку**. Если вы хотите удалить график, находясь в режиме выполнения программы, выберите опцию **Очистить развертку** в контекстном меню работающей развертки осциллограммы.
14. Убедитесь, что переключатель находится в состоянии ИСТИНА, и вновь запустите ВП. На этот раз попытайтесь изменить режим обновления графика. Щелкните правой кнопкой мыши и выберите опцию **Режим обновления ⇒ Временная развертка** в контекстном меню работающей развертки осциллограммы. Обратите внимание на различие прорисовки графика. Теперь укажите опцию **Временная развертка с маркером** и посмотрите, что произойдет.

Использование различных механических режимов работы логических переключателей

Давайте немного отвлечемся от выполненного упражнения. Вы наверняка обратили внимание на то, что всякий раз при запуске этого виртуального прибора требовалось включить вертикальный переключатель **Включить**, прежде чем нажать кнопку запуска, – в противном случае цикл выполнится только один раз. Вы можете изменить механическое действие логического элемента управления для изменения его поведения и устранить это неудобство. LabVIEW предлагает шесть возможных вариантов механического действия логического элемента управления.

При работе элемента управления в режиме **Включить при нажатии** (Switch When Pressed) он изменяет свое значение всякий раз, когда вы щелкаете по нему инструментом управления. Это действие принято по умолчанию для логических элементов управления и похоже на действие управляющих кнопок с подсвечиванием обычных физических приборов. На него не влияет частота, с которой виртуальный прибор считывает полученные значения.

При работе элемента управления в режиме **Включить при отпускании** (Switch When Released) он изменяет свое значение только тогда, когда вы отпускаете кнопку мыши после щелчка по нему. На элемент, работающий в этом режиме, также не влияет частота, с которой виртуальный прибор считывает его значения. Данный режим аналогичен тому, когда вы щелкаете мышью по кнопке выбора варианта в диалоговом окне. Кнопка выделяется, изменения не вносятся до тех пор, пока вы не отпустите кнопку мыши.

При работе элемента управления в режиме **Включить до отпускания** (Switch Until Released) он изменяет свое значение, когда вы щелкаете по нему мышью. Элемент сохраняет новое значение до тех пор, пока вы не отпустите кнопку мыши. Затем элемент управления возвращается к своему первоначальному значению. Этот режим похож на работу дверного звонка. На элемент, работающий в таком режиме, также не влияет частота, с которой виртуальный прибор считывает его значения.

При работе элемента управления в режиме **Сработать при нажатии** (Latch When Pressed) он изменяет свое значение, когда вы щелкнете по нему мышью, и сохраняет новое значение до тех пор, пока виртуальный прибор не считает его один раз. Это происходит независимо от того, держите вы кнопку мыши нажатой или отпустили ее. Вариант **Сработать при нажатии** используется тогда, когда ваш виртуальный прибор должен выполнить какое-либо действие только один раз – например, остановить цикл по условию, когда вы нажимаете кнопку **Стоп**.

При работе элемента управления в режиме **Сработать при отпускании** (Latch When Released) он изменяет свое значение после того, как вы отпустите кнопку мыши. Как только ваш виртуальный прибор прочитает это значение один раз, элемент управления возвращается в прежнее

состояние. Такая операция гарантирует, по крайней мере, одно новое значение. Данный режим аналогичен поведению кнопок в диалоговом окне: кнопка подсвечивается, когда вы щелкнете по ней мышью, и фиксирует выбор в момент отпускания кнопки мыши.

При работе элемента управления в режиме **Сработать до отпускания** (Latch Until Released) он изменяет свое значение, когда вы щелкнете по нему мышью, и сохраняет новое значение до тех пор, пока ваш виртуальный прибор не прочитает его один раз или пока вы не отпустите кнопку мыши.

Рассмотрим, например, вертикальный переключатель – его значением по умолчанию является ЛОЖЬ.

15. Модифицируйте вертикальный переключатель в вашем ВП таким образом, чтобы не возникало необходимости переключать его в значение ИСТИНА всякий раз, когда вы запускаете виртуальный прибор:
    a) включите вертикальный переключатель (значение ИСТИНА);
    b) щелкните правой кнопкой мыши по переключателю и выберите опцию **Операции с данными ⇒ Назначить текущее значение по умолчанию** (Make Current Value Default) в контекстном меню, чтобы перевести значение по умолчанию в положение «включено» (ИСТИНА);
    c) вызовите контекстное меню переключателя и выберите опцию **Механическое действие ⇒ Сработать при нажатии** (Mechanical Action ⇒ Latch When Pressed).
16. Запустите ВП. Щелкните мышью по вертикальному переключателю для остановки получения данных. Переключатель на короткое время перейдет в положение «выключено», а затем, после того как терминал условия выхода из цикла считает его значение ЛОЖЬ, автоматически вернется в исходное положение «включено».

Если вы изменяете значения объектов при помощи локальных переменных, то не сможете использовать различные режимы механического действия логических элементов управления. Мы поговорим о причинах этого, когда будем изучать локальные переменные в главе 12.

Изменение временных параметров выполнения ВП

При запуске виртуального прибора цикл по условию выполняется очень быстро. Если вам нужно получать данные через определенные промежутки времени: секунды, минуты и т.д. – воспользуйтесь функцией **Задержка до следующего кратного интервала мс** (в меню **Время и диалоги**).

17. Преобразуйте ВП для измерения температуры со скоростью одно измерение в 0,5 с, помещая показанный на рис. 8.9 код в цикл по условию.

Рис. 8.9

Функция **Задержка до следующего кратного интервала мс** обеспечивает задержку выполнения каждой итерации на определенное время (в данном случае на 0,5 с).

18. Запустите ВП несколько раз, используя различные значения миллисекунд.
19. Закройте и сохраните ВП. Назовите его **Temperature Monitor.vi** и поместите в директорию MYWORK или в библиотеку виртуальных приборов.

8.3. Графики осциллограмм

В отличие от разверток, которые рисуют данные интерактивно, графики сразу визуализируют сформированные массивы данных. Они не обладают способностью добавлять новые значения к уже созданным. LabVIEW предлагает несколько видов графиков для обеспечения больших возможностей работы: *графики осциллограмм, двухкоординатные графики, графики интенсивности, трехмерные графики, графики цифровых осциллограмм и некоторые особые виды графиков* (*кривые Смита, графики в полярных координатах, кривые максимумов-минимумов и кривые распределения*). В этой книге мы будем подробно говорить о графиках осциллограмм и двухкоординатных (XY) графиках. На лицевой панели ВП графики осциллограмм и графики XY выглядят идентично, но имеют совершенно разные функции.

На рис. 8.10 изображен график с несколькими графическими инструментами.

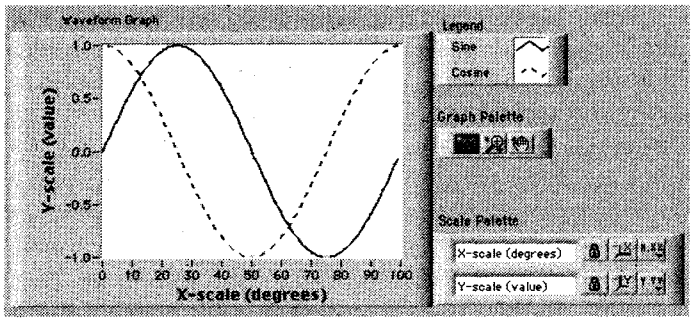


Рис. 8.10

Оба типа графических индикаторов вы можете вызвать из подпалитры **График** палитры **Элементы управления**. График осциллограммы рисует только однозначные функции (одно значение Y соответствует определенному значению X) с однородно расположенными точками, такие как считанные через постоянные интервалы времени значения сигнала переменной амплитуды из канала платы ввода/вывода. График осциллограммы является идеальным инструментом отображения массивов данных, в которых точки распределены равномерно.

График XY представляет собой декартовый график, используемый для отображения массивов данных с изменяющимися временными интервалами или данных с несколькими значениями координаты Y для каждого значения X, например график окружности. Оба типа графиков выглядят одинаково, но имеют различные типы входных данных, поэтому соблюдайте осторожность и не перепутайте их.

8.3.1. Однолучевая осциллограмма

Для построения однолучевых осциллограмм вы можете подключить массив значений Y непосредственно к терминалу графика осциллограммы (рис. 8.11). Этот метод основан на предположении, что начальное значение по оси X равно 0 и что значение ΔX (то есть приращение координаты X) равно 1. Обратите внимание, что терминал графика на блок-диаграмме появляется в виде элемента отображения массива.

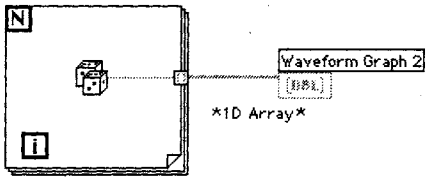


Рис 8.11

Иногда нужно изменить временные параметры графика. Например, вы начали собирать данные во время, отличное от начального X = 0 (или $X_0 = 0$), или расстояние между выборками не равно стандартному ΔX = 1. Для изменения временных параметров объедините значения X_0, ΔX и массив данных в кластер, а затем соедините кластер с графиком. На рис. 8.12 показано, что терминал графика в этом случае превращается в индикатор кластера.

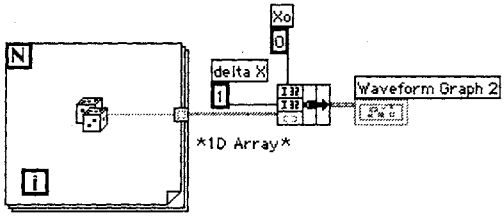


Рис 8.12

8.3.2. Многолучевая осциллограмма

Для построения на графике осциллограммы нескольких кривых необходимо создать массив (или двумерный массив) данных, используемых в примерах с однолучевой осциллограммой. Обратите внимание, что терминалы графиков меняют

свой внешний вид в зависимости от структуры данных, подключенных к ним (массив, кластер, массив кластеров и т.д.), и типа данных (I16, DBL и т.п.).

На рис. 8.13 предполагается, что начальное значение X равно 0 и значение ΔX равно 1 для обоих массивов. Функция **Создать массив** создает двумерный массив из двух одномерных массивов. Обратите внимание, что этот двумерный массив имеет две строки и 100 столбцов – массив 2×100. По умолчанию графики отображают каждую строку в виде отдельной осциллограммы. Если ваши данные организованы столбцом, транспонируйте массив перед тем, как его построить. Транспонирование означает замену индексов столбцов на индексы строк и наоборот. Например, если вы транспонируете массив с тремя строками и десятью столбцами, то получите массив с десятью строками и тремя столбцами. LabVIEW облегчает эту процедуру – просто вызовите контекстное меню графика и выберите опцию **Транспонировать массив** (Transpose Array) – эта опция меню имеет серый цвет, если к ней не подключен двумерный массив. Вы также можете использовать функцию **Транспонировать 2D-массив** (Transpose 2D Array), находящуюся в подпалитре **Массив** палитры **Функции**.

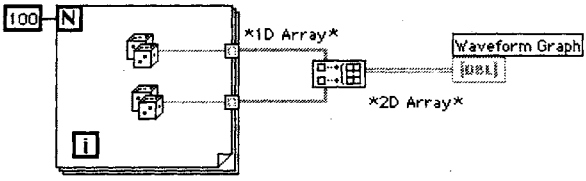


Рис. 8.13

На рис. 8.14 показан случай, когда значения X_0 и ΔX специально задаются. Эти параметры оси X не обязательно должны быть одинаковыми для обоих наборов данных.

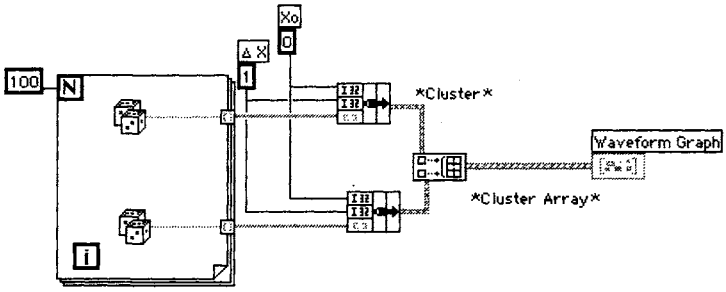


Рис. 8.14

На представленной блок-диаграмме функция **Создать массив** создает массив из двух входных кластеров. Каждый входной кластер состоит из массива и двух скалярных чисел. В результате получился массив кластеров, который подается на

терминал графика. В данном случае определенные значения X_0 и ΔX совпадают со значениями по умолчанию, но вы можете использовать любые другие значения.

8.4. Упражнение 8.2: построение синусоиды на графике осциллограммы

В этом упражнении вы создадите ВП, который будет генерировать массив данных синусоидальной кривой и вычерчивать его на графике осциллограммы.

1. Откройте новый ВП и создайте лицевую панель, как показано на рис. 8.15.

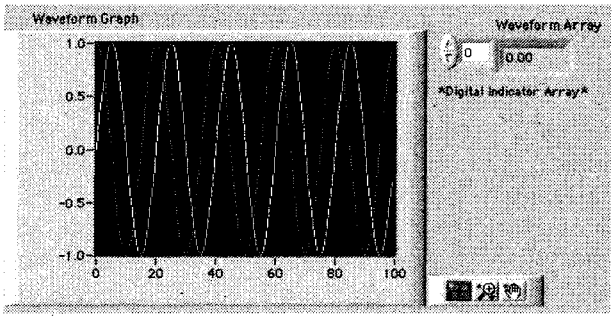


Рис. 8.15

2. Поместите шаблон массива в окне лицевой панели. Назовите его **Массив осциллограммы**. Поместите числовой индикатор внутрь окна данных массива для демонстрации его содержимого.
3. Поместите график осциллограммы в окне лицевой панели. Назовите его **График осциллограммы** и увеличьте его размер путем перемещения угла инструментом перемещения.
   Спрячьте панель редактирования графика, вызвав его контекстное меню и выбрав опцию **Видимые элементы ⇒ Панель редактирования графика** (Plot Legend).
   Отключите автоматическую установку масштаба, щелкнув правой кнопкой мыши и выбрав опцию **Шкала Y ⇒ Автомасштабирование Y**. Измените пределы отображения оси Y путем их выделения с помощью инструмента ввода текста (A) и указания новых чисел: задайте минимальное значение оси Y равным –1.0, максимальное – равным 1.0.
4. Постройте блок-диаграмму, как показано на рис. 8.16.


Sine Function

   Функция **Синус** (Sine) палитры **Числовые ⇒ Тригонометрические** (Trigonometric) вычисляет sin(x) и возвращает одну точку синусоиды. Для этой функции необходимо входное скалярное число в радианах. В данном упражнении вход *x* изменяется с каждой итерацией цикла, и результатом является синусоида.

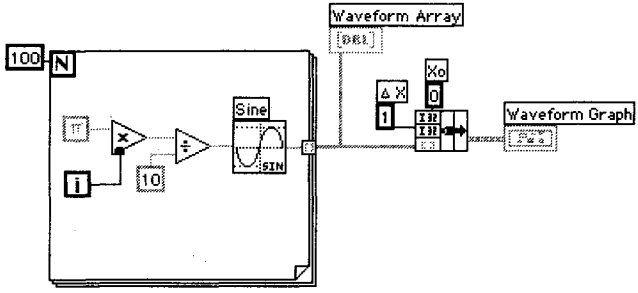


Рис. 8.16

Константа **Pi** палитры **Числовые ⇒ Дополнительные числовые константы** (Additional Numeric Constant) является значением числа π.

Функция **Объединить** объединяет компоненты луча в кластер. Компоненты включают в себя начальное значение X (равное 0), значение ΔX (1) и массив Y (данные осциллограммы). Измените размер функции с помощью инструмента перемещения, передвинув один из ее углов.

В каждом повторении цикла с фиксированным числом итераций создается одна точка графика и сохраняется на границе цикла в массиве осциллограммы. После завершения выполнения цикла функция **Объединить** объединяет начальное значение X, значение ΔX и массив для последующего построения на графике.

5. Вернитесь к лицевой панели и запустите виртуальный прибор. В данный момент график будет состоять из одного луча.
6. Измените значение ΔX на 0,5 и начальное значение X на 20, а затем снова запустите ВП. Обратите внимание, что график теперь показывает те же 100 точек с начальным значением 20 и значением ΔX равным 0,5 (показано на оси X).
7. Инструментом перемещения в виде сетки увеличьте размер индикатора массива, чтобы он отображал несколько элементов. Индекс этих элементов меняется – возрастает слева направо (или сверху вниз), начиная с индекса, показанного на индикаторе индексов массива (рис. 8.17). Вы можете увидеть любой элемент в массиве путем введения индекса этого элемента в индикатор индексов. Если вы укажете число, превышающее размер массива, то изображение элемента будет серым.

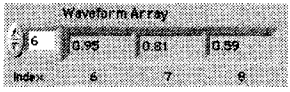


Рис. 8.17

На блок-диаграмме, изображенной на рис. 8.16, задайте для осциллограммы начальное значение X и ΔX. Зачастую X_0 равно 0, а значение ΔX равно 1. В этих случаях вы можете подключить массив осциллограммы прямо к терминалу графика, как это показано на рис. 8.18, и воспользоваться значениями по умолчанию для ΔX и X_0.

8. Вернитесь к окну блок-диаграммы. Удалите функцию **Объединить** и подключенные к ней константы, затем выберите функцию **Удалить неисправные проводники** (Remove Broken Wires) в меню **Правка** (Edit) или нажмите соответствующую клавишную комбинацию. Завершите подключение элементов блок-диаграммы (рис. 8.18).

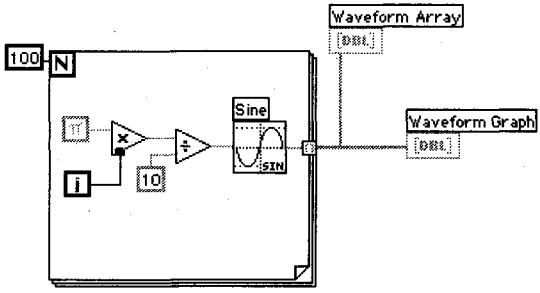


Рис. 8.18

9. Запустите виртуальный прибор. Обратите внимание, что он вычерчивает осциллограмму с начальным значением X = 0 и значением ΔX = 1.

Многолучевые осциллограммы

Многолучевые осциллограммы можно создать путем подключения данных, обычно поступающих на однолучевую осциллограмму, вначале к функции **Создать массив**, а затем к терминалу графика. Хотя нельзя создать массив массивов, допустимо создать двумерный массив, в котором каждый входной массив будет представлять собой строку.

1. Создайте блок-диаграмму, как показано на рис. 8.19.

Функция **Создать массив** создает надлежащую структуру данных для построения двух массивов на графике осциллограммы. С помощью инструмента перемещения увеличьте размер функции **Создать массив**,

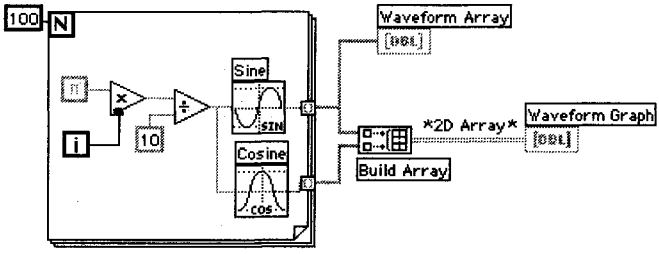


Рис. 8.19

чтобы у нее было два входа. Убедитесь, что для функции **Создать массив** вы *не выбрали* опцию **Объединить входы** (которая в контекстном меню принята по умолчанию), чтобы на выходе получить двумерный массив.

Cosine Function

Функция **Косинус** (Cosine) вычисляет cos(x) и возвращает одну точку косинусоиды. На вход этой функции необходимо подать скалярную величину в радианах. В данном упражнении входная величина x изменяется с каждой итерацией цикла, а результатом является синусоидальная кривая.

2. Переключитесь на лицевую панель и запустите ВП. Обратите внимание, что обе кривые рисуются на одном и том же графическом индикаторе. Для обоих массивов данных по умолчанию приняты одинаковые начальные значения X (0) и приращения координаты ΔX (1).
3. Посмотрите, что произойдет, если выбрать в контекстном меню графика опцию **Транспонировать массив**. При перемене строк на столбцы график меняется радикально! Вновь выберите опцию **Транспонировать массив** для возвращения в прежнее состояние.
4. Закройте и сохраните ВП под именем **Graph Sine Array.vi** в директории MYWORK или в библиотеке виртуальных приборов.

8.5. Двухкоординатные графики

Графики, используемые вами в предыдущих главах, создаются путем построения осциллограмм, точки которых получены через регулярные интервалы времени. Однако, если вы получаете данные через нерегулярные интервалы или хотите построить математическую функцию, имеющую несколько значений Y для одного значения X, то вам необходимо задать точки графика, используя их координаты (X,Y). *Двухкоординатные* (XY Graphs) графики строят подобные кривые. На их вход необходимо подавать типы данных, отличные от тех, которые поступали на графики осциллограмм. Однолучевая двухкоординатная осциллограмма (XY) и соответствующий терминал на блок-диаграмме показаны на рис. 8.20 и 8.21.

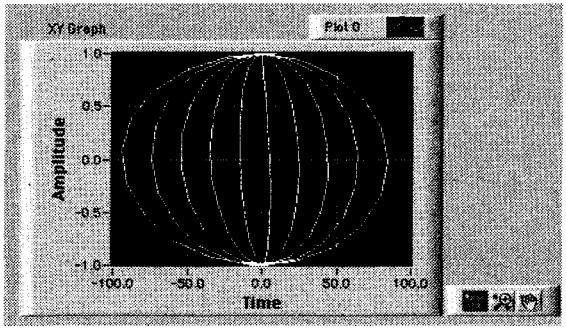


Рис. 8.20

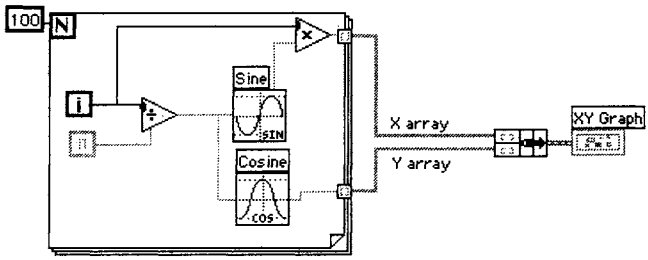


Рис. 8.21

На вход двухкоординатного графика подаются массивы X (верхний вход) и Y (нижний вход), объединенные в кластер. Функция **Объединить** объединяет массивы X и Y в кластер, подключенный к двухкоординатному графику. Терминал двухкоординатного графика выглядит теперь как элемент отображения кластера.

Для получения многолучевых двухкоординатных графиков следует создать массив кластеров значений X и Y, используемых для одиночных графиков, как это показано на рис. 8.21.

Очень легко перепутать функции **Объединить** и **Создать массив**, когда вы формируете данные для построения на графике. Так что уделите этому вопросу особое внимание.

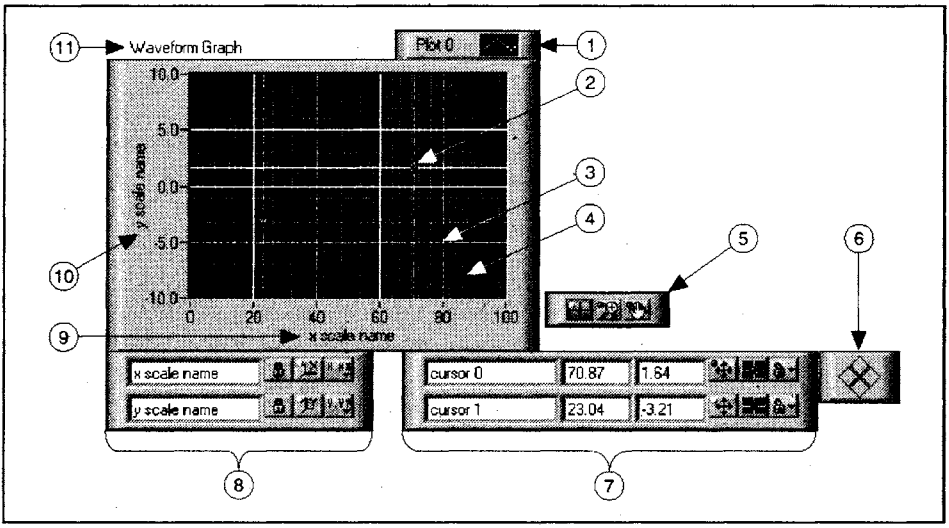
Чтобы поучиться на примерах, взгляните на виртуальные приборы **Waveform Graph.vi** и **XY Graph.vi** в библиотеке `CH8.LLB` директории `EVERYONE`.

8.6. Компоненты разверток и графиков осциллограмм

Графики и развертки осциллограмм обладают замечательными свойствами, которые вы можете использовать для улучшения отображения данных. Ниже рассказывается о настройке этих опций.

8.6.1. Работа с масштабами

Развертки и графики осциллограмм могут автоматически перестраивать свои шкалы по вертикали и горизонтали, чтобы отобразить точки кривых. То есть масштабы осей сами настраиваются для отображения всех точек на графике с максимально возможной разрешающей способностью. Вы можете включать и выключать эту функцию, используя опции **Автомасштабирование X** (AutoScale X) и **Автомасштабирование Y** (AutoScale Y) на вкладках **Шкала X** (X Scale) или **Шкала Y** (Y Scale) контекстного меню объекта. Допустимо также управлять автоматической настройкой масштабов с помощью панели редактирования шкалы (Scale Legend). Режим автоматической настройки шкалы в LabVIEW по умолчанию



1	Plot legend	4	Mini-grid mark	7	Cursor legend	10	Y-scale
2	Cursor (graph only)	5	Graph palette	8	Scale legend	11	Label
3	Grid mark	6	Cursor mover	9	X-scale		

Рис. 8.22

включен для графиков и отключен для разверток осциллограмм. Применение режима автоматической настройки шкал может замедлить обновление развертки или графика осциллограмм в зависимости от используемого вами компьютера и монитора, так как каждый график требует расчета новых масштабов.

Если вы не хотите использовать режим автоматической настройки масштабов, попробуйте изменять масштаб по горизонтали и вертикали непосредственно – с помощью инструментов управления («палец») или ввода текста (A) для указания нового числа, как это делается со всеми элементами управления или отображения в LabVIEW, и выключить режим автоматической настройки шкал.

Меню шкал X и Y

Каждая шкала X и Y имеет подменю опций, как показано на рис. 8.23.

Используйте опцию **Автомасштабирование** для включения режима автоматической настройки шкал.

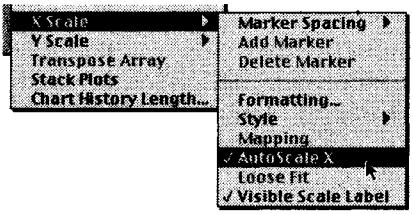


Рис. 8.23

Когда вы активизируете эту опцию, шкалы точно устанавливаются для отображения полного диапазона изменения данных. Задействуйте опцию **Скользящая подгонка** (Loose Fit), если хотите, чтобы LabVIEW округлил масштаб. При включении этой опции числа округляются до множителя инкремента, используемого для шкалы. Например, если метки шкалы кратны пяти, то минимальный и максимальный пределы шкалы будут установлены как множители 5, а не в точный диапазон изменения данных.

Опция **Задание формата** (Formatting) выводит диалоговое окно, показанное на рис. 8.24, которое позволяет настроить следующие параметры:

- опция **Выбор шкалы** (Select Scale) – выпадающее меню, позволяющее выбрать шкалу, которую вы изменяете, по имени. На рис. 8.24 можно видеть, что шкала X названа «Time»;
- меню **Тип шкалы** (Scale Style) предлагает указать главные и второстепенные маркерные отметки для шкалы или вообще не задавать никаких отметок. Щелкните мышью по иконке, чтобы сделать выбор. Главные маркерные отметки соответствуют числам шкалы, а второстепенные обозначают внутренние точки между числами. Это меню позволяет скрыть или отобразить маркеры данной шкалы;
- опция **Режим отображения** (Mapping Mode) дает возможность выбрать либо линейную, либо логарифмическую шкалу для отображения данных;
- с помощью меню **Опции координатной сетки** (Grid Options) можно сделать видимыми или скрыть линии сетки координат, линии сетки вдоль основных маркерных отметок или основных и второстепенных маркерных отметок. Также разрешается изменить цвет линий сетки координат. Щелкните мышью по кнопкам сетки, чтобы увидеть выпадающее меню доступных опций.

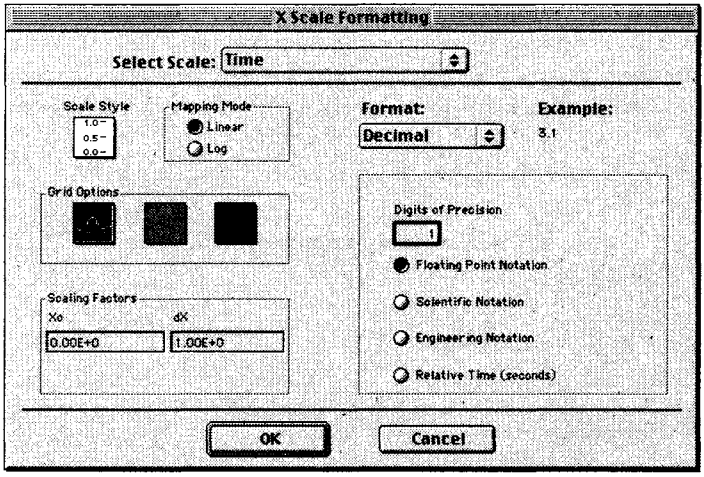


Рис. 8.24

С помощью $\mathbf{X}_0$ из раздела **Масштабные коэффициенты** (Scalling Factors) вы можете установить значение X, с которого хотите начать построение графика, и **dX** – инкремента значений X (такое же, как ΔX).

В разделе **Формат и точность** (Format and Precision) допустимо выбрать формат **Числовой** (Numeric) или **Дата и время** (Time&Date). Если вы задали формат **Числовой**, то можете указать точность отображения (число точек после запятой) и систему обозначений отображения шкалы: **Плавающая запятая** (Floating Point), **Научная** (Scientific), **Техническая** (Engineering) или **Относительное время** (Relative Time). Определяют числа в следующих форматах: **Десятичный** (Decimal), **Десятичный без знака** (Unsigned Decimal), **Шестнадцатеричный** (Hexadecimal), **Восьмеричный** (Octal) или **Двоичный** (Binary). Если вы остановились на формате **Дата и время**, то можете выбрать тип отображения времени и даты.

Панель редактирования шкалы

Панель редактирования шкалы (scale legend) дает возможность создать ярлыки для шкал X и Y (или для множества двухкоординатных шкал, если их у вас больше, чем одна) и посредством контекстного меню осуществлять легкий доступ к их конфигурациям. На панели редактирования шкалы присутствуют кнопки для масштабирования осей X и Y, изменения формата их отображения и автоматической настройки шкал.

Вызвав контекстное меню графика или развертки осциллограмм и выбрав опции **Видимые элементы ⇒ Панель редактирования шкалы**, вы создадите окно, показанное на рис. 8.25.

В текстовом окне панели редактирования можно напечатать имена шкал. Этот текст появляется в окне графика или развертки вдоль осей X или Y.

Пощелкайте мышью по кнопкам панели для настройки таких же опций, какие были только что описаны, на вкладке контекстного меню **Задание формата шкалы X|Y** (X|Y Scale Formatting). Для переключения между ручной и автоматической настройками каждой шкалы, управления видимыми параметрами шкал и т.д. в режиме инструмента управления щелкните мышью по кнопке **Блокировка шкалы** (Scale Lock).

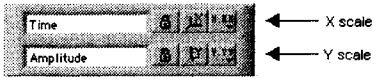


Рис. 8.25

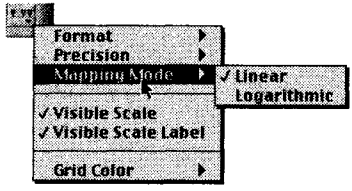


Рис. 8.26

Для создания, визуализации или сокрытия дополнительных шкал графика или развертки необходимо изменить размеры панели редактирования шкал с помощью инструмента перемещения, как и в случае массивов.

8.6.2. Панель редактирования графика

До тех пор пока вы не используете собственный стиль для диаграмм и разверток осциллограмм, в них применяется стиль по умолчанию. *Панель редактирования графика* (plot legend) дает возможность озаглавить, окрасить, выбрать тип линии и точек для каждого луча. Сделать видимой или спрятать панель редактирования можно посредством опции **Видимые элементы ⇒ Панель редактирования графика** в контекстном меню графика или развертки. Также в панели редактирования можно ввести имя для каждого луча. Пример панели редактирования графика показан на рис. 8.27.

Рис. 8.27

При выборе опции **Панель редактирования графика** в окне панели появляется только один луч. Сделать видимым большее количество лучей можно перемещением угла панели редактирования. После введения характеристик луча в панели редактирования графика эта настройка сохраняется вне зависимости от того, является панель редактирования видимой или нет. Если график или развертка содержат большее количество лучей, чем определено в панели редактирования, то LabVIEW вычерчивает дополнительные лучи в стиле по умолчанию.

При перемещении всего графика или развертки панель редактирования перемещается вместе с ними. Вы можете изменить местоположение панели редактирования относительно графика путем перемещения одной панели редактирования в новое место. *Чтобы увеличить место для названия луча, растяните панель редактирования влево, а для увеличения размера образца графика – вправо.*

По умолчанию каждый луч имеет номер, начинающийся с 0. Вы можете изменить эту надпись аналогично тому, как это делалось для других ярлыков LabVIEW – просто начните печатать в режиме инструмента ввода текста (A). У каждого луча есть собственное меню для изменения стилей, линий, цвета и типа точек. Вы можете получить доступ в это меню, щелкнув инструментом управления по образцу графика в панели редактирования.

- опция **Шаблоны графиков** (Common Plots) позволяет использовать для графика любой из шести распространенных стилей, включая график рассеяния, график в виде полос и заливку от нулевого уровня. Опции в этой подпалитре одновременно настраивают типы точек, стиль линии

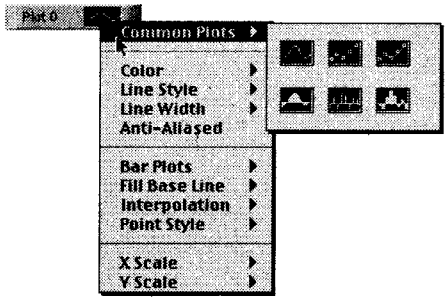


Рис. 8.28

и тип заливки графика (вместо индивидуальной настройки, как будет рассказано ниже);

- опция **Цвет** (Color) отображает цветовую палитру, что дает возможность выбрать цвет луча. Допустимо изменить цвет графиков в панели редактирования с помощью инструмента раскрашивания. Цвет лучей можно изменить и во время выполнения виртуального прибора;
- опции **Тип линии** (Line Style) и **Толщина линии** (Line Width) предлагают различные типы и размеры вычерчивающих линий;
- опция **Сглаживание** (Anti-Aliased) делает линии графиков более гладкими, но имейте в виду, что эта операция требует повышенных вычислительных мощностей и ухудшает быстродействие;
- опция **Диаграммы** (Bar Plots) дает возможность создать графики в виде полос со 100-, 75- или 1-процентной шириной в горизонтальном или вертикальном направлениях. Опция **Заполнение** (Fill Baseline) управляет базисной линией полос. Графики могут не иметь заполнения или быть заполненными до 0, $+\infty$, $-\infty$;
- опция **Интерполяция** (Interpolation) определяет способ вычерчивания линии между точками данных. В первом режиме не вычерчиваются никакие линии, что удобно для графика рассеяния (другими словами, вы получаете только точки). Режим в нижнем левом углу вычерчивает прямую линию между точками. Четыре ступенчатых режима соединяют точки с прямоугольным коленом, удобным для гистограмм;

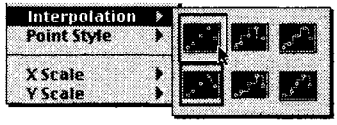


Рис. 8.29

- опция **Тип точки** (Point Style) отображает различные типы точек (отсутствие точек, круглые, квадратные, пустые, заполненные и т.п.), из которых вы можете выбрать нужный;
- опции **Шкала X** и **Шкала Y** необходимы для определения, какой луч соответствует шкале X или Y, когда на диаграмме или развертке имеется большое количество шкал.

8.6.3. Упражнение 8.3: использование двухкоординатного графика для построения окружности

В этом упражнении вы создадите ВП, который вычерчивает окружность на основе данных зависимых массивов X и Y.

1. Откройте новую лицевую панель. Воссоздайте панель, изображенную на рис. 8.30.
2. Разместите **График XY** (палитра **График**) в окне панели. Назовите график **График окружности**.
3. Увеличьте размер графика путем растягивания его угла инструментом перемещения. Попытайтесь сделать область графика приблизительно квадратной.

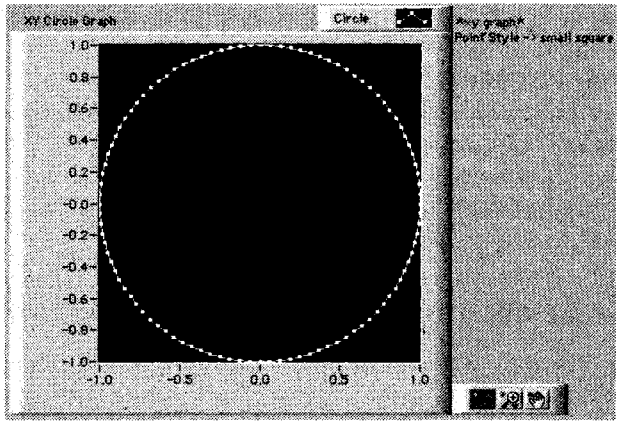


Рис. 8.30

Чтобы удержать пропорции графика, вы можете перетаскивать по диагонали его угол, одновременно нажимая клавишу <shift>.

4. Щелкните правой кнопкой мыши по графику и выберите опцию **Видимые элементы ⇒ Панель редактирования графика**. Измените размер панели редактирования с левой стороны и введите ярлык `Окружность`, используя инструмент ввода текста. Щелкните правой кнопкой мыши по линии в панели редактирования и выберите маленький квадрат в палитре **Тип точки**. Затем задайте новый цвет луча в палитре **Цвет**.
5. Постройте блок-диаграмму, как показано на рис. 8.31.

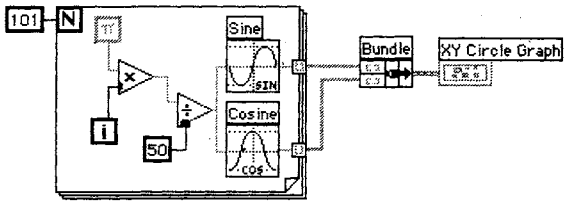


Рис. 8.31

Sine Function

Cosine Function

Bundle

Функции **Синус** (Sine) и **Косинус** (Cosine) вычисляют синус и косинус входного числа. Используйте эти функции в цикле с фиксированным числом итераций для создания массива точек, представляющего один цикл синусоиды и косинусоиды.

Функция **Объединить** объединяет массив синуса (значения X) и массив косинуса (значения Y) для построения массива синуса относительно массива косинуса.

Константа **Pi** служит для преобразования входных чисел функций в радианы.

Используя функцию **Объединить**, вы можете построить массив синуса относительно массива косинуса на двухкоординатном графике, который отображает окружность.

6. Вернитесь к лицевой панели и запустите виртуальный прибор. Сохраните его как **Graph Circle.vi** в директории MYWORK или в библиотеке виртуальных приборов.

8.6.4. Использование палитры элементов управления графиком

Палитра элементов управления графиком (graph palette) представляет собой небольшое окно, которое появляется справа внизу развертки или графика осциллограммы. С помощью палитры элементов управления графиком вы можете плавно прокрутить изображение и сфокусироваться на определенной области, используя кнопки палитры (увеличение). Также допустимо перемещать курсор. Палитра, доступ к которой можно получить через опцию **Видимые элементы** контекстного меню развертки или графика, показана на рис. 8.32.

Рис. 8.32

Появившиеся три кнопки позволяют управлять режимом работы графика. Обычно график находится в стандартном режиме, что дает возможность щелкать мышью по курсорам графика для их пространственного перемещения. Если вы нажмете кнопку прокрутки, то переключитесь в режим прокрутки данных путем перетаскивания частей графика при помощи курсора прокрутки. Если вы щелкнете мышью по кнопке увеличения (zoom), то получите контекстное меню, которое позволит выбрать методы увеличения изображения.

Вот несколько примеров работы:

- **Увеличить прямоугольную область** (Zoom by Rectangle). Переместите курсор, чтобы охватить прямоугольником область, которую нужно увеличить. Когда вы отпустите кнопку мыши, оси изменят масштаб, чтобы показать лишь выбранную область;

- **Увеличить прямоугольную область вдоль X** (Zoom by Rectangle in X) обеспечивает увеличение вдоль оси X (шкала Y остается неизменной);

- **Увеличить прямоугольную область вдоль Y** (Zoom by Rectangle in Y) обеспечивает увеличение вдоль оси Y (шкала X остается неизменной);

- опция **Отменить последнее увеличение** (Undo Last Zoom) является очевидной: она возвращает прежний масштаб;

- **Увеличить около точки** (Zoom in about a Point). Если вы удерживаете кнопку мыши в нажатом положении на определенной точке, то график будет постоянно увеличиваться, пока вы не отпустите кнопку;

- **Уменьшить около точки** (Zoom out about a Point). Если вы удерживаете кнопку мыши в нажатом положении на определенной точке, то график будет постоянно уменьшаться, пока вы не отпустите кнопку.

Последние два режима – **Увеличить** около точки и **Уменьшить около точки** – взаимно обратимы: чтобы перейти из одного режима в другой, просто удерживайте нажатой клавишу <shift>.

8.6.5. Курсоры графика

Графики осциллограмм LabVIEW имеют *курсоры*, чтобы отмечать точки данных для ускорения работы с ними. На рис. 8.33 показан график с курсорами и панелью редактирования курсора (Cursor Legend). На развертках осциллограмм курсоров вы не обнаружите.

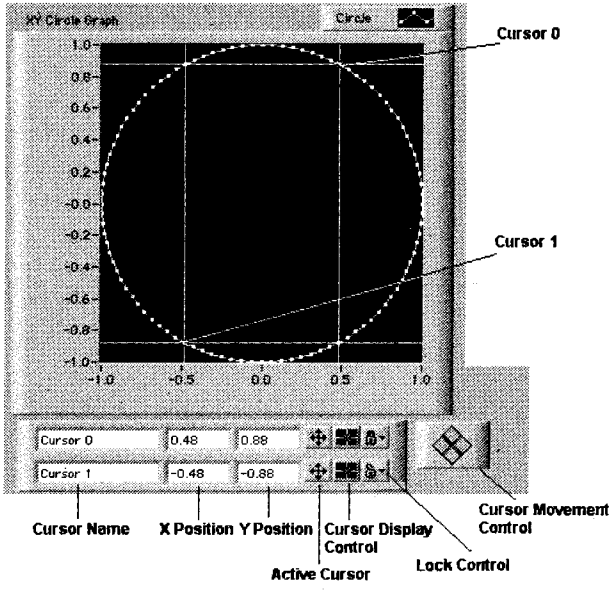


Рис. 8.33

Палитру курсора можно сделать видимой путем выбора опции **Видимые элементы ⇒ Панель редактирования курсора** в контекстном меню графика. Когда панель появляется в первый раз, она имеет серый цвет. Щелкните мышью по кнопке **Включить курсор** (Active Cursor), чтобы активизировать курсор (или введите текст в одно из полей **Название курсора** (Cursor Name) с помощью инструмента ввода текста). Вы можете перемещать курсор вручную или программно, используя узлы свойств (см. главу 12).

Для передвижения курсора вручную применяйте инструмент управления («палец»). Перемещать точку пересечения допустимо во всех направлениях, а горизонтальную или вертикальную линии курсора – лишь в горизонтальном или вертикальном направлениях соответственно.

Вы также можете использовать набор кнопок **Управление перемещением курсора** (Cursor Movement Control) для перемещения курсора вверх, вниз, влево и вправо.

График может иметь столько курсоров, сколько вы хотите. Панель редактирования курсора помогает отслеживать их, и вы можете раздвинуть ее, чтобы отобразить большее количество курсоров.

Для чего нужна панель редактирования курсора? Вы можете ввести название курсора в первом окне слева на панели редактирования. Следующее окно показывает координату по оси X и далее по оси Y. Щелкните мышью по кнопке с изображением перекрестия **Включить курсор** для активизации курсора на этой линии и управления его перемещением (зеленый цвет соответствует активному курсору). Когда вы щелкнете по кнопке консоли **Управление перемещением курсора**, все активные курсоры начнут двигаться.

Щелкните инструментом управления (но не правой кнопкой мыши) по кнопке **Управление отображением курсора** (Cursor Display Control) – следующая кнопка с изображением перекрестия – для установки типа курсора, типа точки и цвета.

Кнопка с изображением замка «привязывает» курсор к графику, что ограничивает его перемещение только одной кривой. Щелкните инструментом управления по этой кнопке, чтобы войти в меню данной опции (не щелкайте правой кнопкой мыши, иначе вы попадете в другое меню).

Для того чтобы убрать курсор, щелкните правой кнопкой мыши по панели редактирования курсора и выберите опцию **Операции с данными ⇒ Удалить элемент** (Delete Element).

8.7. Упражнение 8.4: анализ данных температуры

Создайте виртуальный прибор, который измеряет температуру приблизительно через каждые 0,25 с в течение 10 с. Во время получения данных ВП демонстрирует результаты измерений в реальном времени на развертке осциллограммы. После завершения сбора данных ВП вычерчивает данные на графике и вычисляет минимальное, максимальное и среднее значение температуры.

1. Откройте новую лицевую панель и создайте виртуальный прибор, как показано на рис. 8.34.
2. Измените масштаб графика для получения диапазона от 70,0 до 90,0. Убедитесь, что режим автоматической настройки шкал включен для обеих осей графика.
3. Используя инструмент ввода текста, впечатайте слово `Темп` на панели редактирования графика. А теперь щелкните правой кнопкой мыши (или инструментом управления) по образцу графика **Темп** на панели редактирования и измените тип точки (опция **Тип точки**) до маленьких квадратиков. Вы также можете придать лучам нужный цвет.

   Развертка **Температура** отображает температуру по мере получения данных. После завершения сбора информации о температуре ВП вычерчивает данные на **Графике температуры**. Цифровые элементы отображения

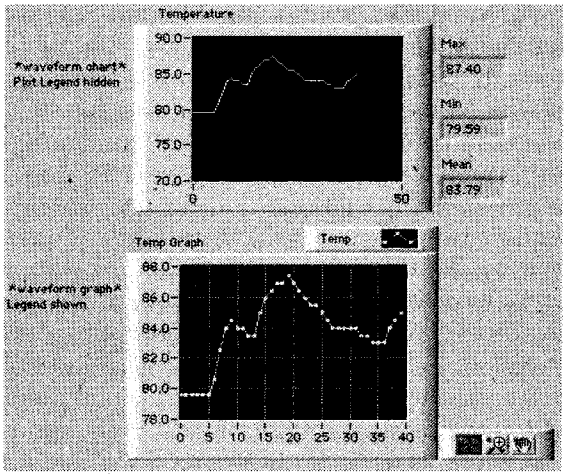


Рис. 8.34

Среднее, **Max** и **Min** покажут соответственно среднее, максимальное и минимальное значение температуры.

4. Постройте блок-диаграмму, как показано на рис. 8.35. Обязательно обратитесь к окну контекстной помощи для отображения вводов и выводов этих функций и смотрите на концы проводников, иначе вы можете ошибочно подключиться к ненужному терминалу.

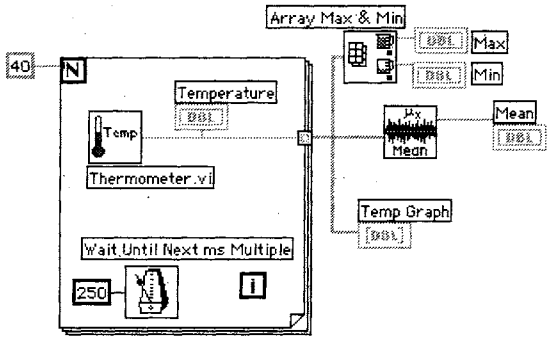


Рис. 8.35

Используйте палитру **Выбрать ВП** для того, чтобы получить доступ к ВПП **Thermometer.vi**. По всей вероятности, он находится в директории `MYWORK`. Если у вас его нет, воспользуйтесь функциями **Thermometer.vi** в `CH5.LLB` или **Digital Thermometer.vi**, которые находятся в подпалитре **Учебник** палитры **Функции**. ВПП **Thermometer** возвращает одно значение температуры всякий раз, когда он вызывается.

Функция **Задержка до следующего кратного интервала мс** заставляет цикл с фиксированным числом итераций выполняться каждые 0,25 с. Функция **Массив Max&Min** возвращает максимальное и минимальное значения массива. В данном случае она возвращает максимальное и минимальное значения температуры, измеренной во время сбора данных. Функция **Mean.vi** палитры **Математика ⇒ Вероятность и статистика** (Mathematics ⇒ Probability and Statistics) возвращает среднее значение температуры.

5. Цикл с фиксированным числом итераций выполняется 40 раз. Функция **Задержка до следующего кратного интервала мс** заставляет каждую итерацию выполняться приблизительно каждые 0,25 с. Виртуальный прибор сохраняет результаты измерений температуры в массиве, созданном на границе цикла с фиксированным числом итераций, используя автоиндексирование. После завершения цикла массив переходит к различным узлам данных. Функция **Массив Max&Min** возвращает максимальное и минимальное значения температуры, функция **Mean.vi** – среднее значение.
6. Вернитесь к лицевой панели и запустите виртуальный прибор.
7. С помощью панели редактирования шкалы (сделайте ее видимой, используя опцию **Видимые элементы ⇒ Панель редактирования шкалы** из контекстного меню графика) измените точность значений шкалы Y так, чтобы график отображал числа с тремя знаками после запятой.

8. Используя палитру элементов управления графиком, увеличьте его, щелкнув по кнопке **Увеличение** (Zoom) и выбрав какой-либо режим.
9. Вызовите контекстное меню графика и отметьте опцию **Видимые элементы ⇒ Панель редактирования курсора**. Вначале курсор будет серого цвета; щелкните по кнопке **Включить курсор** в верхней строке, чтобы активизировать первый курсор.

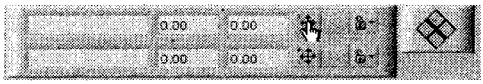


Рис.. 8.36

Используйте инструмент управления для перемещения курсора по графику; обратите внимание на то, как изменяются значения X и Y на дисплее курсора. Эти значения помогут вам определить координаты любой точки на графике. А теперь с помощью кнопок управления курсором подвигайте курсор в разных направлениях. Активизируйте второй курсор с помощью тех же кнопок и перемещайте оба курсора одновременно. Щелкнете инструментом управления по кнопке **Управление отображением курсора** (если вы щелкнете правой кнопкой мыши, то войдете в другое меню) и измените цвет одного из курсоров.

Теперь вы видите, что могут делать курсоры? При желании допустимо выделить текст `Cursor 0` и заменить его другим. Наконец, щелкните инструментом управления по кнопке блокировки («замок») и уберите

выделение опции **Разрешить перетаскивание** (Allow Drag) из появившегося меню. Затем выберите опцию **Прикрепить к графику** (Lock to Plot) из того же меню. Теперь вы не сможете перемещать курсор в любых направлениях на графике с помощью мыши. Попробуйте использовать кнопки управления движением курсора и вы увидите, что он будет следовать графику, к которому прикреплен.

10. Закройте и сохраните ВП. Назовите его **Analysis.vi** и поместите в директорию MYWORK или в библиотеку виртуальных приборов.

8.8. Развертки и графики интенсивности – цвет как третье измерение

Что вы будете делать, если понадобится построить три переменные друг относительно друга? Если вы работаете в Windows, можете воспользоваться функцией ActiveX 3D Graph Controls, которую мы будем обсуждать ниже.

Во всех версиях LabVIEW вы можете построить график интенсивности. *Развертки* и *графики интенсивности* (Intensity charts и graphs) отображают данные в трех измерениях на плоскости путем использования цвета как третьего измерения данных (значения по оси Z). Так же как развертка осциллограммы, развертка интенсивности имеет полосу прокрутки, в то время как график интенсивности фиксирован. Развертки интенсивности являются необходимым инструментом для отображения таких данных, как топографические или температурные карты, где цвет представляет высоту или распределение температуры.

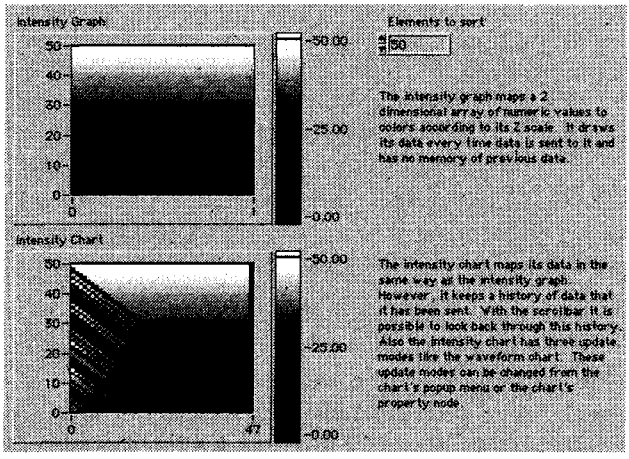


Рис. 8.37

Графики и развертки интенсивности действуют по аналогии с двумерными развертками и графиками, где цвет представляет третью переменную. С помощью опции **Цветовая шкала** (Color Scale) вы можете установить и отобразить

карту-схему цветов. Курсор графика интенсивности дополнительно показывает значение Z.

На вход разверток и графиков интенсивности необходимо подавать двумерные массивы чисел, где каждое число в массиве является значением цвета. Индексы каждого элемента массива представляют местоположение этого цвета на графике. Вы не только сможете определить соответствие чисел и различных цветов при помощи шкалы цвета, но и осуществлять это программно, используя узлы свойств (об этом вы узнаете в главе 12). Простой пример на рис. 8.38 изображает массив 3×4, построенный на графике интенсивности. Отображаемые цвета – красный (1.0), коричневый (2.0) и зеленый (3.0).

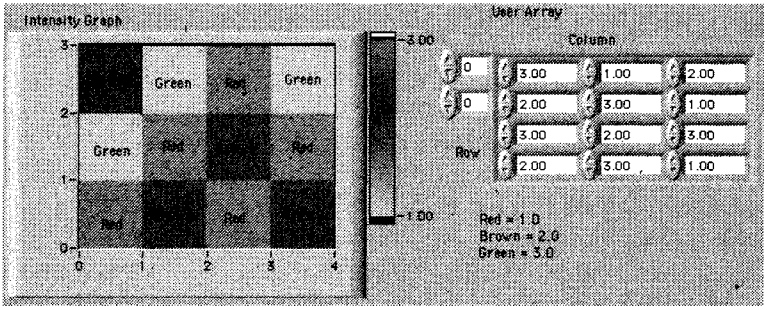


Рис. 8.38

Чтобы назначить цвет значению на цветовой шкале, щелкните правой кнопкой мыши по соответствующему маркеру и выберите опцию **Цвет маркера** (Marker Color). Шкала Z разверток и графиков интенсивности по умолчанию находится в режиме произвольного расположения маркера (Arbitrary Marker Spacing), поэтому вы можете изменить «градиент» окрасок, перемещая маркер инструментом управления. Создайте новые маркеры, выбрав функцию **Добавить маркер** (Add Marker) из контекстного меню цветовой шкалы, затем перенесите их в любое удобное место и назначьте им новый цвет.

8.8.1. Упражнение 8.5: график интенсивности

В этом упражнении вы увидите ВП, который отображает картину интерференции волн. Данный ВП также показывает, каким образом график интенсивности отображает входной двумерный массив на дисплее.

1. Для того чтобы понять, как работают графики и развертки интенсивности, откройте и запустите ВП **Intensity Graph Example VI**, находящийся в директории `EVERYONE\CH8.LLB`. Вы увидите на графике сложную интерференционную картинку. Цветовой диапазон определяется на блок-диаграмме с помощью узла свойств графика интенсивности. Измените цветовой диапазон, щелкнув инструментом управления по

окнам управления цветом в первом кадре структуры последовательности (Sequence Structure) и выбрав новый цвет в появившейся палитре. Вновь запустите виртуальный прибор.

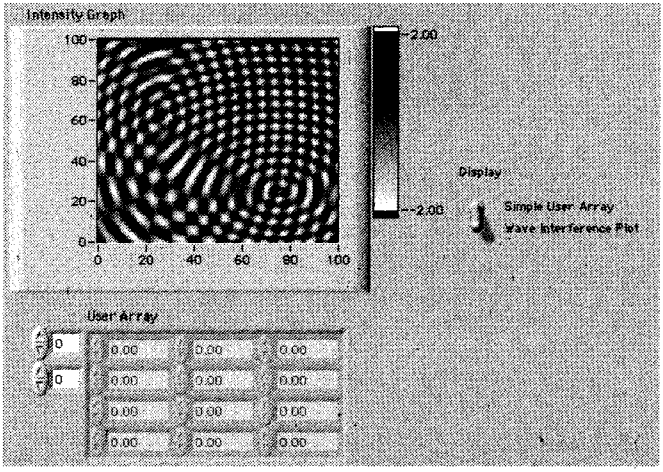


Рис. 8.39

2. Переведите переключатель **Дисплей** (Display) в положение **Simple User Array** и введите значения между 0.0 и 10.0 в элемент управления **User Array** (диапазон цветов для графика был установлен на блок-диаграмме с использованием узлов свойств графика – от голубого (0.0) до красного (10.0)). После ввода всех значений запустите ВП. Обратите внимание, как значение каждого элемента располагается на графике. Измените значения и вновь запустите виртуальный прибор.
3. Взгляните на блок-диаграмму, чтобы увидеть, как работает виртуальный прибор.
4. Закройте виртуальный прибор без сохранения каких-либо изменений. Для более подробного ознакомления с графиками интенсивности откройте ВП, поставляемые с LabVIEW и находящиеся в библиотеке `example\general\graphs\intgraphs.llb`. Выясните также назначение приборов **Simulation of Tomography** и **Heat Equation Example** из библиотеки `examples\analysis\mathxmpl.llb`.

8.8.2. Трехмерные графики

При создании трехмерных графиков LabVIEW предлагает для профессионального варианта Windows несколько функций: **3D-график поверхности** (3D Surface Graph), **3D-параметрический график** (3D Parametric Graph), **3D-график кривой** (3D Curve Graph) из палитры **График**.

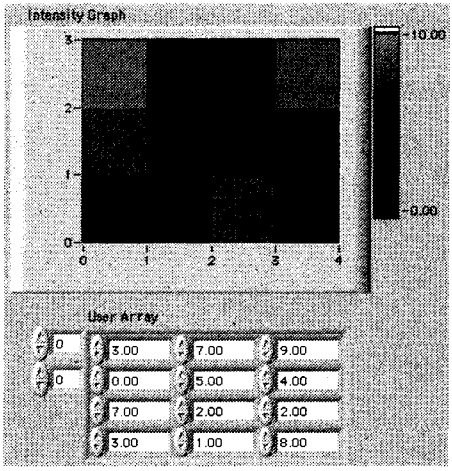


Рис. 8.40

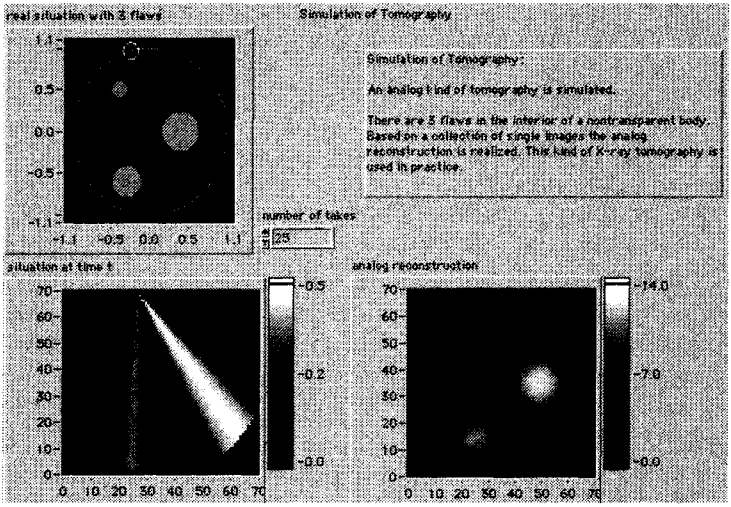


Рис. 8.41

Эти функции построения 3D-графиков работают только под Windows и не представлены в базовом комплекте LabVIEW.

На рис. 8.42 и 8.43 показаны лицевая панель и блок-диаграмма ВП, который представляет пример тора, построенного при помощи функции **3D-параметрический график**. В отличие от ранее рассмотренных разверток и графиков трехмерные графики не имеют простого терминала на блок-диаграмме. Они используют

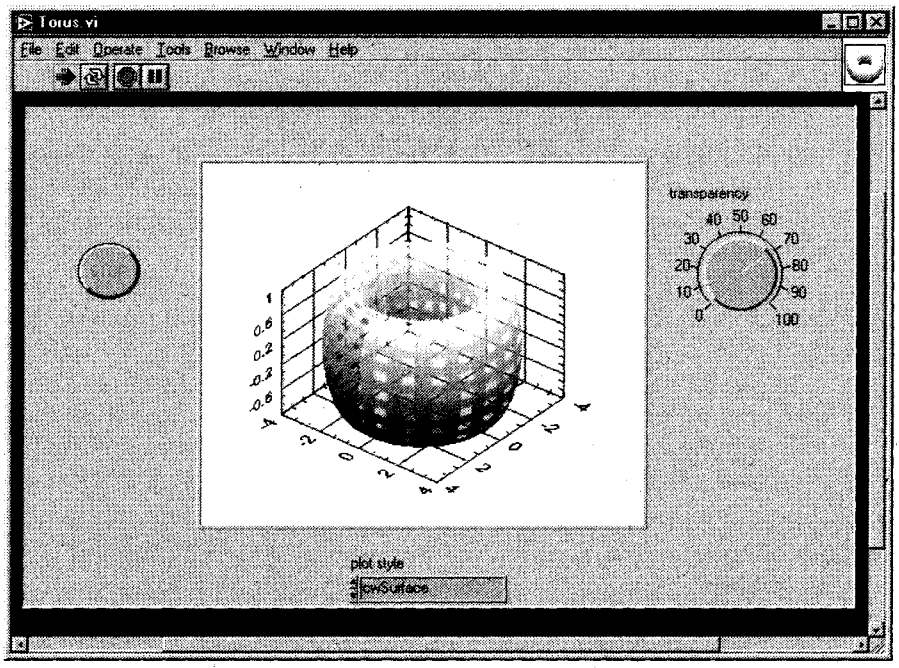


Рис. 8.42

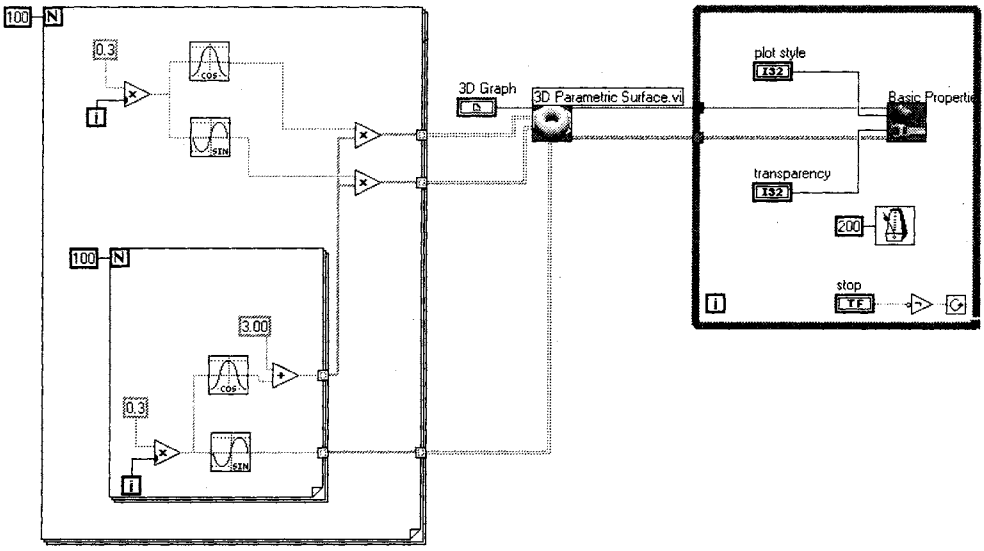


Рис. 8.43

специальные функции, которые создаются автоматически во время помещения трехмерного графика на лицевую панель.

Три типа инструментов построения трехмерных графиков могут вычерчивать в трех измерениях, но каждый из них работает по-своему:

- **3D-график поверхности** вычерчивает простую поверхность, определяемую матрицей z. Поверхность строится за счет изменения векторов *x* и *y*. На вход данной функции необходимо подавать двумерный массив и два необязательных одномерных массива.
- **3D-параметрический график** вычерчивает поверхность на основе плоскостей *x*, *y* и *z*. На вход данной функции необходимо подавать три двумерных массива или матрицы, которые определяют каждую из плоскостей *x*, *y* и *z*.
- **3D-график кривой** описывает линию на основе точек *x*, *y* и *z*. Ввод этого ВП состоит из трех одномерных массивов или векторных вводов, которые определяют каждую точку на графике.

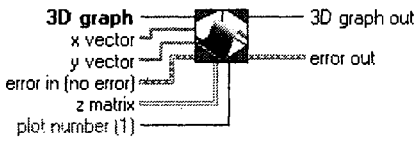


*Рис. 8.44. Функция **3D-график поверхности***

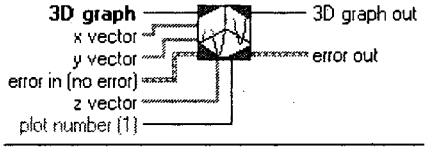


*Рис. 8.46. Функция **3D-параметрический график***

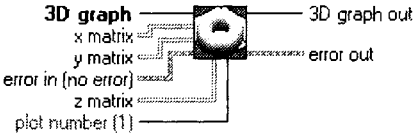


*Рис. 8.45. Функция **3D-график кривой***

Трехмерные графики могут быть сложнее, чем простые развертки и графики, с которыми вы работали раньше. Они представляют собой нечто более совершенное, и мы не будем более говорить о них. Помните лишь, что они имеются в вашем распоряжении в случае необходимости. Для того чтобы понять, как пользоваться трехмерными графиками LabVIEW, посмотрите примеры.

8.8.3. Графики цифровых осциллограмм

Существует еще один тип сложных графиков, о котором мы кратко поговорим, – **График цифровой осциллограммы** (Digital Waveform Graph), который находится в палитре **График**. Все инструменты построения графиков, которые мы до сих пор использовали, очень хорошо работают с *аналоговыми* данными. Если вы хотите использовать LabVIEW для отображения и анализа *цифровых* сигналов (сигналы,

которые обычно имеют лишь два состояния, такие как ИСТИНА или ЛОЖЬ, 0 или 5 В, «включено» или «выключено» и т.д.), то увидите, что этот специальный тип графика весьма полезен, особенно если вы работаете с временными диаграммами или логическими анализаторами.

На рис. 8.47 показан пример графика цифровой осциллограммы, который строит восемь цифровых сигналов с течением времени.

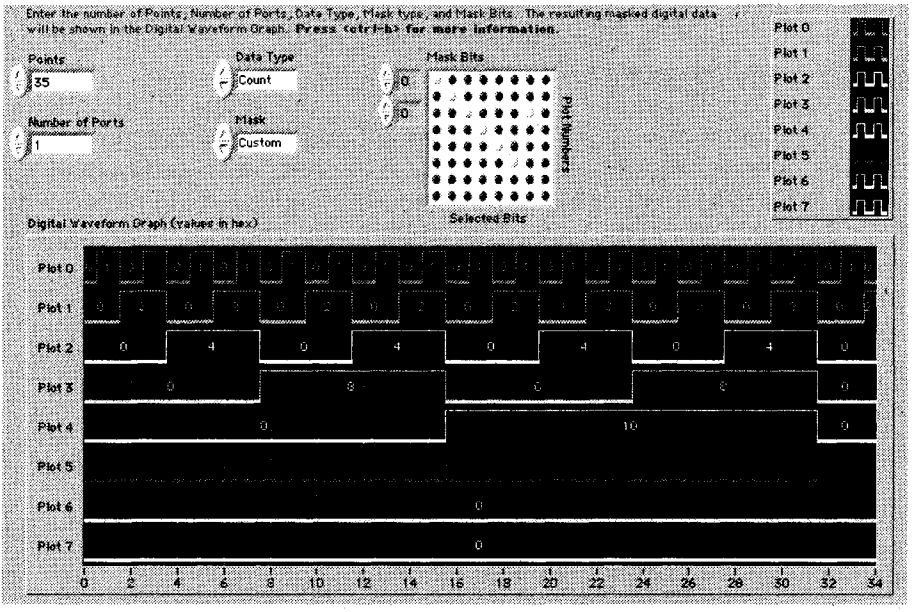


Рис. 8.47

На блок-диаграмме (рис. 8.48) график цифровой осциллограммы имеет кластерный терминал, вводами которого являются:

- X_0 (начальное значение X);
- ΔX;

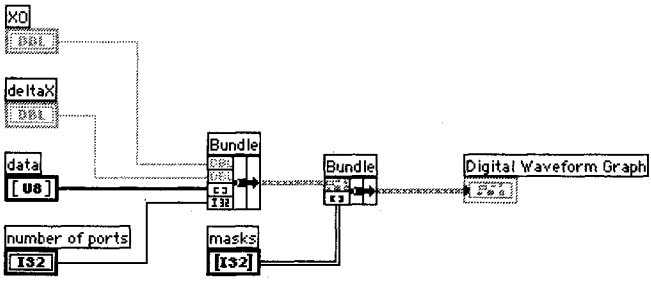


Рис. 8.48

- данные (массив беззнаковых байтов);
- количество портов;
- маски (массив целых чисел) – необязательный ввод.

Если вы уже работали с цифровой логикой раньше, то для вас не будет сложным использование графика цифровых осциллограмм. Несколько хороших примеров применения графиков цифровых осциллограмм вы можете найти в библиотеке `examples\general\graphs\DigitalWaveformGraph.llb`.

8.9. Осциллограммы

Во многих технологических и научных исследованиях большая часть данных, с которыми вы работаете, является набором значений, изменяющихся во времени. Например, аудиосигналы представляют собой изменения давления с течением времени, электрокардиограмма – изменение напряжения со временем, колебания поверхности жидкости в момент падения в нее камешка – это изменения координат *x, y, z* со временем. В LabVIEW есть удобный способ организации и работы с подобными зависящими от времени данными, – *осциллограммы* (waveforms). Они дают возможность сохранить не только основные значения данных, но также отметку о времени получения первого значения, временную задержку между точками данных и комментарии к данным. Как и в случае массивов и кластеров, здесь можно складывать, вычитать и осуществлять многие другие действия непосредственно с осциллограммами. Допустимо создать элемент управления **Осциллограмма** (Waveform) на лицевой панели, взяв его из палитры **Ввод/вывод** (I/O) – рис. 8.49. Соответствующее отображение на блок-диаграмме – это терминал коричневого цвета (рис. 8.50).

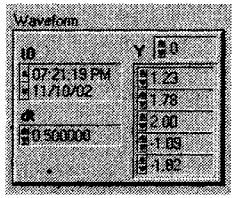


Рис. 8.49. Элемент управления осциллограммой

Рис. 8.50. Терминал элемента управления осциллограммой

Тип данных осциллограммы представляет собой особый тип кластера, который состоит из четырех компонентов – **Y**, $\mathbf{t_0}$, **dt** и **Свойства** (Attributes):

- **Y** – одномерный массив данных, которыми могут быть либо точка, либо другая осциллограмма в зависимости от действия. Представлением одномерного массива является DBL;
- **t0** – скалярная величина, которая представляет время (в соответствии с системными часами) получения первой точки в массиве Y. Эту величину можно назвать *начальное время*;
- **dt** (Δt) – скалярная величина, показывающая время между точками массива Y;
- **Свойства** – по умолчанию этот компонент скрыт (вы можете увидеть его, щелкнув правой кнопкой мыши и выбрав опцию **Видимые элементы** ⇒

Свойства). Это строковый тип данных, который дает возможность присоединить к данной осциллограмме другую информацию, такую как номер прибора или номер канала системы получения данных.

На рис. 8.49 элемент управления осциллограммой показывает, что первая точка осциллограммы была создана в 7:21:19 PM 10 ноября 2002 года и что время между каждой точкой составляет 0,5 с.

8.9.1. Сравнение осциллограмм и массивов

Во многих случаях можно рассматривать осциллограммы как одномерные массивы, в которых содержатся данные, – с некоторой дополнительной информацией о времени и временных интервалах точек данных. Осциллограммы чаще всего используются при получении аналоговых данных (см. главы 10 и 11).

Естественно, нет необходимости применять тип данных осциллограммы – для хранения данных вы всегда можете использовать массивы. Однако здесь имеется некоторое преимущество перед массивами:

- *наличие t0.* При отсутствии типа данных осциллограммы нельзя определить время получения данных. Осциллограммы автоматически записывают время и дату в компонент **t0**, что дает реальное время получения ваших данных;
- *упрощенное построение.* Тип данных осциллограммы также упрощает вычерчивание данных на графике. В предыдущих версиях LabVIEW вы были вынуждены объединять в кластер значения начальной точки (X_0) и времени между точками (ΔX) с вашими данными (массив Y). Поскольку тип данных осциллограммы уже содержит указанные элементы, то все, что нужно сделать, – это соединить его с графиком;
- *упрощенное построение многолучевых графиков.* Тип данных осциллограммы упрощает построение графика с множеством лучей. В предыдущих версиях LabVIEW требовалось объединять в кластер значения X_0, ΔX и массив Y для каждого луча, а затем комбинировать их в массив для создания многолучевого графика. Используя тип данных осциллограммы, вы лишь подключаете одномерный массив осциллограмм к такому графику. Если, например, вы осуществляете сбор данных из многих каналов с помощью виртуального прибора аналогового ввода, то последний автоматически возвращает одномерный массив, и вам достаточно лишь подключиться к графику.

8.9.2. Функции для работы с осциллограммами

В палитре **Функции** имеется целая подпалитра, посвященная работе с осциллограммами, – **Осциллограмма** (Waveform) – рис. 8.51.

Поскольку на самом деле осциллограммы представляют собой особый тип кластера, вы столкнетесь с функциями, аналогичными функциям **Разделить** и **Объединить**: **Вывод компонентов осциллограммы** (Get Waveform components) и **Создать осциллограмму** (Build Waveform).

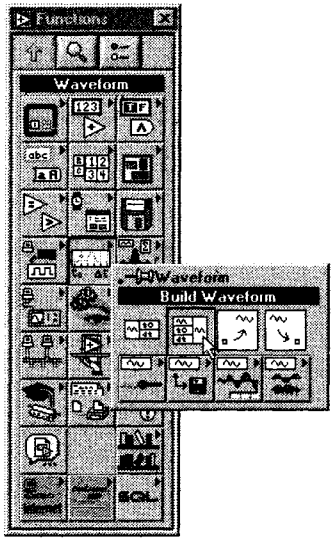


Рис. 8.51. Палитра ***Осциллограмма***

Функция **Вывод компонентов осциллограммы** возвращает заданные компоненты осциллограммы. Задать их можно нажатием правой кнопки мыши, выбором опции **Добавить элемент** и созданием элемента отображения. Эту функцию также можно расширять инструментом перемещения.

Функция **Создать осциллограмму** создает или модифицирует существующую осциллограмму. Если вы не подключили входную осциллограмму, то функция создает новую осциллограмму на базе введенных вами компонентов. Если же вы подключили входную осциллограмму, она модифицируется с использованием введенных компонентов. Эту функцию также можно расширять инструментом перемещения.

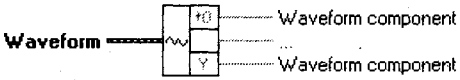


Рис. 8.52. Вывод компонентов осциллограммы

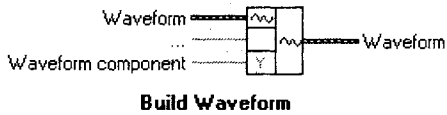


Рис. 8.53. Создать осциллограмму

Палитра **Осциллограмма** имеет подпалитры, содержащие много полезных функций и операций, которые вы можете использовать при работе с осциллограммами.

Функции, содержащиеся в подпалитре **Операции с осциллограммами** (Waveform Operations), позволяют осуществлять арифметические действия и действия сравнения, такие как сложение, вычитание, умножение, определение точек максимума и минимума, объединение и т.д. Обратите внимание, что при большинстве действий, затрагивающих две или более осциллограммы, последние должны иметь одни и те же значения **dt**.

Функции, содержащиеся в подпалитре **Ввод/вывод осциллограммы в/из файл(а)** (Waveform File I/O), позволяют записать в файл и считать из файла данные осциллограммы.

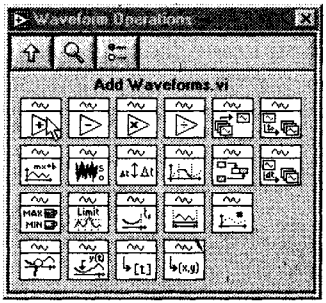


Рис. 8.54

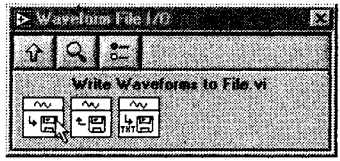


Рис. 8.55

Палитра **Измерения параметров осциллограммы** (Waveform Measurements) дает возможность измерять постоянную составляющую (DC), среднеквадратичное значение (RMS), нелинейные искажения, тоны частоты/амплитуды/фазы, отношение сигнал/шум (SINAD), усредненное быстрое преобразование Фурье (FFT).

Палитра **Создание осциллограммы** (Waveform Generation) позволяет создавать различные типы одно- и многотоновых сигналов, сигналов функционального генератора и шумовых сигналов. Например, вы можете создать синусоидальную волну, задав амплитуду, частоту и т.д.

И еще одно важное замечание: для построения осциллограммы необходимо подключить ее прямо к функциям **Развертка осциллограммы** или **График осциллограммы**. Терминал автоматически адаптируется для принятия типа данных осциллограммы и отразит информацию о временных параметрах на оси X.

В следующем упражнении мы рассмотрим пример использования и построения осциллограммы.

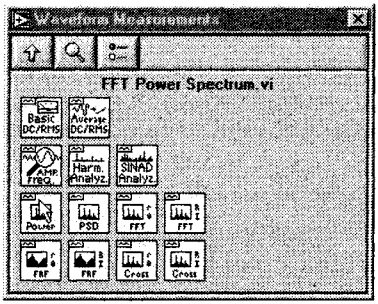


Рис. 8.56

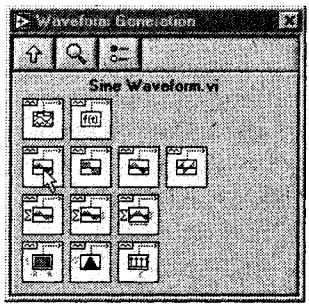


Рис. 8.57

8.9.3. Упражнение 8.6: создание и построение осциллограммы

В этом упражнении вы создадите синусоидальную осциллограмму, зададите начальный отсчет времени относительно текущего времени и построите ее на графике.

1. Откройте новую лицевую панель.
2. Поместите на лицевой панели круглый элемент управления (из палитры **Числовые**) и элементы отображения графика и осциллограммы. Назовите ручку управления **Частота**. Общий вид показан на рис. 8.58.

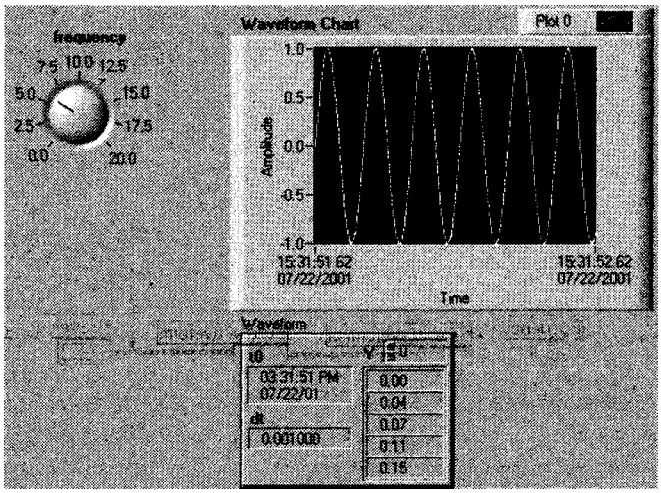


Рис. 8.58

3. Постройте блок-диаграмму, которая будет генерировать синусоидальную осциллограмму и строить ее. Поскольку функции создания осциллограммы не дают значение для компонента **t0** (по умолчанию они возвращают значение 7:00:00 PM 12/31/1903), то используйте функцию **Создать осциллограмму** для установки в **t0** текущего момента времени. Для этого потребуются следующие функции:
   - функция **Осциллограмма Синус** (Sine Waveform) из палитры **Осциллограмма ⇒ Создание осциллограммы** позволяет создавать синусоидальную волну с установленными параметрами;
   - функция **Вызвать дату/время в секундах** (Get Date/Time in Seconds) возвращает текущее время и дату;
   - функция **Создать осциллограмму** позволяет модифицировать компоненты осциллограммы. В данном упражнении это **t0**.

   Блок-диаграмма должна иметь вид, показанный на рис. 8.62.
4. Запустите и испытайте ваш виртуальный прибор на разных частотах. Вы можете щелкнуть правой кнопкой мыши по графику и выбрать функции

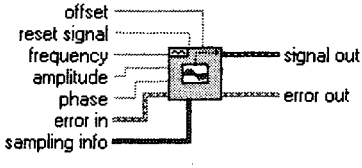


Рис. 8.59. Функция
Осциллограмма Синус

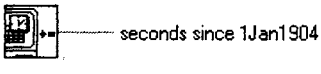


Рис. 8.60. Функция
Вызвать дату/время в секундах

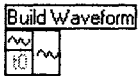


Рис. 8.61. Функция
Создать осциллограмму

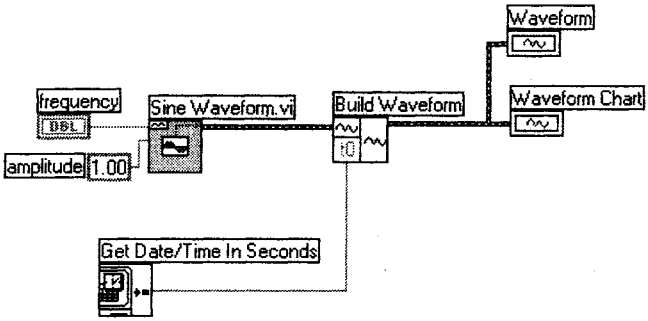


Рис. 8.62

Шкала X ⇒ Автомасштабирование X и **Шкала Y ⇒ Автомасштабирование Y**.

5. Сохраните виртуальный прибор как **Waveform Exercise.vi**.

Мы встретимся с осциллограммами вновь в главе 11, где они будут представлять собой тип данных, используемых в аналоговых функциях ввода/вывода.

8.10. Итоги

Используя развертки и графики LabVIEW, вы можете создавать визуальные изображения данных. *Развертки* постоянно обновляют старые данные, интерактивно вычерчивая одну точку (или набор точек) за единицу времени. Таким образом, легко увидеть текущее значение в контексте предыдущих значений данных. *Графики*, наоборот, отображают все данные сразу после их создания. *Осциллограммы* – новый тип данных – могут использоваться как с развертками, так и с графиками.

LabVIEW предлагает несколько типов графиков: графики *осциллограмм*, *двухкоординатные* графики, графики *интенсивности*, *трехмерные* графики и графики *цифровых осциллограмм*.

График осциллограммы строит лишь точки с единственными значениями, которые равномерно распределены вдоль шкалы X, как у изменяющейся во времени осциллограммы. Другими словами, график вычерчивает массив Y относительно установленной временной базы.

Двухкоординатный график (декартова система координат) дает возможность строить многозначные функции типа окружности. Он вычерчивает массив Y относительно массива X.

Графики интенсивности используются в основном для отображения смоделированных данных, поскольку они могут строить три переменных относительно друг друга на двумерном дисплее. Развертки и графики интенсивности применяют цвет для представления третьей переменной. На их вход подается двумерный массив чисел, где каждому числу соответствует цвет, а индекс этого числа в массиве служит для определения местоположения данного цвета на графике или развертке. В большинстве других случаев графики интенсивности работают как стандартные развертки и графики для двух переменных.

Трехмерные графики (только для Windows) – более сложный, но и более перспективный тип графиков, которые позволяют отображать координаты *x*, *y*, *z* в трехмерном пространстве. В LabVIEW имеется множество функций для манипулирования трехмерными графиками.

График цифровых осциллограмм – это специальный тип графиков, используемый для построения цифровых данных с течением времени. Он особенно полезен для визуализации состояний ИСТИНА/ЛОЖЬ, меняющихся со временем.

Разрешается модифицировать внешний вид разверток и графиков с помощью панели редактирования луча, панели редактирования шкалы и палитры управления графиком. Вы также можете изменить масштабы для удобного отображения данных и задействовать курсоры для маркировки лучей.

Развертки и графики способны вычерчивать несколько лучей. Типы данных, используемых ими, могут оказаться достаточно сложными, поэтому вам следует обращаться к примерам этой главы или находящимся в директории `examples` как к образцу во время написания вашего ВП, применяющего графики.

Механическое действие логических переключателей дает возможность управлять их поведением во время щелчка по ним мышью. Настройте переключатель так, чтобы он возвращался к значению по умолчанию после того, как новое значение считано, – вот способ подготовить его к новому использованию. Такой тип действия называется *защелкой*. Также допустимо настроить выключатель, чтобы он срабатывал после нажатия или отпускания кнопки мыши.

Осциллограммы являются специальным типом данных LabVIEW, которые сохраняют информацию о начальном отсчете времени и временном интервале между точками в последовательности данных. Они находятся в палитре **Осциллограммы**, в которой содержатся также все типы функций для работы с осциллограммами. Вы можете подавать осциллограмму прямо на развертку или график для ее построения

8.11. Дополнительные упражнения

8.11.1. Упражнение 8.7: лимит температуры

Создайте ВП, который измерял бы температуру один раз в секунду и отображал результаты на развертке осциллограммы в режиме временной развертки (scope mode). Если температура будет выше или ниже установленных пределов, то ВП включит светодиод на лицевой панели. Развертка должна вычерчивать температуру и ее верхний и нижний пределы. Обеспечьте возможность установки пределов с лицевой панели. Для сравнения взгляните на лицевую панель, изображенную на рис. 8.63. Назовите виртуальный прибор **Temperature Limit.vi**.

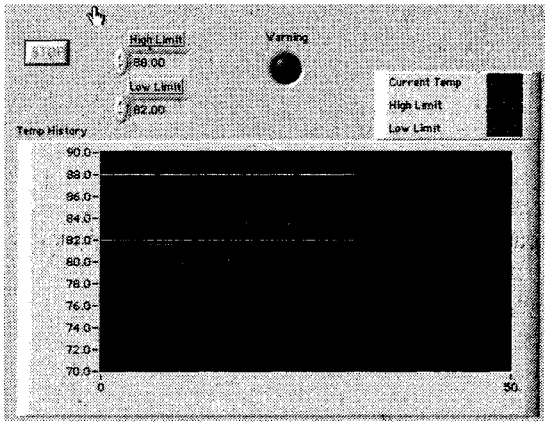


Рис. 8.63

8.11.2. Упражнение 8.8: максимальный и минимальный пределы температуры

Измените созданный вами в упражнении 8.6 ВП для отображения максимального и минимального значений температуры. Назовите ВП **Temp Limit (max/min).vi**.

Вам нужно будет использовать сдвиговые регистры и функцию **Массив Max&Min**, находящуюся в палитре **Массив**.

8.11.3. Упражнение 8.9: вычерчивание случайных массивов

Создайте ВП, который генерировал бы двумерный массив (три строки по десять столбцов), содержащий случайные числа. После создания массива выделите каждую строку и постройте ее на собственном графике. Лицевая панель вашего прибора должна содержать три графика. Назовите виртуальный прибор **Extract 2D Array.vi**.

Обзор

В данной главе рассматриваются некоторые полезные операции со строками. В LabVIEW содержится много встроенных функций, аналогичных функциям массивов и позволяющих управлять строковыми данными. Вы также научитесь сохранять данные в файл и считывать их из файла.

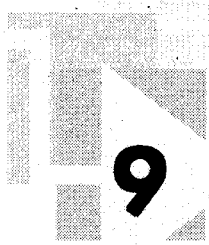
ЗАДАЧИ

- Изучить опции элементов управления строками и их отображением
- Понять, как использовать функции LabVIEW для работы со строками
- Научиться преобразовывать числовые данные в строковые и наоборот
- Научиться использовать ВП ввода/вывода файлов для сохранения данных на диск и считывания их в LabVIEW

ОСНОВНЫЕ ТЕРМИНЫ

- Полоса прокрутки
- Таблица
- Файл табличного формата (spreadsheet file)
- Форматирование строки (formatting string)
- Заданное выражение (regular expression)

ИЗУЧЕНИЕ СТРОК И ПОДПРИБОРЫ ВВОДА/ВЫВОДА



9.1. Еще раз о строках

Мы ввели понятие о строках в главе 4. Строка представляет собой набор символов ASCII[1]. Часто строки используются для простых текстовых сообщений. Например, при управлении инструментом вы передаете числовые данные в виде строк символов, а затем для обработки данных вы превращаете эти строки в числа. Для хранения числовых данных на диске также можно использовать строки. Прежде чем сохранить числовые значения в файле во многих подпрограммах ввода/вывода в файл/из файла LabVIEW переводит их в строковые переменные.

9.1.1. Выбор типа отображения

Элементы управления и индикаторы строк имеют несколько полезных опций. Например, они могут отображать и принимать символы, которые обычно являются неотображаемыми, в частности знак возврата на один символ назад, знак возврата каретки и табуляция. Если вы выбрали функцию '\' **Кодированное отображение** ('\' Codes Display) вместо функции **Нормальное отображение** (Normal Display) из контекстного меню строк, то скрытые символы появятся в виде обратного слэша, за которым следует соответствующий код. В табл. 9.1 показаны значения этих кодов.

Нужно использовать заглавные буквы для шестнадцатеричных символов и строчные буквы для специальных символов, таких как знаки смещения формы и знаки возврата. LabVIEW интерпретирует последовательность `\BFare` как шестнадцатеричный BF, идущий перед словом `are`, тогда как последовательности `\bFare` и `\bfare` интерпретируются как знак возврата, идущий перед словами `Fare` и `fare`.

[1] На момент написания книги LabVIEW 6.0 не поддерживает функцию Unicode, хотя, возможно, такая функция вскоре появится.

Таблица 9.1. Слэш-коды LabVIEW

Код	Выполнение в LabVIEW
\00-\FF	Шестнадцатеричное значение 8-битового символа; буквы должны быть заглавными
\b	Знак возврата на один символ назад (ASCII BS, эквивалент \08)
\f	Знак смещения формы (ASCII FF, эквивалент \0C)
\n	Новая строка (ASCII LF, эквивалент \0A)
\r	Знак возврата каретки (ASCII CR, эквивалент \0D)
\t	Табуляция (ASCII HT, эквивалент \09)
\s	Пробел (эквивалент \20)
\\	Обратный слэш (ASCII \, эквивалент \5C)

В последовательности \Bfare символы\B не являются кодом знака возврата и \Bf не является действительным шестнадцатеричным кодом. В данном случае, когда за обратным слэшем следует лишь часть действительного шестнадцатеричного символа, LabVIEW подставляет 0 сразу после обратного слэша, поэтому \B интерпретируется как шестнадцатеричное 0 B. Когда же за обратным слэшем не идет действительный символьный код, LabVIEW игнорирует этот символ.

Данные в строке не изменяются при выборе режима отображения; меняется лишь вид некоторых символов. Режим **'\' Кодированное отображение** необходим для отладки программ и определения скрытых символов, которые требуются для работы с приборами, последовательным портом и другими инструментами.

После переключения в режим **'\' Кодированное отображение** строка показывает пробелы, введенные оператором.

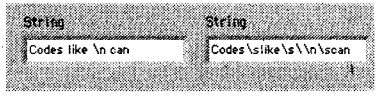


Рис. 9.1. Строка показывает код обратного слэша для символа новой строки в режиме ***Нормальное отображение***

Строки также имеют опцию **Скрытое отображение** (Password Display), которая заставляет элемент управления или отображения показывать «звездочку» вместо каждого введенного символа, поэтому никто не сможет увидеть, что вы печатаете. В то время как на лицевой панели видна лишь последовательность знаков «*****», на блок-диаграмме в строке считываются реальные данные.

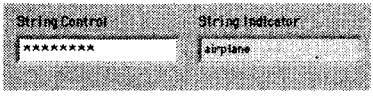


Рис. 9.2

Рис. 9.3

Очевидно, такое отображение полезно, если вы хотите защитить весь ВП или его часть.

Если вы хотите увидеть строку в виде шестнадцатеричных символов, а не алфавитно-цифровых знаков, используйте опцию **Шестнадцатеричное отображение** (Hex Display).

9.1.2. Одинарные строки

Если вы выберете функцию **Ограничить до одной строки** (Limit to Single Line), то строка всегда останется единственной. Если вы отметите эту функцию в контекстном меню строки, размер строки не превысит одной строчки текста, то есть в ней не будет символов возврата каретки. Если вы нажмете клавиши <enter> или <return>, то ввод теста будет автоматически закончен. Если число строк не лимитировано, нажатие клавиши <return> перемещает курсор на новую строку, и вы можете продолжать печатать текст.

9.1.3. Обновление строки во время ввода текста

Обычно элементы управления строками не изменяют их значения на терминале блок-диаграммы до тех пор, пока вы не закончите печатать текст и не нажмете клавишу <enter>. Щелкните мышью за пределами окна строки или по кнопке √ инструментальной линейки ВП для указания завершения ввода строки. Эту операцию вы будете использовать в большинстве случаев, так как нежелательно, чтобы блок-диаграмма начинала работать со строкой, прежде чем пользователь закончил ввод текста.

Если вы хотите, чтобы значения данных обновлялись во время набора текста (как, например, в случае кнопки управления), вызовите контекстное меню элемента управления строками и выберите функцию **Обновлять значение во время ввода** (Update Value While Typing).

9.1.4. Полоса прокрутки

Если выбрать опцию **Видимые элементы ⇒ Полоса прокрутки** (Scrollbar) в контекстном меню строк, то на элементе отображения или управления появляется вертикальная полоса прокрутки. Вы можете использовать эту опцию для минимизации пространства, занимаемого на лицевой панели элементами управления строками, которые содержат большое количество текста. Обратите внимание: опция будет серой до тех пор, пока вы не увеличите в достаточной мере размер вашего текста, чтобы полоса прокрутки стала необходимой.

9.1.5. Таблицы

Таблица в LabVIEW представляет собой специальную структуру, которая отображает двумерный массив строк. Вы можете найти ее в подпалитре **Списки и таблицы**

(Lists&Tables) палитры **Элементы управления** (Controls). Пример таблицы показан на рис. 9.4.

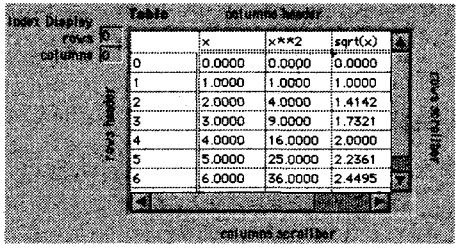


Рис. 9.4. Пример таблицы

В таблицах есть заголовки строк и столбцов, которые можно сделать видимыми или спрятать; заголовки отделены от данных тонкой двойной линией. Вы можете изменять заголовки текста, используя инструменты ввода текста (A) или управления («палец»). Также допустимо обновлять или считывать заголовки, при помощи узлов свойств.

Как и в случае массивов, элемент отображения индекса таблицы показывает, какая ячейка видна в верхнем левом углу таблицы. Для того чтобы научиться работать с таблицей, откройте и запустите пример **Building Tables.vi**, расположенный в библиотеке `EVERYONE\CH9.LLB`.

9.1.6. Окна списков

В LabVIEW имеется два вида окон, содержащих списки: **Окно-список** (Listbox) и **Многостолбцовое окно-список** (Multi-column Lisbox), находящиеся в палитре **Списки и таблицы**.

Окно списков в LabVIEW является аналогом таблицы, однако во время работы программы ведет себя по-другому. В режиме редактирования вы можете впечатать в данное окно любой текст. Когда вы запускаете ВП, окно списков действует в качестве меню «выбора из множества», где нужно щелкнуть мышью по любой строке для ее выделения и выбора. На рис. 9.5 изображено окно списка с несколькими столбцами.

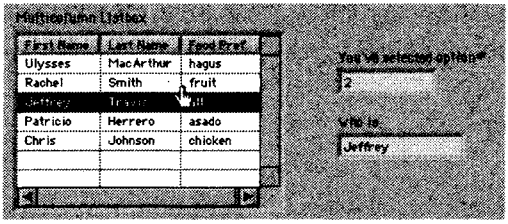


Рис. 9.5. Окно списка

Окна списков целесообразно использовать, когда вы хотите представить данные (в одном или нескольких столбцах) пользователю путем выбора опции, а не ввода текста.

Подробнее о работе окон списков вы узнаете, открыв пример **Listbox Example.vi**, который находится в библиотеке EVERYONE\CH9.LLB.

9.2. Использование функций обработки строк

Строки, как и массивы, могут быть весьма полезными при использовании встроенных функций, предлагаемых LabVIEW. В данном разделе речь пойдет о нескольких функциях, находящихся в подпалитре **Строки** (String) палитры **Функции**. После изучения данного раздела вы можете просмотреть палитру, чтобы узнать, какие еще функции она содержит.

Функция **Длина строки** (String Length) возвращает число символов в данной строке.

Функция **Объединение строк** (Concatenate Strings) объединяет все входные строки в одну выходную.

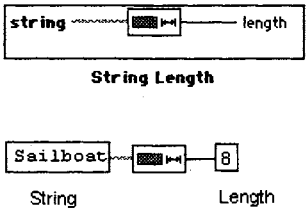


Рис. 9.6. Функция ***Длина строки***

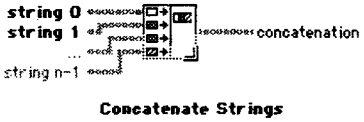


Рис. 9.7. Функция ***Объединение строк***

Функция, помещенная на блок-диаграмму, имеет вид иконки, изображенной слева. Для увеличения числа входов вы можете изменить размер функции с помощью инструмента перемещения.

Кроме простых строк в качестве входных данных вы можете подключить одномерный массив строк. В этом случае на выходе будет одна строка, содержащая объединение строк массива.

Во многих случаях вам понадобится преобразовывать строки в числа или числа в строки. Функции **Преобразовать в строку** (Format Into String) и **Просмотр**

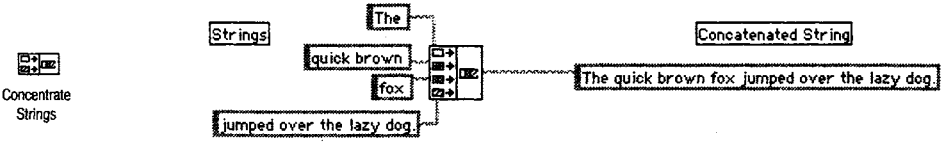


Рис. 9.8

строки (Scan From String) обладают такими возможностями. Мы поговорим об этих функциях позднее. Попросту говоря, функция **Преобразовать в строку** преобразует числовые данные в строковые.

В примере, изображенном на рис. 9.11, эта функция преобразует число с плавающей запятой `1.28` в 6-байтовую строку `1.2800`.

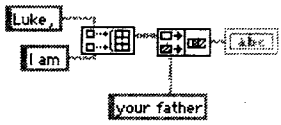


Рис. 9.9

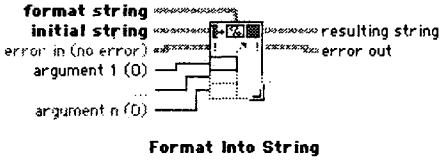


Рис. 9.10. Функция **Преобразовать в строку**

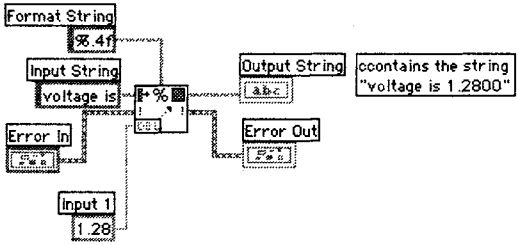


Рис. 9.11. Преобразование числа в строку

Функция **Преобразовать в строку** форматирует входной аргумент (который записан в числовом формате) как строку в соответствии с определением формата, заданным во входной переменной **формат строки** (format string). Спецификации подробно приводятся в руководстве по работе с LabVIEW. Данная функция применяет сформированный шаблон к входным данным, подключенным к вводу **начальная строка** (initial string), и выдает результаты на терминал **результирующая строка** (resulting string). В табл. 9.2 приводятся некоторые примеры работы функции **Преобразовать в строку**.

Таблица 9.2. Примеры работы функции **Преобразовать в строку**

Начальная строка	Формат строки	Число	Результирующая строка
(пустая)	счет= %2d%%	87	счет= 87%
результат =	%2d%%	87	счет= 87%
(пустая)	уровень=%7.2eV	0.03642	уровень= 3.64E-2V
(пустая)	%5.3f	5.67 N	5.670 N

Символ `%` говорит о начале форматирования. Например в случае `%число1.число2число1` определяет длину результирующей строки, а `число2` определяет точность (то есть количество цифр после десятичной запятой). `f` форматирует входное число как число с плавающей запятой, `d` – как целое и `e` – как число с плавающей запятой в научной системе обозначений (экспоненциальное представление чисел).

Размер функции **Преобразовать в строку** может быть изменен для одновременного преобразования множества значений в одну строку.

Функция **Вызвать строку даты/времени** (Get Date/Time String) из палитры **Время и диалоги** выводит строку даты, содержащую текущую дату, и строку времени, содержащую текущее время. Эта функция полезна при задании времени создания/модификации данных. Обратите внимание, что вам не нужно подключать входы к функции **Вызвать строку даты/времени**: она может использовать значения по умолчанию.

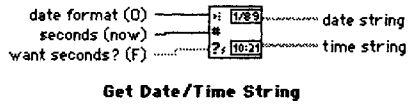


*Рис. 9.12. Функция **Вызвать строку даты времени***

9.3. Упражнение 9.1: создание строк

Создайте ВП, который преобразует число в строку и объединяет эту строку с другими строками для получения одной строки на выходе. Виртуальный прибор также должен определять длину выходной строки.

1. Создайте лицевую панель, как показано на рис. 9.13.
   Виртуальный прибор объединит данные двух строковых и числового индикатор в одну выходную строку, которая будет демонстрироваться на элементе отображения строковых данных. Числовой индикатор покажет ее длину.
2. Постройте блок-диаграмму, как показано на рис. 9.14.
   Функция **Преобразовать в строку** преобразует заданное число в строку смешанного формата с четырьмя знаками точности.

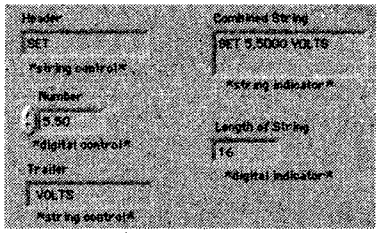


Рис. 9.13

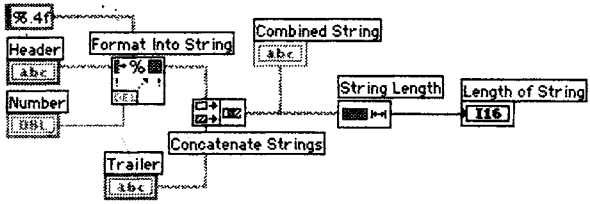


Рис. 9.14

Функция **Объединение строк** объединяет все входные строки в одну выходную. Для увеличения количества входов растяните иконку с помощью инструмента перемещения («стрелка»).

Функция **Длина строки** (String Length) возвращает количество символов в объединенной строке.

3. Вернитесь к лицевой панели и введите текст в оба строковых элемента управления и число в цифровой элемент управления. Не забудьте добавить пробелы в конце заголовка и начале строки завершения. Запустите виртуальный прибор.
4. Сохраните и закройте виртуальный прибор. Назовите его **Build String.vi** и поместите в директорию MYWORK или в библиотеку виртуальных приборов.

9.4. Функции анализа

Иногда требуется разделять строки или преобразовывать их в числа. Все эти операции выполняются с помощью функций анализа.

Функция **Выделение подстроки** (String Subset) осуществляет доступ к отдельной части строки. Она возвращает подстроку, начиная с символа с заданным номером **смещение** (offset), с заданным числом символов **длина** (length). Помните, что номером первого символа является 0.

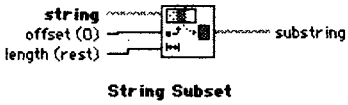


Рис. 9.15. Функция ***Выделение подстроки***

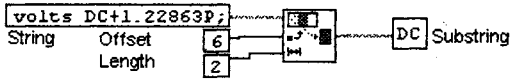


Рис. 9.16

Функция **Шаблон строки** (Match Pattern) используется для поиска заданной структуры символов в строке. Она ищет и возвращает найденную подстроку. Эта функция ищет выражение в соответствии с **заданным выражением** (regular expression), начиная с символа с определенным **смещением**. Как только она находит выражение, она разбивает строку на три подстроки. Если же выражение не обнаружено, то содержание вывода **подходящая подстрока** (match substring)

оказывается пустым, а значение **смещение после совпадения** (offset past match) устанавливается в –1.

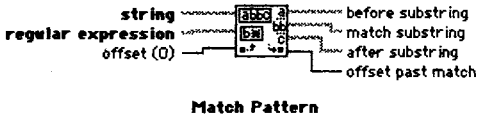


*Рис. 9.17. Функция **Шаблон строки***

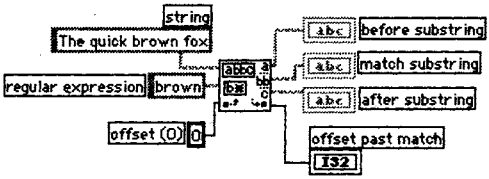


Рис. 9.18

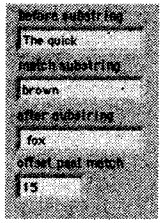


Рис. 9.19

Функция **Просмотр строки** (Scan from String) – обратная функции **Преобразовать в строку** – преобразует строку, содержащую действительные числовые символы (от 0 до 9, +, –, e, E и период) в числовые данные. Эта функция начинает просмотр **входной строки** (input string) с **места начала поиска** (initial search location) и преобразует данные в соответствии со спецификацией формата строки. Данную функцию можно увеличить для одновременного преобразования нескольких значений.

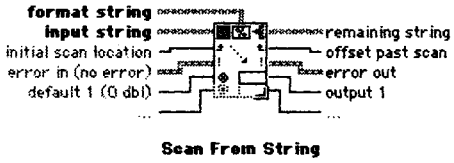


*Рис. 9.20. Функция **Просмотр строки***

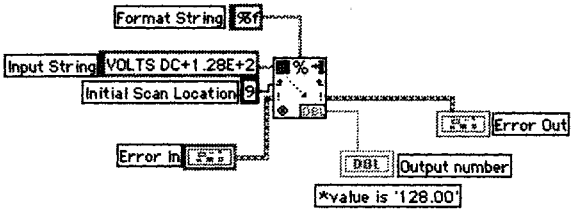


Рис. 9.21

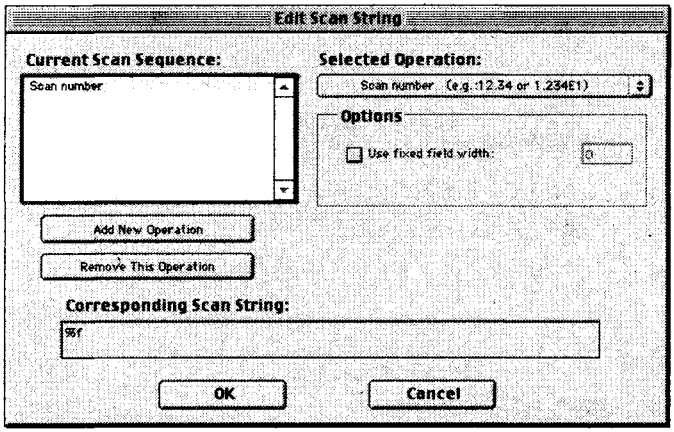


Рис. 9.22

В данном примере функция **Просмотр строки** преобразует строку `VOLTS DC+1.28E+2` в число `128.00`. Она начинает просмотр с восьмого символа строки (который в данном случае является знаком +; не забудьте, что первый символ имеет порядковый номер 0).

Как функция **Преобразовать в строку**, так и функция **Просмотр строки** имеют панель **Редактирование выделения/создание строки** (Edit Scan/Format String), с помощью которой вы можете задать формат строки. В этом диалоговом окне допустимо установить формат, точность, тип данных и длину преобразованного значения. В панель редактирования выделения/создания строки можно попасть, дважды щелкнув мышью или вызвав контекстное меню соответствующей функции и выбрав опцию **Редактирование выделения/создание строки**.

После создания формата строки и щелчка мышью по кнопке **Создать строку** (Create String) будет создана строковая постоянная и подключена к вводу **формата строки** (format string).

9.5. Упражнение 9.2: и снова об анализе строк

Создайте ВП, анализирующий данные длинной строки путем выделения подстроки и преобразования числовых символов в этой подстроке в числовое значение.

1. Создайте лицевую панель, как показано на рис. 9.23.
2. Настройте строку **Пароль** (Password) так, чтобы она отображала лишь звездочки, путем выбора функции **Скрытое отображение** (Password Display) в контекстном меню.
3. Постройте блок-диаграмму, как показано на рис. 9.24.


String Subset

Функция **Выделение подстроки** возвращает из входной строки подстроку заданной длины, начиная с определенного символа.

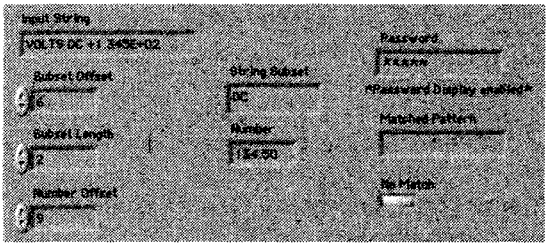


Рис. 9.23. Лицевая панель ВП

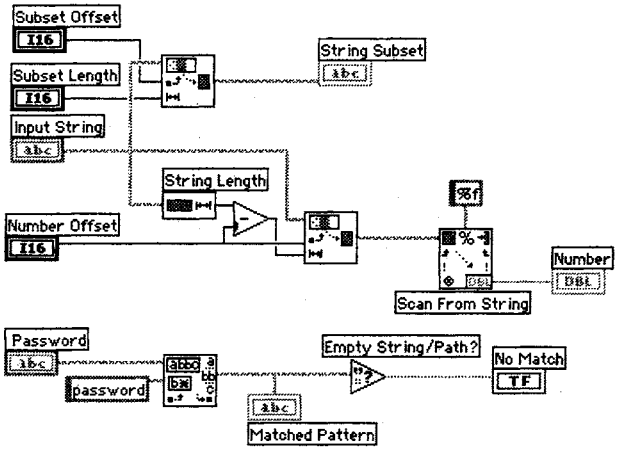


Рис. 9.24

Функция **Просмотр строки** преобразует строку, содержащую действительные числовые символы (от 0 до 9, +, –, E и период) в число.

Функция **Шаблон строки** сравнивает строку пароля пользователя со строкой заданного пароля. Если имеется совпадение, то строка отображается, если же совпадения нет, то строковый элемент отображения покажет пустую строку.

Функция **Пустая строка/путь?** (Empty String/Path?) из палитры **Сравнение** возвращает логическое значение ИСТИНА, если находит пустую строку в шаблоне подстроки вывода в функции **Шаблон строки**.

Функция **Длина строки** (String Length) возвращает число символов в строке.

4. Запустите виртуальный прибор с данными, показанными на рис. 9.23. Обратите внимание, что выделение подстроки DC осуществляется из входной строки. Также отметьте, что числовая часть строки была

проанализирована и преобразована в число. Вы можете попробовать различные значения элементов управления, только помните, что строки, как и массивы, индексируются, начиная с 0.

Обратите внимание, что строка пароля показывает лишь «звездочки». Функция **Шаблон строки** сравнивает входной пароль с паролем в строке (который в данном случае содержит символы, «password») и возвращает его, если он обнаружен. Если же совпадения не было, функция возвращает пустую строку.

5. Закройте ВП, выбрав функцию **Закрыть** в меню **Файл**. Сохраните ВП в директории MYWORK или в библиотеке виртуальных приборов под именем **Parse String.vi**.

9.6. Ввод/вывод данных в файл/из файла

Операции ввода и вывода (I/O) запрашивают информацию из файла и сохраняют информацию в файле на диске. LabVIEW имеет ряд гибких функций ввода и вывода наряду с простыми функциями, которые обеспечивают выполнение практически всех операций ввода и вывода. В этой главе мы поговорим о простых функциях. Все они находятся в подпалитре **Ввод/вывод файлов** (File I/O) палитры **Функции**.

9.6.1. Как они работают

Для функций, работающих с файлами, необходимо ввести путь к размещению файла, который выглядит как строка. Путь является особым видом данных для работы с файловой системой, указывая местоположение файла (см. главы 4 и 12). Если вы не подключили переменную пути к файлу, то функции вызовут диалоговое окно и попросят вас выбрать файл или ввести его имя. Будучи вызванными, функции откроют или создадут файл, прочитают или запишут данные, а затем закроют файл. Файлы, созданные виртуальными приборами, о которых мы сейчас говорим, представляют собой обыкновенные текстовые файлы. Как только вы запишете информацию в файл, вы можете открыть его, используя любой текстовый редактор для просмотра данных.

Одним из наиболее распространенных способов хранения данных в файле является форматирование текстового файла. Это удобно, поскольку вы можете открыть его в любой программе, работающей с таблицами. В большинстве табличных форматов столбцы отделяются табуляцией, а строки – символами конца строк (EOL). Функции **Записать в файл табличного формата** (Write To Spreadsheet File) и **Считать из файла табличного формата** (Read From Spreadsheet File) имеют дело с файлами табличного формата.

Функция **Сохранить символы в файл** (Write Characters To File) сохраняет строковый символ в новый файл или добавляет строку в существующий файл (рис. 9.25).

Функция **Считать символы из файла** (Read Characters From File) считывает определенное количество символов из файла, начиная с заданного смещения (рис. 9.26).

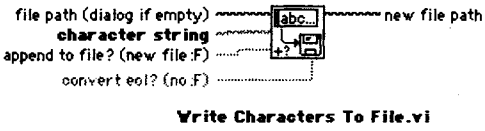


*Рис. 9.25. Функция **Сохранить символы в файл***

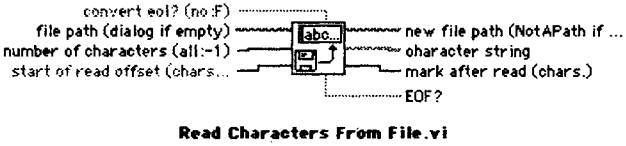


*Рис. 9.26. Функция **Считать символы из файла***

Функция **Считать строки из файла** (Read Lines From File) считывает определенное количество строк из файла, начиная с заданного смещения (рис. 9.27).

Функция **Записать в файл табличного формата** (рис. 9.28) преобразует двумерный или одномерный массив чисел с одинарной точностью в текстовую строку, а затем записывает строку в новый файл или добавляет в существующий. Дополнительно вы можете транспонировать данные. Не подавайте данные одновременно на

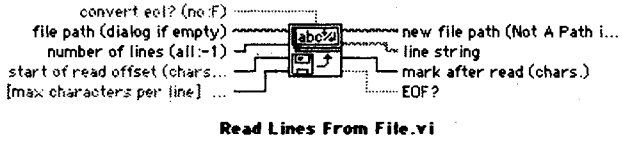


*Рис. 9.27. Функция **Считать строки из файла***

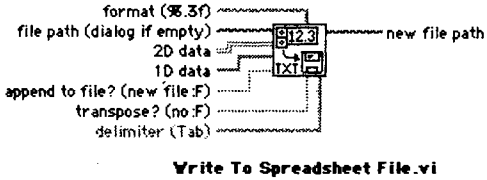


*Рис. 9.28. Функция **Записать в файл табличного формата***

входы одномерного и двумерного массивов: один из них будет проигнорирован. Текстовые файлы, созданные этим виртуальным прибором, прекрасно считываются большинством программ, работающих с таблицами.

Функция **Считать из файла табличного формата** (рис. 9.29) считывает определенное количество строк или столбцов из числового текстового файла, начиная с заданного символа, и преобразует данные в двумерный массив чисел с одинарной точностью. Этот виртуальный прибор может читать файлы табличного формата, сохраненные в текстовом формате.

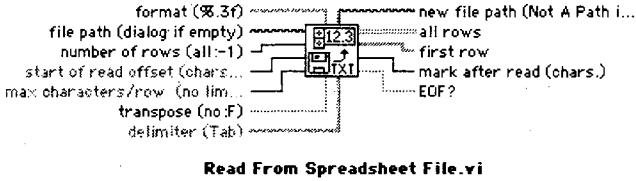


Рис. 9.29. Функция ***Считать из файла табличного формата***

Описанные функции для работы с файлами имеют достаточно высокий уровень, однако легки в использовании. Все они находятся в палитре **Ввод/вывод файлов**. В LabVIEW есть и другие функции – более гибкие, но довольно сложные в применении. Мы поговорим о них в главах 12 и 15.

9.7. Упражнение 9.3: запись в файл табличного формата

В данном упражнении вы измените имеющийся ВП для сохранения данных в новый файл формата ASCII. Позднее вы можете редактировать этот файл из любого приложения, работающего с таблицами.

1. Откройте ВП **Graph Sine Array.vi,** созданный в упражнении главы 8. Если вы не закончили создание этого виртуального прибора, воспользуйтесь его полной версией в библиотеке `EVERYONE\CH8.LLB`. Если вы помните, этот ВП генерирует два массива данных и строит их на графике. Вы можете видоизменить ВП для записи двух массивов в файл, в котором каждый столбец содержит массив данных.
2. Откройте блок-диаграмму **Graph Sine Array.vi** и измените виртуальный прибор путем добавления элементов, изображенных внутри овала.

Функция **Записать в файл табличного формата** преобразует двумерный массив в строку таблицы и записывает ее в файл. Если путь к файлу не установлен (как в этом упражнении), то во время выполнения появится диалоговое окно, где вам предложат выбрать имя файла.

Логическая константа (Boolean Constant) управляет транспонированием двумерного массива перед его записью в файл. Для того чтобы

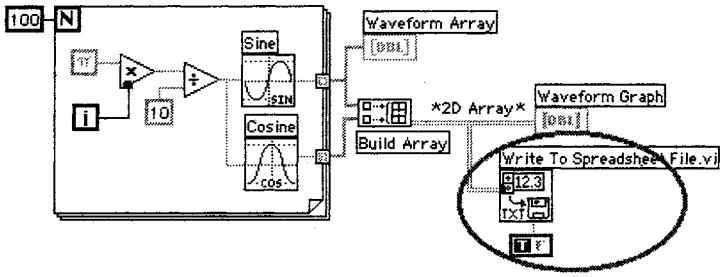


Рис. 9.30

изменить значение на ИСТИНА, щелкните инструментом управления по этой константе. Вам понадобится транспонировать данные, так как массивы данных определяются строками (каждая строка двумерного массива представляет собой массив данных). Поскольку желательно, чтобы каждый столбец таблицы в файле содержал данные для одной осциллограммы, то двумерный массив сначала должен быть транспонирован.

3. Вернитесь к лицевой панели и запустите ВП. После того как массивы данных будут созданы, появится диалоговое окно, где нужно указать имя нового файла. Введите имя файла (если вы не видите эту опцию, щелкните мышью по кнопке **Новый** (New) в диалоговом окне и выберите **Файл** – File) и щелкните по кнопке **ОК**. Запомните имя и местоположение файла, поскольку вам придется считывать данные из него в следующем упражнении.

Не пытайтесь с помощью функций для работы с файлами записать файл в библиотеку виртуальных приборов. В противном случае вы можете переписать библиотеку и потерять сохраненную работу.

4. Сохраните ВП в директории MYWORK или в библиотеке виртуальных приборов, под именем **Graph Sine Array to File.vi**. Закройте ВП.
5. Используйте редакторы таблиц, если таковые имеются, или простой текстовый редактор для просмотра только что созданного файла. Вы увидите два столбца по 100 элементов в каждом.

9.8. Упражнение 9.4: считывание из файла

Создайте ВП для считывания данных из файла, сохраненного в последнем упражнении, и постройте их на графике.

1. Откройте новый ВП и поместите график осциллограммы на его лицевую панель (рис. 9.31). Убедитесь, что автоматическая настройка шкалы включена.
2. Постройте маленькую блок-диаграмму, как показано на рис. 9.32. Примените функцию **Считать из файла табличного формата** для ввода данных в прибор и отображения их на графике.

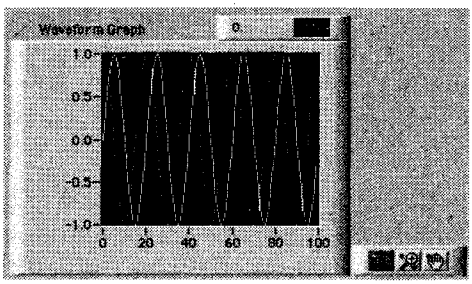


Рис. 9.31

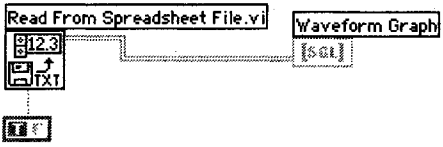


Рис. 9.32

3. Используя значение ИСТИНА логической константы, транспонируйте массив после его считывания. Это необходимо, поскольку графики строят данные построчно, а они были сохранены в файле столбцами. Обратите внимание: если вы не транспонировали массив данных в последнем упражнении в файл, то нужно транспонировать его при чтении из файла.
4. Запустите ВП. Поскольку вы не задали путь к файлу, программа попросит вас ввести имя файла. Выберите файл, созданный в упражнении 9.3. ВП считает данные из файла и построит обе осциллограммы на графике.
5. Сохраните виртуальный прибор в директории MYWORK или в библиотеке виртуальных приборов под именем **Read File.vi**.

9.9. Итоги

LabVIEW содержит много функций для манипулирования строками. Эти функции можно найти в подпалитре **Строки** палитры **Функции**. С их помощью легко

определить длину строки, объединить две строки, стереть подстроку из строки, преобразовать строку в число (или наоборот) и выполнить многие другие операции.

Используя функции в подпалитре **Ввод/вывод файлов**, вы можете записать данные в файл или считать данные из файла. Функция **Сохранить символы в файл** сохраняет текстовую строку в файл. Функции **Считать символы из файла** и **Считать строки из файла** способны вновь ввести содержимое этого файла в LabVIEW. Если вы хотите сохранить массив чисел, примените функцию **Записать в файл табличного формата**. Вы можете считать эти данные и преобразовать их в числовой формат с помощью функции **Считать из файла табличного формата**.

Наши поздравления! Вы освоили основные методы работы в LabVIEW. Раздел 9.10 познакомит вас с довольно сложными функциями LabVIEW, которые упрощают создание программ.

9.10. Дополнительные упражнения

9.10.1. Упражнение 9.5: температуры и отсчет времени

Создайте ВП, который производит 50 измерений температуры внутри цикла – одно измерение через каждые 0,25 с – и строит их на развертке осциллограммы. Также он преобразовывает каждое значение в строку, а затем объединяет эту строку с символом табуляции, отсчетом времени и символом конца строки, после чего записывает все полученные данные в файл. Сохраните ВП под именем **Temperature Log.vi**.

Используйте константы **Табуляция** (Tab) и **Конец строки** (End of Line) из палитры **Строки**; функцию **Объединение строк** для объединения всех строк; функцию **Сохранить символы в файл** для сохранения данных. Вы можете записать данные в файл в виде одной строки за каждую итерацию, но будет значительно быстрее и эффективнее, если вы соберете все данные в одной большой строке, используя сдвиговые регистры и функцию **Объединение строк**, а затем сохраните их в файл. Ваш файл в текстовом редакторе должен выглядеть следующим образом:

```
78.9 11:34:38
79.0 11:34:39
79.0 11:34:50
```

9.10.2. Упражнение 9.6: работа с таблицей символов

Создайте ВП, который создавал бы двумерный массив (три строки и 100 столбцов) случайных чисел и записывал транспонированные данные в файл таблицы символов. Файл должен содержать заголовки для каждого столбца, как показано на рис. 9.33. В этом упражнении вам потребуются подпрограммы из палитр **Строки** и **Ввод/вывод файлов**. Сохраните ВП под именем **Spreadsheet Exercise.vi**.

	A	B	C
1	Waveform 1	Waveform 2	Waveform 3
2	0.668	0.601	0.04
3	0.164	0.884	0.695
4	0.799	0.827	0.685
5	0.775	0.1	0.008
6	0.723	0.264	0.464
7	0.253	0.179	0.145
8	0.749	0.227	0.036
9	0.745	0.133	0.347
10	0.063	0.02	0.358
11	0.308	0.566	0.392

Header

Рис. 9.33

Используйте функцию **Сохранить символы в файл** для написания заголовка, а затем функцию **Записать в файл табличного формата** (с подачей на ввод **применить к файлу** (append to file) постоянной ИСТИНА) для записи числовых данных в тот же файл.

Дополнительные сведения о LabVIEW

Поздравляем вас с завершением изучения первого раздела данной книги. Вы ознакомились с основными методами создания виртуального прибора – выбором элементов управления и индикаторов лицевой панели, соединением элементов на блок-диаграмме, использованием таких структур, как цикл по условию или структура варианта, и созданием простых, удобных виртуальных приборов. Теперь пришло время познакомиться с более мощными функциями и возможностями, предлагаемыми LabVIEW.

В главах 10–16 данной книги речь пойдет о многих оставшихся функциях палитр, причем внимание в основном будет сосредоточено на методиках и инструментах написания усовершенствованных программ. Вы также познакомитесь с основами получения данных в LabVIEW с использованием подключенных многофункциональных плат ввода/вывода. Вы не найдете в этом разделе каких-либо упражнений, поскольку весь материал ориентирован на *информирование* («О, я и не знал, что это можно сделать с помощью LabVIEW!»). Поскольку в LabVIEW есть огромное количество функций и особенностей, часть II послужит вам хорошим гидом для ознакомления с ними.

И наконец, последний совет: вы можете спокойно пропустить некоторые подразделы, не читая их, если они не представляют для вас никакого интереса.

Обзор

В этом разделе вы более углубленно познакомитесь с темами, затронутыми в главе 2: получение данных и управление приборами. LabVIEW дает возможность пользователям превратить компьютеры в виртуальные инструменты, собирающие данные из окружающего мира. Это является одной из главных причин, по которой люди используют LabVIEW. Мы кратко рассмотрим различные варианты получения и создания данных, включая последовательную коммуникацию, интерфейс КОП, съемные многофункциональные платы ввода/вывода. Вы узнаете о некоторых концепциях теории сигналов и типах оборудования, применяемых для создания измерительных систем.

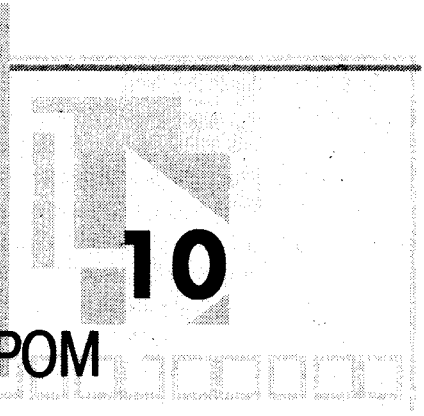
ЗАДАЧИ

- Наконец-то узнать значение тех сокращений, о которых все думают, что вы их знаете
- Познакомиться с опциями оборудования, которое вы будете использовать для получения или создания данных
- Изучить основы теории сигналов, включая классификацию сигналов, типы измерений, обработку сигналов и концепцию выборок
- Получить краткие указания по съему и установке многофункциональных плат ввода/вывода
- Познакомиться с интерфейсом КОП
- Проанализировать последовательную коммуникацию

ОСНОВНЫЕ ТЕРМИНЫ

- Сбор данных (DAQ)
- КОП (GPIB)
- Последовательный
- Сигналы
- Аналоговый
- Цифровой
- Частота
- Сигнал с общей «землей»
- Плавающий сигнал
- Опорная «земля»
- Частота выборки
- Теорема Найквиста
- Обработка сигналов
- MAX
- NI-DAQ
- Виртуальный канал
- Дифференциальная схема измерений
- Схема измерений с общим проводом
- Драйвер инструмента

ВВОД/ВЫВОД ДАННЫХ В КОМПЬЮТЕР: ПОЛУЧЕНИЕ ДАННЫХ И УПРАВЛЕНИЕ ПРИБОРОМ



10

10.1. Аббревиатура

«Давайте пойдем дальше и применим инструменты CASE и UML для создания интерфейса PCI, используя новый стандарт XML».

При обсуждении технических проблем часто употребляются аббревиатуры. Многие делают вид, что все понимают, и никто не осмеливается спросить, что означает то или иное сокращение, чтобы не прослыть неграмотным. В этой главе вы познакомитесь со значениями всех аббревиатур, которые применяются в данной книге.

AC: переменный ток. Эта аббревиатура первоначально указывала на источник питания прибора: переменный ток от сети (AC) и постоянный ток от батареи (DC). Теперь она используется в более широком плане, указывая на любой вид сигнала (не только ток), который быстро изменяется во времени.

ADC (АЦП) или **A/D**: преобразование аналогового сигнала в цифровой. Берется реальный аналоговый сигнал и преобразуется в цифровую форму (последовательность битов), которую компьютер может понять. Во многих случаях чип, используемый в этой операции, называется АЦП.

DAQ: сбор данных. Это сокращение вообще указывает на получение данных, обычно путем аналого-цифрового преобразования. Значение термина иногда включает и генерацию данных. Не путайте, пожалуйста, термины DAQ и DAC, которые звучат одинаково (DAC (ЦАП) – цифро-аналоговый преобразователь).

DC: постоянный ток. Термин, противоположный AC; далее в тексте обозначает ток. Иногда эта аббревиатура используется для обозначения постоянного сигнала. В других случаях DC указывает на сигнал с очень низкой частотой, например меняющийся реже одного раза в секунду. Вообще, граница, разделяющая области применимости данных терминов, субъективна.

DMA: прямой доступ к памяти. Вы можете использовать съемные платы ввода/вывода, которые имеют встроенный канал прямого доступа к памяти, или

приобрести отдельную плату прямого доступа к памяти. Этот механизм дает возможность хранить полученные данные прямо в оперативной памяти компьютера, увеличивая, таким образом, скорость передачи данных.

GPIB: канал общего пользования (КОП), менее известен как HB-IB и IEEE 488.2. Он стал мировым стандартом, обеспечивающим связь прибора с компьютером. КОП разработан компанией Hewlett Packard в 1960 году для программирования инструментов через персональный компьютер на языке Basic. Институт IEEE помог ввести строго регламентированные аппаратные протоколы, обеспечивающие единообразие работы с приборами.

IVI: взаимозаменяемые виртуальные приборы. Это стандарт для инструментальных драйверов (программное обеспечение, используемое для управления внешними устройствами), который может работать с весьма разнообразными инструментами.

MXI: интерфейс мультимедийного расширения. Является стандартом для соединения модуля VXI и обычного компьютера аналогично КОП.

PXI: расширение PCI для работы с инструментами. PCI – стандартная шина, используемая в большинстве компьютеров для подключения внешних устройств. PXI – обозначение открытой архитектуры аппаратной части, применяемой компанией National Instruments для интегрирования высокоэффективных модульных компонентов для сбора данных, управления приборами, обработки изображения и т.д.

RS-232: рекомендуемый стандарт № 232. Предложен Instrument Society of America для последовательной передачи данных. Используется как синоним термина «последовательная передача данных», хотя последний в большей мере относится к передаче одного бита в единицу времени. Другими стандартами, с которыми вы можете встретиться, являются RS-485, RS-422 и RS-423.

SCXI: модули расширения для преобразования сигнала для работы с оборудованием. Это высокоэффективная система формирования и преобразования сигнала, изобретенная National Instruments, которая использует внешний блок, содержащий модули ввода/вывода для обработки сигнала, его умножения и т.д. Блок соединен с платой ввода/вывода в персональном компьютере.

USB: универсальная последовательная шина. Является стандартной шиной во многих персональных компьютерах для подключения внешних периферийных устройств.

VISA: стандартная архитектура виртуального прибора. Это программная архитектура драйверов, разработанная National Instruments. Ее целью является попытка унифицировать стандарты программного обеспечения в случае использования приборов КОП, DAQ, VXI или RS-232.

VXI: расширение VME для работы с инструментами. Весьма эффективно для работы с приборами. Вы можете приобрести VXI-приборы, которые представляют собой маленькие модули (вместо обычного большого прибора с лицевой панелью), вставляющиеся в блок VXI. Блоки VXI часто имеет встроенную материнскую плату компьютера, поэтому нет необходимости использовать внешний персональный компьютер. VXI – это открытый промышленный стандарт, что означает поддержку различных компаний, включая National Instruments.

10.2. Как соединить компьютер с окружающим миром

У вас имеется хороший персональный компьютер и вам хочется опробовать LabVIEW, чтобы создать программу, которая выполнила бы что-нибудь за пределами компьютера. Возможно, вам захочется снять электроэнцефалограмму или построить кривые зависимости силы от смещения при проверке прочности нового пластика. Или вам понадобится нечто более сложное, как, например, управление процессом производства полупроводников, и нужно будет подавать управляющие сигналы на предприятие.

Вне зависимости от применений необходимо найти способ поступления данных в компьютер. Обычно для этого имеется несколько возможностей, но лучшее решение, несомненно, зависит от того, какие усилия вы при этом приложите и какую пользу можете получить от реализации той или иной программы. Прежде чем приобрести аппаратную часть, проанализируйте, какой вид сигналов вы хотите измерить (в некоторых случаях, таких как последовательная передача данных, вам не понадобится дополнительная аппаратная часть). Рис. 10.1 показывает наиболее общие способы поступления данных в ваш компьютер.

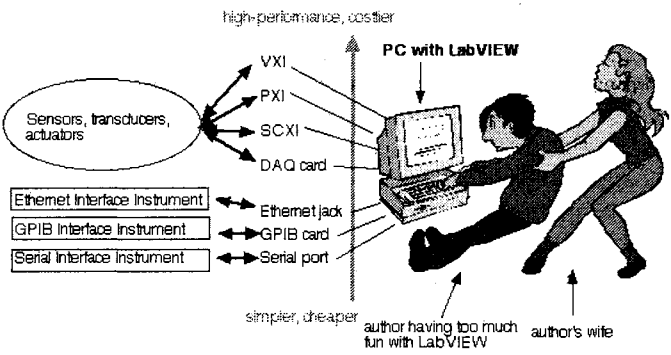


Рис. 10.1

Первое, что вы должны решить во время создания системы сбора данных, – нужно ли использовать традиционные внешние приборы, такие как мультиметр. Хотите ли вы применить физический прибор для сбора и обработки данных или создадите виртуальный прибор LabVIEW, который выполнит все операции посредством встроенной платы сбора данных?

Такие понятия, как стоимость, работа по расписанию и гибкость, также будут играть роль при принятии решения. Например, если нужно считать некоторые данные, определяемые электрическим сигналом с низким напряжением, воспользуйтесь простой платой ввода/вывода. Вы можете написать программу специально для данного приложения, которая создаст виртуальный прибор, точно

подходящий для этой цели. В общем, вам будет намного дешевле приобрести встраиваемую плату ввода/вывода, чем отдельный инструмент. Однако если у вас уже имеется инструмент (например, мультиметр) с КОП, то, наверное, лучше использовать его для получения данных о напряжении, а затем отправить их через КОП в компьютер.

Иногда вообще не возникает вопроса о применении внешнего прибора. Если, например, требуется масс-спектроскопия, ни один спектрометр нельзя как плату ввода/вывода вставить в компьютер.

Еще одним фактором, который может повлиять на решение использовать внешний прибор, является наличие *драйвера инструмента*. Вопреки распространенному мнению драйвер не является «виртуальной» копией лицевой панели прибора в LabVIEW. На самом деле большинство драйверов выглядят довольно скучно. Драйвер прибора обычно состоит из набора подпрограмм, каждая из которых посылает команду или набор команд в инструмент. В этих подпрограммах записан низкоуровневый код, поэтому вы можете быстро скомпилировать приложение. National Instruments (NI) имеет базу данных драйверов для сотен известных приборов, которые используют КОП, RS-232 или VXI. Вы можете получить ее бесплатно у NI. Драйверы других приборов приобретаются у третьих лиц, например у консультантов по программному обеспечению или непосредственно у изготовителей.

И наконец, если вы планируете купить или уже купили встраиваемую плату ввода/вывода, используйте ее на все 100%. Большинство людей не осознают полностью потенциал, который имеет их компьютер со встроенной плата ввода/вывода. Хотите посмотреть на сигнал переменного тока? Не берите осциллограф. Взгляните на сигнал на экране, используя один из примеров ВП получения данных, поставляемых вместе с LabVIEW. С помощью одной платы вы можете создать столько виртуальных приборов, сколько необходимо. А когда придет время усовершенствовать вашу плату или перейти на другую платформу, то не нужно будет ничего менять в блок-диаграмме. Виртуальные приборы LabVIEW для получения данных *функционируют независимо от карты, с которой работает компьютер* (за малым исключением).

Остальная часть этой главы поделена на два раздела: сбор данных и управление прибором. Раздел, посвященный сбору данных, описывает концепции, связанные с аппаратной частью, такие как теория сигналов, тип аппаратных средств и инструкции по конфигурации. Вторая часть, посвященная управлению приборами, имеет дело с протоколами последовательной передачи данных и КОП.

10.3. Сигналы

Прежде чем полностью окунуться в изучение сбора данных, нам необходимо поговорить о том, *что* мы вообще собираемся получать. Сигнал – это просто физическая величина, чья амплитуда и изменение со временем (или какая-либо другая переменная) содержат информацию.

10.3.1. Временные параметры – самое главное

Хотя это может быть неочевидным, *время* обычно является решающим фактором практически во всех измерениях. Хотим ли мы увидеть, как изменяется во времени температура двигателя, как выглядит отфильтрованный аудиосигнал, или закрыть некоторые клапаны при достижении смесью газа своего оптимального соотношения, время во всех этих случаях является решающим фактором при сборе информации и управлении. Нам нужно знать не *что* происходит, а *когда* это происходит. Даже сигналы с постоянной амплитудой не являются установившимися. Если бы они не изменялись во времени, мы бы всегда знали их величину. Но тогда зачем их измерять?

Временные параметры сигнала необходимы при создании программ получения данных по двум важным причинам. Во-первых, вам нужно решить, какую установить частоту выборки или, по-другому, как часто компьютер будет осуществлять измерение. Во-вторых, необходимо распределить время процессора для выполнения других задач, таких как запись данных в файл или считывание их из файла.

Если требуется считывать данные один-два раза в секунду или реже и вам не нужна повышенная точность синхронизации между выборками, вы можете использовать функции LabVIEW для управления частотой выборок прямо из программы. Поместите функцию **Задержка** в ВП, который однократно считывает данные из канала, например в цикле. Для более точных измерений или измерений переменных сигналов необходимо дать возможность аппаратной части и низкоуровневому программному обеспечению установить частоту выборки на плате ввода/вывода. Мы поговорим об этом подробнее в главе 11.

10.3.2. Классификация сигналов

Предположим, что вам необходимо произвести какое-либо измерение. Чтобы преобразовать сигнал аппаратурой преобразования или измерить сигнал с помощью платы ввода/вывода, нужно вначале преобразовать сигнал в электрический – напряжение или ток. Такую задачу выполняют преобразователи (датчики). Например, если вы хотите измерить температуру, вы должны каким-то образом представить ее как напряжение, которое может считываться платой ввода/вывода. Существует большое разнообразие преобразователей температуры, использующих некоторые принципы термодинамики и физические свойства материалов для преобразования температуры в электрический сигнал.

Как только физическая величина представлена в форме электрического сигнала, вы можете измерять ее для получения полезной информации, передаваемой через один или более параметров: состояние, величину, скорость, форму и частотный спектр (рис. 10.2).

Строго говоря, все сигналы аналоговые и изменяются во времени. Однако для обсуждения методов измерения вы должны классифицировать данный сигнал. Классификацию принято производить по способу передачи информации. Существует пять видов сигналов. Прежде всего, сигналы могут быть *аналоговыми* или

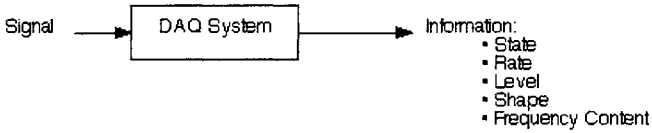


Рис. 10.2. Параметры сигнала

цифровыми. Цифровой (или двоичный) сигнал имеет лишь два возможных дискретных уровня – высокий и низкий. Аналоговый сигнал, наоборот, содержит информацию в непрерывно изменяющейся во времени амплитуде.

Специалисты часто сводят классификацию цифровых сигналов к двум видам, а аналоговых – к трем. Два вида цифровых сигналов – это сигнал перехода от высокого (on) к низкому (off) уровню (или наоборот) и сигнал в виде серии импульсов. Три вида аналоговых сигналов представлены постоянным сигналом, переменным сигналом во *временной области* (time domain) и переменным сигналом в *частотной области* (frequency domain) – рис. 10.3. Все эти виды сигналов по-своему уникальны в плане передачи информации. Пять видов сигнала соответствуют пяти основным видам информации, переносимой ими: состояние, скорость, уровень, форма и частотный спектр.

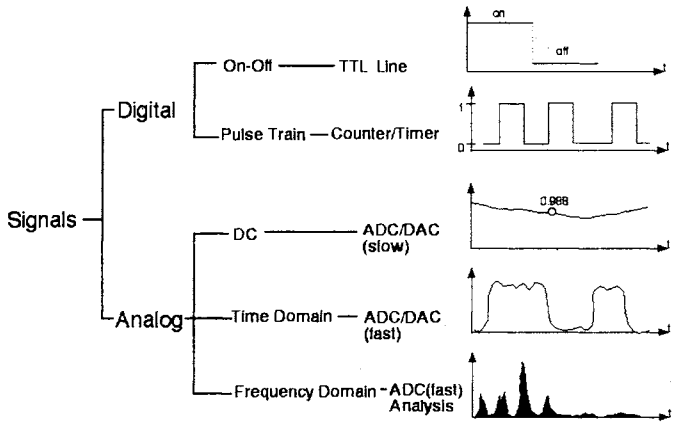


Рис. 10.3. Виды сигналов

Цифровые сигналы

Первым типом цифрового сигнала является сигнал on-off, или сигнал *состояния* (state), который передает информацию о цифровом уровне. Таким образом, необходимым прибором для измерения этого типа сигнала служит простой цифровой детектор. Выход транзисторно-транзисторной логической (TTL) схемы можно рассматривать как пример цифрового сигнала состояния. Другим примером является работа светодиода, как показано на рис. 10.4.

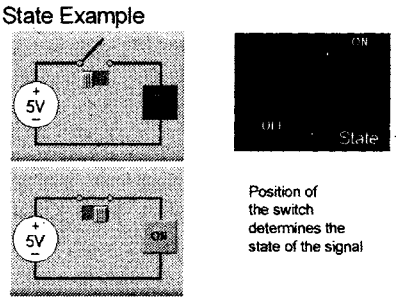


Рис. 10.4. Пример сигнала состояния

Второй вид цифрового сигнала – это серия импульсов или сигнал, пропорциональный скорости (rate). Сигнал состоит из последовательности переходов из одного состояния в другое. Информация заключена в количестве переходов, скорости, с которой меняются состояния, или времени между одним или несколькими переходами из одного состояния в другое. Выходной сигнал оптического кодировщика, установленного на валу двигателя, является примером сигнала, состоящего из серии импульсов. В некоторых случаях для того, чтобы приборы работали, на их вход необходимо подать цифровой сигнал. Например, скорость вращения и положение ротора шагового двигателя управляется последовательностью цифровых импульсов.

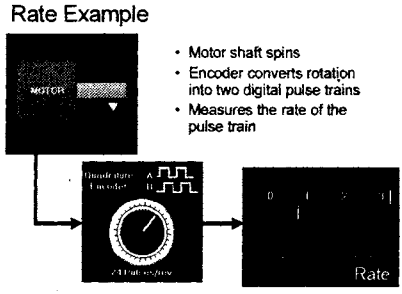


Рис. 10.5. Пример серии импульсов

Аналоговые уровневые сигналы

Аналоговыми уровневыми сигналами называются статические или медленно меняющиеся аналоговые сигналы. Наиболее важной характеристикой этого сигнала является уровень или амплитуда, которые несут информацию в данный момент времени. Поскольку аналоговый сигнал такого вида меняется медленно, то точность измеряемого уровня представляет больший интерес, чем время или скорость, при которой осуществляется измерение. Прибор или плата ввода/вывода, которая измеряет сигналы постоянного тока, действует как аналого-цифровой

преобразователь, который преобразует аналоговый электрический сигнал в цифровое значение, используемое в компьютере.

Как показано на рис. 10.6, уровневыми сигналами могут быть температура, напряжение батареи, давление и статические нагрузки. В каждом случае система сбора данных отслеживает сигнал и возвращает одно значение, отображающее его амплитуду в данный момент времени. Таким образом, для этих сигналов достаточно элементов отображения LabVIEW типа числовых индикаторов, манометров и разверток осциллограмм.

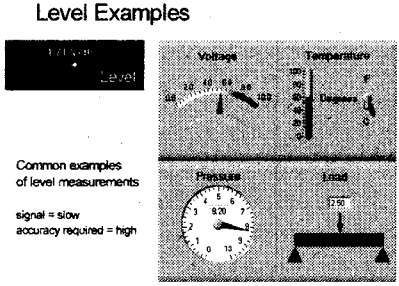


Рис. 10.6. Уровневые сигналы

Ваша система сбора данных должна обладать следующими харакиеристиками при подаче на нее аналоговых уровневых сигналов:

- высокой точностью/разрешающей способностью – для точного измерения уровня сигнала;
- полосой пропускания в нижней части спектра – для измерения сигнала при низкой частоте выборки (синхронизация программного обеспечения должна быть достаточной).

Переменные аналоговые сигналы во временной области

Аналоговые сигналы во временной области отличаются от других сигналов тем, что их полезная информация заключена не только в уровне сигнала, но и в изменении этого сигнала во времени. При измерении сигнала такого типа (его часто называют осциллограммой) интерес представляют такие характеристики его формы, как крутизна, местоположение и форма пиков и т.д.

Для измерения формы осциллограммы нужно использовать жестко синхронизированную по времени последовательность отдельных измерений амплитуд. Эти измерения должны быть сделаны с частотой, позволяющей адекватно воспроизвести форму осциллограммы. Кроме того, последовательность измерений надо начать в строго определенное время, чтобы гарантированно получить полезную часть сигнала. Таким образом, прибор или встроенная плата ввода/вывода, измеряющие сигналы подобного типа, должны состоять из аналого-цифрового

преобразователя, таймера и триггера. Таймер точно регистрирует преобразование аналогового сигнала в цифровой. При получении необходимого количества выборок триггер запускает измерение в установленное время в соответствии с некоторым внешним условием.

Существует неограниченное количество различных осциллограмм, некоторые из них показаны на рис. 10.7. Все они имеют одну общую особенность: форма осциллограммы (изменение уровня со временем) является самым интересным для нас элементом.

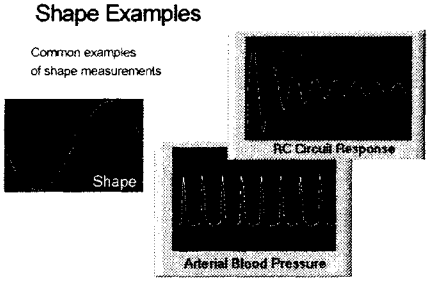


Рис. 10.7. Примеры сигналов различной формы

Система сбора данных, применяемая для считывания осциллограмм, должна иметь следующие особенности:

- более широкую полосу пропускания – для измерения сигнала при высокой частоте выборки;
- точный таймер – для измерения сигнала через точные интервалы времени;
- триггер – для старта измерений в точно определенное время.

Переменные аналоговые сигналы в частотной области

Аналоговые сигналы в частотной области похожи на осциллограммы, поскольку они также несут информацию о том, как сигналы изменяются во времени. Однако информация, извлекаемая из такого сигнала, содержится в его частотной составляющей в отличие от формы или изменяющейся во времени характеристики осциллограммы.

Так же как и при измерении осциллограммы, прибор, используемый для измерения частотного сигнала, должен включать в себя аналого-цифровой преобразователь, таймер и триггер для своевременного захвата осциллограммы. Кроме того, прибор должен уметь анализировать, чтобы выделить информацию о частоте сигнала. Вы можете осуществить цифровую обработку сигнала с помощью программного обеспечения или специальной аппаратной части цифрового сигнального процессора, созданного для быстрого и эффективного анализа сигнала.

Система сбора данных, применяемая для получения сигналов в частотной области, должна иметь:

- более широкую полосу пропускания – для измерения сигнала при высокой частоте выборки;
- точный таймер – для измерения сигнала через определенные интервалы времени;
- триггер – для начала измерений в определенное время;
- функции анализа – преобразование временной информации в частотную.

На рис. 10.8 показано несколько примеров сигналов в частотной области. Поскольку вы можете проанализировать любой сигнал в частотной области, некоторые сигналы и области применений очень удобны для такого анализа. Сюда относятся голосовые, акустические и геофизические сигналы, вибрация и т.д.

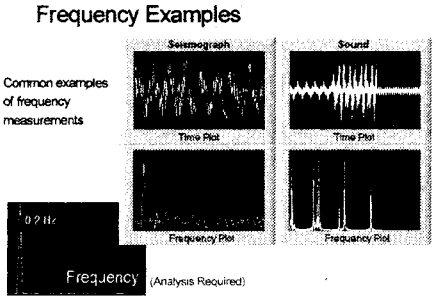


Рис. 10.8. Примеры частотных сигналов

Один сигнал – пять видов измерений

Пять видов сигналов, представленных в данном разделе, не являются взаимоисключающими. Определенный сигнал способен нести более одного вида информации. Таким образом, сигнал можно классифицировать по-разному и, следовательно, измерить несколькими способами. Для цифровых on-off сигналов, сигналов в виде серии импульсов и уровневых сигналов допустимы упрощенные методы измерений, так как они являются простыми вариантами аналоговых сигналов во временной области.

Вы можете измерить один и тот же сигнал с помощью различных систем, начиная от простой цифровой платы ввода/вывода и заканчивая сложной системой частотного анализа. Выбираемый метод измерения зависит от информации, которую вы хотите выделить из сигнала. На рис. 10.9 показано, как один сигнал – последовательность импульсов напряжения – может предоставить информацию, характерную для всех пяти типов сигналов.

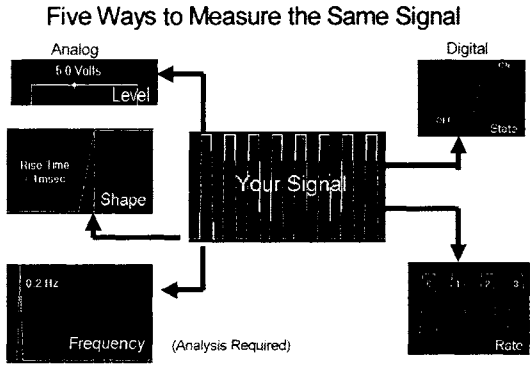


Рис. 10.9. Пять способов измерить один и тот же сигнал

Упражнение 10.1: задание по классификации сигналов

Классифицируйте следующие сигналы. В некоторых случаях сигнал может относится к нескольким видам. Выберите тип, который, по вашему мнению, больше соответствует сигналу. Обведите номер в левом поле в соответствии с обозначениями:

1: аналоговый уровневый сигнал
2: переменный аналоговый сигнал во временной области
3: цифровой сигнал состояния
4: цифровой импульсный сигнал
5: переменный аналоговый сигнал в частотной области

Таблица 10.1

1	2	3	4	5	Уровень напряжения батареи
1	2	3	4	5	Состояние реле
1	2	3	4	5	Сигналы на параллельном порте компьютера во время печати
1	2	3	4	5	Кратковременная импульсная помеха источника питания
1	2	3	4	5	Передаточная функция цифрового фильтра
1	2	3	4	5	Данные, поступающие через Internet
1	2	3	4	5	Относительная влажность
1	2	3	4	5	Число оборотов двигателя в минуту во время поездки по городу
1	2	3	4	5	Электроэнцефалограмма
1	2	3	4	5	Речевые сигналы через микрофон
1	2	3	4	5	Абсолютное давление в цилиндре двигателя

Преобразователи

Во время наладки системы сбора данных помните: все, что вы собираетесь измерить, должно превратиться в электрическое напряжение или ток. Способ, с помощью которого вы трансформируете измеряемые явления, такие как температура, сила, звук, свет и т.п., основан на использовании преобразователя. В табл. 10.2 приведены несколько устройств, применяемых для преобразования физических явлений в измеряемую величину.

Таблица 10.2

Явление	Вид преобразователя
Температура	Термопары, датчик температуры, термисторы, интегрированный в схему датчик
Свет	Вакуумные фотосенсоры, фотосопротивления
Звук	Микрофон
Сила и довление	Тензодатчики, пьезоэлектрические преоброзователи, датчик напряжений
Местоположение	Потенциометры, линейный датчик на основе дифференциального трансформатора, оптические кодировщики
Поток жидкости	Измерители нопора, расходомер роторного типа, ультразвуковой расходомер
pH	pH-электроды

10.3.3. Формирование и преобразование сигнала

Теперь, когда вы выяснили, какой вид сигналов хотите получить, можете подключить выход преобразователя прямо к плате ввода/вывода, правильно? Неправильно! Почти в 50% или более случаев ответ будет отрицательным: мы живем в мире, заполненном электрическим шумом.

К тому времени, когда сигнал достигнет платы сбора данных, он может обрасти шумом или характеризоваться другими негативными факторами, что сводит на нет все ваши усилия.

Вам придется осуществить некоторые виды преобразования аналоговых сигналов, представляющих физические явления. Что означает «преобразование сигнала»? Это определенные действия, производимые с вашим сигналом, с целью подготовки его для оцифровки в плате ввода/вывода. Сигнал должен поступить в плату с минимальным количеством шумовых помех в пределах напряжения (обычно от +5 до –5 В или 0 до 10 В) и тока (порядка 20 мА), ограниченных параметрами платы, и с достаточной точностью для данного вида измерений. Но, прежде всего, нужно определить, какой тип преобразователя требуется. Например, обработка аудиосигнала с микрофона может заключаться лишь в надежном заземлении системы и, возможно, в использовании фильтра нижних частот. Однако, если вы

хотите измерить уровни ионизации в плазменной камере при напряжении 800 В и при этом не спалить компьютер, вам необходимо использовать более сложную электрическую схему, которая включала бы внешние усилители с понижающим коэффициентом усиления.

Для сигналов, которые нуждаются в специальной обработке, или для установок, которые генерируют очень много сигналов, в National Instruments разработали систему SCXI. Там, где необходимо создать сложную систему в соответствии с требованиями, очень удобно использовать блок SCXI и подключаемые модули. В качестве модулей выступают аналоговые мультиплексоры, аналоговые платы вывода, макеты электронных схем, модули обработки сигнала для термопар и т.п. Помните, что не обязательно задействовать эту систему для преобразования сигнала.

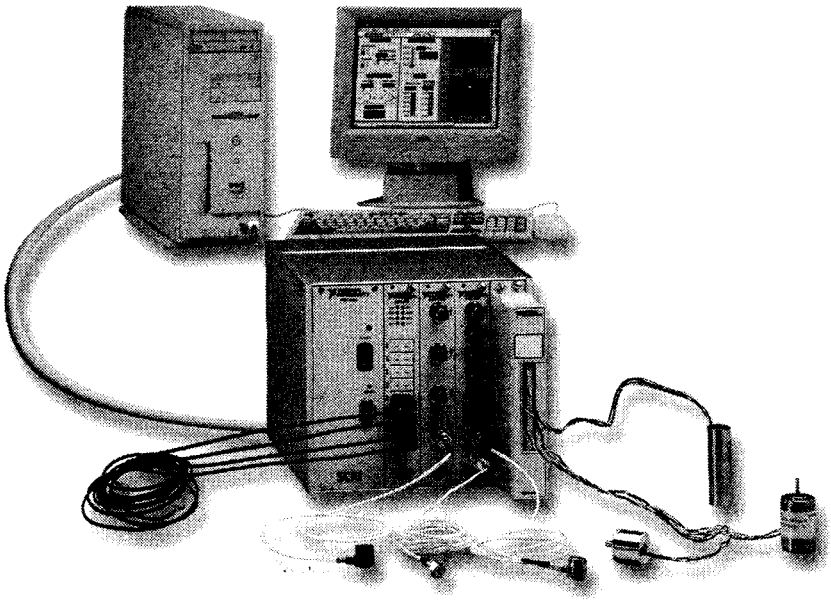


Рис. 10.10. Шасси SCXI с несколькими модулями обработки сигналов, присоединенное к компьютеру с LabVIEW

Наиболее распространенными типами обработки сигнала являются следующие:

- усиление;
- управление преобразователем;
- линеаризация;
- развязка и изоляция электрических цепей;
- фильтрация.

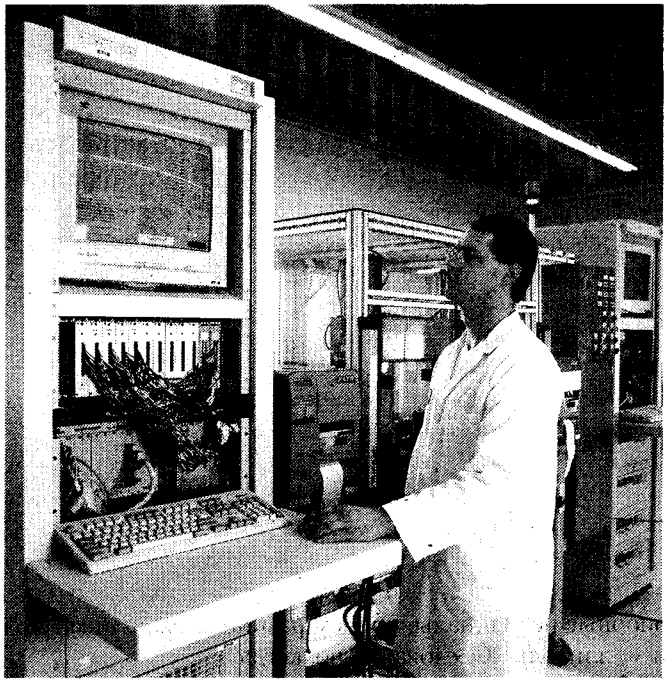


Рис. 10.11. Системы SCXI часто применяются в измерительных приборах, предназначенных для установки в стойке и использующих многоканальные приложения, как показано на этой фотографии

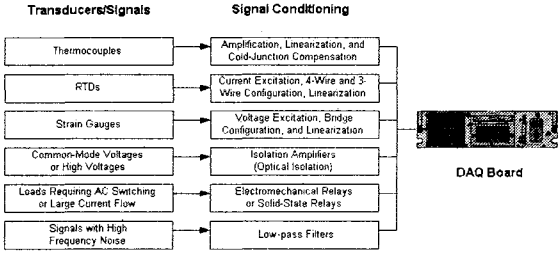


Рис. 10.12

10.3.4. Проблема заземления

Некоторые физические величины являются абсолютными: яркость света или масса (я знаю, что физики с этим не согласятся – вещи ведут себя странно, когда вы

приближаетесь к скорости света, и поэтому почти ничто не является абсолютным, за исключением самой скорости света). Но в любом случае напряжение не является абсолютным. Чтобы иметь значение, оно всегда требует опорного напряжения. Напряжение является мерой разности потенциалов между двумя телами. Одно из этих тел обычно выбирается в качестве опорного, и считается, что его потенциал имеет значение 0 В. Таким образом, разговор о сигнале напряжением 3,47 В не значит ничего до тех пор, пока мы не узнаем, относительно чего оно измерялось. Если вы заметили, часто об опорном напряжении не говорят. Это потому, что напряжение 0 В принято называть «землей». Именно здесь причина путаницы, так как понятие «земля» используется в разных контекстах для указания на разные опорные потенциалы.

Заземление (Earth ground) указывает на потенциал земли под вашими ногами. Большинство электрических штепсельных розеток имеет контакт, который заземлен и соединен с электрической системой для обеспечения безопасности. Многие приборы также являются заземленными, поэтому часто употребляется термин *заземление системы*. Это заземление осуществляется с помощью третьего провода в электрических розетках. Основной причиной заземления подобного рода является обеспечение безопасности, а не то, что оно служит опорным потенциалом. В действительности вы можете с кем-нибудь поспорить, что два заземленных источника имеют разные опорные потенциалы, причем разница между ними может составить несколько сотен милливольт. Следовательно, когда нужно определить опорное напряжение, мы должны говорить не просто о заземлении.

Опорная «земля» (Reference ground, иногда называется *общий провод*) часто создает интересующий нас опорный потенциал. При этом обычное заземление может быть, а может и не быть. Дело в том, что многие инструменты, приборы и источники сигналов создают опорное напряжение (отрицательный контакт, общий контакт и т.д.) с помощью которого определяется значение измеряемого напряжения.

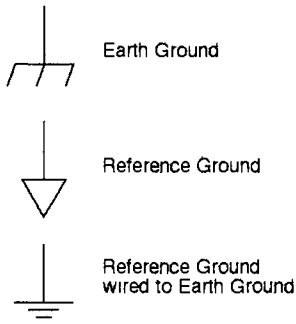


Рис. 10.13. Обозначение заземления

В схемах данной книги используются символы заземления, показанные на рис. 10.13. Однако знайте, что эти символы используются инженерами довольно непоследовательно.

Платы ввода/вывода в компьютере также предназначены для измерения напряжения относительно некоторого опорного напряжения. Его выбор зависит от типа источника сигнала. Сигналы могут быть классифицированы по двум широким категориям:

- заземленные (Grounded);
- «плавающие» (Floating).

Заземленный источник сигнала

Заземленным источником является такой, сигналы которого отсчитываются относительно заземления прибора, представляющего собой потенциал земли или здания. В этом случае источники имеют общее заземление с платой ввода/вывода. Наиболее распространенными примерами заземленных источников являются приборы, такие как генераторы сигналов и источники напряжения, которые заземляются через систему электропитания здания (рис. 10.14).

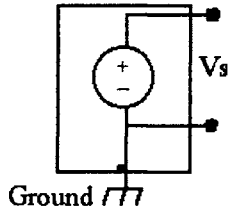


Рис. 10.14

«Плавающий» источник сигнала

«Плавающим» источником является такой, в котором напряжение не соотносится с общим заземлением, создаваемым землей или зданием. Примерами таких источников являются аккумуляторы, термопары, трансформаторы и отдельные усилители. На рис. 10.15 показано, что ни один выходной терминал источника не подключен к заземляющему проводу электрической розетки. Таким образом, ни один терминал не зависит от заземления.

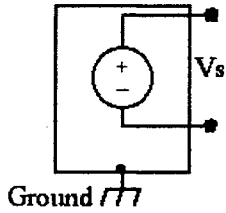


Рис. 10.15

10.3.5. Схемы измерений

Для измерения сигнала вы почти всегда можете сконфигурировать плату ввода/вывода, чтобы измерение подпадало под одну из следующих схем:

- дифференциальная (Differential);
- с общим заземленным проводом (Referenced single-ended – RSE);
- с общим незаземленным проводом (Nonreferenced single-ended – NRSE).

Дифференциальная схема измерений

В дифференциальной (независимой) схеме измерений ни один вход не соединен с фиксированным заземлением, таким как земля или здание. Большинство плат ввода/вывода с измерительными усилителями (специальный тип электрической схемы, в которой выходное напряжение относительно «земли» пропорционально разнице между напряжениями на ее двух входах) можно сконфигурировать в соответствии с дифференциальной схемой измерения. На рис. 10.16 изображена восьмиканальная дифференциальная схема, используемая в платах ввода/вывода серии E. Аналоговые мультиплексоры увеличивают число каналов измерения посредством одного усилителя. Контакт этой платы AIGND (заземление аналогового входа) является заземлением измерительной системы.

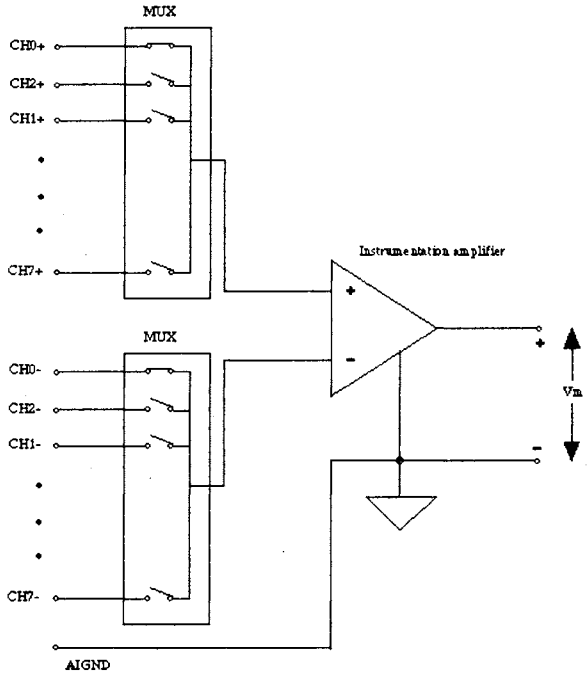


Рис. 10.16

Прежде чем мы будем говорить о схемах с общим проводом, следует отметить, что системы SCXI всегда используют дифференциальную схему измерения, в то время как встраиваемые платы ввода/вывода предоставляют вам определенный выбор в этом плане.

Для любознательных читателей

С помощью идеальной дифференциальной схемы можно измерить лишь разность потенциалов между двумя терминалами – входом (+) и входом (–). Поэтому любое напряжение на входах усилителя относительно его заземления, называемое

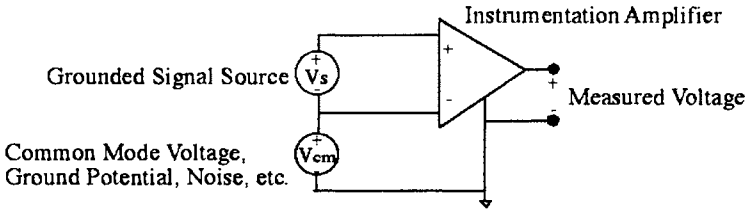


Рис. 10.17

синфазным, с помощью этой схемы измерить нельзя. Однако используемые на практике приборы частично устраняют эту особенность. Дело в том, что диапазон *синфазного* напряжения ограничивает допустимую амплитуду напряжения на каждом входе относительно заземления измерительной системы. Нарушение этого ограничения приводит не только к ошибке в измерениях, но и к возможному повреждению компонентов платы.

Вычислить и измерить синфазное напряжение V_{cm} относительно заземления платы ввода/вывода можно с помощью следующей формулы

$$V_{cm} = \frac{V^+ + V^-}{2},$$

где

V^+ – напряжение на *неинвертирующем* входе измерительной системы относительно заземления усилителя.

V^- – напряжение на *инвертирующем* входе измерительной системы относительно заземления усилителя.

Когда вы производите измерение с помощью платы ввода/вывода, синфазное напряжение не должно быть произвольно высоким. Встраиваемые платы имеют определенное максимальное рабочее напряжение, то есть максимальное синфазное напряжение, которое выдерживает плата, и измерения все еще остаются точными.

Схема с общим заземленным проводом

Схема измерений с общим заземленным проводом аналогична заземленному источнику сигнала, в котором измерения осуществляются относительно заземления. На рис. 10.18 показана 16-канальная измерительная система в схеме с общим заземленным проводом.

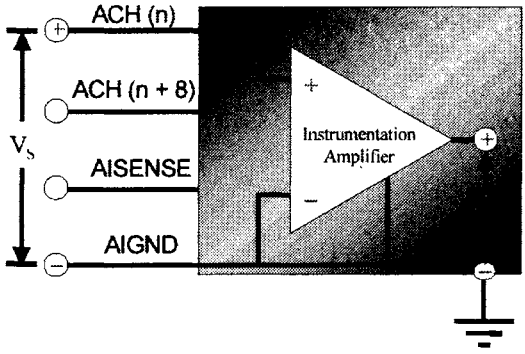


Рис. 10.18

Будьте осторожны при соединении источника тока или напряжения с платой ввода/вывода. Убедитесь, что напряжение источника сигнала не превышает максимально допустимого напряжения или тока, выдерживаемого платой. В противном случае вы можете повредить плату и компьютер.

Схема с общим незаземленным проводом

При работе с платами ввода/вывода часто используют вариант схемы измерения с общим проводом, известный как схема с *общим незаземленным проводом*. В этой схеме все измерения осуществляются относительно общего базового заземления, напряжение которого может меняться относительно заземления измерительной системы. На рис. 10.19 изображена схема с общим незаземленным проводом, на которой контакт AISENSE представляет собой общий провод (опорное заземление) при измерениях, а контакт AIGND осуществляет заземление системы.

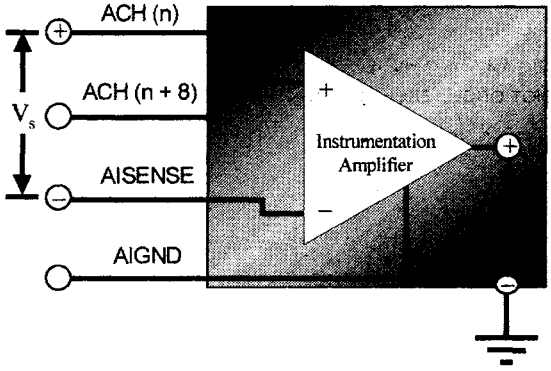
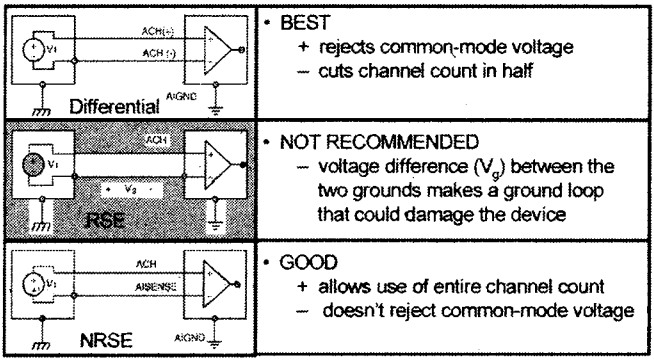


Рис. 10.19

Кстати, измерительная система определяется тем, как сконфигурирована плата ввода/вывода. Большинство плат компании National Instruments можно сконфигурировать для проведения измерения по всем трем схемам с помощью программы NI-MAX. Некоторые из устаревших плат также допустимо использовать, но только после перестановки их перемычек в определенное положение. Когда вы сконфигурировали плату ввода/вывода для определенного типа измерительной системы, все ее входные каналы будут подчиняться данной схеме измерений. Следует отметить, что вы не сможете изменить ситуацию при помощи LabVIEW, поэтому заранее решите, какую схему измерений вы собираетесь реализовать.

При решении проблемы, связанной с выбором схемы, всегда измеряйте сигналы заземленных источников с помощью дифференциальной схемы или схемы

с общим незаземленным проводом, а сигналы «плавающих» источников – с помощью схемы с общим заземленным проводом. В случае использования схемы с общим заземленным проводом при измерении сигнала от заземленных источников возникает опасность появления *паразитного контура с замыканием через «землю»*, что может явиться источником ошибки при измерениях. Точно так же применение дифференциальной схемы или схемы с общим незаземленным проводом для измерения сигнала «плавающего» источника наверняка будет невозможно из-за *токов смещения*, которые заставляют входное напряжение выходить за пределы входного диапазона платы ввода/вывода (хотя вы можете устранить эту проблему, установив резисторы смещения между входами и заземлением).



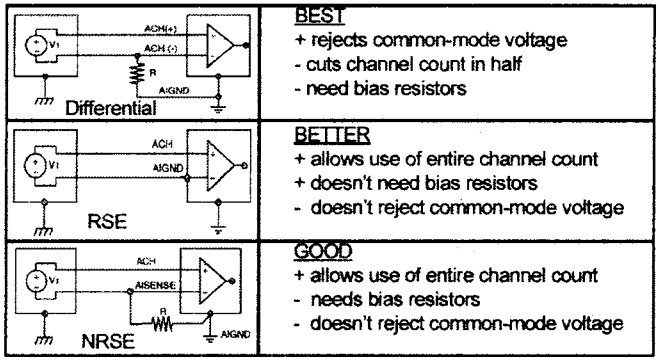


Рис. 10.20

10.3.6. Дискретизация, появление ложной частоты и мистер Найквист

Последней и, возможно, наиболее важной частью теории сигнала является процесс дискретизации.

Реальные сигналы, как правило, непрерывны. Для представления этих сигналов в компьютере плата ввода/вывода должна достаточно часто измерять уровень сигнала и присваивать ему дискретное число, которое компьютер будет принимать. Такой процесс называется аналого-цифровым преобразованием. Затем компьютер осуществляет что-то вроде «соединения точек» и выдает нечто похожее на реальный сигнал (вот почему мы говорим, что он *представляет* сигнал).

Частота дискретизации (sampling rate) системы отражает, насколько часто происходит аналого-цифровое преобразование. Каждый вертикальный штрих на рис. 10.21 представляет собой одно преобразование. Если система сбора данных делает одно преобразование в течение 0,5 с, мы говорим, что частота дискретизации составляет 2 выборки в секунду, или 2 Гц. В качестве альтернативы можно ввести период дискретизации, который равен обратному значению частоты (в данном примере 0,5 с). Оказывается, частота дискретизации сильно влияет на то, похож ли оцифрованный сигнал на реальный.

Если частота дискретизации недостаточно высока, возникает неприятное явление – появление ложной частоты (aliasing). Ее наличие всегда легко заметить (рис. 10.22).

В результате в ваших данных появляются высокочастотные компоненты, которые отсутствуют в реальном сигнале и таким образом искажают его. Ложную частоту удалить невозможно, вот почему так важно производить выборку (дискретизацию) с достаточно высокой скоростью.

Как определить частоту дискретизации? Один ученый по имени Найквист решил эту проблему, и его принцип, называемый теоремой Найквиста, звучит очень легко: во избежание появления ложной частоты частота дискретизации должна быть в два раза больше максимальной частотной компоненты в воспринимаемом сигнале.

Например, если известно, что сигнал, который вы измеряете, может изменяться 1000 раз в секунду (1000 Гц), следует выбрать частоту дискретизации большей 2 кГц.

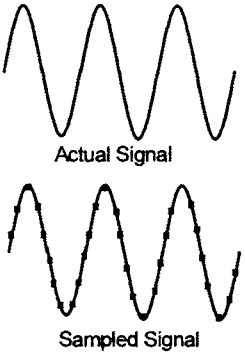


Рис. 10.21

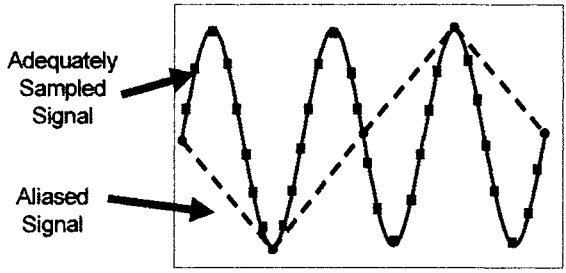


Рис. 10.22

Обратите внимание, что теорема Найквиста предполагает, что вам известна самая высокая частотная компонента сигнала. Если же она наперед не известна, пропустите сигнал через фильтр, чтобы избавиться от потенциальных высокочастотных компонент (об этом будет рассказано ниже).

Теорема Найквиста говорит только о безашибочном представлении частоты сигнала, но не о точном воспроизведении его формы. Если при дискретизации вам необходимо воспроизвести форму сигнала, то частота выборок должна превышать частоту Найквиста. В основном достаточно превышения в 5–10 раз от максимальной частотной компоненты аналогового сигнала.

Другой причиной необходимости знания частотного диапазона вашего сигнала является выбор *фильтров защиты от наложения спектров* (фильтров пропускания нижних частот). Во многих реальных сферах применения сигналы «обрастают» большим количеством высокочастотных шумов, которые сильно превышают теоретический частотный предел измеряемого сигнала. Например, обычным биомедицинским сигналом является электрокардиограмма – напряжение, отображающее работу сердца. Хотя эти сигналы редко имеют компоненты с частотой выше 250 Гц, электроды легко детектируют радиочастотный шум с диапазоном от 100 кГц и выше! Чтобы не делать дискретизацию на чрезмерно высоких частотах, системы сбора данных содержат фильтры нижних частот, которые не пропускают волны с частотой выше 250 Гц. В этом случае плата ввода/вывода сможет работать легче и делать выборки на частоте, скажем, 600 Гц.

Единственным случаем, когда частота дискретизации не имеет значения, является измерение сигналов постоянного тока, таких как температура или давление. Физическая природа этих сигналов такова, что они не могут изменяться быстрее, чем один или два раза в секунду. В этих случаях используется частота выборки, не превышающая 10 Гц.

10.3.7. И в заключение...

Мы рассмотрели большое количество материала, начиная от физических явлений до плат ввода/вывода. Если вы что-то не поняли, не отчаивайтесь. Теория сигналов является довольно сложной. Если у вас нет достаточного опыта в сфере электротехники, понадобится некоторая практика, прежде чем вы полностью освоите материал.

Вы познакомились с классификацией сигналов, типами преобразователей, важностью преобразования и обработки сигнала, различными схемами измерений для заземленных и «плавающих» источников сигналов и, наконец, с теорией дискретизации Найквиста. Может быть, вам следовало бы прослушать пару курсов по электротехнике, чтобы основательно изучить раздел, связанный со сбором данных и контрольно-измерительными приборами. Тем не менее полученных знаний будет вполне достаточно, чтобы начать процесс измерений.

10.4. Выбор и конфигурация измерительной аппаратной части систем сбора данных

10.4.1. Выбор аппаратной части

Если вы знаете, какой вид сигналов нужно измерить и/или воспроизвести, то настала пора выбрать встраиваемую плату ввода/вывода (при условии, что она будет удовлетворять вашим требованиям). Если вы собираетесь использовать LabVIEW, мы рекомендуем работать с платами, созданными компанией National Instruments. Хотя существует возможность применения LabVIEW совместно с платами других производителей, вы можете оказаться вовлеченными в длительный марафон низкоуровневого программирования. National Instruments предлагает широкий выбор всех видов плат с хорошим набором платформ, эксплуатационных качеств, функциональностью и диапазоном цен. Вы можете посмотреть весь каталог (http://ni.com), где под заголовком «DAQ Designer» найдете раздел, который поможет определить, какой тип аппаратной части больше подходит для ваших целей.

Чтобы подобрать наиболее оптимальную аппаратную часть, необходимо понять, каким требованиям должна отвечать создаваемая система. Наиболее существенным требованием является количество входов и выходов (I/O count). Следующий список вопросов может оказаться полезным при выборе платы:

- Какой тип операционной системы используется (Windows, Linux и т.д.)?
- Какой тип шины или разъема имеется в наличии (PCI, PC-card и т.д.)?
- Как много аналоговых входов необходимо? (Умножьте на 2, если требуются дифференциальные входы.)

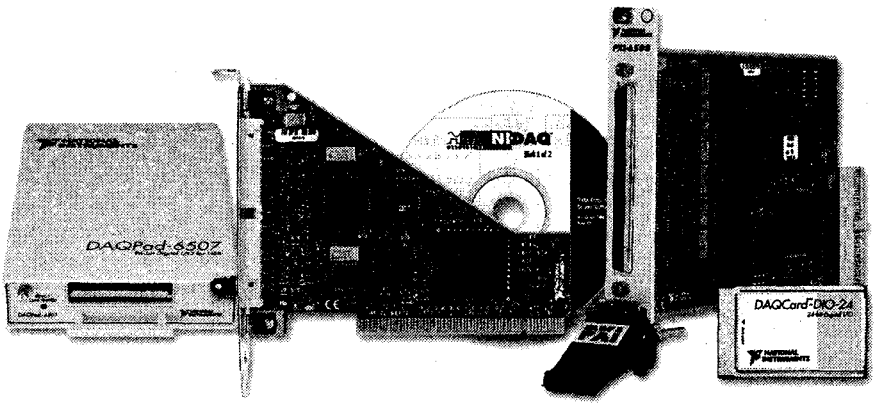


Рис. 10.23

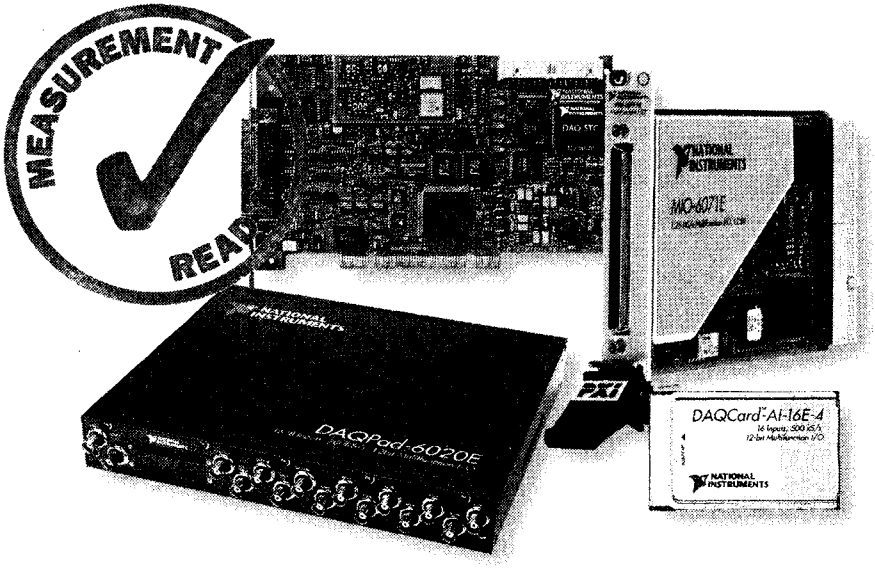


Рис. 10.24. Компания National Instruments производит большое количество плат сбора данных, совместимых с LabVIEW

- Как много аналоговых выходов нужно?
- Аналоговые входные сигналы являются напряжением, или током, или и тем и другим? Какие у них диапазоны?
- Как много цифровых входов и выходов необходимо?
- Нужны ли сигналы счетчика или синхронизации? Как много?
- Требует ли какой-либо аналоговый сигнал ввода/вывода (например, температура, давление, механическое напряжение) специальной обработки?
- Превысит ли какой-либо сигнал ввода/вывода допустимое напряжение ввода/вывода или ток в 20 мА?
- Какова минимальная частота дискретизации в одном канале?
- Какова минимальная частота сканирования по всем каналам?
- Какая точность или разрешающая способность (12 или 16 бит) необходима?
- Играет ли роль компактность, неприхотливость, цена? Если да, то каково компромиссное решение?
- Предусмотрел ли я будущие перспективы (расширение, новые системы сбора данных и т.д.)?

Теперь вы готовы выбрать плату ввода/вывода. Наиболее популярным типом плат компании National Instruments являются платы серии E, такие как PCI-MIO-16E-4, которые имеют 16 аналоговых входов, два аналоговых выхода, восемь цифровых каналов ввода и вывода и два счетчика. В зависимости от спецификации

эта плата содержит различные комбинации АЦП, ЦАП, линий цифрового ввода/вывода, схемы счетчика/таймера, что делает ее пригодной для широкой сферы использования. Разъемы внутри платы, такие как шина системной интеграции в реальном времени, передают сигналы таймера и сигналы запуска между платами, обеспечивая, таким образом, синхронизацию операций среди многочисленных устройств. Эта плата хорошо подходит для таких областей применения, где обычно требуется несколько аналоговых входов и один аналоговый выходной или цифровой сигнал. Если вам потребуется больше аналоговых входов, то поставьте плату NI-6704, которая имеет 32 аналоговых входа. Вы также можете приобрести платы таймера, быстрые платы цифрового ввода/вывода, плат для портативных компьютеров, внешние платы ввода/вывода – практически любую спецификацию, которую можно себе представить.

Аналоговый вход платы MIO состоит из аналого-цифрового преобразователя и *схемы аналогового ввода*. Последняя содержит аналоговый мультиплексор, измерительный усилитель и схему выборки и хранения.

Кроме элементов, изображенных на рис. 10.25, устаревшие платы ввода/вывода содержат несколько компонентов, например перемычки и переключатели, которые необходимы при их конфигурации. Эти компоненты, задающие такие параметры, как базовый адрес, прямой доступ к памяти (DMA) и уровни прерываний, будут рассматриваться в следующем разделе. Стандарт Plug&Play сделал их ненужными.

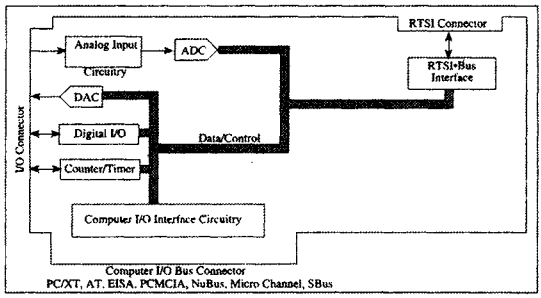


Рис. 10.25

Что делать, если вы озабочены ценой плат? Большинство встраиваемых плат стоит в пределах $1000–2000. Два фактора являются прямо пропорциональными цене платы: частота дискретизации и количество аналоговых каналов ввода/вывода. Таким образом, если вы измеряете постоянный ток, нет необходимости выкидывать лишние деньги на приобретение платы с частотой оцифровки в несколько мегагерц. Кроме того, обычно канал цифрового ввода/вывода недорог. Вообще же стоимость плат не должна быть решающим фактором, особенно если рассмотреть, что вы приобретаете. Сколько будут стоить осциллограф, или анализатор спектра, или сотня других приборов, если вы вынуждены купить их в качестве невиртуальных инструментов?

Одним из лучших решений вопроса о том, какая аппаратная часть вам необходима, является бесплатная программа National Instruments – DAQ Designer. Вы можете найти ее на сайте http://ni.com. Естественно, она может рекомендовать лишь собственные аппаратные продукты NI. Если вы хотите изучить более широкий диапазон опций, проконсультируйтесь с опытной компанией – системным интегратором (такой, как Rayodyne, http://www.rayodyne.com), которая посоветует лучший вариант аппаратной части.

И наконец, имейте в виду, что платы, изготовленные не National Instruments, будут работать с LabVIEW только в том случае, если производитель предоставит драйвер LabVIEW (или если вы напишете его сами).

10.5. Упражнение 10.2: анализ измерительной системы

Здесь представлена пара более сложных примеров, связанных с измерениями сигнала. Вашей задачей является определение необходимой информации.

Ответьте на следующие вопросы:

1. Какой вид сигналов необходимо измерить?
2. Какой тип измерения сигнала вы рекомендуете?
3. Какой должна быть частота дискретизации сигналов? Что такое теорема Найквиста?
4. Нужна ли обработка сигнала? Если да, то какого типа?
5. Какую аппаратную часть вы хотите использовать?

А. Профессор Гарри Фейс из биомедицинской лаборатории хочет получить сигналы сердца человека, не убивая их электрическим током. Он хочет снять электрокардиограммы двух субъектов одновременно. Каждый субъект имеет четыре электрода, подключенных к его телу в разных местах. Задачей является измерение в реальном времени потенциала между каждым из трех электродов; четвертый электрод играет роль заземления. Проводники, соединяющие электроды с системой сбора данных, не экранированы и не заземлены. Максимально информативными компонентами осциллограмм являются сегменты длительностью 2 мс. Сигналы генерируются в диапазоне 0,024 мВ.

Б. Мисс И. М. Эйнорд необходимо измерить изменение сопротивления гибкого материала под воздействием нагрузки и высокой температуры. Для этого у нее имеется специальная камера с механизмом, который выкручивает и вытягивает материал. Камера работает также в качестве печи с управляемой температурой. Мисс Эйнорд хочет увидеть в реальном времени, как изменяется сопротивление каждого из 48 проводов этого материала внутри камеры. Сопротивление измеряется путем подачи известного напряжения к каждому проводу и измерения тока. Температура в камере контролируется с помощью термопары. Механизм подачи нагрузки включается и выключается посредством реле. Наконец, количество циклов механизма подачи нагрузки тоже должно быть измерено.

Ответы смотрите в конце главы.

10.6. Установка плат

Все встраиваемые платы имеют *драйверы*, то есть низкоуровневый код, убеждающий ваш компьютер, что платы действительно находятся внутри него и готовы к работе. Хорошо, если драйверы установлены правильно и вам не нужно проверять это, чтобы использовать плату. Все платы компании National Instruments поставляются с драйверами, собранными в единый пакет. Этот продукт называется NI-DAQ. Фактически NI-DAQ инсталлируется по умолчанию в процессе установки LabVIEW. Поэтому, если у вас полная версия LabVIEW, все необходимые драйверы уже находятся в компьютере.

Между NI-DAQ и LabVIEW функционирует связывающая программа, называемая MAX (Measurement and Automation Explorer – программа анализа измерений и автоматизации) – рис. 10.26. MAX является программным интерфейсом Windows, который дает возможность доступа ко всем платам NI (будь то платы ввода/вывода, КОП, VXI и т.д.). MAX используется в основном для конфигурации и тестирования аппаратной части. Это необходимо сделать перед тем, как использовать данную плату в LabVIEW. MAX инсталлируется по умолчанию во время установки LabVIEW. После установки вы увидите иконку MAX на Рабочем столе Windows.

Рис. 10.26

Давайте поговорим об этой программе подробнее.

MAX – программа, доступная только в Windows. Для настройки встраиваемых плат в других ОС, таких как MacOS или Linux, используйте программу конфигурации NI-DAQ, поставляемую для соответствующей ОС.

На рис. 10.27 показана взаимосвязь между NI-DAQ, MAX и LabVIEW.

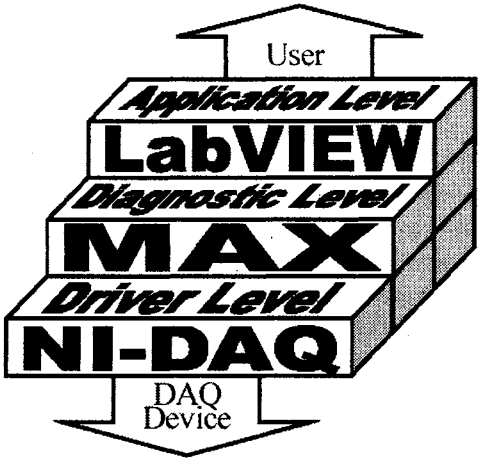


Рис. 10.27

10.6.1. Настройки аналогового канала ввода/вывода

Плата сбора данных имеет несколько параметров аналогового ввода/вывода, которые управляют действиями АЦП и ЦАП. Практически во всех платах вы можете настроить эти параметры при помощи программного обеспечения MAX (в устаревших платах необходимо использовать перемычки). К числу изменяемых параметров относятся диапазон, схема измерения, источник опорного напряжения и полярность (табл. 10.3).

Таблица 10.3. Изменяемые параметры плат E-серии

Диопазон входного напряжения АЦП	Однополярное от 0 до 10 В
	Биполярное ±5 В
	Биполярное ±10 В (по умолчанию)
Режим работы АЦП	С общим заземленным проводом
	С общим незоземленным проводом
	Дифференциальный (по умолчанию)
Источник опорного напряжения ЦАП	Внешний (по умолчанию)
	Внутренний
Полярность сигнала ЦАП	Однополярный – чисто двоичный режим
	Биполярный – режим дополнения до двух (по умолчанию)

10.6.2. Программа анализа измерений и автоматизации

MAX в основном используется для настройки и тестирования аппаратной части National Instruments. Но он также предлагает и другую функцию, такую как обновление до последней версии NI-DAQ.

Функциональность MAX подразделяется на четыре категории:

- окружение данных;
- устройства и интерфейсы;
- масштабы;
- программное обеспечение.

Рассмотрим каждую из этих категорий подробнее.

Окружение данных

Этот раздел показывает все настроенные *виртуальные каналы* и дает возможность их тестирования и перенастройки. Также через него запускается Мастер создания каналов (Channel Wizard), который позволяет создавать новые виртуальные каналы.

Виртуальный канал – это краткое обозначение настроенного канала системы. Вы начинаете создание канала, задавая ему тип и имя. Далее можете ввести описание канала, решить, какой тип преобразователя этот канал будет использовать, установить рабочий диапазон (определить усиление), выбрать вид заземления, использовать масштаб и дать ему описательное имя вместо номера. Позднее это имя вам пригодится при работе в LabVIEW для получения доступа к каналу и к его

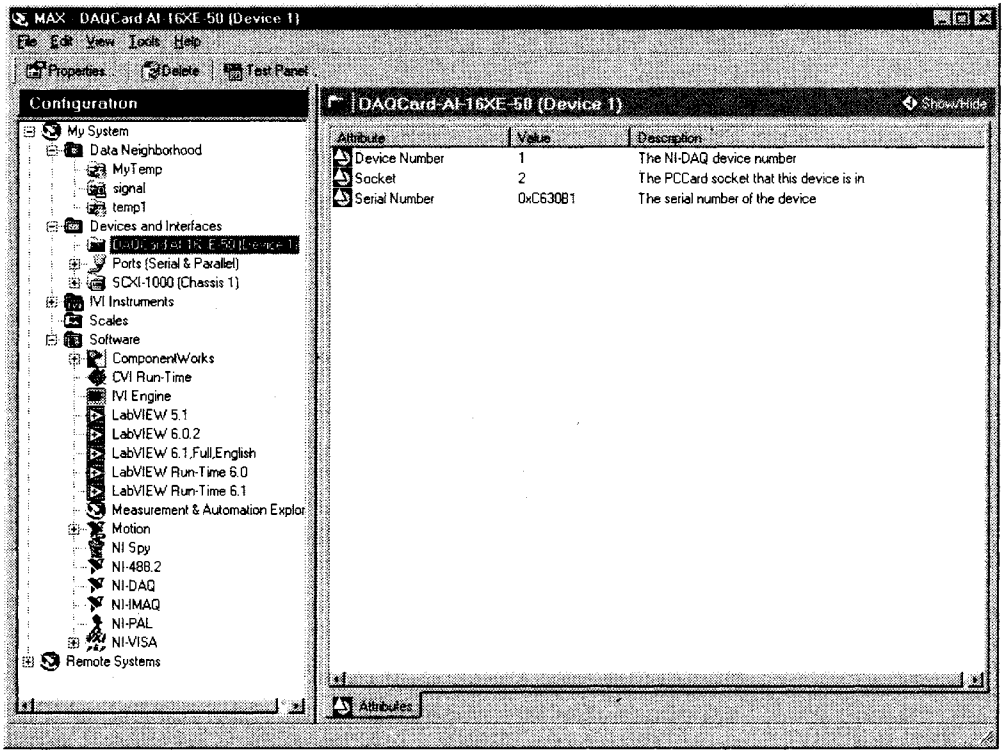


Рис. 10.28. Программа настройки MAX

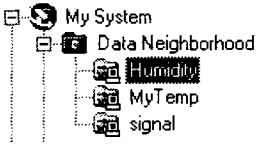


Рис. 10.29

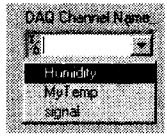


Рис. 10.30

информации о настройках. Например, если канал 0 в плате ввода/вывода был подключен к датчику температуры, вы можете создать виртуальный канал для канала 0 и назвать его **Датчик температуры** (Temperature Sensor). Допустимо создать виртуальные каналы для аналогового ввода, аналогового вывода и цифрового ввода/вывода. В этом случае обращение к каналу по имени (Temperature Sensor) вместо числа 0 поможет вам запомнить функцию канала.

Конечно, вы не обязаны использовать виртуальные каналы. В приборе LabVIEW разрешается обращаться к каналам по их номерам (0, 1, 2, ...). Но их первоначальная настройка в качестве виртуальных каналов удобна, так как (об этом рассказывается в следующей главе) вы можете использовать элементы управления с выпадающим

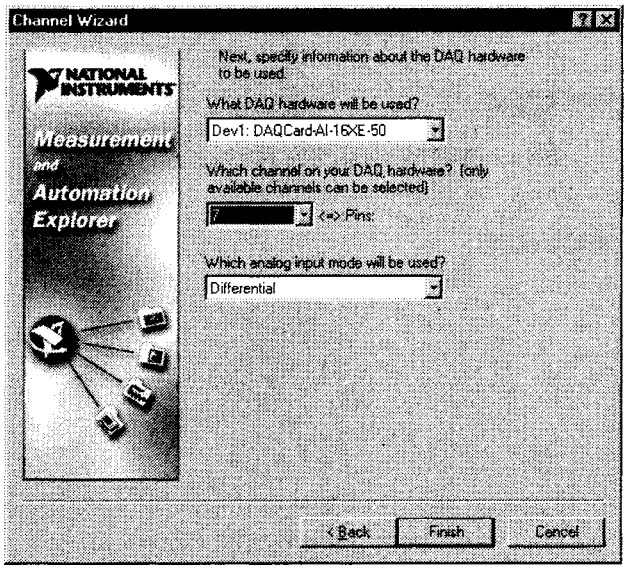


Рис. 10.31. Мастер создания канала

текстовым меню на лицевой панели (или блок-диаграмме), которые отображают имена всех существующих виртуальных каналов.

Для того чтобы создать виртуальный канал, щелкните правой кнопкой мыши по иконке **Окружение данных** (Data Neighborhood) в MAX и выберите опцию **Создать новый** (Create New). Появится Мастер создания канала (DAQ Channel Wizard), который проведет вас через все этапы создания виртуального канала.

Устройства и интерфейсы

Следующим разделом MAX является раздел **Устройства и интерфейсы**. В соответствии со своим названием эта категория показывает любые установленные и обнаруженные устройства National Instruments, такие как встраиваемые платы ввода/вывода, блоки SCXI и PXI, платы КОП и т.д. Вкладка также позволяет провести настройку и тестирование ваших приборов.

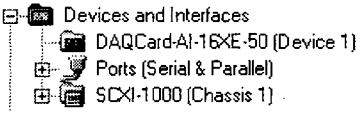


Рис. 10.32

Если вы щелкнете правой кнопкой мыши по установленной плате или прибору, в контекстном меню отобразятся следующие опции: **Свойства** (Properties), **Панели тестирования** (Test Panels) и **Удалить** (Delete).

В панели **Свойства** настраиваются платы ввода/вывода, как показано на рис. 10.33. В верхней части панели имеются закладки, которые вы можете использовать в процессе конфигурирования устройства:

- **Система** (System) позволяет изменить номер прибора и предлагает две кнопки для тестирования устройства сбора данных. Кнопка **Тестирование**

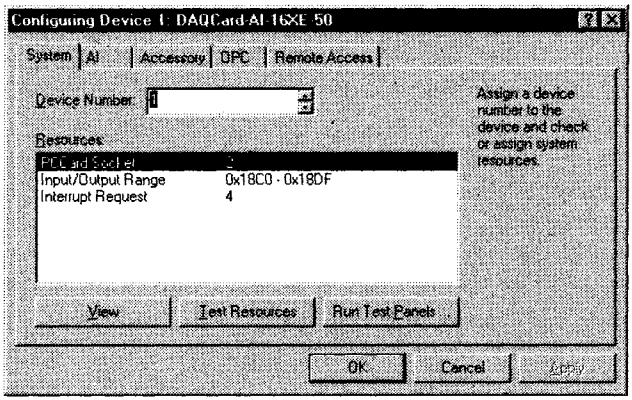


Рис. 10.33

ресурсов (Test Resources) позволяет произвести основное тестирование ресурсов системы, которые этот прибор занимает. К тестируемым ресурсам системы относится базовый адрес ввода/вывода, запрос на прерывание и прямой доступ к памяти:

- *базовый адрес ввода/вывода* (Base I/O Address). Устройство сбора данных взаимодействует с компьютером в основном через регистры. NI-DAQ записывает в настроечные регистры платы информацию для конфигурации устройства и считывает данные из регистров для определения статуса устройства или измерения сигнала. Настройки базового адреса ввода/вывода определяют, в какой области памяти, обслуживающей работу устройств ввода/вывода компьютера, хранятся регистры устройства;
- *запрос прерывания* (IRQ). Другой способ взаимодействия устройства сбора данных с компьютером – прерывания, которые дают процессору возможность быстро реагировать на периферийные устройства компьютера. Представить работу прерывания можно по аналогии с дверным звонком. Если у вашей двери нет звонка, вам придется периодически подходить к двери, чтобы узнать, есть ли кто за ней. Если имеется звонок, вам нужно подойти к двери лишь тогда, когда этот звонок звенит и вы уверены, что кто-то к вам пришел. В случае с устройством сбора данных процессору нет необходимости непрерывно проверять готовность данных к считыванию с устройства. Устройство сбора данных может использовать прерывание в качестве дверного звонка, который дает процессору сигнал о том, что в устройстве есть данные для считывания. Каждый запрос прерывания имеет свой номер. Например, устройству, изображенному в окне **Свойства**, было дано прерывание номер 4;
- *прямой доступ к памяти* (DMA) – третий способ взаимодействия устройства сбора данных с компьютером. Это метод передачи данных, при котором данные переносятся прямо с периферического

устройства в память компьютера, минуя процессор. Такой способ обычно требуется для достижения максимальной скорости передачи данных, что весьма удобно для высокоскоростных устройств сбора информации;

Большую часть времени вам не надо заботиться об установках базового адреса ввода/вывода, прерывания или канала прямого доступа к памяти, поскольку это автоматически делается Windows и NI-DAQ при установке платы в компьютер. Менять эти параметры необходимо только при наличии конфликтов устройств или других подобных проблемах.

- **Аналоговый вывод** (AO) позволяет настроить полярность аналогового выходного сигнала по умолчанию, а также задать источник опорного напряжения ЦАП;
- **Аналоговый ввод** (AI) позволяет настроить аналоговый входной сигнал, давая возможность определить тип сигнала, пределы изменения напряжения и масштаб при его отображении;
- **Дополнительно** (Accessory) позволяет настроить дополнительные возможности при работе с устройством сбора данных, например TBX-68 (блок со встроенной температурной компенсацией). Если NI-DAQ нет необходимости знать о дополнительных возможностях, то их не будет в перечне. В этом случае выберите опцию **None**;
- **OPC** позволяет установить период перекалибровки аналогового ввода, если вы используете сервер NI-DAQ OPC. *Open Process Control* (OPC – открытое управление процессом) – протокол, используемый в промышленной автоматизации. Если вы не знаете, как работает OPC, можете проигнорировать эту вкладку.

После того как устройство прошло тестирование ресурсов и вы настроили параметры вкладок **Система**, **AI**, **AO** и **OPC**, вернитесь к вкладке **Система** и щелкните по кнопке **Панель тестирования** (Test Panels). Появится окно, изображеное на рис. 10.34. Панель тестирования применяется для тестирования аналогового ввода, аналогового вывода, цифрового ввода/вывода и счетчика устройства сбора данных.

Панель тестирования широко используется для поиска и устранения неисправностей, так как она дает возможность проверить работоспособность устройства непосредственно через NI-DAQ. Если ваш прибор не работает в режиме тестирования, он не будет работать и в LabVIEW. Если у вас возникла необъяснимая проблема с получением данных в LabVIEW, воспользуйтесь кнопкой **Тестирование ресурсов** (Test Resources) и панелью тестирования, чтобы убедиться, что устройство функционирует нормально.

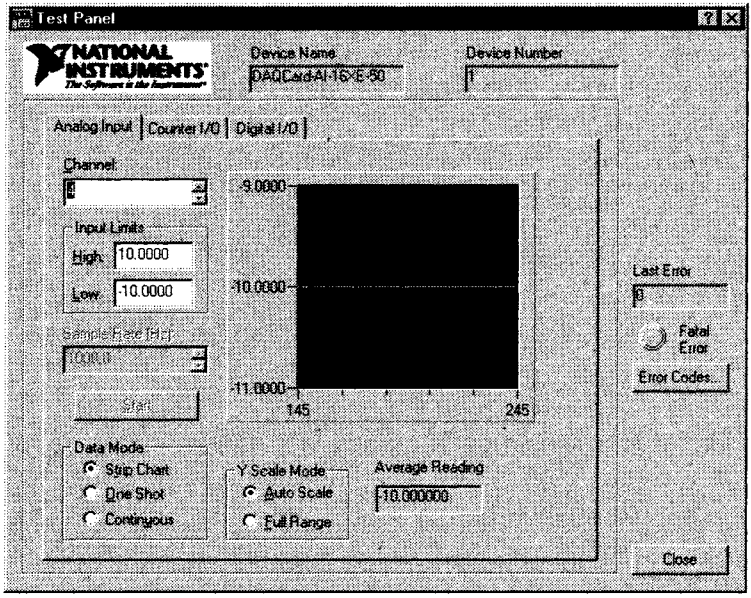


Рис. 10.34

Масштабы

Масштабы (Scales) являются следующей опцией в MAX. Здесь вы можете задать масштабы, которые будут использоваться в имеющихся виртуальных каналах. Эта опция иногда необходима для датчиков, которые не являются линейными. Также она полезна в случаях, когда надо напрямую измерять величину в действительных единицах измерения (например, температуру) вместо того, чтобы преобразовывать напряжение или ток к нужной единице. Задаваемый масштаб может быть одного из трех видов: линейного, полиномиального и табличного.

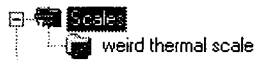


Рис. 10.35

Линейный (linear) – масштаб, определяемый формулой $y = mx + b$.

Полиномиальный (polynomial) – масштаб, определяемый формулой $y = a_0 + a_1x + a_2x^2 + ... + a_nx^n$.

Табличный (table) – масштаб, в котором вы вводите в строке значение и соответствующую масштабную величину в формате таблицы.

Программное обеспечение

Последним разделом программы анализа измерений и автоматизации является **Программное обеспечение** (Software). Этот раздел показывает все установленные

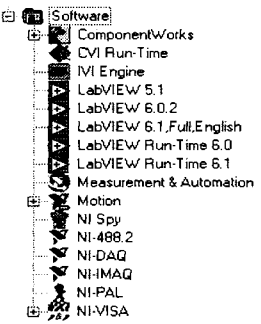


Рис. 10.36

версии программного обеспечения NI. Иконка каждого пакета программного обеспечения является одновременно кнопкой его быстрого запуска. Например, если вы щелкнули мышью по иконке LabVIEW, запустится программа LabVIEW. Категория **Программное обеспечение** также включает в себя Агент обновления программного обеспечения (Software Update Agent), который проверяет, является ли программное обеспечение последней версией, доступной в NI. Если это не последняя версия, Агент соединит вас со страницей на ni.com для загрузки последней версии ПО.

10.6.3. Платы ввода/вывода в MacOS и Linux

Если вы используете операционные системы MacOS, Linux, Solaris или HP-UX, то программа MAX, представленная выше, здесь неприменима (по крайней мере на момент написания книги), так как она функционирует только под Windows. Тем не менее NI обеспечивает драйверную поддержку большинства своих плат для работы под другими операционными системами, включая MacOS и Linux.

Пакет драйверов (NI-DAQ) особый для каждой операционной системы, и вам следует тщательно ознакомиться с информацией по установке и настройке плат.

10.7. Использование платы КОП

Плата, работающая с каналом общего пользования, применяется для управления и взаимодействия с одним или несколькими внешними измерительными приборами, имеющими этот интерфейс. КОП, изобретенный Hewlett Packard в 60-х годах, стал наиболее популярным стандартом взаимодействия с измерительными системами. Все приборы компании HP поддерживают КОП, так же как и тысячи приборов других компаний. КОП был модернизирован и стандартизирован IEEE, получив имя IEEE 488.2. Он обладает следующими особенностями:

- данные по каналу общего пользования передаются параллельно, то есть один байт (8 бит) в единицу времени;
- аппаратная часть отслеживает установление связи, синхронизацию и т.д.;
- по одной шине можно связать вместе несколько измерительных приборов (до 15);
- передача данных осуществляется быстро: 800 Кб/с и более.

Встраиваемые платы КОП есть почти для каждой платформы и шины, а также для внешних интерфейсов, преобразующих последовательный порт, параллельный порт или порт USB в канал общего пользования.

Другой характерной особенностью, которая делает КОП популярным, является то, что компьютер и измерительный прибор «говорят» друг с другом, используя

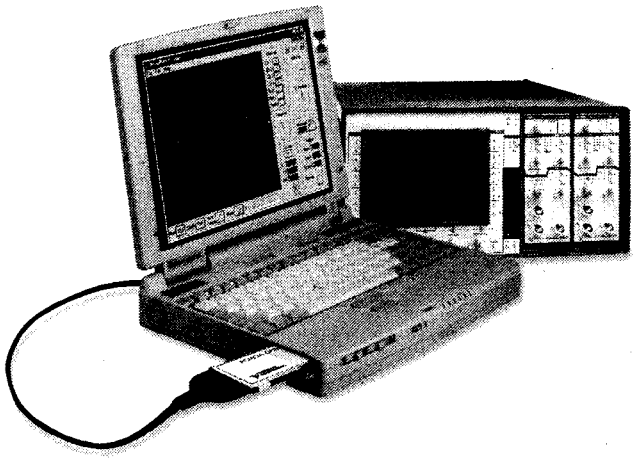


Рис. 10.37. Платы КОП обычно используются для взаимодействия с внешними приборами, такими как осциллограф

интуитивные ASCII-команды. Например, персональный компьютер, подключенный к цифровому авометру HP 3458A, способен передать что-то вроде:

```
PC: IDN? ; [Identity? - Кто вы?]
HP: HP3458A
PC: RMEM 1; [Возвратить содержание регистра памяти 1.]
HP: +4.23789
```

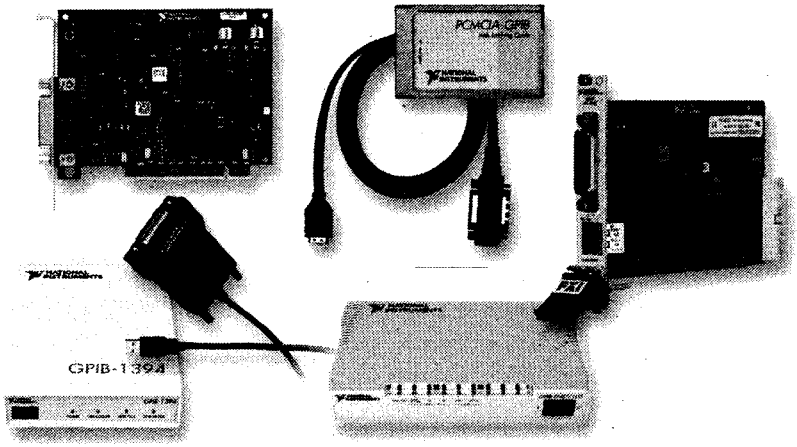


Рис. 10.38. Контроллеры КОП поставляются с интерфейсами на любой вкус: PCI, PC Card, Ethernet, USB, PXI и т.д.

Большое количество драйверов КОП, которые можно получить бесплатно, делают взаимодействие с внешним прибором таким же простым, как и написание нескольких подпрограмм виртуального прибора, создающего все необходимые ASCII-команды.

Установка платы КОП является довольно простой операцией. Платы КОП, изготовленные компанией NI, интегрируются с NI-MAX, поэтому вы легко их настроите с помощью раздела **Устройства и интерфейсы**. А лучше всего воспользоваться инструкцией по эксплуатации и установке платы КОП, которая к ней прилагается.

10.8. Подготовка к последовательной коммуникации

У последовательной передачи данных есть одно преимущество: она является простой и дешевой. Вам не нужна какая-либо дополнительная аппаратная часть, поскольку большинство компьютеров имеют, по крайней мере, один последовательный порт. Однако это не означает легкость и правильность подключения приборов. Хотя такие известные стандарты, как RS-232 и RS-485, определяют кабельное соединение, разъем, синхронизацию и т.п., изготовители подобных устройств довольно часто игнорируют эти стандарты. Если вы только что соединили кабелем последовательный порт компьютера и измерительный прибор, то у вас только один шанс из трех, что система будет работать нормально.

Хотя RS-232 постепенно заменяется более новым протоколом типа USB, до сих пор выпускается большое количество приборов, работающих в стандарте RS-232/485. Если вы хотите их правильно использовать, необходимо познакомиться с приборами и их параметрами, такими как скорость передачи сигнала, четность, стоповые биты и т.п. Вам также следует знать, для чего существуют различные контакты разъема последовательного порта.

Во многих случаях используются только некоторые линии. Вы можете столкнуться со следующими:

- *Transmit (TxD)* пересылает данные из персонального компьютера в прибор;
- *Receive (RxD)* пересылает данные из прибора в компьютер;
- *Ground (GND)* – заземление. Никогда не забывайте его подключать;
- *Clear-to-Send (CTS)* – персональный компьютер использует эту линию для оповещения прибора о готовности принять данные;
- *Ready-to-Send (RTS)* – персональный компьютер использует эту линию для оповещения прибора о готовности отослать данные.

Если у вас возникают проблемы при подготовке устройства к последовательной передаче данных, попробуйте следующие действия:

- поменяйте линии передачи и приема данных. Для этого вы можете использовать специальный «безмодемный» кабель;
- проверьте скорость передачи данных, четность, стоповые биты и т.д. в вашем компьютере. Затем проверьте те же параметры в приборе. Если есть разница, они не будут взаимодействовать друг с другом;

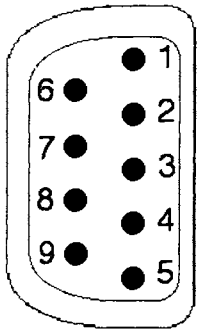


Рис. 10.39. Цоколевка последовательного порта (разъем DB-9). RxD (2): получение данных; TxD (3): передача данных; DTR (4): готовность терминала данных; CTS (8): сброс передатчика; GND (5): «земля» или общий провод; DSR (6): готовность набора данных; RTS (7): готовность к отправке; RLSD (1): обнаружение сигнала в линии приема; RI (9): индикатор вызова

- устройство с последовательной передачей данных всегда имеет источник питания. Проверьте, включен ли он;
- убедитесь в правильности соединения каналов CTS и RTS;
- убедитесь, что последовательный порт не применяется для другого вида операций;
- проверьте, используете ли вы необходимый вам последовательный порт, а не какой-либо другой;
- убедитесь, что вы посылаете правильные символы окончания строки.

Наиболее удобным способом выяснения правильности настройки компьютера для последовательной передачи данных является использование второго персонального компьютера и соединение их последовательных портов. Затем с помощью программы, работающей с терминалами ввода/вывода на каждом компьютере, вы сможете проверить, появляются ли данные, которые вы печатаете на одном компьютере, на мониторе другого и наоборот.

10.9. Итоги

Уф! Эта глава была довольно сложной – если вы прочитали ее до конца, то заслужили отдых.

Мы рассмотрели теорию сигналов и основы сбора данных. Различные сигналы могут быть классифицированы в зависимости от целей измерения. Сигналы разделяют на следующие виды: аналоговые постоянного тока, аналоговые переменного тока, цифровые сигналы состояния, последовательности импульсов и частотные. Источники сигналов могут *заземляться* или работать в *«плавающем»* режиме. *Заземленные сигналы* обычно поступают от устройств, которые

соединены с заземлением здания. Примерами источников, работающих в «*плавающем*» режиме, являются многие виды датчиков, такие как термопары или акселерометры (измерители скорости). В зависимости от типа источника сигнала, характеристик сигнала и количества сигналов могут использоваться три типа измерительных схем: *дифференциальная*, с *общим заземленным* и *общим незаземленным проводом*. Категорически нельзя применять схему с общим заземленным проводом при измерении сигнала от заземленного источника. Частота дискретизации системы сбора данных (для сигналов переменного тока) является весьма важным фактором. В соответствии с *теоремой Найквиста* частота дискретизации должна, по крайней мере, в два раза превышать максимальную частотную составляющую измеряемого сигнала.

Следующий шаг – выбор платы или системы сбора данных. Существует большое количество плат ввода/вывода для различных платформ, сфер применения и бюджета. Системы SCXI и PXI, созданные NI, дают возможность работать с очень большим числом каналов, а также *обрабатывать сигнал*. Установить плату ввода/вывода теперь гораздо легче, чем раньше, но она до сих пор требует определенных знаний для настройки параметров с помощью программы NI-MAX. Набор драйверов для плат, изготовленных NI, называется NI-DAQ. Эта программа на функциональном уровне взаимодействует с MAX.

Наконец, мы кратко рассмотрели некоторые аспекты аппаратной части, призванной взаимодействовать с внешними приборами через *КОП* и интерфейс *последовательной передачи данных*. КОП – широко используемый стандарт для многих измерительных приборов. И при работе в LabVIEW вы практически всегда сможете найти драйвер для вашего прибора. Последовательная передача данных является дешевой и довольно простой, но на практике она связана с постоянным поиском и устранением неисправностей.

Если у вас есть желание побольше узнать о сборе данных при помощи LabVIEW, почитайте книгу Брюса Михуры «LabVIEW – система сбора данных и управления» (Bruce Mihura «LabVIEW for Data Acquisition», 2001, Prentice Hall).

10.10. Ответы к упражнениям

10.1. 1, 3, 4, 2, 5, 4, 1, 2, 2, 2, 1

10.2.

А.

1. Аналоговые сигналы переменного тока, маленькая амплитуда, «плавающий» режим.
2. Дифференциальный (из-за маленькой амплитуды относительно «земли»).
3. Частота Найквиста f_n = 1/(2 мс) = 500 Гц. Делайте дискретизацию на частоте более 1000 Гц, например 5 кГц.

4. Да. Необходимы усиление (маленькая амплитуда сигнала), изоляция (безопасность) и, возможно, фильтры нижних частот против шума дискретизации.
5. Подойдет любая плата серии E.

Б.

1. Аналоговый ввод постоянного тока (сопротивление, показания термопары), аналоговый вывод постоянного тока (возбуждение напряжения), цифровой вывод on-off (реле для включения механизма подачи нагрузки), цифровой вход счетчика (для счета циклов).
2. С общим заземленным проводом.
3. Все измерения постоянного тока. Частота в 10 Гц будет достаточной частотой дискретизации.
4. Да, преобразование посредством термопары.
5. Система SCXI, так как необходима специальная обработка сигнала.

Обзор

В этой главе мы изучим то, для чего LabVIEW чаще всего используется: сбор данных и управление приборами. Вы познакомитесь с некоторыми ВП из палитр ***Сбор данных*** *и* ***Связь с прибором****. Будут рассмотрены аналоговый ввод/вывод, цифровой ввод/вывод, управление приборами с КОП и последовательная передача данных. Мы пройдем через основные этапы, необходимые для начала сбора данных в LabVIEW, и покажем вам нужное направление в изучении сбора данных и управления приборами для дальнейшего самообразования.*

ЗАДАЧИ

- Исследовать палитру **Ввод/вывод** лицевой панели: тип данных осциллограммы, имена каналов сбора данных
- Познакомиться с основными ВП палитры **Сбор данных**
- Изучить последовательность использования виртуальных приборов, необходимых для простейших аналоговых и цифровых измерений
- Опробовать некоторые примеры и Мастера сбора данных
- Познакомиться с основными ВП палитры **Связь с прибором**
- Изучить ВП VISA для управления инструментами
- Проверить ряд примеров управления через КОП
- Получить общее представление о последовательной передаче данных

ОСНОВНЫЕ ТЕРМИНЫ

- Устройство
- Каналы
- Порт
- VISA
- Аналоговый ввод
- Аналоговый вывод
- Линия
- Мастер
- Буфер
- Запуск
- КОП
- IVI
- Осциллограмма выборок
- Последовательная передача данных
- Опрос
- Идентификатор задачи
- Осциллограмма

СБОР ДАННЫХ И УПРАВЛЕНИЕ ПРИБОРАМИ В LabVIEW

11

11.1. Определения, драйверы и приборы

Взгляните на палитры, изображенные на рис. 11.1 и 11.2. Виртуальные приборы этих палитр выполняют функции, отличные от любых других в LabVIEW. Они позволяют взаимодействовать, считывать, вводить, измерять, включать и выключать внешнее оборудование посредством плат ввода/вывода и КОП. Мы кратко остановимся на приборах LabVIEW, работающих с аналоговыми и цифровыми сигналами, последовательным портом, каналом общего пользования и универсальным средством для последовательной передачи данных VISA. Для более эффективного изучения вам вначале нужно понять роль интерфейса между LabVIEW и платами.

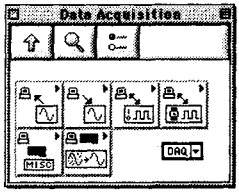


Рис. 11.1

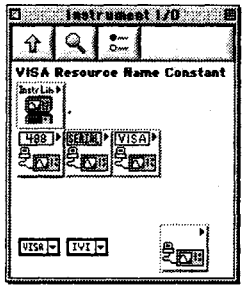


Рис. 11.2

Из рис. 11.3 видно, что инициализация операции по сбору данных вовлекает LabVIEW, вызывая NI-DAQ, который в свою очередь дает сигнал аппаратной части начать операцию ввода/вывода. В качестве промежуточного места для хранения поступающих данных платы ввода/вывода используют управляемые *буферы*

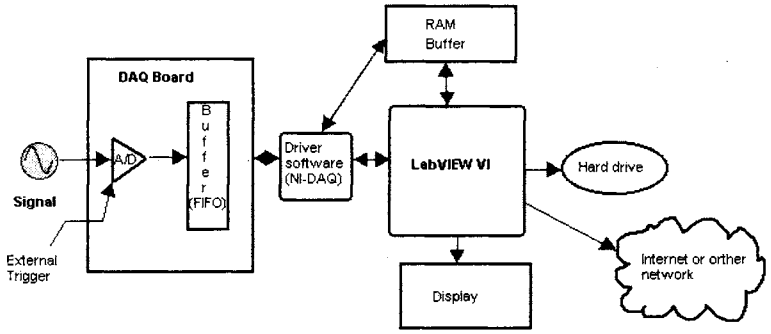


Рис. 11.3

плат (FIFO, First-In, First-Out – «первым вошел – первым вышел») и буферы оперативной памяти компьютера. Обратите внимание, что программное обеспечение не является единственным местом, где происходит инициализация операций ввода/вывода: внешняя часть оборудования также может запустить эту операцию.

Для классификации типа операции по сбору данных служат две важные характеристики:

- использование буфера;
- использование внешнего пускового устройства для запуска, остановки или синхронизации операции.

11.1.1. Буферы

Буфер, используемый в данном контексте, является областью памяти персонального компьютера (не FIFO-буфер карты), зарезервированной для временного хранения данных. Например, вы получили несколько тысяч выборок в течение одной секунды. Было бы затруднительно отобразить или построить все данные в течение такого короткого промежутка времени. Но, давая команду плате записать эти данные в буфер, вы сохраните их, а затем можете извлечь для отображения или анализа. Помните, что буферы зависят от скорости и объема сбора данных. Если плата ввода/вывода имеет прямой доступ к памяти, то операции аналогового ввода используют самый короткий маршрут от аппаратной части к оперативной памяти – прямую передачу данных в память компьютера.

Если вы не применяете буфер, то места для хранения нескольких точек данных нет. Это означает, что вы должны обрабатывать (строить на графике, сохранять на диске, анализировать и т.д.) *одну точку в единицу времени* по мере их поступления.

Используйте *буферизованный ввод/вывод* в следующих случаях:

- когда нужно считать или сгенерировать много выборок данных с частотой, превышающей действительно необходимую для отображения, либо

сохранить их на жестком диске или проанализировать в реальном времени;
- когда нужно непрерывно считывать или генерировать постоянный ток (со скоростью более 10 выборок в секунду) и иметь возможность анализировать или отображать часть данных «на лету»;
- когда период дискретизации должен быть точным и однородным во всех выборках.

Не используйте буфер ввода/вывода в следующих случаях:

- когда набор данных небольшой и короткий (например, получение одной точки с каждого из двух каналов в течение одной секунды);
- когда необходимо уменьшить загрузку памяти (буфер «забирает» память).

11.1.2. Запуск

Запуск (triggering) означает любой способ, с помощью которого вы запускаете, останавливаете или синхронизируете процесс сбора данных. Триггером обычно служит цифровой или аналоговый сигнал, состояние которого анализируется для определения следующего действия. Запуск при помощи программы является наиболее легким и интуитивно понятным: вы управляете триггером прямо из программы. Примером может служить логический элемент управления на лицевой панели, используемый для начала или остановки процесса сбора данных. Запуск с помощью оборудования дает возможность электрической схеме платы управлять пусковыми устройствами, вследствие чего увеличивается точность и синхронизация процессов сбора данных. Запуск этого вида можно разделить на *внешний* и *внутренний*. Пример внутреннего запуска – программирование платы на вывод цифрового импульса, когда напряжение в аналоговом канале достигает определенного уровня. Все платы ввода/вывода, изготовленные компанией National Instruments, имеют внешний контакт триггера, представляющий собой цифровой вход, который применяется для запуска. У многих приборов есть цифровой выход, служащий специально для запуска других приборов или инструментов, в данном случае платы ввода/вывода.

Используйте *программный запуск*, когда:

- пользователь нуждается в наличии простого и ясного управления операциями по сбору данных *и;*
- синхронизация явления (такого, как начало операции аналогового ввода) не обязательно должна быть очень точной.

Используйте запуск с помощью *аппаратной части*, когда:

- синхронизация процесса сбора данных должна быть точной;
- вы хотите упростить программную часть (например, цикл по условию, который наблюдает за появлением определенных данных, может быть удален);

- все процессы по сбору данных должны быть синхронизированы каким-либо внешним устройством.

В следующей главе вы научитесь использовать виртуальные приборы сбора данных для настройки операций ввода/вывода и типа запуска.

11.2. Аналоговый ввод/вывод

Все примеры, приведенные ниже, предполагают, что вы установили и настроили плату ввода/вывода. Если вам необходима помощь в этом, прочитайте главу 10 и документацию к плате. Примеры и упражнения также предполагают, что у вас есть источник сигнала, который вы хотите измерить.

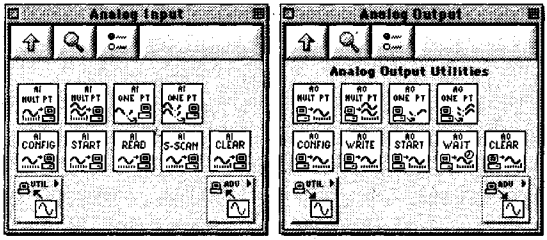


Рис. 11.4

Прежде чем подробнее познакомиться с виртуальными приборами сбора данных, необходимо ввести несколько определений. Следующие термины относятся соответственно к входам и выходам виртуальных приборов, поэтому они важны для понимания функции приборов. Для ясности на рис. 11.5 изображен один из ВП сбора данных с его входами и выходами.

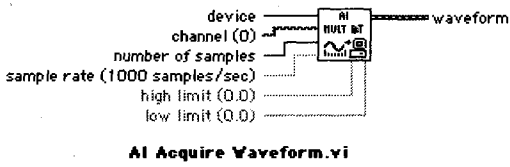


*Рис. 11.5. ВП **АЦП Получить осциллограмму***

Ввод **прибор** (device) является «номером прибора», который NI-DAQ присвоил плате. Вы можете выяснить, что это такое, ознакомившись со свойствами платы в настройках MAX. Данный параметр говорит LabVIEW, какой тип платы вы используете, сохраняя независимость самого ВП от типа платы (если позже вы

будете применять другую совместимую плату и присвоите ей тот же самый номер, то все ваши виртуальные приборы будут работать без изменения).

Выборка (sample) представляет одно преобразование АЦП. Это всего лишь одна точка – одно числовое значение, соответствующее реальному аналоговому сигналу во время проведения измерения.

Канал (channel) определяет физический источник выборки (выборок). Например, плата с 16 каналами аналогового ввода означает, что единовременно вы можете получить 16 наборов данных. В виртуальных приборах LabVIEW канал или набор каналов устанавливается с использованием типа данных **Имя канала сбора данных** (DAQ Channel Name). Этот тип данных очень похож на строковый. Элемент управления/отображения такого типа находится в палитре **Ввод/вывод**.

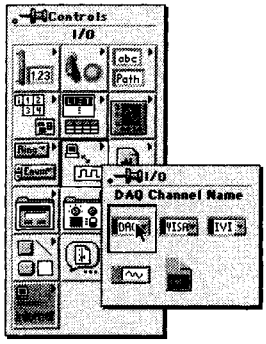


Рис. 11.6

Применение опции **Имя канала сбора данных** необходимо для определения каналов, в которых вы проводите измерения. Определить каналы можно несколькими способами:

- если вы сконфигурировали виртуальные каналы в MAX, то щелкните мышью по стрелке контекстного меню **Имя канала сбора данных**, чтобы увидеть список виртуальных каналов и выбрать соответствующий;
- можно непосредственно ввести имя виртуального канала или номер канала (либо диапазон номеров каналов).

На рис. 11.7 и 11.8 показаны различные способы представления имен каналов сбора данных и массив **Имя канала сбора данных**.

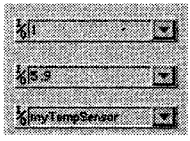


Рис. 11.7. Имя канала сбора данных

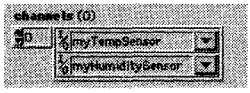


Рис. 11.8. Массив имен каналов сбора данных

Наиболее легким способом при получении данных является, прежде всего, определение виртуального канала в MAX, а затем выбор его с помощью элемента управления каналом сбора данных. Но если вы желаете напрямую ввести номера каналов, то легко можете это сделать, установив один, несколько или целый диапазон номеров каналов (табл. 11.1).

Таблица 11.1

Каналы	Строка имени канала сбора данных
5	5
0–4	0:4
1, 8 и 10–13	1, 8, 10:13

Опрос (scan) – это набор, состоящий из выборок, по одной из каждого используемого канала.

Осциллограмма (waveform) – набор выборок из *одного* канала, собранных в течение определенного промежутка времени и расположенных в порядке получения. Обычно (но не всегда) интервал времени между точками является постоянным для данной осциллограммы.

Новички часто смешивают понятия опроса и осциллограммы. Опрос – это зависимость выборок от *каналов* (одна выборка из каждого канала в единицу времени); осциллограмма представляет собой набор выборок (из одного и того же канала) в зависимости от *времени*.

Обратите внимание, что выход осциллограммы представляет собой *тип данных* осциллограммы. Мы уже говорили об осциллограммах в главе 8. Вы можете вновь обратиться к ней, если возникли затруднения с пониманием термина, поскольку практически все функции аналогового ввода/вывода используют этот тип данных.

Верхний предел (high limit) и **нижний предел** (low limit) – пределы напряжения, которые предполагаются в сигнале. Путем изменения этих значений относительно принятых по умолчанию (±10 В) вы можете установить коэффициент усиления системы сбора данных. Например, если вы задали пределы +5 и –5 В, то коэффициент усиления равен 2. Если пределы +1 и –1 В, то коэффициент усиления равен 10. Используйте эти опции, если уверены, что ожидаемый *диапазон* входного сигнала отличается от принятого по умолчанию. Для определения коэффициента усиления служит следующая формула:

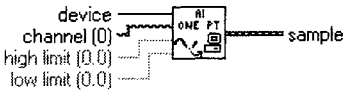
Коэффициент усиления = 20 / (верхний предел – нижний предел).

Имейте в виду, что многие платы ввода/вывода поддерживают лишь, заранее определенные величины коэффициентов усиления. Если вы укажете теоретический коэффициент усиления, который не воспринимается платой, то LabVIEW автоматически установит его на ближайшую заранее определенную величину. Обычная плата имеет следующие коэффициенты усиления: 0,5, 1, 2, 5, 10, 20, 50 и 100.

Идентификатор задачи (taskID) представляет собой 32-битовое целое число, которое используется некоторыми ВП сбора данных для идентификации выполняемой операции ввода/вывода. Многие ВП требуют **входного идентификатора** (taskID in) и возвращают **выходной идентификатор задачи** (taskID out). При подключении такого идентификатора вам не нужно снабжать каждый ВП информацией о плате, частоте выборки, пределах и т.д. Первый виртуальный прибор может передать всю информацию в виде идентификатора задачи для другого виртуального прибора. Если вы немного сконфужены, наберитесь немного терпения и посмотрите на примеры. В действительности это не так сложно.

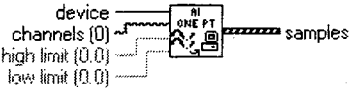
11.2.1. Простой аналоговый ввод/вывод: верхний уровень

Если вам не терпится опробовать плату ввода/вывода, попытайтесь для начала использовать какой-либо простой виртуальный прибор. Если даже вы перейдете к более сложному процессу сбора данных, то вам не помешает знание следующих примеров как средства тестирования платы. Все ВП верхнего уровня палитры **Аналоговый ввод** (Analog Input) и **Аналоговый вывод** (Analog Output) легки в применении. Их можно задействовать в качестве самостоятельных виртуальных приборов с минимальной настройкой и конфигурацией.



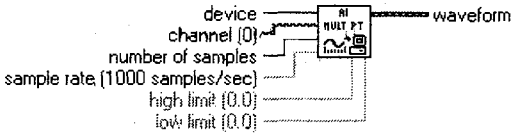
AI Sample Channel.vi

Рис. 11.9. ВП АЦП Выборка из канала



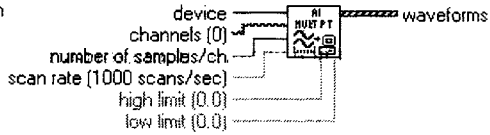
AI Sample Channels.vi

Рис. 11.10. ВП АЦП Выборка из каналов



AI Acquire Waveform.vi

Рис. 11.11. ВП АЦП Получить осциллограмму



AI Acquire Waveforms.vi

Рис. 11.12. ВП АЦП Получить осциллограммы

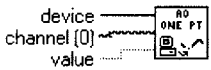
Эти ВП сбора данных могут осуществлять следующие операции:

1. Аналоговый ввод:
   - делать одну выборку из определенного канала;
   - делать по одной выборке из каждого канала (channel), поданного на его вход. Выборки затем возвращаются в массиве осциллограмм выборки (samples), упорядоченных по каналам;
   - получать одну осциллограмму (набор выборок за промежуток времени) из одного канала при определенной частоте выборки. Выборки возвращаются в виде осциллограммы (waveform);
   - получать осциллограммы из каждого канала, поданного на вход. Выборки возвращаются в массиве осциллограмм (waveforms), упорядоченных по каналам, в порядке поступления. Данные каждого канала сохраняются в компонентах отдельной осциллограммы.

Хотя по умолчанию выход приборов аналогового ввода данных представляет собой осциллограмму или массив осциллограмм, вы можете подключить к выходу скалярную величину (в случае получения одной выборки), одномерный

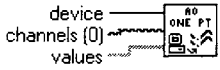
массив (в случае нескольких выборок) или двумерный массив (в случае осциллограмм). Это допустимо, поскольку выводы данных ВП являются полиморфными, то есть легко приспосабливаются к другим типам данных. Вы увидите, что проще работать с выходными данными в виде осциллограмм, как установлено по умолчанию.

2. Аналоговый вывод:
   - устанавливать напряжение в определенном выходном канале. Это напряжение остается постоянным до тех пор, пока его не изменят или прибор не будет включен повторно;
   - устанавливать величины напряжения в определенных выходных каналах. Эти величины напряжения остаются постоянными до тех пор, пока их не изменят или прибор не будет включен повторно;
   - генерировать осциллограмму в определенном выходном канале. Данные осциллограммы (в вольтах) должны иметься в наличии как компоненты входной осциллограммы. Число обновлений напряжения в секунду – частота обновления (update rate) – определяет время между точками (параметры $\mathbf{t_0}$ и **dt** осциллограммы игнорируются).



AO Update Channel.vi

Рис. 11.13. ВП ЦАП Обновить канал



AO Update Channels.vi

Рис. 11.14. ВП ЦАП Обновить каналы

AO Generate Waveform.vi

Рис. 11.15. ВП ЦАП Генерация осциллограммы

AO Generate Waveforms.vi

Рис. 11.16. ВП ЦАП Генерация осциллограмм

Такая функция, как **ЦАП Генерация осциллограммы**, может быть создана одновременно для каждого канала. В качестве входных данных функции **ЦАП Генерация осциллограммы** и **ЦАП Генерация осциллограмм** могут принять одномерный и двумерный массивы соответственно, следовательно, входы осциллограмм являются полиморфными.

Наконец, необходимо знать, что предыдущие ВП аналоговых ввода и вывода *синхронны* с вводом/выводом на плате сбора данных: виртуальный прибор будет продолжать выполняться до тех пор, пока все данные не будут считаны или записаны.

11.2.2. Упражнение 11.1: аналоговый ввод

1. Подключите источник напряжения типа функционального генератора (или батарейки с напряжением 1,5 В, если это все, что у вас имеется) к каналу 0 на плате ввода/вывода. Убедитесь, что плата сконфигурирована для дифференциальной схемы измерения или схемы с общим проводом.
2. Создайте лицевую панель и блок-диаграмму, как показано ниже. ВП **АЦП выборка из канала** (AI Sample Channel) находится в палитре **Аналоговый ввод**. Если ваша плата не является прибором № 1 в MAX, то измените константу устройства (device), чтобы она соответствовала его номеру.
3. Сохраните ваш виртуальный прибор как **Quick Analog In.vi**.

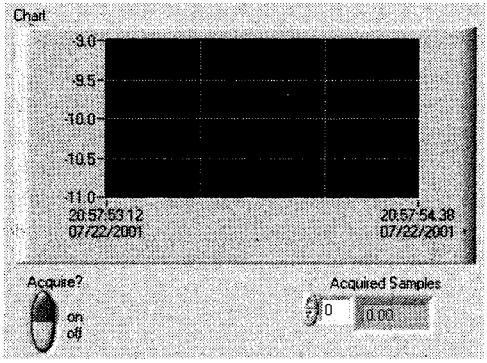


Рис. 11.17

4. Запустите ВП с включенным переключателем **Acquire**. Через несколько секунд выключите его.
5. Исследуйте данные в массиве полученных выборок (**Acquired Samples**).

Обратите внимание, что мы подключили массив осциллограмм к одномерному массиву **Acquired Samples**. LabVIEW автоматически извлекает компоненты **Y** из осциллограмм и размещает их в этом массиве.

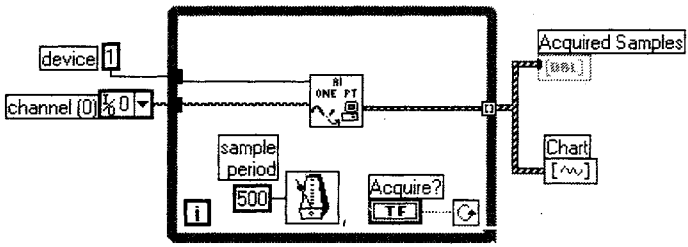


Рис. 11.18

Цикл по условию в упражнении 11.1 использовался для простоты: он не сильно загружает процессор. Теперь нужно бегло взглянуть на ваши данные. Но на большой блок-диаграмме вы вряд ли захотите, чтобы программа управляла «индивидуальным» получением каждой выборки. Для этого служат платы ввода/вывода и низкоуровневые драйверы.

Описаный метод сбора данных прекрасно работает при следующих условиях:

- частота выборки – низкая (раз в секунду или медленнее);
- отсутствуют другие действия операционной системы во время запуска ВП;
- небольшие изменения периода выборок являются допустимыми.

Предположим, вы запускаете ВП, используя в цикле функцию интервалов времени LabVIEW, осуществляя одну выборку в секунду, и одновременно хотите перемещать окно по экрану. Ваш ВП не получит никаких данных во время этого перемещения! Допустим, вы перемещаете окно в течение 5 с. Когда вы наконец отпустите кнопку мыши, ВП продолжит выполнение, но *никак не отобразит*, что в течение этих 5 с не было сделано выборок данных. При использовании виртуальных приборов, применяющих циклический механизм LabVIEW для создания временных интервалов, необходимо убедиться, что одновременно не запущены какие-либо другие программы. Также следует избегать возникновения других событий в операционной системе (активность мыши, жесткого диска, сети и т.д.) во время запуска ВП. Чтобы предотвратить случайности, воспользуйтесь такой функцией LabVIEW, как **Счетчик времени**, для контроля точности временных интервалов цикла.

Рассмотрим еще один пример «легкого» ВП. Упражнение 11.2 даст вам возможность получить несколько осциллограмм и отобразить их на графике.

11.2.3. Упражнение 11.2: еще раз об аналоговом вводе

1. Подключите четыре источника постоянного тока или четыре источника с низкочастотным напряжением к каналам 0–3. Если у вас нет такого количества источников напряжения, подключите один ко всем каналам или используйте цепь резисторов для изменения амплитуды напряжения в каждом канале.
2. Постройте лицевую панель и блок-диаграмму, как показано на рис. 11.19 и 11.20. Из контекстного меню графика выберите функцию **Игнорировать отметку времени** (Ignore Timestamp). Функция аналогового ввода возвращает массив осциллограмм – по одной для каждого канала.
3. Установите параметры **скорость опроса** (scan rate), **каналы** (channels) и **количество опросов для получения данных** (number of scans to acquire), как указано.
4. Сохраните виртуальный прибор с именем **Acquire Multiple Channels.vi**.

Блок-диаграмма очень проста. Функция **АЦП Получение осциллограмм** (AI Acquire Waveforms) выполнит все операции. Обратите внимание, что этот ВП

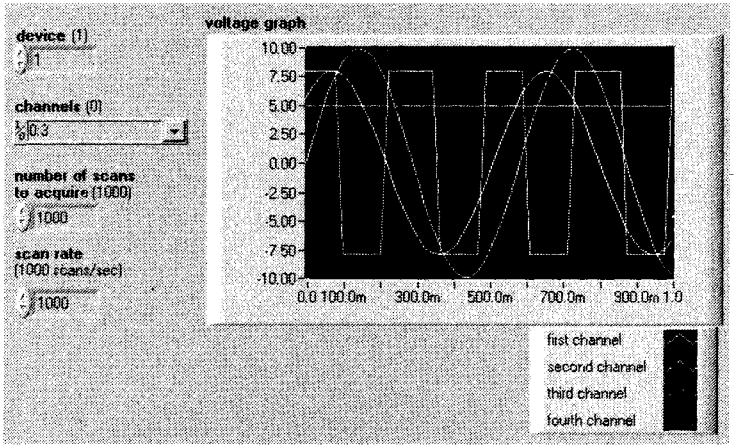


Рис. 11.19

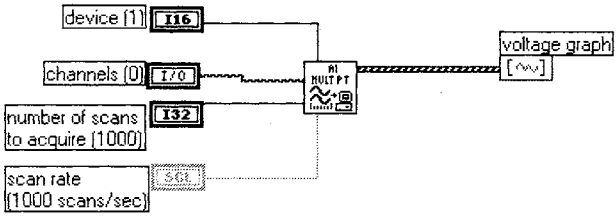


Рис. 11.20

осуществляет небуферизованное аналого-цифровое преобразование с программным запуском.

Отметьте также, что мы использовали строку `0:3` для определения каналов. Мы могли бы определить их как виртуальные каналы, если бы они были установлены таким образом в MAX. Например, если бы каналу 0 было присвоено имя `MySensor1`, а каналу 1 – `MySensor2` и т.д., разрешалось бы подключить вместо этой строки массив виртуальных каналов.

Необходимо упомянуть об одном важном ограничении при работе с несколькими каналами ввода/вывода. Если вы установите высокую скорость опроса нескольких каналов и посмотрите данные из каждого канала с течением времени (а не зависимость от индексов), то заметите существенную *задержку по фазе* между каналами. Почему это происходит? Большая часть плат может выполнять только одно аналого-цифровое преобразование в единицу времени. Вот почему этот процесс называется *опросом*. Данные во входных каналах оцифровываются

последовательно по одному каналу. Задержка, называемая *межканальной*, возникает между выборками из каждого канала. Как правило, межканальная задержка очень мала, но это сильно зависит от платы.

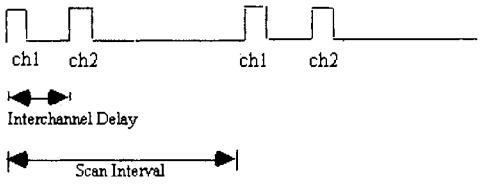


Рис. 11.21

В сигналах постоянного тока и низкочастотных сигналах задержка по фазе в опросах не является проблемой. Межканальная задержка иногда настолько меньше периода опроса, что плата в состоянии делать выборки из каждого канала *почти* одновременно. Например, межканальная задержка может находиться в микросекундном диапазоне, а частота выборки составлять 1 опрос в секунду (как показано на рис. 11.22). Однако на более высоких частотах задержка бывает достаточно заметной и создает некоторые проблемы в процессе измерений, особенно если ваша система зависит от синхронизированных сигналов.

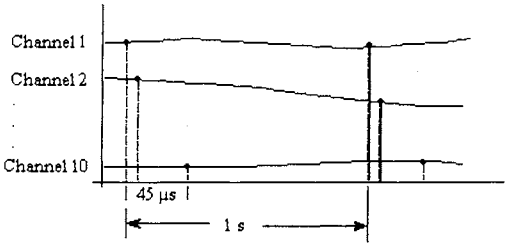


Рис. 11.22

11.2.4. Улучшенный аналоговый ввод/вывод: средний уровень

Если вы больше не хотите получать данные при помощи LabVIEW, то можете пропустить оставшуюся часть этой главы.

Одним из основных ограничений виртуальных приборов верхнего уровня, рассмотренных ранее, является избыток выполняемых задач по сбору данных. Каждый раз, когда вы вызываете, например, функцию **АЦП выборка из канала**, вы настраиваете аппаратную часть на особый вид измерений, определяете частоту выборки и т.д. Очевидно, если вы собираетесь делать выборки очень часто, то не

нужно настраивать измерения в каждой итерации. Виртуальные приборы верхнего уровня предназначены для легкого и быстрого программирования, но они могут загружать процессор ненужными задачами и не проявляют гибкости в тех приложениях, где необходимо обработать много данных.

Виртуальные приборы среднего уровня предлагают большую функциональность, гибкость и эффективность в использовании. Эти виртуальные приборы обладают такими способностями, как управление межвыборочной частотой, применение внешних пусковых устройств и осуществление непрерывных операций ввода/вывода. Ниже кратко описывается каждый из виртуальных приборов ввода/вывода среднего уровня. Эффективное использование виртуальных приборов аналогового ввода/вывода возможно лишь при подключении необходимых входов. В большинстве случаев вам не нужно беспокоиться о второстепенных входах (серого цвета), которые показаны в окне помощи.

Аналоговый ввод

Функция **АЦП Конфигурация** (AI Config) настраивает работу аналогового ввода для определенного набора каналов, конфигурирует аппаратную часть и выделяет место под буфер в памяти компьютера (рис. 11.23). **Устройство** (Device) определяет номер платы ввода/вывода, **Каналы** (Channels) – номера каналов аналогового ввода. Опция **Пределы входного сигнала** (Input limits) задает диапазон входного сигнала и устанавливает коэффициент усиления аппаратной части. **Размер буфера** (Buffer size), измеряемый в опросах (scans), управляет объемом памяти компьютера, зарезервированным для полученных данных. **Межканальная задержка** (Interchannel Delay) устанавливает межканальный фазовый сдвиг для интервалов опроса.

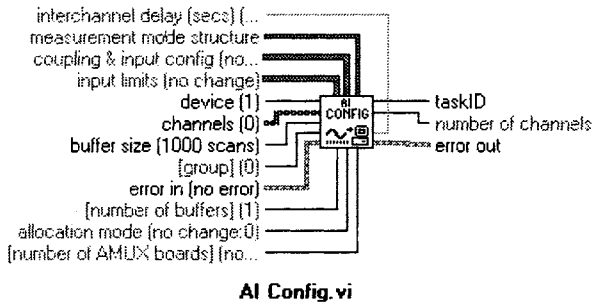


Рис. 11.23. ВП АЦП Конфигурация

АЦП Старт (AI Start) начинает работу буферизованного аналогового ввода. Этот ВП управляет скоростью сбора данных, количеством точек данных и использованием опций запуска аппаратной части. Двумя важными входами ВП **АЦП Старт** являются **частота опроса** (scan rate) – количество опросов каждого канала в секунду – и **количество опросов** (number of scans to acquire) – сколько раз опрашивать весь набор каналов.

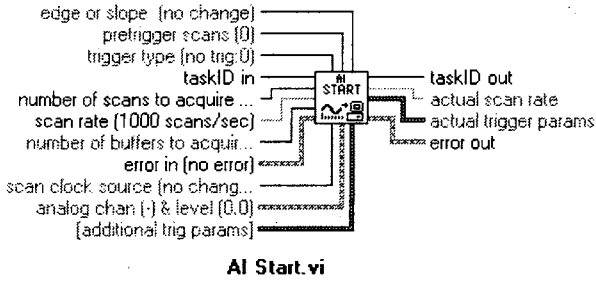


Рис. 11.24. ВП АЦП Старт

АЦП Чтение (AI Read) считывает данные из буфера, размещенного в памяти компьютера прибором **АЦП Конфигурация**. Этот ВП может управлять количеством точек для считывания из буфера, их расположением в считываемом буфере и тем, возвращать ли значения напряжения в виде бинарных или масштабированных данных. На выходе ВП генерируется массив осциллограмм, по одной для каждого канала (рис. 11.25).

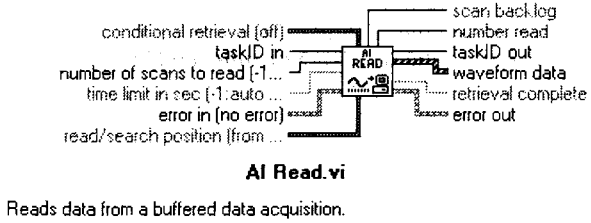


Рис. 11.25. ВП АЦП Чтение

АЦП Единичный опрос (AI Single Scan) возвращает данные, полученные в результате одного опроса (рис. 11.26). Выходные данные в виде осциллограммы (waveform data) представляют собой значения напряжения, считанные из каждого канала набора. Этот ВП применяется в сочетании с **АЦП Конфигурация**; при использовании **АЦП Единичный опрос** нет необходимости в виртуальных приборах **АЦП Старт** и **АЦП Чтение**.

Рис. 11.26. ВП АЦП Единичный опрос

АЦП Очистка (AI Clear) останавливает работу аналогового ввода, удаляет буфер из памяти компьютера и освобождает ресурсы платы ввода/вывода, такие как счетчики (рис. 11.27).

Рис. 11.27. ВП АЦП Очистка

Первым виртуальным прибором, всегда используемым при настройке аналогового ввода, является **АЦП Конфигурация**, который создает идентификатор задачи и кластер ошибок. На вход всех остальных виртуальных приборов аналогового ввода/вывода идентификатор задачи поступает и служит для определения прибора и рабочих каналов. По завершении выполнения каждого ВП идентификатор задачи поступает на его выход. Поскольку идентификатор задачи является входным и выходным параметром для других виртуальных приборов аналогового ввода/вывода, он формирует зависимость данных среди виртуальных приборов. Это позволяет управлять потоком данных на диаграмме и дает возможность убедиться, что виртуальные приборы выполняются в установленном порядке.

Аналоговый вывод

ЦАП Конфигурация (AO Config) настраивает работу аналогового вывода для определенного набора каналов, конфигурирует аппаратную часть и выделяет место под буфер в памяти компьютера (рис. 11.28). **Устройство** (Device) определяет номер платы ввода/вывода, **Каналы** (Channels) – номера каналов аналогового вывода. Опция **Установки пределов** (Limit settings) задает диапазон выходных сигналов. **Идентификатор задачи** (taskID) используется всеми последующими виртуальными приборами аналогового вывода для идентификации прибора и рабочих каналов.

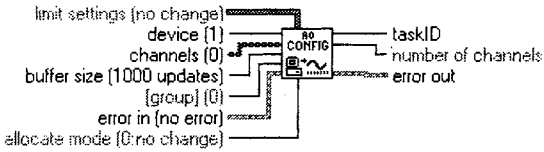


Рис. 11.28. ВП ЦАП Конфигурация

ЦАП Запись (AO Write) записывает данные массива осциллограмм (waveform data) в буфер, используемый при работе аналогового вывода (рис. 11.29). Данные должны быть представлены в виде одной осциллограммы для каждого канала из списка каналов.

ЦАП Старт (AO Start) запускает буферизованный аналоговый вывод (рис. 11.30). **Скорость обновления** (Update rate) – число обновлений генерируемого напряжения в секунду. Если подключить 0 к вводу **количество итераций буфера** (number of buffer iterations), то плата станет непрерывно выводить данные из буфера до тех пор, пока не будет запущена функция **ЦАП Очистка**.

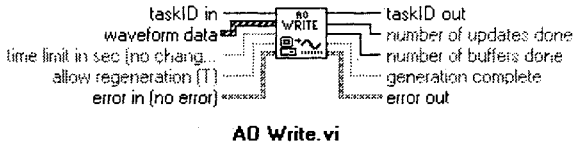


Рис. 11.29. ВП ЦАП Запись

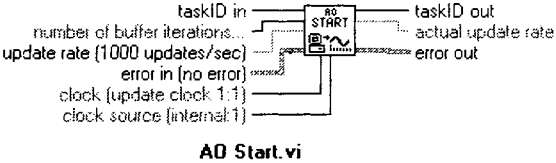


Рис. 11.30. ВП ЦАП Старт

ЦАП Задержка (AO Wait) ждет завершения генерации осциллограммы и после этого возвращает идентификатор (рис. 11.31). Этот ВП проверяет статус задачи через равные промежутки времени в асинхронном режиме, чтобы освободить процессор для других операций. Интервал ожидания рассчитывается путем деления данных терминала **проверка каждые N обновлений** (check every N updates) на значение **Скорость обновления**.

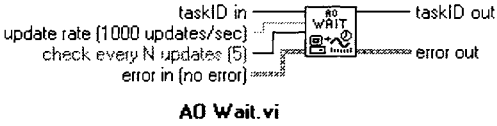


Рис. 11.31. ВП ЦАП Задержка

ЦАП Очистка (AO Clear) останавливает работу аналогового вывода, удаляет буфер из памяти компьютера и освобождает ресурсы платы сбора данных, такие как счетчики (рис. 11.32).

Рис. 11.32. ВП ЦАП Очистка

Все виртуальные приборы, показанные выше, сильно зависят от потока данных. Как вы уже, вероятно, заметили, их расположение диктуется логикой. Аналоговые функции связаны вместе идентификатором задачи и кластерами ошибок. На рис. 11.33 и 11.34 показана необходимая последовательность виртуальных приборов для получения и генерации аналоговой осциллограммы.

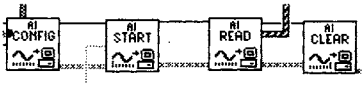


Рис. 11.33. Получение

1. **АЦП Конфигурация**: конфигурирует каналы и буфера.
2. **АЦП Старт**: запускает процесс получения данных.
3. **АЦП Чтение**: считывает данные с буфера.
4. **АЦП Очистка**: очищает буфер от данных и освобождает ресурсы.

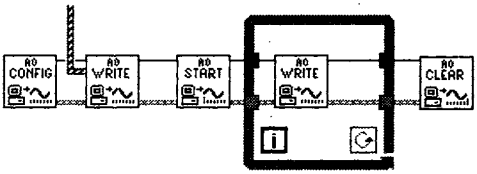


Рис. 11.34. Генерация

1. **ЦАП Конфигурация**: конфигурирует каналы и буфер.
2. **ЦАП Запись**: заносит данные в буфер.
3. **ЦАП Старт**: запускает генерацию.
4. **ЦАП Запись**: заносит новые данные в буфер.
5. **ЦАП Задержка**: [необязателен] ждет опустошения буфера.
6. **ЦАП Очистка**: очищает буфер и освобождает ресурсы.

Буферизованный аналоговый ввод

Вы помните буферизованный ввод/вывод, описанный в начале этой главы? Ранее мы применили цикл по условию, чтобы собрать множество точек, а теперь воспользуемся виртуальными приборами промежуточного ряда. Это можно сделать путем программирования аппаратной части для получения заданного количества точек при определенной скорости выборки.

11.2.5. Упражнение 11.3: сбор данных с использованием буфера

1. Вначале создайте лицевую панель, как показано на рис. 11.35, с простой разверткой для просмотра данных.
2. Подключите источник напряжения к каналу 0. После этого соедините ВП аналогового ввода/вывода (см. рис. 11.36) в порядке, описанном ранее.
3. Сохраните ваш виртуальный прибор как **Buffered Analog In.vi**.

Вот и все – вы только что создали очень простой осциллограф. Следует сделать несколько замечаний относительно новой блок-диаграммы:

- элемент управления **Пределы входного сигнала** (Input Limits) используется для регулировки коэффициента усиления платы. Например, если сигналы имеют амплитуду 100 мВ, то коэффициент усиления платы равен 100. Однако если вы уже установили пределы напряжения для виртуального канала в MAX, то вам не следует пользоваться элементом управления **Пределы входного сигнала**;
- виртуальный прибор **АЦП Старт** запускает процесс сбора данных. Как только он закончит выполняться, все данные будут загружены в буфер.

Данные из буфера не будут доступны в LabVIEW, пока вы не решите их считать. Хотя мы подключили виртуальный прибор **АЦП Чтение** сразу после этого виртуального прибора, допустимо считать данные позднее при условии, что данные в буфере не были удалены или переписаны каким-либо другим процессом;

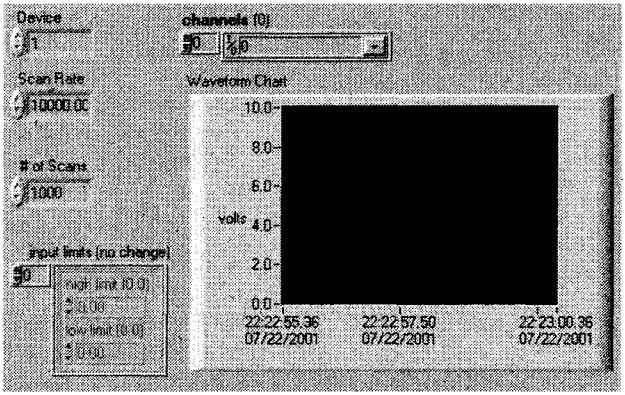


Рис. 11.35

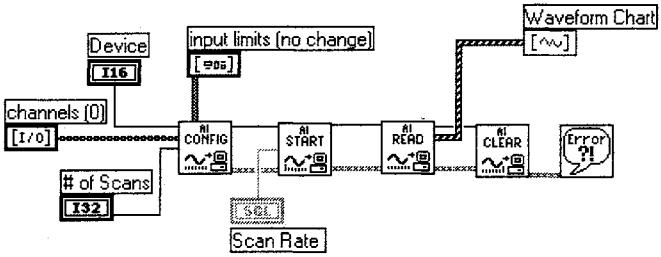


Рис. 11.36

- виртуальный прибор **АЦП Очистка** необходим для работы, так как он очищает буфер. В противном случае виртуальный прибор **АЦП Чтение** будет считывать старые данные.

11.2.6. Упражнение 11.4: еще раз о процессе сбора данных

Данное упражнение является модификацией предыдущего.

1. Создайте ВП отображения и хранения осциллограммы. Этот ВП должен иметь лицевую панель, аналогичную показанной на рис. 11.35, но с дополнительной способностью хранения всех данных в файле (если в этом вам необходима помощь, посмотрите еще раз функции по работе

с осциллограммами в главе 8 и операции ввода/вывода в файл в главе 9). У пользователя должна быть возможность назвать файл. Этот же ВП должен быть в состоянии считать файл и отобразить на той же развертке прежние данные.

Для того чтобы единовременно выполнялась только одна операция, используйте логический элемент управления для выбора «считать/записать» и структуру варианта.

2. Создайте ВП аналогового вывода в виде «синтезатора осциллограммы». Этот виртуальный прибор позволит вам выбирать между такими генерируемыми формами осциллограммы, как синусоида, треугольник, прямоугольник и зуб пилы. Применяйте функцию **Генератор функций** (Function Generator) из палитры **Осциллограмма ⇒ Генерация осциллограмм** (Waveform Generation). Работайте с виртуальными приборами аналогового вывода так же, как с приборами аналогового ввода. В качестве дополнительной возможности этот прибор должен использовать осциллограмму из внешнего источника наряду с другими генерируемыми функциями.

Решения для этих упражнений вы найдете на CD в директории `CH11.LLB`: **Acquire and Save.vi** и **Function Generator.vi**.

11.2.7. «Интеллектуальные» приборы аналогового ввода/вывода

В этом разделе мы рассмотрим на примерах некоторые сложные, но полезные на практике понятия, такие как кольцевой буфер (непрерывный сбор данных), синхронизация при помощи аппаратной части и потоковая передача данных.

Непрерывный сбор данных

Непрерывный сбор данных, или сбор данных в реальном времени, возвращает данные по мере поступления, не прекращая их получения. Этот метод обычно основан на использовании схемы кольцевого буфера, как показано на рис. 11.37. Вы устанавливаете размер кольцевого буфера. Плата ввода/вывода собирает и хранит данные в этом буфере. Когда буфер заполняется, плата начинает записывать данные в начало буфера (поверх ранее сохраненных данных – независимо от того, были они считаны LabVIEW или нет). Затем продолжается до тех пор, пока система не получит определенное количество выборок. После этого LabVIEW останавливает процесс сбора данных, в противном случае появится ошибка. Непрерывный сбор данных используется в основном для потоковой передачи данных на диск и их отображения в реальном времени.

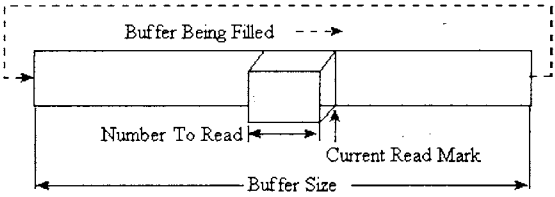


Рис. 11.37. Кольцевой буфер

11.2.8. Упражнение 11.5: непрерывный сбор данных

1. Создайте лицевую панель, изображенную на рис. 11.38.
2. Сохраните этот виртуальный прибор как **Continuous Acquistion.vi**. Затем соедините виртуальные приборы аналогового ввода, как показано на рис. 11.39.

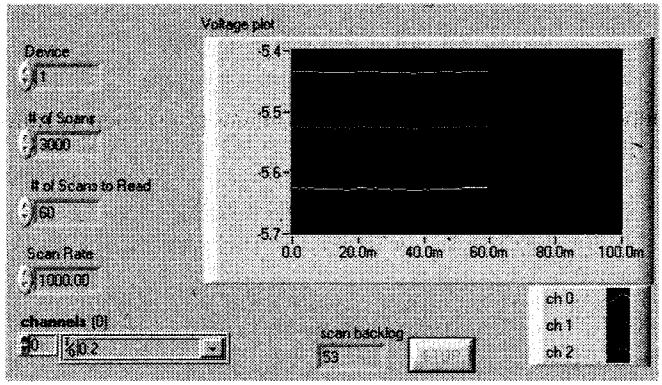


Рис. 11.38

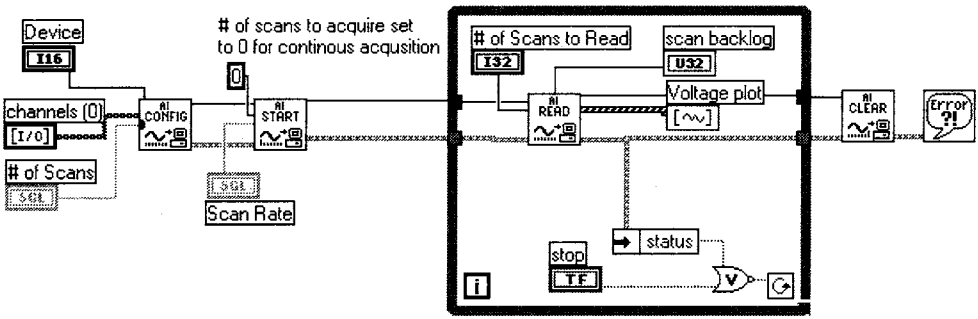


Рис. 11.39

3. Сконфигурируйте LabVIEW для непрерывного сбора данных путем соответствующей настройки ВП **АЦП Старт**. Установите опцию **количество опросов сбора данных** (number of scans to acquire) виртуального прибора **АЦП Старт** на 0. Такой режим получения данных называется *асинхронным*. Это означает, что в процессе сбора данных другие операции LabVIEW также могут выполняться. На рис. 11.39 показана блок-диаграмма для лицевой панели, изображенной на рис. 11.38. Виртуальный прибор **АЦП Чтение** вызывается внутри цикла для считывания данных из буфера. Отсюда вы можете отослать данные для построения графика. Виртуальный прибор **АЦП Очистка** останавливает сбор данных, стирает содержимое буфера из памяти компьютера и освобождает ресурсы платы. Обратите внимание, что этот виртуальный прибор (осуществляющий *непрерывный* сбор данных) отличается от ВП для получения осциллограммы (обычный виртуальный прибор буферизованного сбора данных) наличием цикла по условию, константы (0), подключенной к терминалу **количество опросов для сбора данных** виртуального прибора **АЦП Старт** и параметра **количество опросов для чтения** (number of scans to read).

Этот ВП создан для непрерывного выполнения. Чтобы во время его работы не происходило переполнения памяти, в нее помещается буфер установленной длины, заполненный выборками от начала до конца. Затем данные в начале буфера перезаписываются новыми до тех пор, пока буфер вновь не оказывается заполненным. В качестве примера предположим, что ВП выполняется с размером буфера 10 и частотой выборки, равной одной выборке в секунду. Ниже показано, как буфер наполняется данными:

1. Когда виртуальный прибор не запущен, буфер пуст.
2. По истечении 1 с в нем появляется одна выборка.
3. По истечении 9 с буфер почти заполнен.
4. По истечении 12 с буфер наполнен, и вновь поступающие данные накладываются на старые.

Чтобы воспользоваться синхронизацией аппаратной части и возможностями управления памятью в LabVIEW, мы применили виртуальные приборы промежуточного ряда. Если во время выполнения ВП **Single Channel AI.vi** операционная система перехватит работу CPU, то ВП промежуточного уровня будут использовать буфер платы ввода/вывода и прямой доступ к памяти (DMA) – если таковой имеется – для продолжения сбора данных без помощи CPU. Прямой доступ к памяти дает возможность аппаратной части передать данные непосредственно в память компьютера, даже если процессор занят другой задачей. LabVIEW начнет терять выборки данных только тогда, когда операционная система задействует CPU дольше времени обработки FIFO-буфера платы ввода/вывода и буфера DMA. Эту ситуацию можно лучше понять, если обратиться к предыдущим четырем рисункам, изображающим заполнение буфера. Вначале предположим, что у платы ввода/вывода нет FIFO-буфера, однако имеется прямой доступ к памяти. В идеале

плата записывает данные в буфер непрерывно, а LabVIEW и непрерывно считывает несколько выборок перед последней, поступившей в буфер. Предположим, что через секунду процессор будет использоваться другой задачей, а LabVIEW уже считал первую выборку. Когда процессор занят, плата ввода/вывода может сохранять данные в буфере, но LabVIEW будет не в состоянии считать их с этого буфера. Если по истечении 12 с процессор все еще занят, LabVIEW потеряет данные второго отрезка слева, хотя он содержит свежие данные (светло-серый цвет). Четыре виртуальных прибора промежуточного уровня, используемые в предыдущем примере, отметят это как ошибку.

Почему нам нужен индикатор **Scan Backlog**? Весьма полезно знать, успевает ли LabVIEW считывать данные. Если буфер заполняется быстрее, чем ВП считывает из него данные, то вы начнете терять информацию, поскольку буфер переполняется и новые данные накладываются на старые.

Запуск и синхронизация аппаратной части

Существует два способа запуска операции сбора данных: посредством программного обеспечения и посредством аппаратной части.

При запуске с помощью программного обеспечения операция по сбору данных начинается тогда, когда выполняется определенная функция программы, инициирующая сбор данных. Например, «легкие» виртуальные приборы сбора данных используют программный запуск. Как только LabVIEW начинает выполнение ВП, идет получение или генерация данных. Все платы ввода/вывода поддерживают программный запуск.

Другим известным способом запуска операции сбора данных является ожидание некоторого внешнего события. Обычно начинают сбор данных в зависимости от характеристик цифрового или аналогового сигнала, таких как состояние, уровень или амплитуда. Электронная схема съемной платы использует этот аналоговый или цифровой сигнал для включения таймеров платы, которые управляют сбором данных. Большая часть плат ввода/вывода поддерживает *цифровой* запуск процесса сбора данных, другие платы – *аналоговый* запуск. Входной контакт платы ввода/вывода, на который подается сигнал запуска, обычно помечается как **EXTTRIG** либо **Start Trig**. Все платы серии E компании NI поддерживают цифровой запуск.

Вы можете использовать внешнее аналоговое устройство для запуска *считывания* данных из буфера в LabVIEW, а не просто *инициировать* операцию по сбору данных. Этот вид запуска называется *считыванием по условию* (conditional retrieval). При считывании по условию плата ввода/вывода собирает данные и сохраняет их в буфере посредством программного запуска. Однако плата не запрашивает данные до тех пор, пока не получит выборку, удовлетворяющую определенным условиям по уровню и крутизне. (Не спутайте считывание по условию с аналоговым запуском аппаратной части. Системы, использующие аналоговый запуск, не сохраняют данные в буфере до появления определенного события; системы, применяющие считывание по условию, обязательно сохраняют данные в буфере, но эти данные не считываются в LabVIEW, пока не произойдет запуск.)

На рис. 11.40 показан кластер считывания по условию, который устанавливает критерий для считывания данных из буфера. Этот кластер поступает на вход ВП **АЦП Чтение**. Как только сбор данных начался, плата непрерывно делает выборки сигнала и сравнивает их с условиями считывания. После того как условия выполнены, ВП **АЦП Чтение** возвращает количество данных, определенное на вводе **числа опросов для считывания** (number of scans to read).

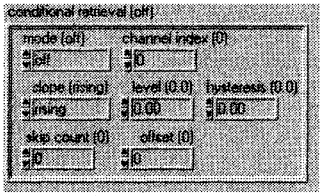


Рис. 11.40. Кластер считывания по условию

11.2.9. Упражнение 11.6: инициирование сбора данных

Создайте виртуальный прибор аналогового ввода, который использует аппаратный запуск для начала и остановки процесса сбора данных.

1. Подключите источник ТТЛ сигнала к разъему внешнего пускового устройства платы ввода/вывода и подайте на входы платы пару аналоговых сигналов.
2. Создайте лицевую панель, как показано на рис. 11.41. Элемент управления **тип триггера** (trigger type) является элементом с выпадающим меню, которое позволяет задать тип пускового устройства:

   0: Нет запуска (по умолчанию)
   1: Аналоговый запуск

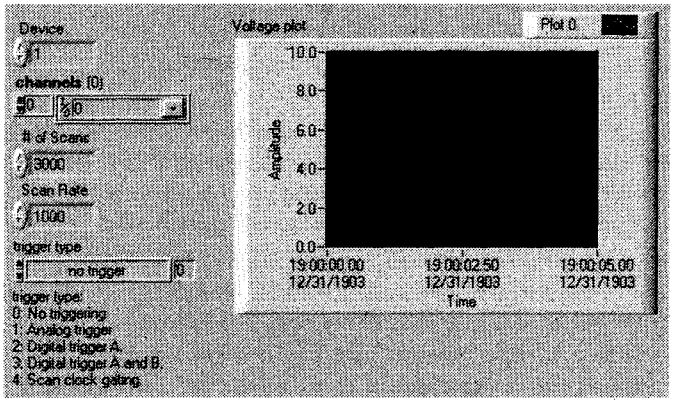


Рис. 11.41. Лицевая панель

2: Цифровой запуск (триггер A)
3: Цифровой запуск (триггеры A и B)
4: Стробирование синхронизации развертки

Вы можете задействовать один из этих типов, посмотрев пример на компакт-диске, или просто использовать числовое управление.

3. Создайте блок-диаграмму, как показано на рис. 11.42. Обратите внимание, что единственным отличием этой блок-диаграммы и простого буферизованного аналогового ввода является подключение ввода **тип триггера** (trigger type) в виртуальном приборе **АЦП Старт**. Сохраните этот виртуальный прибор как **Triggered Analog In.vi**.

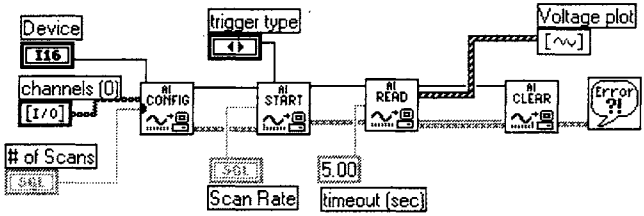


Рис. 11.42

Потоковая запись данных в файл

В ранее рассмотренных упражнениях вы научились с помощью LabVIEW преобразовывать данные в формат таблицы символов и после завершения процесса сбора данных сохранять их в файле ASCII. Другим и, возможно, более эффективным подходом является сохранение небольших объемов информации на жестком диске непосредственно во время процесса сбора данных. Этот тип операции записи/чтения файла называется *потоковой передачей данных*. Преимуществом потоковой передачи данных в файл является ее быстрота. Следовательно, вы можете непрерывно собирать информацию и всегда иметь сохраненную копию всех полученных данных.

При непрерывной работе ВП решающим фактором становится скорость, с которой LabVIEW считывает данные из буфера, а затем переносит их на диск. У вас должна быть возможность считывать и переносить данные достаточно быстро, чтобы не происходило переполнения кольцевого буфера. Для увеличения эффективности считывания данных во время процесса сбора необходимо избегать выполнения других функций, таких как анализ данных. С этой же целью вместо вывода данных из виртуального прибора **АЦП Чтение** в аналоговом виде применяют вывод данных в двоичном виде. Когда вы настраиваете виртуальный прибор **АЦП Чтение** для создания только бинарных данных (это задается терминалом **единицы вывода** – output units), он может записать их на диск быстрее, чем при использовании виртуальных приборов из папки **Ввод/вывод файлов**

после считывания массива данных. Недостатком считывания и потоковой передачи двоичных данных являются трудности при последующем использовании файла.

Вы можете модифицировать виртуальный прибор непрерывного сбора данных, чтобы применять потоковую запись файла. Хотя раньше вы, вероятно, не использовали бинарные файлы, вы легко создадите ВП, описываемый в следующем примере. Более подробно о бинарных и других файлах мы поговорим в главе 15.

11.2.10. Упражнение 11.7: потоковая запись на диск

Из-под LabVIEW откройте пример виртуального прибора **Cont Acq to File (binary)**, который находится в библиотеке `LabVIEW\Examples\daq\analogin\strmdsk.llb`. Вы увидите, как он реализует запись на диск.

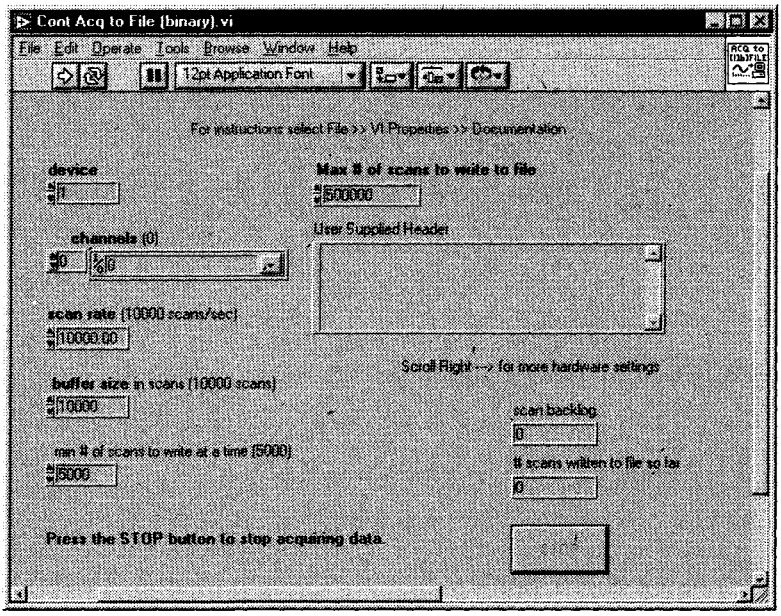


Рис. 11.43

Запустите этот ВП в течение нескольких секунд. Если хотите, можете вначале ввести заголовок файла в окне **User Supplied Header**.

Для того чтобы увидеть данные, сохраняемые на диск, используйте сопутствующий пример ВП **Display Acq'd File (binary)**, находящийся в той же директории, что и предыдущий виртуальный прибор.

Запустите этот ВП, применив имя файла из предыдущего ВП. Обратите внимание, что первый ВП, записывающий данные, даже не пытается построить

график. Целью потоковой передачи является *быстрая* запись данных на диск, а затем их просмотр.

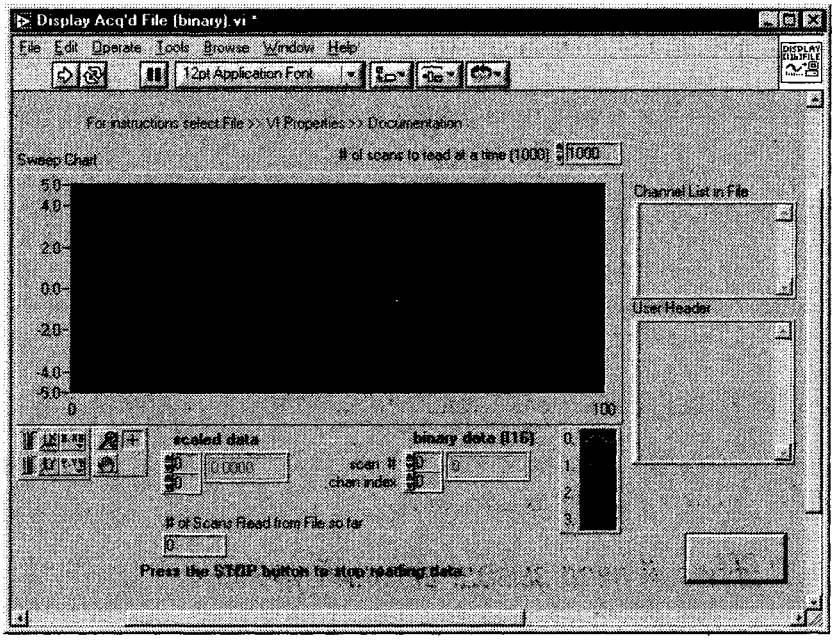


Рис. 11.44

11.3. Цифровой ввод/вывод

В цифровом мире – за небольшим исключением – все намного проще, чем в аналоговом. Все, с чем работают цифровые устройства, – это 1 или 0, выше или ниже, on или off. В действительности требуются некоторые знания в области двоичного представления и арифметики наряду со знанием следующих определений LabVIEW.

Цифровая линия является эквивалентом аналоговому каналу: это путь, по которому передается цифровой сигнал. Цифровые линии обычно бывают либо *входными*, либо *выходными*, но иногда могут быть *двунаправленными*. В большинстве плат линии должны быть настроены в качестве либо входных, либо выходных; они не могут действовать одновременно в обоих направлениях.

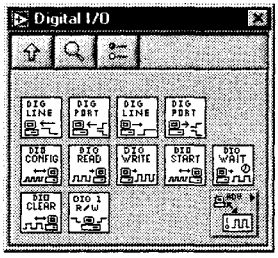


Рис. 11.45

Портом называется набор цифровых линий, сконфигурированных в одном направлении, которыми можно пользоваться в одно и то же время. Количество цифровых линий каждого порта зависит от типа платы, но большая часть портов состоит из четырех или

восьми цифровых линий. Например, многофункциональная плата ввода/вывода может содержать восемь цифровых линий, сконфигурированных в один 8-канальный порт, два 4-канальных порта или даже восемь 1-канальных портов. Порты имеют свойства *цифровых каналов*.

Разрядность порта – это количество его линий.

Состоянием называется один из двух возможных статусов цифрового канала: логическое ИСТИНА (то же, что и логическая 1 или «on») или логическое ЛОЖЬ (то же, что и логический 0 или «off»).

Шаблон представляет собой последовательность цифровых состояний, часто выражаемую в виде двоичного числа, которое описывает состояния каждого из каналов порта. Например, 4-канальный порт может иметь шаблон «1101», означающий, что первый, третий и четвертый каналы находятся в состоянии ИСТИНА, а второй канал – в состоянии ЛОЖЬ. Первый бит (самый младший разряд) располагается с правого края шаблона. Последний бит (четвертый в данном примере – самый старший разряд) расположен слева. Данный шаблон можно преобразовать из двоичного представления в десятичное число 13.

Кстати, платы ввода/вывода компании NI используют положительную ТТЛ логику. Это означает, что нижний логический уровень сигнала лежит в пределах 0–0,8 В, а верхний – в пределах 2,2–5,5 В.

11.3.1. Цифровой ввод/вывод: верхний уровень

Для простого цифрового ввода/вывода можно использовать виртуальные приборы верхнего уровня в палитре **Цифровой ввод/вывод**: применять их легко и интуитивно понятно.

Виртуальный прибор **Считывание из цифровой линии** (Read from Digital Line) считывает логическое состояние цифровой линии (рис. 11.47). Ввод **прибор** (device) определяет номер платы ввода/вывода, а **цифровой канал** (digital channel) – порт, содержащий данную линию. **Линия** (Line) задает цифровую линию для считывания. **Состояние линии** (Line state) возвращает состояние линии: высокое (ИСТИНА) или низкое (ЛОЖЬ).

Виртуальный прибор **Считывание из цифрового порта** (Read from Digital Port) считывает состояние всех линий порта (рис. 11.48). **Цифровой канал** (Digital

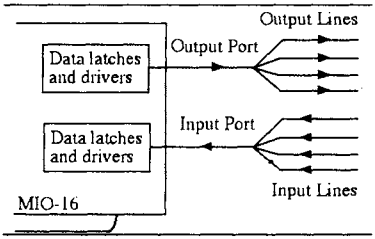


Рис. 11.46

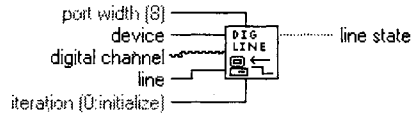


Рис. 11.47. ВП **Считывание из цифровой линии**

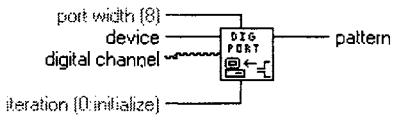


Рис. 11.48. ВП ***Считывание из цифрового порта***

channel) определяет цифровой порт для считывания. **Шаблон** (Pattern) возвращает состояние цифровой линии в виде десятичного числа, которое, будучи преобразованным в двоичное представление, облегчает наблюдение за состоянием каждой отдельной линии порта.

Виртуальный прибор **Запись в цифровую линию** (Write to Digital Line) устанавливает заданную линию порта в высокое или низкое логическое состояние (рис. 11.49). Ввод **устройство** определяет номер платы ввода/вывода, а **цифровой канал** – порт, содержащий данную линию. Ввод **линия** (line) задает цифровую линию для записи, а **состояние линии** (line state) отображает состояние, которое нужно записать в линию: высокое (ИСТИНА) или низкое (ЛОЖЬ).

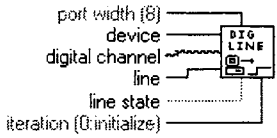


Рис. 11.49. ВП ***Запись в цифровую линию***

Виртуальный прибор **Запись в цифровой порт** (Write to Digital Port) конфигурирует состояние определенного порта (рис. 11.50). **Цифровой канал** (Digital channel) указывает цифровой порт для конфигурации. На вводе **шаблон** задается десятичный эквивалент двоичной конфигурации, которая должна быть записана в линии порта.

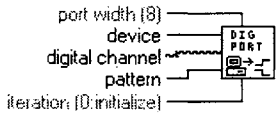


Рис. 11.50. ВП ***Запись в цифровой порт***

Все из вышеперечисленных ВП осуществляют *незамедлительный* ввод/вывод. Это означает, что как только ВП, например **Запись в цифровую линию**, выполнится, то соответствующая линия тотчас устанавливается в состояние ИСТИНА или ЛОЖЬ и остается в том или ином состоянии до тех пор, пока другой запуск ВП не изменит его.

Терминал **итерация** (iteration) данных ВП нуждается в некотором пояснении. Поскольку функция является высокоуровневой цифровой функцией ввода/вывода, то при каждом (по умолчанию) их вызове цифровые порты конфигурируются для соответствующего направления и типа передачи. Обычно такую настройку осуществляют один раз для целой серии считываний и записей. Если вы повторно используете одну из этих функций, например в цикле, то можете избавиться от постоянной настройки портов путем подключения ненулевого числа к терминалу итерации. Нуль (по умолчанию) говорит ВП о необходимости инициализировать конфигурацию при запуске. Следующий пример поможет вам понять работу этих виртуальных приборов.

Предположим, имеется высоковольтный переключатель и цифровое реле, дающее возможность узнать, когда переключатель закрыт. Цифровая линия этого реле обычно находится в состоянии ИСТИНА (переключатель открыт). Когда переключатель закрывается, линия переходит в состояние ЛОЖЬ. Для создания простого ВП, проверяющего состояние цифровой линии 3 в цифровом порте 0 платы ввода/вывода, следует создать что-то похожее на блок-диаграмму, изображенную на рис. 11.51.

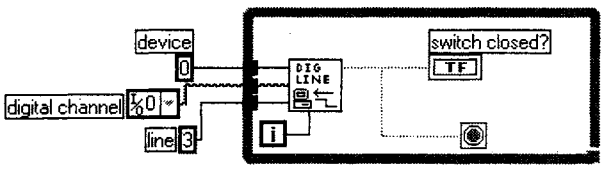


Рис. 11.51

Обратите внимание, каким образом терминал итерации цикла подключен к вводу **итерации** в цифровом ВП. При первом выполнении цикла, когда i = 0, ВП **Считывание из цифровой линии** настраивает цифровой порт. Позже, когда i > 0, ВП будет лишь считывать из линии, позволяя циклу выполниться за меньшее время. Когда переключатель закрывается, цифровая линия переходит в состояние ЛОЖЬ, цикл прекращает выполняться и появляется сигнал «переключатель закрыт».

11.3.2. Упражнение 11.8: цифровой вывод

Создайте ВП, который имел бы массив из четырех логических светодиодов на лицевой панели. Трансформируйте их в элементы управления (путем вызова их контекстного меню и выбора опции **Изменить на элемент управления**). Целью работы этой панели является включение некоторых реальных светодиодов, подсоединенных к цифровым линиям вашей платы. (Не забудьте подключить резистор последовательно со светодиодом.) Пользователь будет иметь возможность включать и выключать реальные светодиоды с помощью виртуальных. Кроме этого, заставьте программу записывать данные в цифровой порт, а не в каждую

отдельную линию для снижения загрузки процессора. Сохраните ваш виртуальный прибор как **Digital Port.vi**.

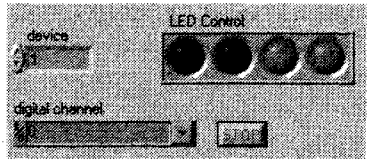


Рис. 11.52

Решение является очень простым. Но оно будет еще проще, если вы используете функцию **Логический массив в число** (Boolean Array to Number) из палитры **Логические**. С ее помощью можно подключить логический массив непосредственно к входу **шаблон**.

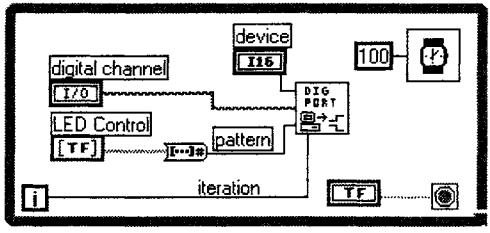


Рис. 11.53

В палитре **Цифровой ввод/вывод** представлено большое количество других функций. Для их применения в более сложных программах пользуйтесь примерами и руководствами LabVIEW.

11.4. Элементы управления приборами: VISA, КОП и последовательная передача данных

Мы могли бы посвятить целую книгу элементам управления приборами в LabVIEW, но сложность этой темы выходит за пределы главы и даже книги; мы лишь наметим правильное направление в изучении данного вопроса. В главе 10 обсуждались некоторые протоколы взаимодействия с приборами, а именно канал общего пользования и последовательная передача данных. Способы коммуникации с прибором могут зависеть от типа прибора. Обычно применяются следующие типы управления:

- канал общего пользования (GPIB);
- последовательный порт;
- расширение VME для работы с инструментами (VXI);

- расширение шины PCI для работы с инструментами (PXI);
- приборы на основе компьютеров.

11.4.1. Архитектура программного обеспечения виртуальных приборов

В попытке создания стандартного интерфейса программного обеспечения, который бы не зависел от протокола или шины аппаратной части, разработчики остановились на архитектуре программного обеспечения виртуального прибора – VISA. VISA является стандартным программным интерфейсом приложения (API), осуществляющего ввод/вывод, для программирования контрольно-измерительного оборудования. VISA может управлять многими типами приборов (VXI, GPIB, PXI или с последовательной передачей), вызывая соответствующие драйверы, которые зависят от типа используемого прибора.

LabVIEW содержит функции VISA в палитре **Управление прибором** (Instrument Control) ⇒ **VISA** (рис. 11.54).

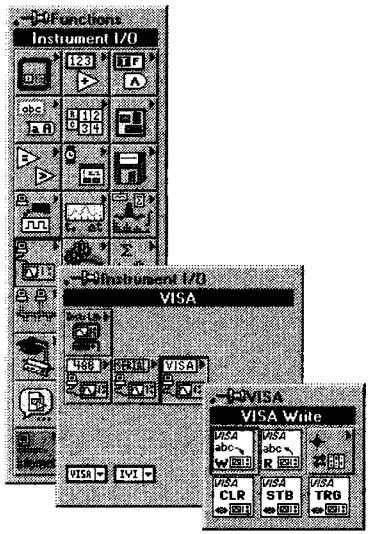


Рис. 11.54

Палитра **VISA** имеет ряд функций, среди которых **VISA Запись**, **VISA Чтение**, **CLR** (Очистка), **STB** (Байт состояния) и **TGR** (Триггер). Последние три являются стандартными функциями в большинстве приборов.

Имя ресурса VISA аналогично каналу сбора данных. Он говорит функциям VISA, с каким прибором или ресурсом вы взаимодействуете. Имена ресурсов VISA можно задать в разделе **Устройства и интерфейсы** программы MAX. В этом

Рис. 11.55. ***VISA Запись*** *(VISA Write). Записывает данные в прибор, определенный на терминале* ***Имя ресурса VISA*** *(VISA resource name)*

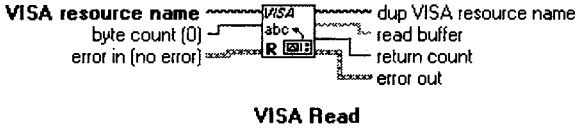


Рис. 11.56. ***VISA Чтение*** *(VISA Read). Считывает данные из прибора*

Рис. 11.57. ***VISA Очистка*** *(VISA Clear). Очистка состояния прибора*

Рис. 11.58. ***VISA Считать БСП*** *(VISA Read STB). Считывает байт состояния прибора*

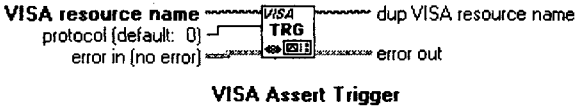


Рис. 11.59. ***VISA Объявить триггер*** *(VISA Assert Trigger). Объявляет программный или аппаратный запуск прибора*

случае для установки прибора служит функция **Постоянная имени ресурса VISA** (VISA Resourse Name Constant) из палитры **Связь с прибором** (Instrument I/O).

Например, у вас есть прибор с последовательной коммуникацией, подключенный к порту 1 (COM1). Вы хотите записать определенную команду в этот прибор и считать 10 байт, которые он возвращает по команде. В таком случае блок-диаграмма вашего ВП должна выглядеть, как показано на рис. 11.60.

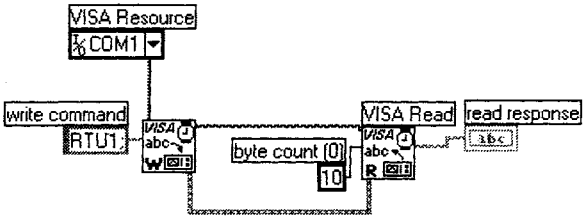


Рис. 11.60

Естественно, командные строки и их длина зависят от особенностей протокола самого прибора.

Иногда при коммуникации с помощью VISA приборы работают не очень хорошо, а иногда вы сами по какой-либо причине не хотите использовать VISA. Во всяком случае LabVIEW предлагает приборы низкого уровня для прямого взаимодействия с использованием протоколов последовательной передачи или КОП.

11.4.2. Канал общего пользования

Подпалитра **КОП** (GPIB) находится в палитре **Связь с прибором**, как показано на рис. 11.61. Набор функций КОП включает инициализацию, посылку команд, возможно считывание ответной реакции, запуск прибора и закрывание канала связи.

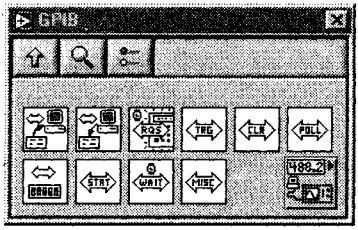


Рис. 11.61

Помните, что все приборы с КОП имеют *адрес* (address), который представляет собой число между 0 и 30, определяющее прибор. Иногда приборы имеют *дополнительный адрес*. Для того чтобы общаться с прибором, во всех виртуальных инструментах, использующих КОП, нужно задать адрес прибора (в формате строки). Тогда становится возможным применение одного и того же виртуального прибора для взаимодействия с различными инструментами по одному каналу общего пользования.

Хорошим способом проверить взаимодействие с вашим прибором посредством КОП является пример **LabVIEW <-> GPIB.vi** из библиотеки `examples\instr\smplgpib.llb`.

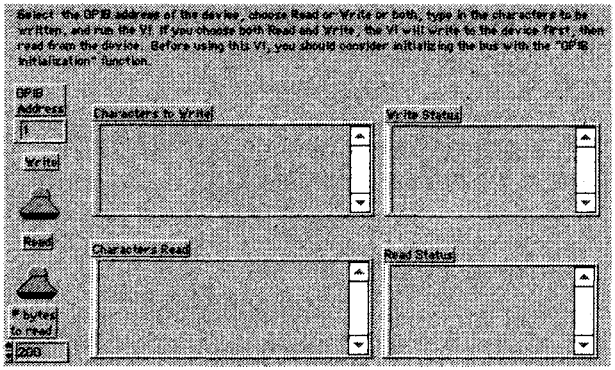


Рис. 11.62

11.4.3. Последовательная коммуникация

Подпалитра **Последовательная коммуникация** (Serial) находится в палитре **Связь с прибором** (рис. 11.63).

Обратите внимание, что эти функции в действительности являются специализированными функциями VISA для последовательных портов. Как и в случае КОП, функции предназначены для считывания и записи. Особенностью последовательных портов является возможность таких операций, как обнаружение количества байтов, ожидающих считывания в последовательном порте, или конфигурирование последовательного порта (например, скорости передачи).

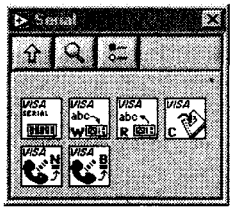


Рис. 11.63

Считается, что последовательная коммуникация наиболее проста для программирования. Однако она страдает от неправильного употребления и игнорирования стандартов аппаратной части, сложных и запутанных протоколов программирования и относительно низкой скорости передачи данных. Трудности, с которыми люди сталкиваются при написании программ для последовательной передачи данных, часто не имеют никакого отношения к LabVIEW! Давайте бегло рассмотрим, как использовать ВП последовательной передачи данных для взаимодействия с прибором.

Сначала необходимо познакомиться с некоторыми основными концепциями работы последовательной коммуникации. Если вы использовали модем и знаете, что такое скорость двоичной передачи, стоповые биты и среднее значение четности, этого уже достаточно, чтобы начать работать. В противном случае, обратитесь к соответствующей литературе о последовательном порте (любая хорошая книга о протоколе RS-232).

Хорошим началом работы при последовательном взаимодействии с прибором, является использование примера **LabVIEW <-> Serial.vi** (рис. 11.64).

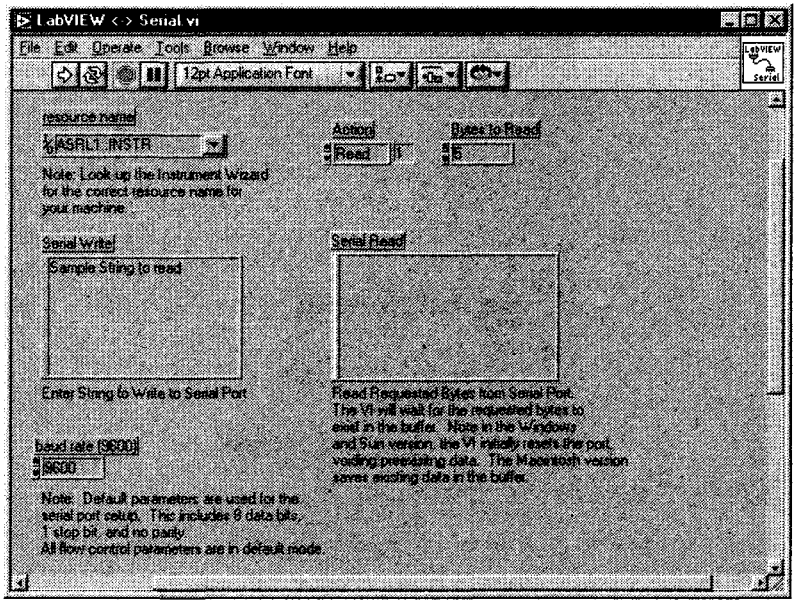


Рис. 11.64

11.4.4. Драйверы приборов

И наконец, никакое обсуждение элементов управления приборами не будет завершенным без упоминания о *драйверах*. Драйвер прибора представляет собой набор функций, которые выполняют команды, необходимые для осуществления операций прибора. Поскольку драйверы в LabVIEW упрощают программирование прибора до высокоуровневых команд, вам не придется заучивать таинственный, трудно запоминаемый низкоуровневый синтаксис, который призван управлять приборами. Применять драйверы необязательно; они просто экономят время при разработке вашего проекта и избавляют от необходимости изучать руководство по эксплуатации прибора перед написанием программы.

Драйверы создают команды для приборов и общаются с ними через последовательную шину, КОП или VXI. Кроме того, драйверы прибора получают, анализируют и масштабируют строки данных для использования в ваших тестовых программах, что значительно снижает время, необходимое для написания программы.

В качестве примера в LabVIEW имеется пара драйверов, таких как драйвер прибора **HP34401A MultiMeter**. Если вы посмотрите на палитру **Драйверы приборовов** (Instrument Drivers), то увидите, как выглядят функции драйвера прибора (рис. 11.65).

Хотя вам потребуется драйвер, который специально создан для вашей модели прибора, в LabVIEW можно использовать более 700 драйверов от более чем 50 компаний. Их список доступен по адресу zone.ni.com/idnet. Вы можете использовать эти драйверы для быстрого создания законченных проектов. Драйверы значитель-

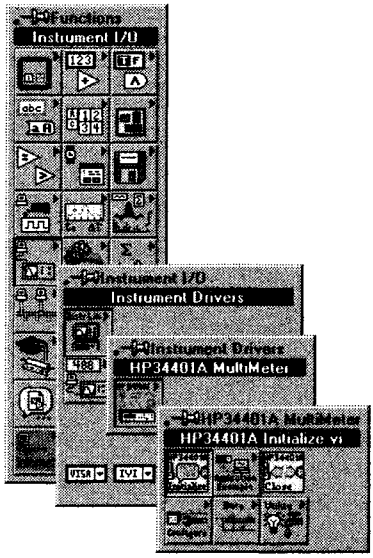


Рис. 11.65. Драйвер прибора HP34401A MultiMeter

но уменьшают время на создание программного обеспечения, поскольку допустимо сразу начать работу с прибором из LabVIEW.

11.5. Итоги

В этой главе ставилась цель помочь вам начать использование LabVIEW для сбора данных и познакомить с элементами управления приборами.

Мы рассмотрели понятия буферизации и запуска. Буферизация – это метод временного хранения выборок данных в определенном месте памяти (буфере). Она используется для достижения точной и быстрой частоты выборки. Запуск – это метод инициализации и окончания процесса сбора данных в зависимости от сигнала запуска. Запуск осуществляется с помощью программного обеспечения или внешнего прибора.

Палитры **Аналоговый ввод** и **Аналоговый вывод** содержат ВП для аналогового ввода/вывода различных уровней сложности. Виртуальные приборы верхнего уровня используются для простого ввода/вывода. Они дают возможность получить одну точку (небуферизованный аналоговый ввод) или целую осциллограмму (буферизованный аналоговый ввод) из одного или нескольких каналов. ВП среднего уровня обеспечивают более контролируемый процесс сбора данных. С помощью этих ВП вы можете выполнить определенную последовательность действий, таких как настройка, запуск, считывание и очистка операции. Эти ВП

позволяют осуществить буферизованный ввод/вывод, аппаратный запуск и потоковую запись данных на диск. Виртуальные приборы нижнего уровня обычно не предназначены для начинающих.

Палитра **Цифровой ввод/вывод** содержит несколько простых виртуальных приборов для считывания и записи данных в цифровые линии плат ввода/вывода. Допустимо считать или записать данные либо в одну линию, либо в целый порт (четыре или восемь линий в зависимости от платы).

Палитра **VISA** дает возможность взаимодействия практически с любым видом приборов, независимо от шины. Иногда вам понадобится использовать подпалитры **GPIB** или **Serial**, находящиеся на один уровень ниже. Если же у вас есть драйвер, разработанный для вашего прибора, то вы уже на полпути к созданию законченного приложения.

Обзор

Эта глава продемонстрирует вам, как использовать некоторые наиболее эффективные расширенные функции и структуры LabVIEW. LabVIEW предоставляет возможность создавать, сохранять и манипулировать локальными и глобальными переменными, подобно традиционным языкам программирования Вы также узнаете, как сделать элементы управления и индикации более гибкими и удобными, используя их узлы свойств (property nodes), которые определяют внешний вид и поведение объектов лицевой панели. Дополнительно эта глава содержит описание ряда таких вспомогательных расширенных функций, как системные вызовы (system calls), вызов и генерация внешнего программного кода и диалогов. Наконец, вам будут представлены расширенные возможности преобразования данных и указаны случаи, в которых они могут понадобиться.

ЗАДАЧИ

- Понять принципы работы с локальными и глобальными переменными, а также как и когда ими следует пользоваться
- Научиться настраивать внешний вид и поведение объектов лицевой панели с использованием узлов свойств
- Изучить некоторые вспомогательные расширенные функции
- Познакомиться с методами взаимодействия LabVIEW с внешним программным кодом
- Научиться проводить расширенные преобразования различных типов данных

ОСНОВНЫЕ ТЕРМИНЫ

- Локальная переменная
- Глобальная переменная
- Режим чтения и записи
- Условие соревнования (Race Condition)
- Узел свойств (Property Node)
- Библиотека динамических связей (Dynamic Links Library – DLL)
- Узел взаимодействия с программным кодом (Code Interface Node – CIN)
- Вызов библиотеки (Call Library)
- ASCII Строка
- Двоичная строка (Binary String)
- Подгонка типов (Typecasting)

РАСШИРЕННЫЕ СТРУКТУРЫ И ФУНКЦИИ LabVIEW

12

Каждая программа содержит по крайней мере одну ошибку и может быть сокращена по крайней мере на одну инструкцию – таким образом, по индукции, каждую программу можно сократить до одной инструкции, которая не будет работать...

Неназванный профессор информатики

12.1. Локальные и глобальные переменные

Локальные и глобальные переменные в LabVIEW с технической точки зрения являются структурами. Если вам когда-либо доводилось программировать на обычных алгоритмических языках типа C или Паскаля, то вы уже знакомы с понятиями локальной и глобальной переменных. До настоящего времени мы считывали данные с объекта лицевой панели либо записывали их через его терминал на блок-диаграмме. Однако объект лицевой панели имеет лишь один терминал на блок-диаграмме, а вам может понадобиться обновлять показания или считывать данные с объекта лицевой панели из различных точек блок-диаграммы или из другого виртуального прибора.

Локальные переменные (local variables, или locals) обеспечивают доступ к объектам лицевой панели из различных точек блок-диаграммы одного и того же виртуального прибора в тех случаях, когда вы не имеете возможности или не хотите подключать проводник к терминалу объекта.

Глобальные переменные (global variables, или globals) предоставляют доступ к данным любого типа (или нескольким типам данных одновременно, если это потребуется) среди нескольких ВП в случаях, когда вы не можете подключиться через узлы виртуальных подпрограмм или когда несколько виртуальных приборов одновременно выполняются и обмениваются данными. Во многом глобальные переменные подобны локальным, однако область их действия не ограничивается

одним ВП, то есть глобальные переменные могут переносить данные между несколькими ВП.

Этот раздел научит вас пользоваться локальными и глобальными переменными, а также покажет возможные ошибки, которых следует избегать.

12.1.1. Локальные переменные

Локальные переменные в LabVIEW представляют собой встроенные объекты, доступ к которым осуществляется из подпалитры **Структуры** (Structures) палитры **Функции**. При выборе объекта **локальная переменная** на блок-диаграмме вначале появляется узел, помеченный знаком вопроса (?), который указывает на неопределенность локальной переменной. Если щелкнуть инструментом управления («палец») по этому узлу, появляется список всех текущих индикаторов и элементов управления; выбор одного из них определяет значение локальной переменной. Вы также можете вызвать контекстное меню этой локальной переменной (щелкнув правой кнопкой мыши) и отметить опцию **Выбрать элемент** (Select Item) для доступа к этому списку. Еще один метод создания локальной переменной заключается в вызове контекстного меню терминала интересующего объекта и выборе опции **Создать** ⇒ **Локальная переменная** (Create ⇒ Local Variable).

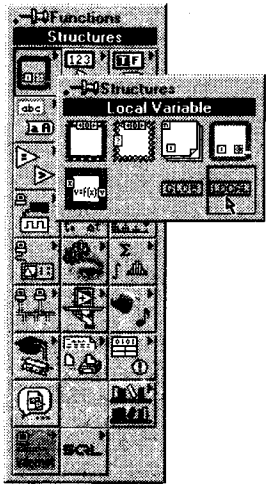


Рис. 12.1

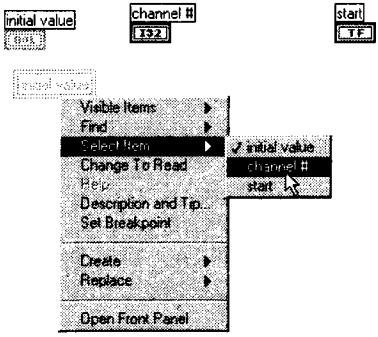


Рис. 12.2

Существует по крайней мере две причины, согласно которым вы будете использовать локальные переменные в виртуальном приборе:

- возможность создать алгоритмы, такие, например, как управление параллельными циклами посредством одной переменной, которые вы не сумеете реализовать другим способом;
- любой элемент управления может виртуально работать как индикатор и наоборот.

Управление параллельными циклами

Ранее мы рассмотрели, каким образом LabVIEW управляет порядком выполнения алгоритма через *поток данных* (dataflow). Идея следования выполнения по потоку данных является одной из причин того, что LabVIEW столь интуитивно понятен и так легок для программирования. Однако могут возникнуть ситуации (а если вы разрабатываете крупное серьезное приложение, то они *наверняка* возникнут), когда потребуется считывать данные с индикаторов или элементов управления на лицевой панели или записывать данные в них без непосредственного подключения проводников к терминалам этих объектов. Классический пример такой проблемы изображен на рис. 12.3. Здесь нужно завершить выполнение двух независимых циклов по условию с помощью одного логического элемента управления **Стоп**.

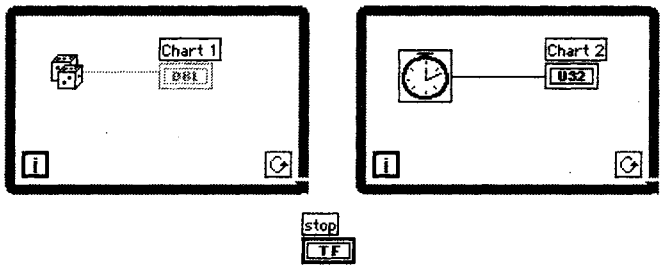


Рис. 12.3

Как это можно осуществить? Кто-то может сказать, что следует просто подключить кнопку **Стоп** к условным терминалам циклов. Однако подумайте о том, как часто циклы будут проверять состояние кнопки **Стоп**, если она подключена за пределами циклов.

Подключение кнопки **Стоп** за пределами циклов к обоим условным терминалам не сработает, поскольку состояние элементов управления, размещенных за пределами циклов, не считывает с момента входа в тело цикла. В этом случае циклы будут выполняться только один раз, если кнопка **Стоп** имеет логическое

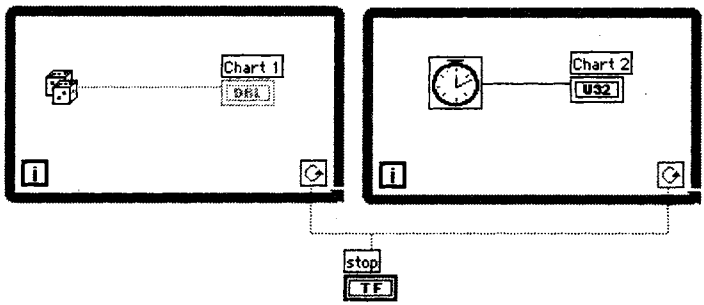


Рис. 12.4

состояние ЛОЖЬ (False) к моменту начала выполнения цикла или будут выполняться бесконечно, если она находится в первоначальном состоянии ИСТИНА (True).

Тогда почему бы не поместить кнопку **Стоп** внутрь одного из циклов, как показано на рис. 12.5? Будет ли эта схема работать?

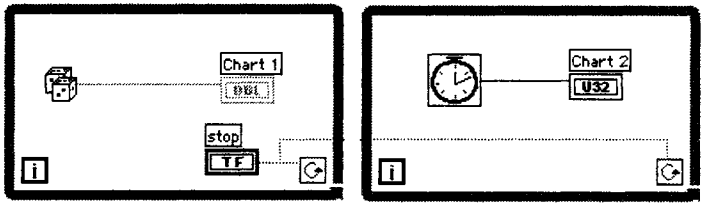


Рис. 12.5

Размещение кнопки **Стоп** внутри одного из циклов и создание проводника от нее к условному терминалу другого цикла не произведет желаемого эффекта по тем же причинам. Более того, второй цикл даже не начнет выполняться до тех пор, пока не завершится первый цикл (наверное, вы уже вспомнили *принцип управления через поток данных?*).

Решением этой дилеммы является использование локальной переменной. Локальные переменные создают «копию» данных из другого терминала на блок-диаграмме и всегда содержат актуальные значения объектов лицевой панели, с которыми они ассоциированы. Таким образом, вы можете получить доступ к элементу управления или индикатору в более чем одной точке вашей блок-диаграммы без подключения проводников к его терминалу.

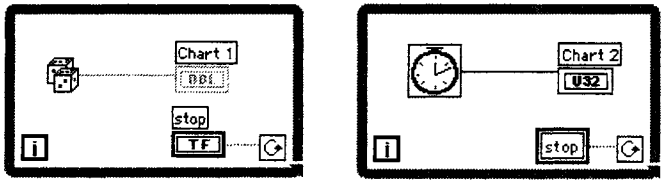


Рис. 12.6

Возвращаясь к предыдущему примеру, мы можем теперь использовать одну кнопку **Стоп** для управления обоими циклами путем подсоединения ее логического терминала к условному терминалу одного из циклов и подключения локальной переменной, ассоциированной с терминалом кнопки **Стоп**, к условному терминалу другого цикла.

Существует условие для создания локальной переменной булевского типа: объект лицевой панели не может быть переведен в режим защелки (Latch), устанавливаемый опцией Механическое действие (Mechanical Action). Хотя на первый взгляд это требование и не очевидно, булевский объект-защелка в совокупности с ассоциированной с ним локальной переменной в режиме чтения приводит к неоднозначной ситуации. Поэтому при попытке создания локальной переменной булевского объекта-защелки LabVIEW переходит в состояние «сломанная стрелка», не позволяющее вам запустить такой виртуальный прибор.

Стираем различия между элементом управления и индикатором

Одной из действительно приятных особенностей локальных переменных является то, что они дают возможность *записывать* данные в *элемент управления* и *считывать* данные с *индикатора*, чего вы не можете сделать с обычными терминалами объекта. Локальные переменные имеют два режима (mode): *чтение* (READ) и *запись* (WRITE). Терминал локальной переменной способен находиться только в одном из этих режимов, но допустимо создать второй локальный терминал к той же самой переменной в другом режиме. Понятие режима является достаточно очевидным: в режиме чтения можно считывать значение с терминала локальной переменной, как вы бы считали значение с обычного элемента управления; в режиме записи можно записать данные в терминал локальной переменной, как если бы вы обновили показания обычного индикатора. Запомните следующее правило для подключения локальных переменных:

Режим ЧТЕНИЯ = ЭЛЕМЕНТ УПРАВЛЕНИЯ	READ mode = CONTROL
Режим ЗАПИСИ = ИНДИКАТОР	WRITE mode = INDICATOR

Еще один вариант интерпретации заключается в рассмотрении локальной переменной в режиме чтения как *источника данных*, а в режиме записи – как *приемника данных*.

Вы можете установить локальную переменную в режим чтения либо записи, щелкнув правой кнопкой мыши по терминалу локальной переменной и выбрав опцию **Изменить на** (Change To). Терминал локальной переменной в режиме чтения обведен более жирной линией, чем в режиме записи (точно так же, как элемент управления имеет толстую окантовку в отличие от индикатора) – рис. 12.7. Будьте внимательны к толщине границ терминала при подключении локальных переменных, чтобы

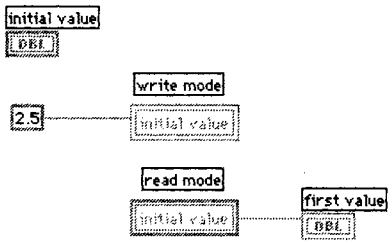


Рис. 12.7

быть уверенными в правильности выбора режима. Если вы попытаетесь записать данные в локальную переменную, установленную в режим чтения, то получите неисправный проводник – и будете долго ломать голову в поисках причины ошибки.

Последнее, но важное замечание: вы должны присвоить ярлык элементу управления или индикатору, которому соответствует локальная переменная. Это значит, что, создавая элемент управления или отображения и не присваивая ему имя-ярлык (что вполне допустимо в LabVIEW), вы впоследствии не сможете ассоциировать с ним локальную переменную.

В качестве простого упражнения создадим ручку управления таймером: пользователь может установить время выдержки и наблюдать, как ручка поворачивается при обратном отсчете времени – таким образом работали старинные кухонные таймеры. Очевидно, ручка, как любой объект лицевой панели, должна быть доступна пользователю для управления (установки выдержки), но затем она должна воспринимать значения, вычисляемые на блок-диаграмме, и поворачиваться соответственно истечению времени.

Упражнение 12.1: использование локальных переменных

1. На лицевой панели создайте ручку управления и выберите опцию **Видимые элементы ⇒ Цифровой дисплей** (Visible Items ⇒ Digital Display), как показано на рис. 12.8.
2. Создайте локальную переменную, выбрав ее в палитре **Структуры** (Structures). Получив на диаграмме иконку локальной переменной, щелкните по ней инструментом управления («палец») и выберите вариант **Таймер (сек)** – Timer (sec). По умолчанию локальная переменная находится в режиме записи.
3. Постройте простую блок-диаграмму, показанную на рис. 12.9.
4. Сохраните ваш виртуальный прибор как **Kitchen Timer.vi**.

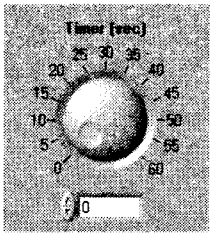


Рис. 12.8

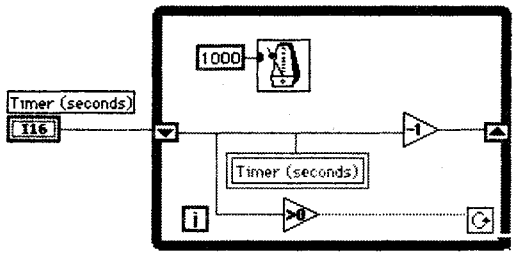


Рис. 12.9

В этом примере сдвиговый регистр инициализируется значением **Таймер (сек)**, которое было установлено пользователем с лицевой панели. Как только цикл начинает выполняться, значение сдвигового регистра уменьшается на единицу один раз в секунду, и это значение передается в локальную переменную **Таймер (сек)**,

которая находится в режиме записи. Ручка управления на лицевой панели вращается, отображая измененное значение.

Можно сделать забавное дополнение к этому примеру – звуковой сигнал, который даст вам знать, что время выдержки истекло, то есть таймер достиг нуля – подобный звуковой сигнал подают настоящие таймеры. Далее в этой главе вы узнаете, как создавать простые звуковые сигналы с помощью LabVIEW.

Локальные переменные представляются отличными структурами для пользования, и это действительно так! Однако будьте внимательны при работе с ними: вас подстерегает распространенная ошибка – условия соревнования. Условия соревнования наступают, когда две или более копии локальной переменной в режиме записи могут быть записаны одновременно.

Существует некоторая опасность при использовании локальных и глобальных переменных: случайное создание условия соревнования. Чтобы понять это, посмотрите на рис. 12.10 и 12.11. Обратите внимание на два цикла по условию, которые управляются кнопкой **Пуск** и локальной переменной **Пуск**. Однако запись в локальную переменную **Ввод осуществлен** производится как в левом, так и в правом цикле. Теперь запустите этот ВП со значением **Пуск**, установленным в логическое состояние ЛОЖЬ и с различными значениями двух ползунковых регуляторов, **Ввод А** и **Ввод Б**. Циклы выполнятся только один раз. Какое число

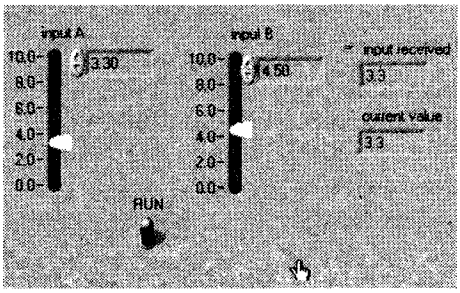


Рис. 12.10

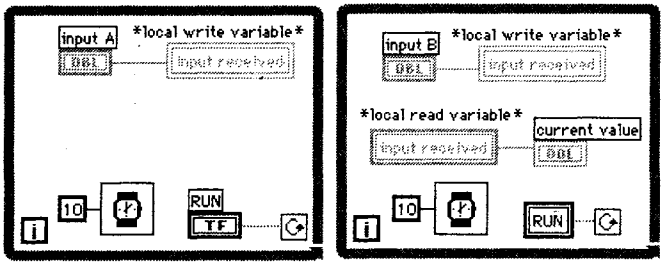


Рис. 12.11

появится на индикаторе **Текущее значение**? Нельзя сказать точно, поскольку это может быть число либо с **Ввод А**, либо с **Ввод Б**. В LabVIEW нет возможности задавать «геометрический» порядок выполнения: ни справа налево, ни сверху вниз.

Если вы циклически запустите этот ВП с включенной кнопкой **Пуск**, то увидите, что показания индикатора **Текущее значение** будут перескакивать со значения одного ввода на значение другого. Чтобы избежать условий соревнования, как в нашем примере, установите порядок выполнения с помощью потока данных, структур последовательности или других, более сложных систем.

Следует также иметь в виду, что при каждом чтении или записи локальной переменной в памяти создается копия этих данных. Таким образом, при работе с локальными переменными необходимо проверять блок-диаграмму и предполагаемый порядок выполнения алгоритма, чтобы избежать условий соревнования: вообще же старайтесь ограничивать использование локальных переменных, чтобы снизить потребность в памяти.

Упражнение 12.2: еще раз об удобствах локальных переменных

Еще один случай, когда удобны локальные переменные: вы хотите создать более дружественный интерфейс виртуального прибора с индикатором статуса. Например, текстовый индикатор, на котором появляются различные сообщения или запросы пользователю каждый раз, когда в системе наступают определенные события. Создайте простой ВП сбора данных, идентичный тем, которые вы создавали в главе 11, но измените его так, чтобы появился текстовый индикатор, выдающий пользователю следующие сообщения:

- «Программа ожидает ввода данных»;
- «Программа осуществляет ввод данных»;
- «Программа отображает данные на графике»;
- «Программа завершает выполнение».

Рис. 12.12–12.14 изображают лицевую панель и два первых кадра структуры последовательности.

Сохраните ваш виртуальный прибор как **Status Indicator.vi**.

Упражнение 12.3: локальные переменные – еще больше удобств

В ряде приложений вам может понадобиться нечто вроде «главного» элемента управления (master control), который изменял бы значения других элементов управления. Например, нужно создать простую программу для управления уровнем звука стереосистемы. Предположим, что компьютер каким-то способом подключен к элементу управления мощностью усилителя. В виртуальном приборе, изображенном на рис. 12.15, созданная на компьютере панель управления звуком имеет три ползунковых регулятора: **Уровень левого канала**, **Уровень правого канала** и **Общая громкость**. Уровни левого и правого каналов могут быть установлены независимо друг от друга; перемещение главного регулятора пропорционально увеличивает или уменьшает мощность правого и левого каналов.

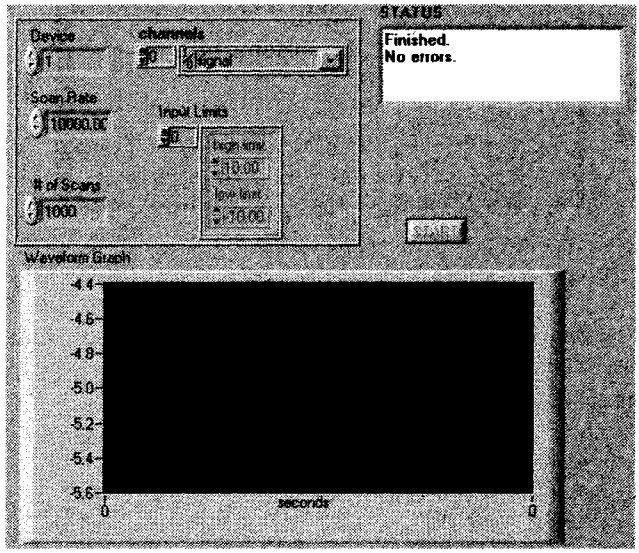


Рис. 12.12

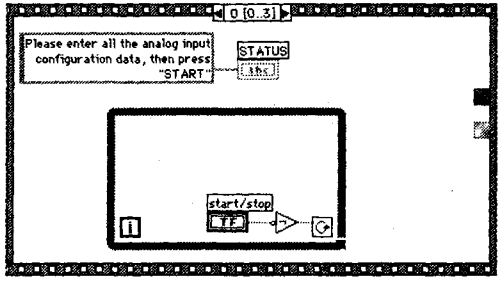


Рис. 12.13

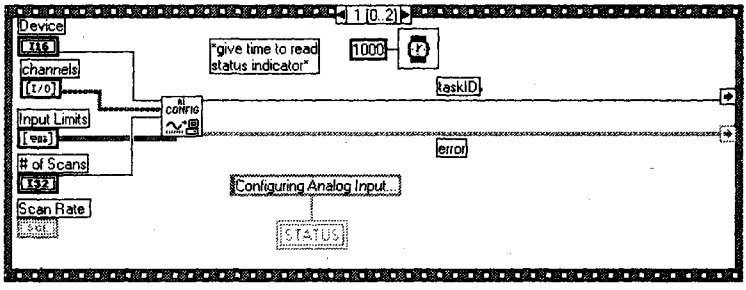


Рис. 12.14

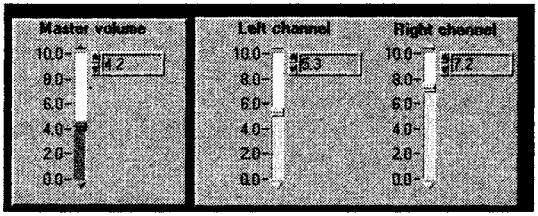


Рис. 12.15

Создайте блок-диаграмму для лицевой панели, изображенной на рис. 12.15. Забавно, что при перемещении ползунка главного регулятора передвигаются и ползунки двух других.

Сохраните ваш виртуальный прибор как **Master and Slave.vi**.

Для выполнения этого упражнения используйте сдвиговые регистры.

12.1.2. Глобальные переменные

Если вы не использовали глобальные переменные ранее – примите наши поздравления! Глобальные переменные являются наиболее редко и наиболее неправильно применяемыми структурами программирования. Они часто являются причиной таинственных ошибок, неожиданного поведения программ, да и вообще эта структура программирования довольно неуклюжа. Хотя вышесказанное – правда, не исключены случаи, когда вам понадобятся глобальные переменные. (Дело не в том, что глобальные переменные плохи как таковые, а в том, что пользоваться ими нужно весьма осторожно.)

Как вы помните, для доступа к объектам лицевой панели, расположенным в разных местах на блок-диаграмме, можно задействовать локальные переменные. Локальные переменные предоставляют доступ к данным лишь внутри одного виртуального прибора. Предположим, вам необходимо передавать данные между несколькими виртуальными приборами, которые запускаются одновременно или их иконки как виртуальных подпрограмм не могут быть соединены проводниками на единой блок-диаграмме. В этой ситуации удобны глобальные переменные. Во многом глобальные переменные сходны с локальными, но их область действия не ограничена одним ВП – глобальные переменные могут передавать данные между несколькими виртуальными приборами.

Рассмотрим следующий пример. Предположим, что имеется два ВП, которые выполняются одновременно. Каждый ВП пишет данные поточечно из некоего источника сигнала в индикатор-осциллограф. Первый виртуальный прибор также содержит логическую кнопку **Питание** (Power) для прекращения выполнения обоих виртуальных приборов. Вспомните, что когда оба цикла находились на одной и той же блок-диаграмме, мы были вынуждены задействовать *локальную* переменную для остановки обоих циклов. Теперь, когда каждый цикл находится в отдельном виртуальном

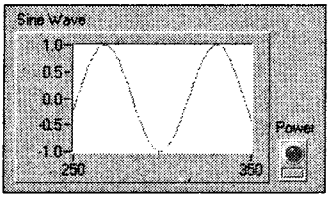


Рис. 12.16. Лицевая панель первого ВП

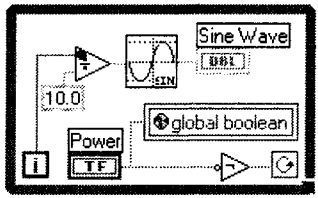


Рис. 12.17. Блок-диаграмма первого ВП

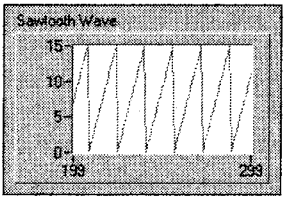


Рис. 12.18. Лицевая панель второго ВП

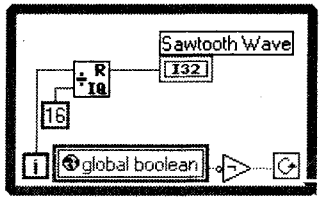


Рис. 12.19. Блок-диаграмма второго ВП

приборе, придется использовать *глобальную* переменную. Обратите внимание, что терминал глобальной переменной внешне похож на терминал локальной, но дополнительно содержит маленькую пиктограмму с изображением земного шара.

Создание глобальных переменных

Глобальные переменные, как и локальные, являются структурой LabVIEW, доступ к которой осуществляется из подпалитры **Структуры**. Подобно локальным переменным, каждый терминал глобальной переменной может быть установлен либо в режим чтения, либо в режим записи. В отличие от локальных переменных, различные ВП могут независимо вызывать одну и ту же глобальную переменную. Глобальные переменные являются эффективным способом обмена данными между несколькими виртуальными приборами без необходимости подключения проводников данными от одного ВП к другому – глобальные переменные хранят информацию независимо от отдельных виртуальных приборов. Если один ВП записал значение в глобальную переменную, то любой ВП или ВПП, осуществляющий чтение из этой глобальной переменной, получит только что записанное значение.

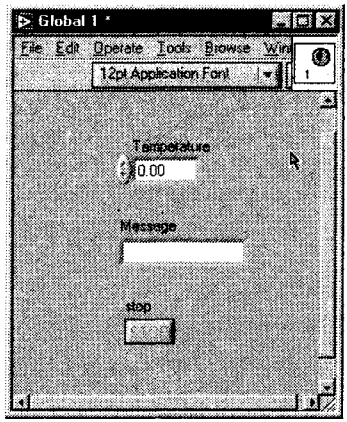


Рис. 12.20

Как только структура глобальной переменной выбрана из палитры, на блок-диаграмме появляется пиктограмма . Она символизирует глобальную переменную, которая еще не определена. Дважды щелкнув мышью по ее иконке, вы вызовете окно, идентичное лицевой панели самостоятельного виртуального прибора. Воспринимайте глобальные переменные как особый вид виртуального прибора – они могут содержать любой тип и любое количество структур данных на лицевой панели, но не имеют соответствующей блок-диаграммы. Глобальные переменные хранят данные, но самостоятельно не производят с ними каких-либо операций. Размещение элементов управления или индикаторов на лицевой панели глобальной переменной осуществляется точно так же, как и для обычного виртуального прибора. Интересной особенностью глобальных переменных является отсутствие принципиальной разницы в выборе элемента управления или индикатора для представления определенного типа данных, поскольку вы можете как считывать из глобальных переменных, так и записывать в них. Опять же, как и в случае с локальными переменными, не забывайте снабжать ярлыками все объекты в ваших глобальных переменных, в противном случае вы не сможете их использовать.

Как показано в следующем примере, глобальная переменная может содержать числовые данные, логическую кнопку **Стоп** и строковый элемент управления.

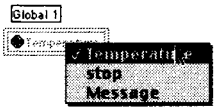


Рис. 12.21

Сохраните глобальную переменную так же, как вы сохраняете виртуальный прибор. Многие разработчики применяют расширение .gbl для глобальных переменных, чтобы их отслеживать. Для того чтобы использовать сохраненную глобальную переменную в блок-диаграмме, укажите опцию **Выбрать ВП** (Select a VI) в палитре **Функции**. При этом на блок-диаграмме появляется терминал, показывающий одну из переменных, которые содержатся в глобальной переменной. Чтобы выбрать нужную переменную, щелкните правой кнопкой мыши по терминалу и укажите опцию **Выбрать элемент** (Select Item) или просто щелкните по терминалу инструментом управления («палец»). С одного терминала глобальной переменной вы можете выбрать только одну из содержащихся в ней переменных. Для использования другой переменной или другого элемента глобальной переменной создайте копию ее терминала с помощью перетаскивания при удерживании клавиши <ctrl> (или <option>).

Так же как и локальные, глобальные переменные могут находиться в режиме чтения или записи. Для выбора режима щелкните правой кнопкой мыши по терминалу и выберите опцию **Заменить на** (Change To). Глобальные переменные в режиме чтения выделены более толстыми границами, чем в режиме записи; они ведут себя в режиме чтения аналогично элементам управления, а в режиме записи – аналогично индикаторам. Соответственно глобальная переменная в режиме чтения является источником данных, а в режиме записи – приемником данных.

Режим ЧТЕНИЯ = ЭЛЕМЕНТ УПРАВЛЕНИЯ	READ mode = CONTROL
Режим ЗАПИСИ = ИНДИКАТОР	WRITE mode = INDICATOR

Вот несколько важных советов по использованию глобальных переменных:

- всегда инициализируйте глобальные переменные в блок-диаграмме. Их начальное значение должно быть определено в коде. Глобальные переменные не сохраняют своих значений по умолчанию, пока вы не перезапустите LabVIEW;
- никогда не осуществляйте чтение и запись одной и той же глобальной переменной в одном и том же месте диаграммы: может наступить неопределенность порядка чтения/записи, которая приведет к неопределенности данных, то есть к *условию соревнования*;
- поскольку глобальные переменные могут хранить несколько переменных различных типов, лучше группировать данные в одну глобальную переменную, а не создавать несколько глобальных переменных.

Важно обращать внимание на имена, которые вы даете переменным внутри глобальных переменных. Все виртуальные приборы, которые вызывают глобальную переменную, будут ссылаться на конкретную переменную по ее имени; поэтому не следует давать одинаковые имена элементам управления или индикаторам.

Пример

Рассмотрим проблему, аналогичную двум независимым циклам по условию. Предположим, в отличие от предыдущих случаев, что вместо двух независимых циклов по условию на одной блок-диаграмме мы имеем два независимых виртуальных подприбора, которые выполняются одновременно.

На рис. 12.22–12.25 показаны иконки и лицевые панели двух ВПП: **Генерация отсчетов времени** (Generate Time) и **Построение графика** (Plot). Эти ВПП созданы для одновременного выполнения – виртуальный подприбор **Генерация отсчетов времени** непрерывно собирает показания внутренних часов в миллисекундах, начиная с момента запуска. ВПП **Построение графика** ежесекундно генерирует случайные числа до тех пор, пока не будет нажата кнопка **Стоп**. После этого он считывает все отсчеты времени из ВПП **Генерация отсчетов времени** и строит график случайных чисел в зависимости от моментов времени, в которые эти числа были получены.

Рис. 12.22

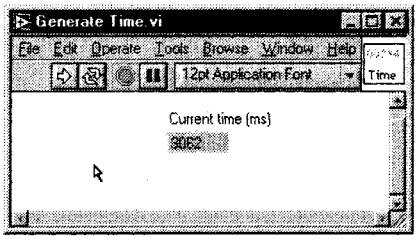


Рис. 12.23

Рис. 12.24

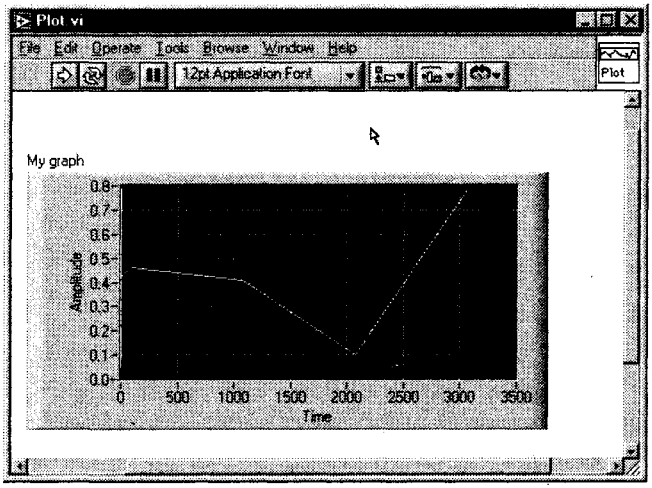


Рис. 12.25

Как показано в следующем примере, глобальная переменная может содержать числовые данные, логическую кнопку **Стоп** и строковый элемент управления.

Способ, которым эти два ВПП обмениваются данными, основан на использовании глобальных переменных. Требуется, чтобы ВПП **Построение графика** собирал массив отсчетов времени, полученных виртуальным подприбором **Генерация отсчетов времени**, и, более того, чтобы оба подприбора были остановлены одним логическим элементом управления.

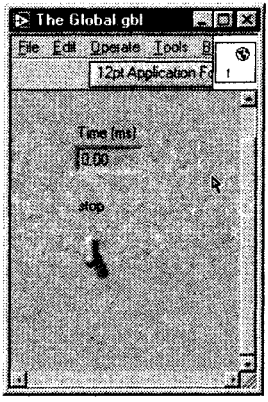


Рис. 12.26

Итак, вначале нужно создать одну глобальную переменную, содержащую две необходимые переменные. Как вы помните, для создания новой глобальной переменной надо выбрать структуру **Глобальная переменная** (Global Variable) в палитре **Структуры** (Structures), затем дважды щелкнуть мышью по иконке глобуса для определения ее компонентов. В этом случае мы определяем числовую переменную **Время (мс)** и булевскую **Стоп**. Сохраните созданную структуру как **The Global.gbl** – рис. 12.26.

Затем мы используем переменные, являющиеся компонентами глобальной, в соответствующих точках обоих виртуальных подприборов.

Обратите внимание на то, как работает логическая переменная **Стоп**. Кнопка **Стоп** виртуального подприбора **Построение графика** записывает логическое состояние в переменную **Стоп** глобальной переменной, которая, в свою очередь, останавливает цикл по условию в ВПП **Генерация отсчетов времени**. Если кнопка **Стоп** нажата в ВПП **Построение графика**, она также остановит цикл в ВПП **Генерация отсчетов времени**. Аналогично значения отсчетов времени из ВПП **Генерация отсчетов времени** передаются в численную переменную

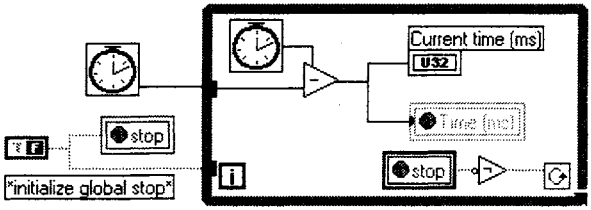


Рис. 12.27

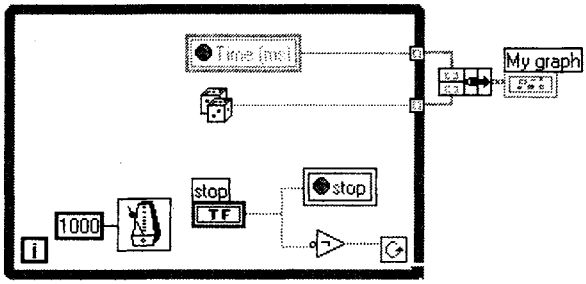


Рис. 12.28

Время (мс) глобальной переменной, которая вызывается виртуальным подприбором **Построение графика** для создания массива.

На примере двух виртуальных приборов вы освоили работу с глобальными переменными. Задачи из этого примера можно было решить и без помощи глобальных переменных, но в данном случае пример преследовал скорее иллюстративные цели.

Если вы взглянете на блок-диаграмму, которая вызывает два ВПП, то обнаружите другую проблему, связанную с использованием глобальных переменных: *нигде нет соединяющих проводников*! Глобальные переменные «размывают» видимость управляющего потока данных, поскольку нельзя увидеть взаимную связь между двумя подприборами. Даже если вы заметите глобальную переменную на блок-диаграмме, вы не будете знать, в каких еще точках она применяется. К счастью, начиная с версии LabVIEW 4.0 это неудобство устранено путем введения функции поиска точек обращения к глобальным переменным.

Вызвав контекстное меню терминала глобальной переменной, вы можете выбрать опцию **Найти ⇒ Определение глобальной переменной** (Find ⇒ Global Definition), которая доставит вас к точке лицевой панели, где определяется глобальная переменная (рис. 12.29). Другая опция, **Найти ⇒ Использование глобальной переменной** (Find ⇒ Global References), предоставит список всех виртуальных приборов, которые применяют эту глобальную переменную. Более подробно о возможностях поиска в LabVIEW рассказано в главе 13.

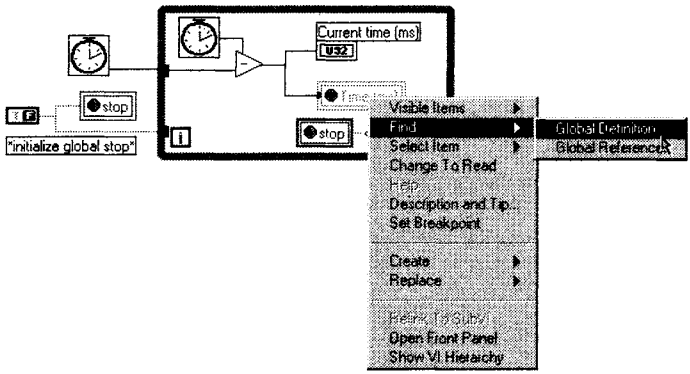


Рис. 12.29

12.2. Узлы свойств

С помощью *узлов свойств* (Property Nodes, в LabVIEW версии 5.1 и более ранних узлы атрибутов) вы можете создавать более мощные программы с более дружественным интерфейсом. Узлы свойств позволяют программно управлять свойствами объектов лицевой панели, такими как цвет, видимость, местоположение, формат представления чисел и т.д. Ключевым здесь является слово *программно*: изменение свойств объекта лицевой панели происходит в соответствии с алгоритмом, определенным на блок-диаграмме. Например, вы можете изменить цвет числового поля индикатора на голубой, зеленый или красный по мере увеличения его числового значения. Или выборочно предоставить пользователю различные элементы управления, определенные наборы которых будут возникать и исчезать в зависимости, например, от того, какие кнопки нажаты. Вы даже можете анимировать экран, заставив некий элемент лицевой панели перемещаться по рабочему пространству, символизируя тем самым некоторый физический процесс.

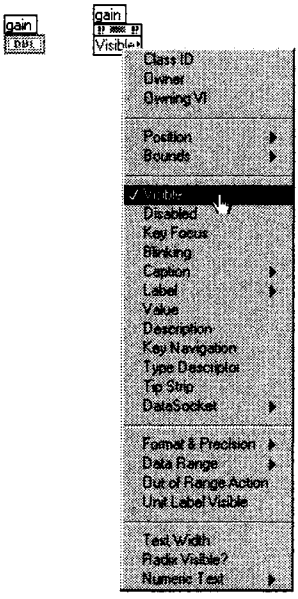


Рис. 12.30

Для того чтобы создать узел свойств, щелкните правой кнопкой мыши по объекту лицевой панели либо по его терминалу и выберите опцию **Создать ⇒ Узел свойств** (Create ⇒ Property node). На блок-диаграмме появится терминал с тем же именем, что и исходный объект. Чтобы увидеть, какие опции могут быть установлены с помощью узла свойств, щелкните по нему инструментом управления или вызовите

его контекстное меню и укажите опцию **Выбрать элемент** (Select Item). Теперь вы можете выбрать свойство или свойства для дальнейшего использования. Каждый объект имеет набор базовых свойств и дополнительный набор специфических свойств, относящихся к данному объекту.

Так же как с локальными переменными, вы можете либо считывать, либо записывать свойство объекта (хотя некоторые свойства предназначены только для чтения). Чтобы изменить режим свойства, вызовите контекстное меню и выберите опцию **Изменить на запись/чтение** (Change to Write/Read). Маленькая стрелка внутри терминала узла свойств укажет, какой режим используется в данный момент. Узел свойств в режиме записи имеет стрелку слева – значит, данные поступают в узел, записывая новое свойство. Узел свойств в режиме считывания имеет стрелку справа, считывая текущее свойство и предоставляя эти данные пользователю. Та же аналогия, которую мы использовали для локальных переменных, элементов управления (режим чтения) и индикаторов (режим записи), справедлива и для узлов свойств.

Интересной особенностью узлов свойств является то, что в одном терминале на блок-диаграмме можно установить или считать несколько свойств, но только одного из элементов управления или индикации. Чтобы добавить дополнительное свойство, нужно *изменить размер* терминала инструментом перемещения на заданное число полей свойств – аналогичным способом добавляются множественные вводы к таким функциям, как **Объединить** (Bundle), **Создать массив** (Build Array) и т.д. На рис. 12.31 показаны два свойства в одном и том же терминале для числового элемента управления **коэффициент усиления**.

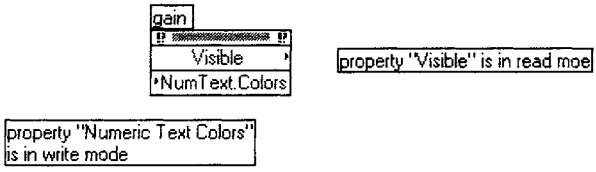


Рис. 12.31

Рассмотрим простой пример. Предположим, что нужно создать лицевую панель, на которой определенные специализированные элементы управления были бы спрятаны до момента, когда они понадобятся. На лицевой панели, изображенной на рис. 12.32, приведен измеритель механического напряжения (тензометр) и логический переключатель сигнала тревоги. Добавим кнопку **Показать/спрятать элементы расширенного управления** (Show/hide advanced controls), которая указывает на возможность появления органов управления некими специальными, на первый взгляд неясными и сложными параметрами.

В этом примере мы добавили два таких элемента управления: регуляторы **коэффициент усиления** и **смещение**, которые можно сделать невидимыми, установив свойство **Видимость** (Visible) в логическое состояние ЛОЖЬ, если кнопка **Показать/**

спрятать элементы расширенного управления не нажата. Если же эту кнопку нажать, появятся (то есть станут видимыми) две ручки управления – рис. 12.33.

Данная блок-диаграмма будет заключена в тело цикла по условию, как показано на рис. 12.34, чтобы оператор имел возможность нажать кнопку управления в ходе выполнения эксперимента.

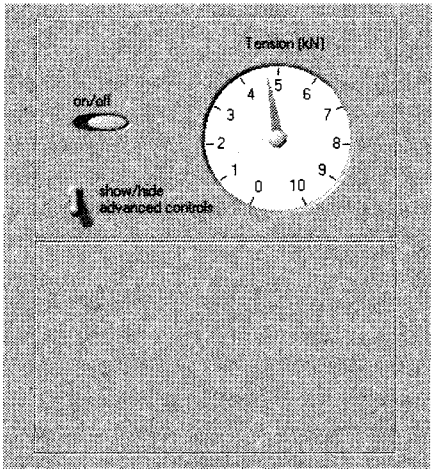


Рис. 12.32

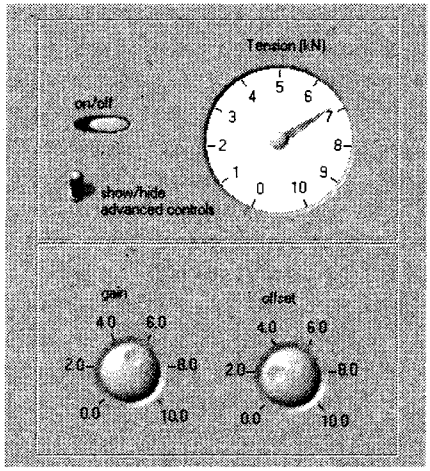


Рис. 12.33

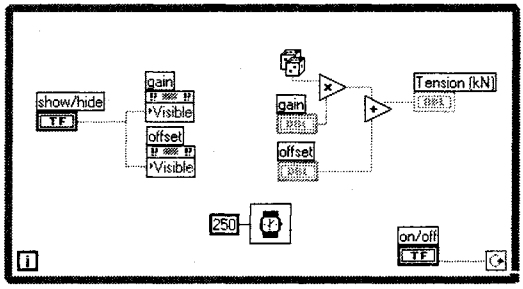


Рис. 12.34

Возможно, вам потребуется несколько опций в узле свойств объекта. Не забывайте, что вместо создания еще одного узла свойств к тому же объекту вы можете адресоваться сразу к нескольким опциям одновременно, просто увеличив терминал узла свойств с помощью инструмента перемещения – точно так же, как вы это делали при увеличении размерности кластера и массива. Изменяя размер, вы увидите последовательное появление новых полей опций; при желании измените их, щелкнув по любой из них инструментом управления («палец»).

Перечислим свойства, которые могут быть установлены основными опциями узла свойств:

- **Видимость** (Visible). Устанавливает или считывает статус видимости объекта на лицевой панели. Объект будет отображаться при установке свойства **Видимость** в состояние ИСТИНА; в состоянии ЛОЖЬ он будет спрятан. Управление видимостью объекта – лучшее решение, чем его «прозрачная» раскраска, поскольку к прозрачному объекту возможен случайный доступ;
- **Запрет** (Disabled). Задает или считывает статус прав доступа пользователя к элементу управления. Нулевое значение опции разрешает доступ пользователя к элементу управления; значение 1 запрещает доступ без какой-либо видимой индикации; значение 2 запрещает доступ к элементу управления и отмечает неактивность элемента серым цветом;
- **Фокусировка на объект** (Key Focus). При установке этого свойства в состояние ИСТИНА активность курсора автоматически переключается в поле указанного элемента управления. Вообще фокусировка переключается при табуляции через поля элементов управления. Эта опция будет полезна при создании приложения, не использующего мышь. Подробнее о фокусировке на объект читайте в главе 13;
- **Положение** (Position). Представляет собой кластер из двух чисел, которые соответственно определяют координаты верхней левой границы объекта на лицевой панели;
- **Размер** (Size). Представляет собой кластер из двух чисел, которые соответственно определяют высоту и ширину изображения данного объекта на лицевой панели;
- **Мерцание** (Blinking). При установке в состояние ИСТИНА изображение объекта на лицевой панели становится мерцающим;
- **Формат и точность** (Format and Precision). Устанавливает или считывает параметры формата и точности у соответствующих числовых элементов управления и отображения. Входной кластер содержит два целых числа: одно для формата, другое для точности. Это то же самое свойство, которое вы можете вызвать из обычного контекстного меню числового объекта;
- **Цвет** (Color). В зависимости от типа объекта свойство **Цвет** может иметь несколько вариантов. На вход свойства допустимо подключить массив полей установки цветов текста, фона, атрибутов и т.д. в зависимости от типа объекта.

Отметьте также, что каждый объект имеет свойства, называемые **Идентификатор класса** (ClassID), **Владелец** (Owner), **Владение ВП** (Owning VI). На данном этапе освоения программы можно не заботиться об этих свойствах. Они предназначены для реализации наиболее совершенных и относительно сложных сценариев программирования в LabVIEW.

Окно справки (Help) может помочь при использовании узлов свойств. Если вы наведете курсор на терминал узла свойств, окно справки покажет, что означает это свойство и какой тип данных оно ожидает. Щелкните правой кнопкой мыши по терминалу узла свойств и выберите опцию **Создать константу** (Create Constant) для создания константы с соответствующим типом данных – эта процедура очень удобна, особенно если ввод представляет собой сложный тип данных, например кластер.

Почти у всех элементов управления и индикации есть базовые свойства. Большинство элементов имеет внушительный набор свойств, особенно таблицы и графики (более 100 свойств!). Ряд свойств мы даже не будем упоминать – потому, что вы можете их не использовать, а также потому, что получить о них подробную информацию легко как во встроенной справочной системе LabVIEW, так и в руководствах пользователя. Лучший способ ознакомиться с узлами свойств – это создать

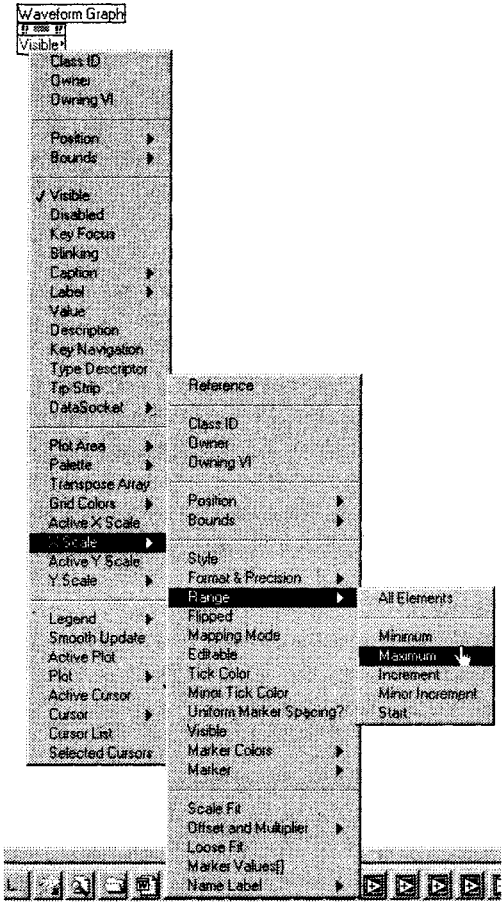


Рис. 12.35

некоторое их количество к элементам управления и индикации в приложении и поэкспериментировать с различными свойствами и опциями. Вы обнаружите, что узлы свойств весьма удобны для придания виртуальным приборам большей динамики, гибкости и дружественности (в особенности для того, чтобы впечатлить вашего шефа-«чайника»).

Другой пример

Графики и диаграммы имеют огромное количество опций в своих узлах свойств, в чем вы можете убедиться, вызвав щелчком правой кнопки мыши контекстное меню на терминале узла свойств развертки (рис. 12.35).

Следующий пример, который встроен в полную версию LabVIEW (группа примеров использования узлов свойств), показывает возможность программного управления одним из многих режимов отображения информации на графике. ВП **Chart Property Node** иллюстрирует возможность установки одного из трех

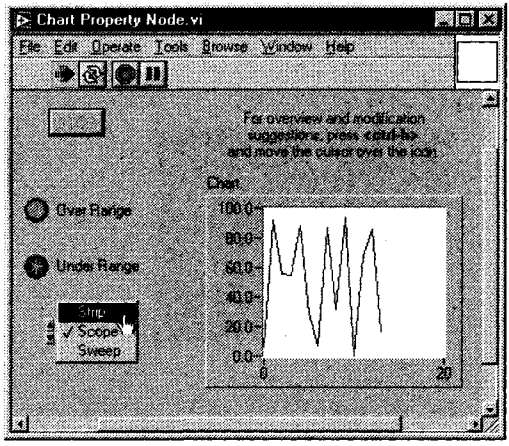


Рис. 12.36

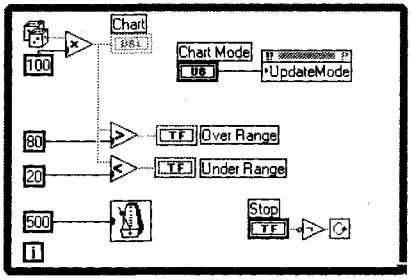


Рис. 12.37. Внутри цикла расположен узел свойств, который меняет режим обновления развертки согласно установке элемента циклического перебора, расположенного на лицевой панели

способов развертки: **Strip**, **Scope** и **Sweep** (подробнее об этих свойствах рассказывалось в главе 8).

Вы можете выбрать режим отображения графика и наблюдать за его изменением непосредственно во время работы виртуального прибора, установив свойство **Режим обновления** (Update Mode) в узле свойств графика.

12.2.1. Упражнение 12.4: использование узлов свойств с развертками

Создайте виртуальный прибор, который отображал бы три канала данных (экспериментальных данных, введенных в реальном времени с платы аналогового ввода/вывода, либо просто случайных чисел). Обеспечьте пользователю возможность «включать» или «выключать» любую из этих кривых с помощью трех кнопок. Поскольку индикатором здесь является развертка, а не график, то данные будут накапливаться, даже если виртуальный прибор закрыт и вновь открыт. Добавьте кнопку **Стереть**, которая очистит развертку.

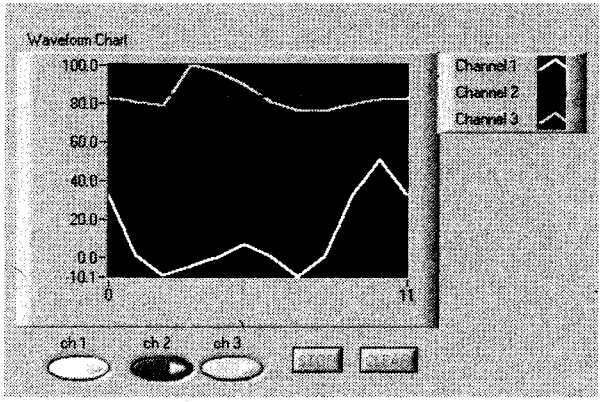


Рис. 12.38

В LabVIEW существует прозрачный цвет. Для задания цветности служит кластер цветовых констант. Цветовые константы имеют цифровой тип данных, представленный в палитре **Числовые ⇒ Дополнительные числовые константы** (Numeric ⇒ Additional Numeric Constants) – рис. 12.39. Для установки цвета вызовите контекстное меню цветовой константы (не забывайте, что «T» означает прозрачный цвет).

При использовании многолучевого графика или диаграммы вы можете изменить свойства лишь одной кривой за раз. Кривые графиков нумеруются как *0, 1, ..., n* именно для применения в узлах свойств. Особое свойство, называемое **Активная кривая** (Active Plot), предназначено для выбора кривой, свойства которой вы

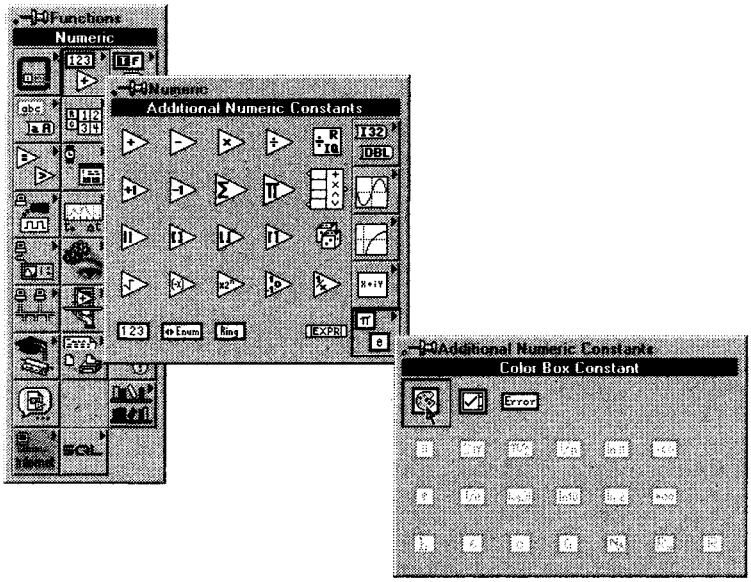


Рис. 12.39

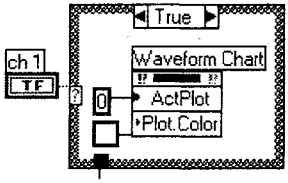


Рис. 12.40

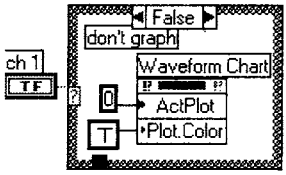


Рис. 12.41

хотите изменить или считать. В этом упражнении вам придется создать структуры вариантов, подобные приведенным на рис. 12.40 и 12.41.

Для очистки диаграммы используйте свойство **История** (History Data). Затем просто подключите к этому свойству пустой массив.

Сохраните ваш виртуальный прибор как **Property Nodes-Graph.vi**.

12.2.2. Упражнение 12.5: использование узлов свойств для создания динамических меню

Циклические элементы управления в сочетании с узлами свойств помогут создать эффективные дружественные виртуальные приборы. Обратимся к примеру **Меню с циклическими элементами управления** (Ring Control Menus).

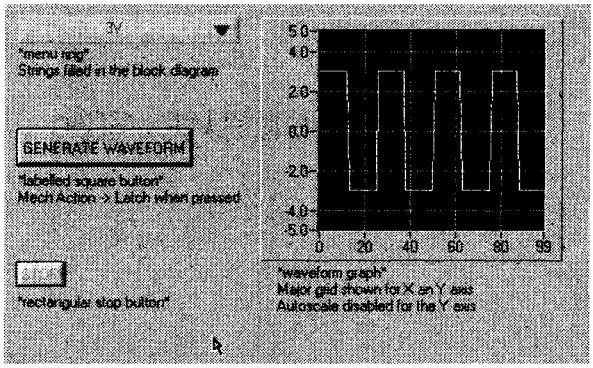


Рис. 12.42

Выпадающее меню с циклическим перебором вариантов, первоначально пустое, используется для того, чтобы запросить оператора о типе выходного сигнала. Как только будет нажата кнопка **Генерировать осциллограмму**, активизируется циклический элемент управления с запросом «Выберите тип осциллограммы», где предлагается несколько вариантов: **Синусоидальная** (Sine Wave), **Прямоугольная** (Square Wave) или **Пилообразная** (Sawtooth Wave) — рис. 12.43.

Затем тот же самый циклический элемент управления *будет изменен* для того, чтобы пользователь смог выбрать амплитуду выходного сигнала. Наконец, осциллограмма отображается на графике и соответствующий выходной сигнал генерируется универсальной платой ввода/вывода (рис. 12.44).

На рис. 12.45 приведен фрагмент блок-диаграммы, реализующей описанный алгоритм.

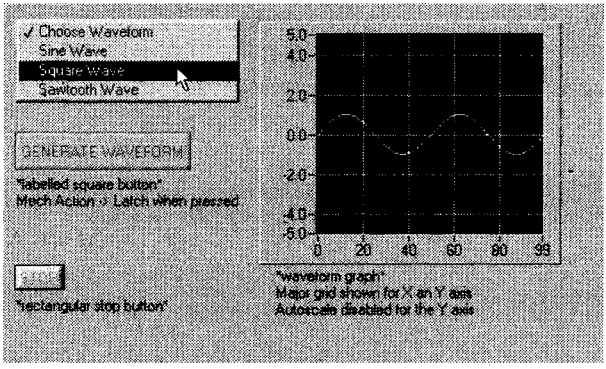


Рис. 12.43

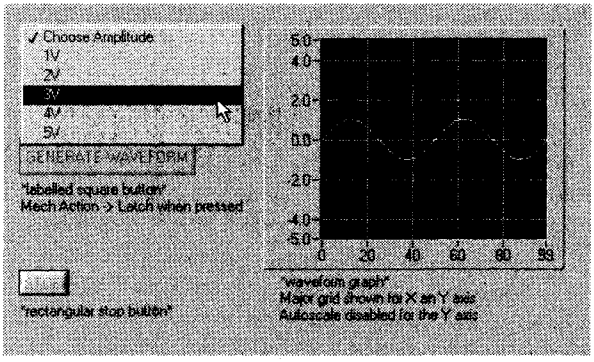


Рис. 12.44

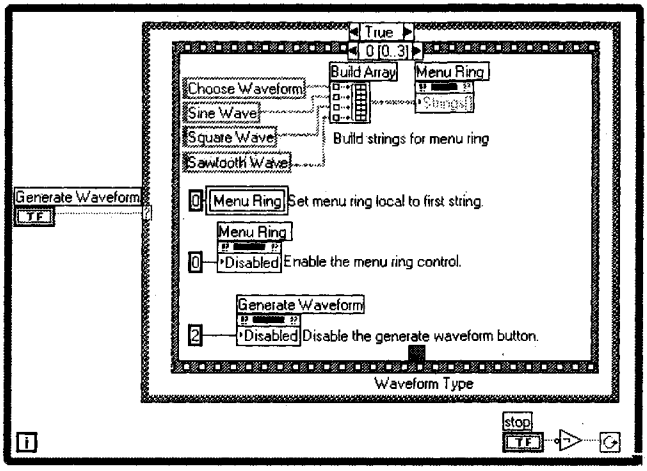


Рис. 12.45

12.3. Другие функции LabVIEW

Категории «прочее» всегда трудно избежать – подобно одноименной, постоянно растущей, категории моего личного бюджета. В палитре **Расширенные** (Advanced) разместилось много разнообразных «прочих» функций. Мы рассмотрим некоторые функции из этой палитры, а также ряд других, которые не совсем точно вписываются в какие-то иные палитры, но тем не менее могут быть весьма полезными.

12.3.1. Диалоги

Способен ли LabVIEW поддерживать с вами диалог? На настоящий момент вы можете вызывать диалоговые окна LabVIEW, которые будут содержать сообщения для пользователя и такие часто используемые кнопки вариантов ответа, как **Да** или **Нет** – мы привыкли видеть их во многих стандартных приложениях. Мы упоминали о диалоговых функциях и ранее, однако полезно сделать их обзор еще раз.

Доступ к функциям диалога осуществляется из палитры **Время и диалог** (Time & Dialog). LabVIEW предлагает два типа всплывающих диалоговых окон: однокнопочный и двухкнопочный. В каждом случае вы можете создать сообщение, которое появится в окне, и установить подписи кнопок. Двухкнопочная диалоговая функция возвращает логическое значение, указывающее на ту из кнопок, которая была нажата. В обоих случаях LabVIEW приостанавливает выполнение виртуального прибора до тех пор, пока пользователь не отреагирует на диалоговое окно. Диалоговое окно является *модальным*. Это означает, что, хотя окна лицевых панелей текущего и других открытых виртуальных приборов продолжают обновляться, пользователь не может ничего сделать до тех пор, пока он не даст ответ активному диалоговому окну.

Предположим, что вы хотите добавить диалоговое окно для подтверждения определенного действия пользователя в ситуации, когда его выбор может оказаться критическим.

На рис. 12.46 и 12.47 показаны пример лицевой панели с диалоговым окном и фрагмент соответствующей блок-диаграммы, когда пользователь нажал кнопку **Самоуничтожение компьютера**.

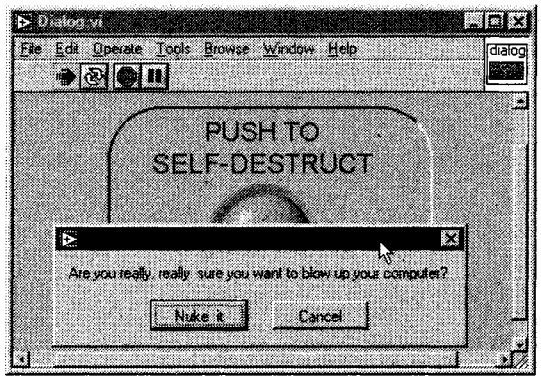


Рис. 12.46

Диалоговые окна являются отличным средством для придания «респектабельности» и дружественности вашему приложению для Windows или Macintosh, но не стоит чрезмерно ими увлекаться. Не забывайте, что приложения LabVIEW уже сами по себе обеспечивают пользовательский графический интерфейс, поэтому изобилие ненужных диалоговых окон может сделать вашу программу запутанной и надоедливой. Диалоги лучше оставить только для наиболее важных сообщений

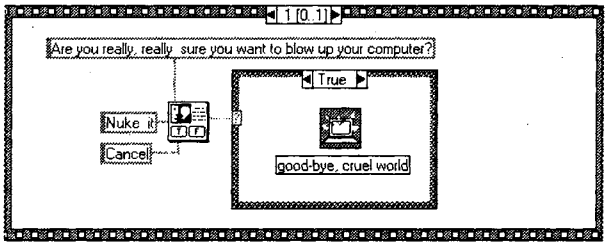


Рис. 12.47

и/или подтверждений действий пользователя. Отметим также, что LabVIEW не позволяет размещать в диалоговом окне более двух кнопок ответа. Для более сложных диалогов рекомендуем создать подпрограмму, где лицевая панель будет содержать диалоговые элементы, которые вам понадобились. Вы можете сделать этот ВПП всплывающим, а также настроить его внешний вид, щелкнув правой кнопкой мыши по его иконке и выбрав опции **Свойства ВП** ⇒ **Появление окна** ⇒ **Диалог** (VI Properties ⇒ Window Appearance ⇒ Dialog).

12.3.2. Говорим жесткое «НЕТ»

LabVIEW предлагает две функции для немедленного программного прерывания выполнения ВП: **Стоп** (Stop) и **Выход из LabVIEW** (Quit LabVIEW). Эти функции находятся в палитре **Управление приложением** (Application Control).

Функция **Стоп** имеет логический ввод. В состоянии ИСТИНА (которое является состоянием по умолчанию, если к входу ничего не подключено) выполнение виртуального прибора и всех его подпрограмм останавливается так же, как если бы вы нажали кнопку **Стоп** на инструментальной панели LabVIEW.

Функция **Выход из LabVIEW** не только вызывает остановку запущенного ВП, но также завершает текущий сеанс работы самой среды LabVIEW, если на вход подано состояние ИСТИНА (которое вновь является состоянием по умолчанию). Будьте осторожны с этой функцией!

Некоторые вспоминают о третьей функции, существовавшей в версии LabVIEW 2.2: **Выключение** (Shutdown). На некоторых моделях компьютеров Macintosh эта функция выключала питание самого компьютера, монитора и ряда периферийных устройств. Мы так и не поняли, зачем эта функция вообще понадобилась, но было забавно оставить запущенным на чьем-нибудь Macintosh красивый цветной виртуальный прибор, снабженный большой кнопкой с надписью «НЕ СМЕЙТЕ НАЖИМАТЬ ЭТУ КНОПКУ!», а затем подождать кого-нибудь достаточно любопытного... Естественно, кнопка была соединена с функцией выключения. Это был хороший способ немножко поразвлечься за счет неопытных пользователей LabVIEW.

Если серьезно, то вы должны пользоваться этими функциями очень внимательно. Считается плохим стилем программирования (на любом алгоритмическом

языке) применять функции «жесткого останова». Всегда старайтесь найти более изящное решение для выхода из программы.

Убедитесь, что ВП завершил все задания (закрыл файлы, завершил выполнение всех ВПП и т.д.), прежде чем вызвать функцию **Стоп**. В противном случае вы можете столкнуться с непредсказуемым поведением файла или ошибкой ввода/вывода и потерять результаты вычислений.

12.3.3. Звук

Очень приятной или, наоборот, весьма надоедливой опцией, которую предлагают многие программы, является звуковое сопровождение – все зависит от того, как именно вы его используете. Звуковое оповещение бывает очень полезным –

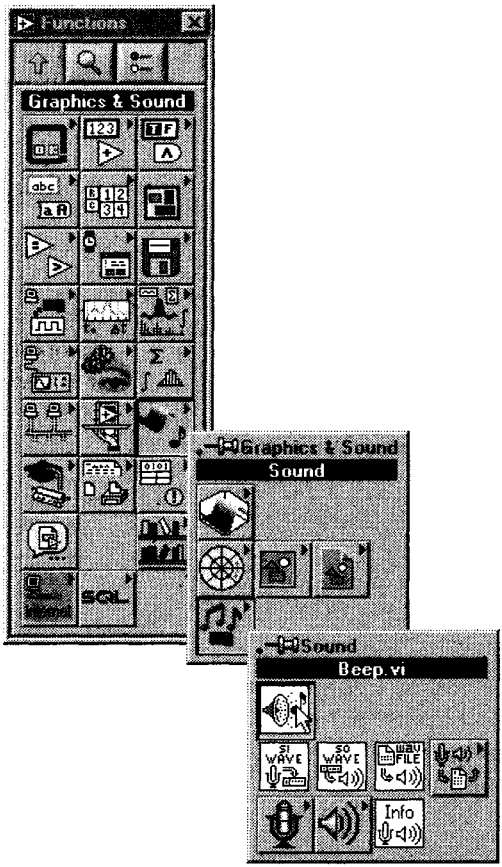


Рис. 12.48

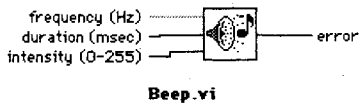


Рис. 12.49

например, когда оператор не может постоянно следить за экраном во время какого-нибудь эксперимента. Однако звуковые сигналы, сопровождающие все уведомления или предупреждения программы, очень быстро станут вас раздражать. Поэтому тщательно продумывайте звуковую схему виртуального прибора.

Звуковые функции доступны из палитры **Графика и звук** ⇒ **Звук** (Graphics & Sound ⇒ Sound).

На очень простом уровне при работе под всеми операционными системами LabVIEW позволяет вызвать системный звуковой сигнал (звук по умолчанию) через функцию **Звонок** (Beep). В операционной системе MacOS вы можете установить частоту, интенсивность и длительность звука. В ОС семейства Windows все эти входные данные игнорируются и вызывается звук по умолчанию, установленный в разделе **Звуки** панели управления Windows.

При работе под ОС Windows и MacOS LabVIEW допускает запись и воспроизведение звуковых файлов (в форматах WAV и AIFF). Вы можете воспользоваться функциями **Звук: Воспроизвести Wave-файл** (Sound Play Wave File), **Звук: Воспроизвести осциллограмму** (Sound Read Waveform) и **Звук: Записать осциллограмму** (Sound Write Waveform) для работы со звуковыми файлами в LabVIEW. Просмотрите примеры работы со звуком в LabVIEW и постарайтесь разобраться в методике их создания и функционирования.

12.4. Вызов кода из других языков программирования

Что делать, если у вас уже есть код, написанный на традиционном языке программирования (C, Pascal, FORTRAN или Basic), который вы хотели бы использовать в виртуальном приборе? Или вы просто соскучились по печатанию любимых текстовых инструкций, заканчивающихся незабвенными точками с запятой, для хорошо знакомого языка программирования? LabVIEW предоставляет средства для взаимодействия с кодом других языков. Если вы планируете написать *весь* код на C или C++, попробуйте LabWindows/CVI (также разработанную и поставляемую National Instruments) – среду программирования, очень схожую с LabVIEW; главным отличием является замена графической блок-диаграммы текстовым кодом на языке C.

Пропустите этот раздел, если вы не знакомы с программированием на традиционных языках вроде C либо не нуждаетесь во взаимодействии LabVIEW с внешним программным кодом этих языков.

Есть три способа вызова внешнего программного кода:

1. Простейшей возможностью является запуск отдельной исполняемой программы, которая делает то, что вам необходимо. В системах Windows, UNIX и MacOS вы можете осуществить это с помощью функции **Системная командная строка** (System Exec) из палитры **Коммуникации** (Communication). В системе MacOS версии 9.x и более ранних задействуйте функции **AppleEvents** (палитра **Коммуникации**) – рис. 12.50.

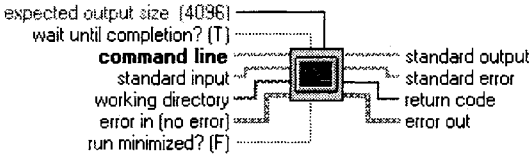


Рис 12.50. Функция ***Системная командная строка***

2. Если у вас имеются уже скомпилированные функции – библиотеки динамических связей (Dynamic Link Libraries – DLL) в системах Windows, разделяемые библиотеки (Shared Libraries) в UNIX и кодовые фрагменты (Code Fragments) в MacOS – используйте ВП **Вызов библиотечной функции** (Call Library Function) из палитры **Расширенные** (Advanced) LabVIEW – рис. 12.51.
3. Наконец, если у вас есть желание написать собственный код на C и внедрить его в LabVIEW, обратитесь к функции **Узел кодового интерфейса** (Code Interface Node – CIN) из палитры **Расширенные** (рис. 12.52).

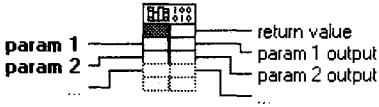


Рис. 12.51. ВП ***Вызов библиотечной функции***

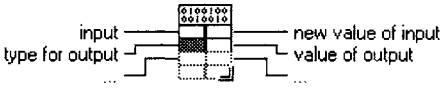


Рис. 12.52. Функция ***Узел кодового интерфейса***

В Windows LabVIEW может экспортировать ВП в исполняемый файл (.exe) или библиотеку динамических связей (.dll), которые, в свою очередь, будут использоваться другими программами. Об этом речь пойдет в главе 13.

12.4.1. Узлы кодового интерфейса

Для использования узла кодового интерфейса (УКИ) вы должны выполнить следующие основные шаги:

1. Вызовите пиктограмму УКИ из палитры **Расширенные** (Advanced) и поместите ее в выбранном месте блок-диаграммы.
2. УКИ имеет терминалы для ввода и вывода данных. По умолчанию у УКИ есть лишь одна пара терминалов. Вы можете переопределить размер УКИ для включения в него заданного числа параметров.
3. По умолчанию каждая пара терминалов представляет собой пару вход/выход: левый терминал является входом, а правый – выходом. Однако если функция возвращает больше выходных значений, чем входных

(или вообще не имеет входных параметров), то вы можете сменить тип терминала на **Только на выход** (Output-Only), вызвав контекстное меню терминала и выбрав эту опцию.

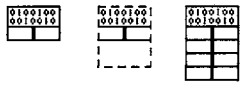


Рис. 12.53

4. Подключите входы и выходы к УКИ. Используйте любой тип данных LabVIEW для соединения с терминалом УКИ (естественно, он *должен* соответствовать типу данных параметра вызываемой функции C). Порядок пар терминалов УКИ соответствует порядку вызова параметров в программном коде. В примере, изображенном на рис. 12.54 и 12.55, мы вызываем УКИ, который проводит фильтрацию входного массива **Исходные данные** и передает эти данные в массив **Отфильтрованные данные**. Отметьте также, что терминалы УКИ при подсоединении к ним проводников приобретают обозначения типов данных, которые через них передаются.

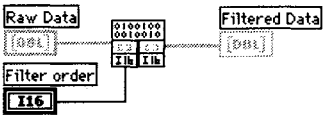


Рис. 12.54

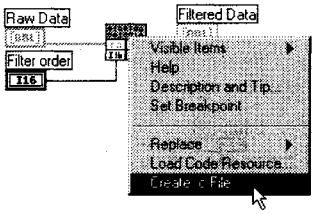


Рис. 12.55

5. Создайте файл `.c`, выбрав эту опцию из контекстного меню терминала УКИ. Файл `.c`, сгенерированный LabVIEW в стиле языка программирования C, является шаблоном, в котором вы будете размещать ваш оригинальный C-код.
6. Скомпилируйте исходный код из УКИ. Этот шаг может оказаться сложным в зависимости от операционной системы, компилятора и утилит. Вы должны в первую очередь скомпилировать код с помощью компилятора, поддерживаемого LabVIEW, а затем, используя встроенную в LabVIEW утилиту, преобразовать полученный объектный код в формат, непосредственно загружаемый в LabVIEW.
7. Загрузите объектный код в память, выбрав опцию **Загрузить кодовый ресурс** (Load Code Resource) из контекстного меню. Укажите файл `.lsb`, который вы создали при компиляции исходного кода.

После успешного выполнения всех описанных операций вызовите код C в УКИ так, как будто это виртуальная подпрограмма, с одним важным исключением: вызовы УКИ *выполняются синхронно*, то есть, в отличие от большинства других задач в LabVIEW, они не разделяют процессорное время с другими задачами LabVIEW. Например, если у вас имеется УКИ и цикл с фиксированным числом итераций (For Loop) на одинаковом уровне в блок-диаграмме, то цикл с фиксированным числом итераций приостанавливает выполнение до тех пор, пока УКИ не

завершится. Это весьма важное обстоятельство, в случае если ваша задача предъявляет повышенные требования к синхронизации процессов.

И наконец, еще один важный факт относительно узлов кодового интерфейса: в отличие от виртуальных приборов они абсолютно *не переносимы* с одной платформы на другую. Если вы скомпилировали код LabVIEW, содержащий УКИ, в PowerMac, а затем пытаетесь запустить его в Windows, у вас ничего не получится. Чтобы обойти эту проблему, перекомпилируйте код УКИ, используя компилятор С для платформы, на которую нужно перенести алгоритм. Компиляторы С, протестированные с LabVIEW, представлены в табл. 12.1.

Таблица 12.1. Компиляторы С

Windows	Microsoft Visual C++, Symantec C
MacOS	Metrowerks CodeWarrior, Apple MPW
Linux, Solaris, HP-UX	GNU C++ (gcc) compiler

Существует много сложных вопросов, связанных с взаимодействием LabVIEW и внешнего программного кода. Поскольку многие из них сильно зависят от процессора, операционной системы и используемого компилятора, мы не будем углубляться в дальнейшее обсуждение УКИ или вызова библиотечной функции.

12.5. «Забивание квадратных шпилек в круглые отверстия»: расширенные преобразования и смена типов данных

Вспомните полиморфизм, о котором мы говорили ранее. Это одно из лучших свойств LabVIEW, которое в большинстве функций позволяет оперировать различными типами данных, даже не задумываясь об этом (любой компилятор с традиционного языка программирования «обругает» вас, если вы попытаетесь, например, сложить константу с массивом, – но только не LabVIEW). LabVIEW обычно осуществляет все преобразования типов автоматически, когда получает на вход некоей функции различные, но совместимые типы данных.

Если вы собираетесь разрабатывать программы, которые включают в себя управление приборами, связь между приложениями или сетевые компоненты, то обязательно будете пользоваться строковыми данными. Вам часто придется преобразовывать числовые данные, такие как массивы чисел с плавающей точкой, в текстовые строки. Мы уже говорили об этом в главе 9. Очень важно теперь подчеркнуть различие между двумя типами строковых данных: *ASCII-последовательностями* (ASCII strings) и *двоичными (бинарными) строками* (binary strings).

Строки ASCII используют отдельный символ для представления каждой цифры в числе. Таким образом, число 145, преобразованное в ASCII-код, будет состоять из символов 1, 4 и 5[1]. Для представления совокупностей чисел – например,

массивов – используются разделители, в частности символ пробела. Такой тип символьного представления чисел является весьма распространенным – например, для обмена данными с внешними приборами по интерфейсу канала общего пользования:

```
"1" "4" "5"
00000001 00000100 00000101
3 байта
```

Двоичные строки являются чуть более тонким инструментом – во-первых, при считывании их в виде ASCII-символов вы не сможете сказать, какой тип данных записан в строке. Для представления числа используется определенный шаблон двоичных разрядов (bit pattern) или *двоичное представление* (bit representation). Таким образом, в двоичной строке число 145 в представлении I8 можно записать одним байтом (который соответствует в ASCII-таблице моего компьютера символу «ё»). Двоичные строки весьма распространены в программах, от которых требуется максимальная производительность, так как преобразование данных в двоичные строки осуществляется быстрее и, кроме того, они занимают меньше места в памяти.

```
"ё"
10010001
1 байт
```

В конечном счете, все данные в компьютерах хранятся в виде двоичных чисел. Как же LabVIEW определяет, является двоичная последовательность строковыми данными, логическими данными, числами с плавающей точкой или массивом целых чисел? Все данные в LabVIEW имеют два компонента: *сами данные* и *дескриптор типа* данных. Дескриптор типа данных представляет собой массив целых (I16) чисел, которые образуют код, идентифицирующий представление данных (целое число, строка, число двойной точности, логическое значение и т.д.). Этот дескриптор содержит информацию о длине (в байтах) данных, а также дополнительную информацию об их типе и структуре. Подробный список кодов типов данных приведен в статье *Application Note 154, LabVIEW Data Types*, которая содержится в руководствах пользователя формата PDF (LabVIEW PDF manuals).

При выполнении типичной функции преобразования, например числа в десятичную строку, изменяется дескриптор типа данных и сами данные преобразуются по определенному алгоритму. На рис. 12.56 приведен пример часто используемой функции преобразования числа в ASCII-строку.

Рис. 12.56

[1] Стандарт Unicode, популярная альтернатива ASCII, использует 4 байта (32 бита) для представления символа, что позволяет кодировать многообразные символы, включая национальные кодировки. На момент написания этой книги поддержка Unicode в LabVIEW отсутствовала

Темы, рассматриваемые ниже, начинающим пользователям могут показаться крайне трудными. Если вам не нужно пользоваться преобразованиями двоичных строк в приложении, можете смело пропустить остаток этого раздела.

В некоторых случаях требуется преобразовать данные в двоичную строку. Двоичные строки занимают в памяти мало места, операции над ними часто выполняются быстрее и, наконец, для некоторых задач данные принципиально требуется представлять в виде двоичных строк (например, при работе с сетевым протоколом TCP/IP или при обмене данными с внешними устройствами). В LabVIEW встроено средство преобразования любых данных в двоичные строки – функция **Перевести в строку** (Flatten To String), расположенная в подпалитре **Манипуляция данными** (Data Manipulation) палитры **Расширенные** (Advanced) – рис. 12.57.

Входными данными для функции **Перевести в строку** могут быть любые типы данных LabVIEW, включая сложные типы, такие как кластеры. Функция возвращает два выхода: **двоичную строку**, представляющую входные данные (data string), и **строку типа данных** (type string), которая, строго говоря, является не строкой, а целочисленным массивом I16. **Строка типа данных** передает дескриптор типа данных.

Преобразованная таким образом двоичная строка содержит не только данные в компактной форме, но и *заголовок* информации. Заголовок, находящийся в начале двоичной строки, содержит сведения о длине, типе, структуре и т.д. данных, записанных в виде двоичной строки. Например, на диаграмме, изображенной на рис. 12.58, показан массив чисел с плавающей точкой двойной точности (DBL), преобразованный в строку. **Строка типа данных**, которая всегда является массивом I16, также содержит информацию о заголовке.

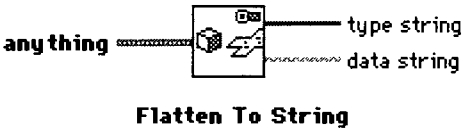


*Рис. 12.57. Функция **Перевести в строку***

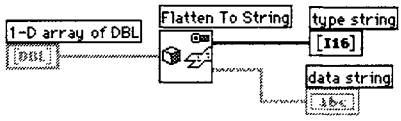


Рис. 12.58

Ключевым моментом в использовании непонятных преобразований в двоичные строки является то, что в большинстве случаев вам не нужно будет оперировать этими строками или наблюдать их непосредственно: вы лишь передаете строки в файл, в сеть, на внешний прибор и т.д. – для повышения производительности приложения. Для чтения подобного рода данных понадобится совершить обратное преобразование, воспользовавшись обратной функцией, называемой **Восстановить из строки** (Unflatten From String) – рис. 12.59. На вход **Тип данных** (type) следует подать шаблон типа данных[1] LabVIEW для данных, зашифрованных во

[1] Например, это может быть константа соответствующего типа данных, в нашем случае она играет роль маски, поэтому ее величина значения не имеет. – *Прим. перев.*

входной **двоичной строке** (binary string). Выходное значение **величина** (value) содержит данные, преобразованные из двоичной строки. В случае если преобразование было неудачным (несовпадение типов), логический терминал **Ошибка типа** (type.err) примет состояние ИСТИНА.

На рис. 12.60 показано, как преобразовать двоичную строку в массив чисел DBL. Подключите на вход двоичную строку и шаблон типа данных – пустой одномерный массив чисел с плавающей запятой двойной точности (DBL), и вы получите исходный 1D-массив DBL-чисел. Обратите внимание, что мы переводим, а затем восстанавливаем данные без непосредственного «просмотра» или «оперирования» двоичными строками.

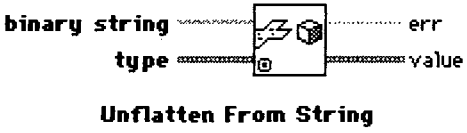


Рис. 12.59. Функция ***Восстановить из строки***

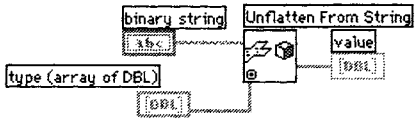


Рис. 12.60

В завершение кратко рассмотрим действие другой, очень мощной функции LabVIEW для осуществления быстрых и эффективных преобразований: **Приведение типа** (Type Cast) – рис. 12.61.

Эта функция дает возможность изменять дескриптор типа данных без модификации самих данных. Данные никоим образом не преобразуются, изменяется лишь их дескриптор. Вы можете взять любой тип данных (строковые, логические, числовые, кластеры и массивы) и обозначить его как-нибудь иначе. Одним из преимуществ этой функции является экономия памяти, поскольку, в отличие от других функций преобразования, она не создает в памяти копии преобразуемых данных.

Обычное использование этой функции показано на рис. 12.62. Здесь некий прибор выдает последовательность отсчетов данных в виде двоичной строки, которую необходимо превратить в массив чисел для последующей обработки. Предположим, что в двоичной строке содержится массив целых чисел I16.

Приведение типа данных от скалярных величин (или массивов скалярных величин) к строке осуществляется так же, как и при помощи функции **Перевести**

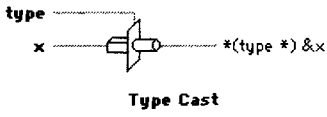


Рис. 12.61. Функция ***Приведение типа***

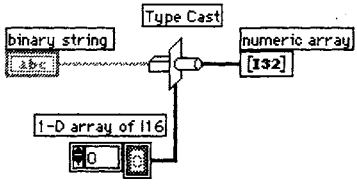


Рис. 12.62

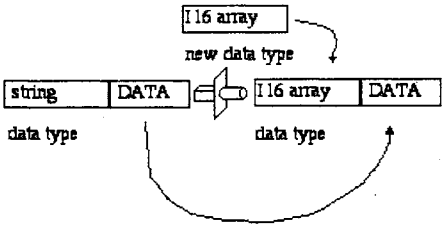


Рис. 12.63

в строку, однако есть отличительная особенность – приведение типов не добавляет и не убирает никакой информации в *заголовке* данных в отличие от перевода в строку и восстановления из строки. Вход **тип данных** в функции **Приведение типа** является исключительно «шаблонным», служащим только для определения типа – любые реальные данные на этом входе игнорируются.

Необходимо быть очень внимательными при использовании этой функции: вы должны понимать, как представляются данные, чтобы приведение типа имело осмысленный результат. Как упоминалось ранее двоичные строки часто содержат информацию заголовка. Функция **Приведение типа** никак не воздействует на эту информацию. Если вы сами не удалите заголовки, они будут восприняты как данные, и в результате получатся ошибочные значения («информационный мусор»).

Удостоверьтесь, что вы понимаете, как представлены данные, подлежащие приведению типов. Приведение типов не включает ни преобразования собственно данных, ни проверки данных на ошибку. Более того, даже порядок следования байтов в числе различен на разных платформах, поэтому необходимо знать и учитывать порядок следования байтов в ваших данных.

Можете поэкспериментировать с ВП **Приведение типов** (Type cast), наблюдая, какие получаются результаты для различных типов данных. Чтобы показать, в какое нелепое положение вы можете попасть, используя приведение типов данных, мы создали кластер, содержащий числовые элементы управления типа I16 и привели эти данные к строке, числовому массиву и логическому состоянию (рис. 12.64).

Лицевая панель (рис. 12.65) показывает результаты, полученные для двух чисел на входе.

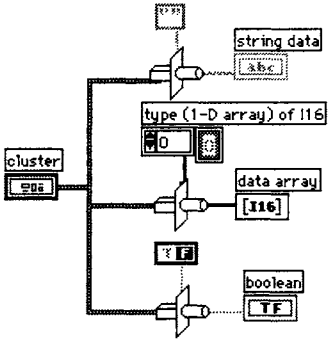


Рис. 12.64

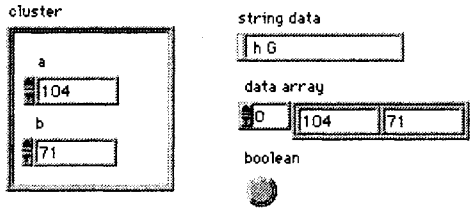


Рис 12.65

Строка вернула два символа, соответствующих ASCII-кодам чисел 104 и 71. Массив просто реорганизовал два числа из элементов кластера в элементы массива. А что же логическая величина? Как двоичная величина восприняла данные кластера? Поскольку ответа на этот вопрос из общих соображений мы дать не смогли, пришлось попробовать... Оказалось, что логическое состояние является ИСТИНОЙ, если числовое значение отрицательное, и наоборот. В случае кластера логическая величина представляет собой результат операции логическое ИЛИ над каждым из приведенных элементов кластера. Таким образом, если любое из чисел в исходном кластере отрицательное, то логическая функция преобразует результат в ИСТИНУ. Довольно странно, не так ли?

12.6. Итоги

В данной главе мы рассмотрели несколько особенностей LabVIEW: локальные и глобальные переменные и узлы свойств. Мы также описали несколько расширенных функций: диалоги, звуковые сигналы, вызов внешнего кода, преобразование двоичных строк и приведение типов данных. Эффективность и гибкость этих структур и функций дают возможность перейти на более глубокий уровень разработки приложений в LabVIEW.

Локальные переменные позволяют создавать копию объекта лицевой панели в блок-диаграмме, которая может быть считана или записана в различных местах. Глобальные переменные подобны локальным, но хранят свои данные независимо от какого-либо конкретного виртуального прибора, что позволяет распределять данные между различными ВП без соединения их проводниками. Локальные переменные очень удобны для управления параллельными циклами и обновления показаний объекта лицевой панели с нескольких точек блок-диаграммы. Глобальные переменные также являются эффективной структурой, но работать с ними следует осторожно, поскольку они способны вызывать проблемы.

Узлы свойств позволяют задать внешний вид и поведение элементов управления и индикации, причем эти характеристики можно изменять программно. Каждый элемент управления и индикации имеет набор базовых свойств (таких, как цвет, видимость и т.д.). У многих объектов – например, графиков – есть множество свойств, которые разрешается устанавливать или считывать соответственно.

LabVIEW позволяет использовать внешние процедуры, написанные на языке C, с помощью узла кодового интерфейса (УКИ). УКИ дает возможность вызывать фрагменты C-программ, обработанные внешними компиляторами. Кроме того, допустимо напрямую вызывать функции из библиотек динамических связей (DLL) в Windows.

Функции **Перевести в строку** (Flatten To String), **Восстановить из строки** (Unflatten From String) и **Приведение типа** (Type Cast) являются эффективными средствами преобразования при работе с различными типами данных. Эти функции могут преобразовывать различные типы данных LabVIEW в двоичные строки и наоборот. Двоичные строки часто используются в различных практических приложениях.

Обзор

Оптимизируя многочисленные настройки, вы можете подогнать внешний вид и функциональность среды LabVIEW для достижения наибольшей производительности работы. Вы также научитесь настраивать свойства виртуального прибора, например устанавливать параметры элементов лицевой панели для управления ими с клавиатуры. Мы рассмотрим очень мощное средство: сервер виртуальных приборов (VI Server) – и покажем, как динамически управлять ВП с его помощью. LabVIEW позволяет представлять числа не только в привычной десятичной, но и в двоичной, восьмеричной, шестнадцатеричной системах счисления. Вы узнаете о наличии в LabVIEW встроенных единиц размерности численных данных, которые позволяют вам проводить некоторые операции преобразования данных автоматически. Вы будете приятно удивлены, узнав, как просто можно превратить фрагмент диаграммы ВП в виртуальный подприбор (ВПП). И наконец, вы познакомитесь с некоторыми полезными инструментами среды разработки LabVIEW из меню ***Инструменты*** *(Tools), такими как функция* ***Найти*** *(Find) – средство быстрого поиска объектов, подпрограмм или текста.*

ЗАДАЧИ

- Использовать **Опции** (Options) для настройки среды LabVIEW
- Познакомиться со свойствами ВП (VI Properties) для выбора особых вариантов оформления и выполнения ВП
- Получить представление о сервере виртуальных приборов (VI Server) и о том, что можно сделать с его помощью
- Научиться устанавливать доступ к элементам управления с клавиатуры
- Научиться представлять числа в различных системах счисления и присваивать им единицы размерности
- Создать виртуальный подприбор из выделенной области блок-диаграммы
- Получить представление об утилитах меню **Инструменты**

ОСНОВНЫЕ ТЕРМИНЫ

- Опции настройки (Options)
- Свойства ВП (VI Properties)
- Сервер виртуальных приборов (VI Server)
- Класс приложения (Application Class)
- Класс ВП (VI Class)
- Методы и свойства (Methods and Properties)
- Приоритет (Priority)
- Выполнение с повторным вхождением (Reentrant Execution)
- Объект фокусировки (Key Focus)
- Система счисления (Radix)
- Единица размерности (Unit)
- Найти (Find)
- Окно оптимизации (Profile Window)
- Предыстория (History)
- Иерархия ВП (VI Hierarchy)

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ LabVIEW

13

13.1. Опции, опции...

Возможно, вы уже пользовались подменю **Опции** (Options) из меню **Инструменты** (Tools). В любом случае будет неплохо бегло просмотреть все имеющиеся опции, поскольку именно здесь вы, например, можете найти решение проблемы с LabVIEW, которая беспокоит вас уже месяц. Рассмотрим такие настройки, как маршруты поиска, цвета и шрифты лицевой панели, форматы представления времени и даты и т.д.

Как показано на рис. 13.1, настройки разбиты на категории. При выборе каждой из категорий содержимое окна опций изменяется, отображая варианты настроек. Детальное описание всех опций настройки было бы довольно долгим, поэтому мы разберем лишь некоторые из них – наиболее часто используемые и полезные:

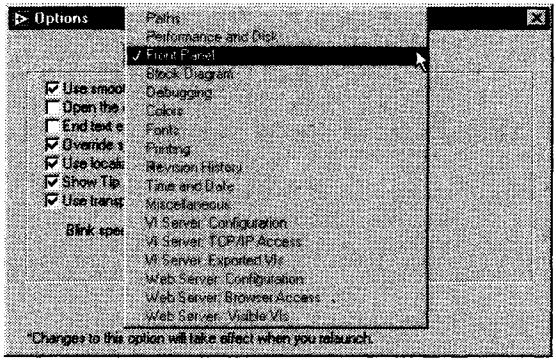


Рис. 13.1

- в опциях **Пути** (Paths) вы можете установить по умолчанию каталоги размещения среды LabVIEW, пути поиска библиотек, пользовательских ВП и каталог для размещения временных файлов. Если вы храните проекты виртуальных приборов в своем пользовательском каталоге[1], то можете изменить маршрут по умолчанию так, чтобы при старте LabVIEW быстро открыть ВП;
- категория **Лицевая панель** (Front Panel) содержит ряд важных настроек. Например, опция **Использовать сглаживание при отрисовке** (Use smooth updates during drawing) устраняет раздражающее мелькание, которое вы видите на графике или диаграмме, если они часто обновляются. Опция **Использовать прозрачные имена ярлыков** (Use transparent name labels) также популярна, поскольку многим не нравятся объемные окна ярлыков элементов управления и индикации. Разрешается установить здесь частоту мигания (blink speed) – в случае, если у объекта задан атрибут **мигание**. Многие думают, что эта опция устанавливается программно – а на самом деле именно здесь, в разделе **Опции**. Наконец, российским пользователям будет нелишне знать, что LabVIEW предоставляет возможность установить опцию локализации разделителя целой и дробной части числа: **Использовать локализованную разделительную точку** (Use localized decimal point). Локализованные операционные системы представляют этот разделитель в виде запятой (а не точки), что может привести к ошибкам в обработке локализованных данных с помощью LabVIEW, если эта опция не установлена;

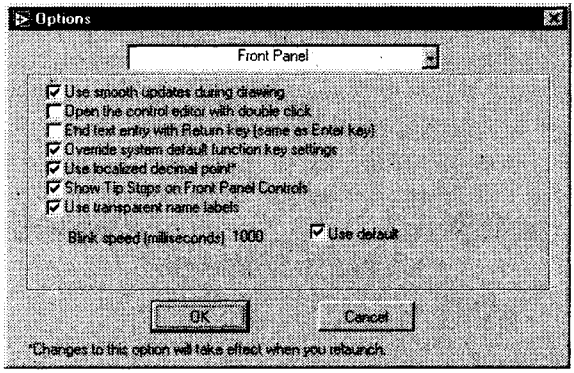


Рис. 13.2

- опция **Цвета** (Colors) дает возможность установить цвета по умолчанию для передней панели, блок-диаграммы и т.д. Если вы мечтаете раскрасить фон панелей LabVIEW в красный цвет, то вам сюда...

[1] В особенности на удаленном сетевом ресурсе. – *Прим. перев.*

- опция **Печать** (Printing) предоставляет выбор между стандартным вариантом печати, растровой картинкой или выводом в формате PostScript. В зависимости от имеющегося в наличии принтера вы можете выбрать вариант, обеспечивающий наилучшие результаты.

Просмотрите оставшиеся категории. Допустимо установить стили шрифтов[1], времени и даты, распределение ресурсов между LabVIEW и другими приложениями и т.д. Вы увидите ряд опций для настройки сервера ВП (VI Server), о котором мы поговорим несколько позже, и для встроенного в LabVIEW Интернет-сервера (Web Server), описание которого приводится в главе 14.

13.2. Конфигурирование виртуального прибора

Часто требуется, чтобы программа выводила новое окно при нажатии определенной кнопки или при наступлении определенного события. Например, вы хотите создать главную лицевую панель с несколькими кнопками, которые задают варианты работы. Нажатие каждой из этих кнопок приводит к появлению нового окна, содержащего, в свою очередь, кнопку завершения, которая закрывает это окно и возвращает пользователя к главной лицевой панели. Поскольку всплывающее окно наиболее вероятно будет лицевой панелью виртуального подприбора, вы также можете настроить ряд его свойств, такие как появление **Инструментальной панели** (Toolbar), или кнопок управления окном (свернуть, изменить размер, закрыть). Допустимо установить опции, которые управляют внешним видом окна и выполнением ВП, из двух различных мест: **Свойства виртуального прибора** (VI Properties) – из контекстного меню иконки ВП, расположенной в верхнем правом углу окон панелей, или из меню **Файл**; и **Установка узла ВПП** (SubVI Node Setup) – из контекстного меню иконки ВПП на блок-диаграмме. Здесь следует подчеркнуть очень важное отличие: настройка опций в **Установке узла ВПП** *действует только на это отдельное вхождение подприбора*, тогда как опции, установленные в **Свойствах виртуального прибора** *действительны всегда*, независимо от того, запускается ВП как самостоятельное приложение или вызывается как ВПП. Начнем рассмотрение с немногочисленных и простых опций окна **Установка узла ВПП**.

13.2.1. Настройки окна Установка узла ВПП

Выбрав опцию настройки свойств из контекстного меню ВПП, вы откроете диалоговое окно, изображенное на рис. 13.3, где можете выбрать любую из следующих опций:

- **Открыть лицевую панель после загрузки** (Open Front Panel when loaded). Когда виртуальный прибор загрузится в память, его лицевая

[1] Очень полезно сразу установить по умолчанию кириллические шрифты. Это сэкономит время при внесении меток на русском языке, а также повысит межплатформенную переносимость ваших ВП. – *Прим. перев.*

панель появится в новом окне (например, если вы открываете виртуальный прибор, в котором данный ВП содержится в виде подприбора);

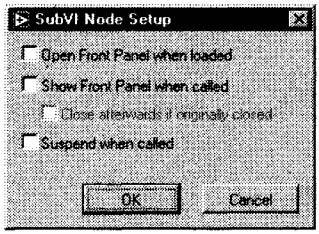


Рис. 13.3

- **Показать лицевую панель после вызова** (Show Front Panel when called). Лицевая панель ВП открывается, когда происходит его выполнение как ВПП. Эта и следующая опции очень полезны для создания интерактивных виртуальных приборов;
- **Закрыть по завершении выполнения, если изначально закрыт** (Close afterwards if originally closed). Опция закрывает лицевую панель ВПП после завершения ее выполнения, создавая эффект выпадающего окна. Эта установка работает только совместно с предыдущей;
- **Приостановить после вызова** (Suspend when called). Установка действует подобно отладочной точке прерывания, заставляя ВПП приостановиться сразу же при вызове. Действительно, можно использовать эту опцию в качестве средства отладки для проверки входных данных ВПП перед тем, как он начнет выполняться.

Помните, что все установки узла виртуального подприбора действуют только на данное вхождение ВПП. Они не оказывают никакого влияния на копии того же виртуального прибора в других точках блок-диаграммы вызывающего ВП.

13.2.2. Упражнение 13.1: использование виртуальных подприборов

В этом упражнении вы воспользуетесь настройками **Установки узла ВПП** для создания программной оболочки процедуры авторизации, которая может быть использована в любых других приложениях.

1. Создайте простую лицевую панель, подобную приведенной на рис. 13.4, которая вызывает всплывающее окно ВПП при нажатии кнопки **Сменить пользователя**. Назовите этот высокоуровневый виртуальный прибор **Shell.vi**.
2. Используйте подпрограмму **Login.vi** (которую вы найдете на приложенном компакт-диске) для построения блок-диаграммы, как показано на рис. 13.5 и 13.6.
3. Щелкните правой кнопкой мыши по подприбору Login и проведите установку узла ВПП, как показано на рис. 13.7. Когда этот виртуальный прибор выполняется, он отображает последнее значение **Текущий пользователь** (из неинициализированного сдвигового регистра) до тех пор, пока не будет вызван ВПП **Login**, который изменит значение.

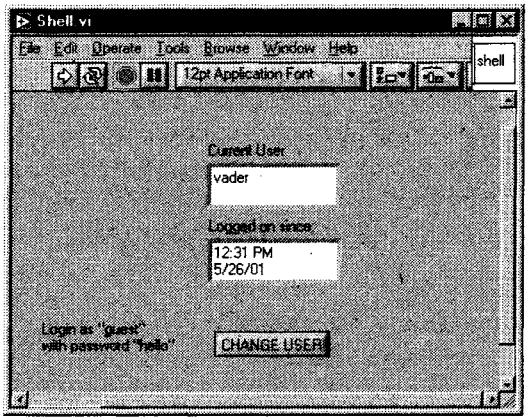


Рис. 13.4

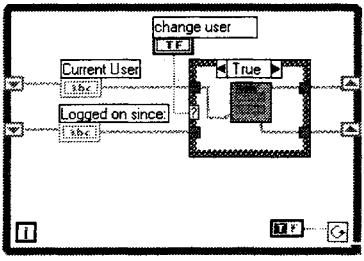


Рис. 13.5

Рис. 13.6

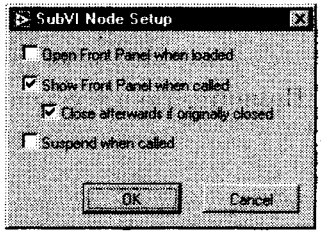


Рис. 13.7

ВПП **Login** должен быть закрыт до запуска этого примера. В противном случае его окно по завершении выполнения закрыто не будет, как следует даже из самого наименования опции **Закрыть по завершении выполнения, если изначально закрыт**.

13.2.3. Опции свойств виртуальных приборов

Группа **Свойства виртуального прибора** более многочисленная. Диалоговое окно, которое появляется при выборе опции **Свойства ВП** из контекстного меню его иконки, предлагает набор категорий свойств: **Общие**, **Использование памяти**, **Документирование**, **История изменений**, **Безопасность**, **Размер окна**, **Внешний вид окна**, **Выполнение** и **Печать**. Некоторые из них предназначены только для чтения (такие как **Использование памяти**); другие могут быть изменены. Ниже мы кратко опишем некоторые из этих настроек.

Общие

Свойство **Общие** (General) виртуального прибора показывает путь к файлу ВП, позволяет отредактировать его иконку и проверить историю изменений ВП.

Использование памяти

Раздел **Использование памяти** (Memory Usage) показывает, какой объем памяти занят вашим ВП.

Документирование

В разделе **Документирование** (Documentation) вы можете документировать виртуальный прибор. В поле **Описание ВП** (VI Description) – рис. 13.8 – введите описание вашего ВП – считается хорошим стилем не пренебрегать этой несложной, но полезной работой. Такой текст появится в окне контекстной справки LabVIEW, если навести указатель мыши на иконку ВП. Дополнительно можно указать **Ссылку на справку** (Help Tag) и **Путь справки** (Help Path) для внешнего файла справки (например, ссылку на Internet-страницу).

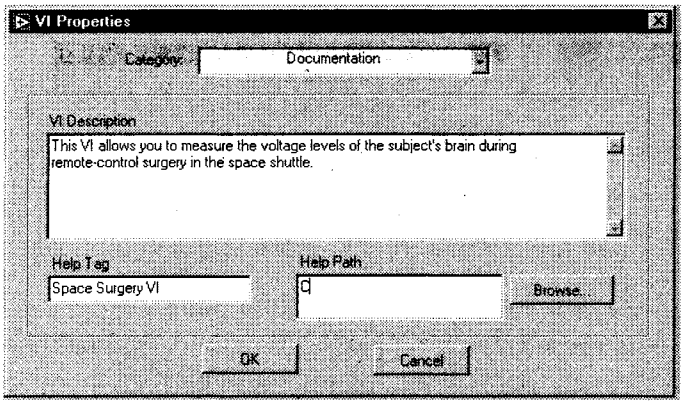


Рис. 13.8

История изменений

История изменений (Revision History) позволяет отслеживать историю изменений кода виртуального прибора. Она информирует, какая версия ВП является текущей и какие изменения внесены туда по сравнению с предыдущими версиями. Вообще эта опция полезна только в случае, если вы регулярно вносите записи (каждый раз при сохранении ВП).

Безопасность (Security)

Виртуальный прибор LabVIEW – открытая система: по умолчанию любой пользователь, в чьем распоряжении находится ваш ВП, может просматривать и изменять

код (блок-диаграмму). Но иногда вы не хотите, чтобы кто-либо вносил изменения или даже просто увидел блок-диаграмму. Настоящая опция дает возможность установить пароль, без знания которого никто не сможет просмотреть или изменить блок-диаграмму.

Если вы решили защитить ваш ВП паролем, постарайтесь не забыть его или хотя бы запишите. LabVIEW не предоставляет никакой возможности открыть диаграмму, если пароль будет утерян...

13.2.4. Внешний вид окна

Настройки раздела **Внешний вид окна** (Window Appearance), приведенные на рис. 13.9, позволяют управлять многими деталями внешнего вида окна виртуального прибора. Вы можете выбрать из предварительно определенных заготовок **Приложение верхнего уровня** (Top-level application), **Диалог** (Dialog) или **По умолчанию** (Default) либо настроить внешний вид окна по своему желанию (Custom).

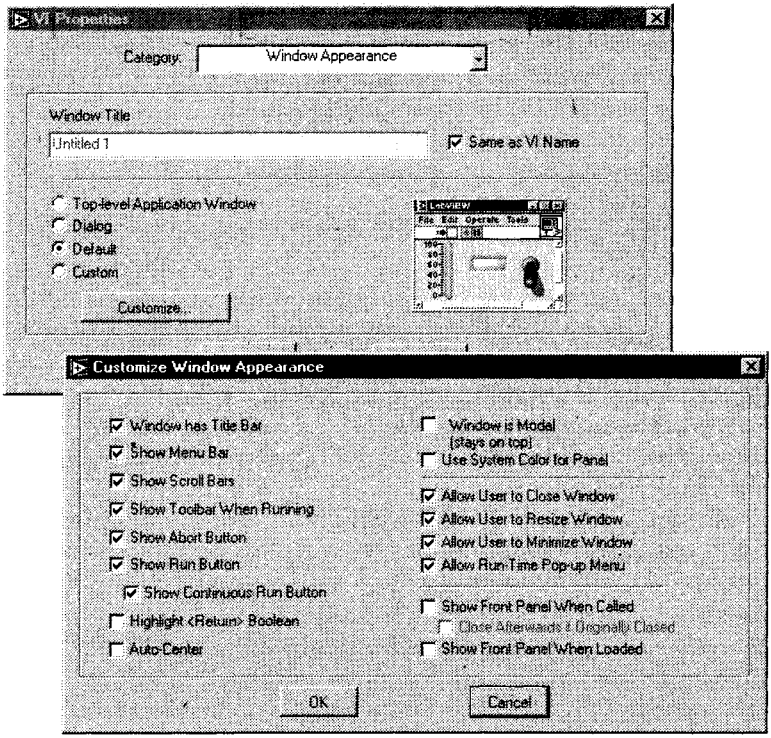


Рис. 13.9

Большая часть настроек понятна сама по себе, но все-таки приведем некоторые комментарии:

- будьте осторожны с такими вариантами, как запрещение кнопки останова: **Показать кнопку останова** (Show Abort Button). Виртуальный прибор, у которого эта опция снята, нельзя принудительно остановить во время выполнения, даже если кнопка дублирована назначенным сочетанием клавиш на клавиатуре! Используйте эту установку только для полностью отлаженных и протестированных ВП, которые, по вашему мнению, гарантированно завершат выполнение программным способом;
- опция **Диалоговое окно** (Dialog Box) придает виртуальному прибору статус системного диалогового окна и препятствует доступу к другим окнам LabVIEW;
- действие опции **Выделить логический <ввод>** (Highlight <Return> Boolean) не столь очевидно. Эта опция создает выделение (черную окантовку) логического элемента управления, которому была назначена клавиша <ввод> (<return> или <enter>). Подробнее назначение клавиш обсуждается в следующем разделе.

Размер окна

Настройка **Размер окна** (Window Size) дает возможность установить определенные размеры окна ВП на экране, а также автоматически масштабировать объекты лицевой панели при изменении размеров окна.

13.2.5. Выполнение

Опция **Выполнение** (Execution) позволяет установить некоторые очень тонкие настройки (рис. 13.10):

- **Приоритет** (Priority): не заботьтесь особо об этой функции, реально она вам не понадобится. Но если вы хотите знать, что она устанавливает, почитайте Руководство пользователя LabVIEW;
- **Требуемая модель выполнения** (Preferred Execution System): это средство управления многопоточностью в LabVIEW. Опция позволяет выбрать предпочтительный поток для выполнения ВП. Вероятно, устанавливать ее также не понадобится;
- **Разрешить отладку** (Allow Debugging): эта опция, установленная по умолчанию, дает вам возможность использовать встроенные средства отладки LabVIEW во время выполнения виртуального прибора. Допустимо запретить их, увеличив тем самым скорость выполнения на 1–2% и снизив загрузку памяти виртуальным прибором;
- **Запустить при открытии** (Run When Opened): опция запускает прибор при открытии;
- **Приостановить после вызова** (Suspend When Called): редко используемая опция отладки;

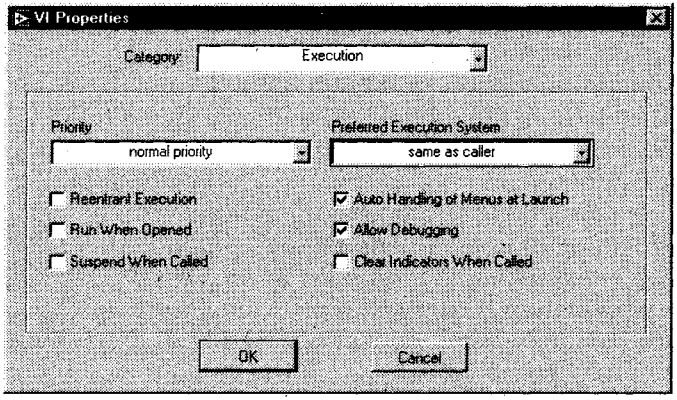


Рис. 13.10

- **Выполнение с повторным вхождением** (Reentrant Execution): эта опция заслуживает отдельного объяснения.

Выполнение с повторным вхождением

Выполнение с повторным вхождением является концептуальным понятием. Обычно, если есть две или более точки вызова одного и того же виртуального подприбора, которые запускаются параллельно, LabVIEW выделяет обеим точкам вызова ВПП общее идентичное пространство памяти для хранения данных, которое они делят между собой. Однако в ряде случаев вам может понадобиться выделение различных участков памяти для каждого из узлов ВПП. На блок-диаграмме, изображенной на рис. 13.11, мы используем ВПП **Бегущее усреднение** (Running Average) в каждом канале данных.

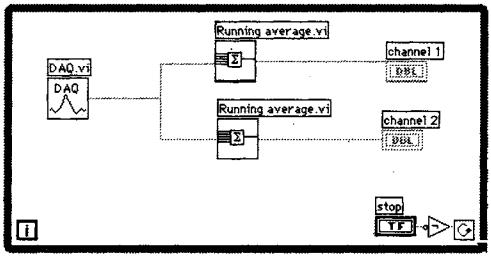


Рис. 13.11

ВПП **Бегущее усреднение** использует *неинициализированные* терминалы сдвиговых регистров в качестве ячеек памяти. Вспомните, что неинициализированные терминалы сдвиговых регистров хранят свое последнее значение даже после остановки и повторного запуска виртуального прибора (в главе 6 мы обсуждали поведение

сдвиговых регистров). Если оставить этот ВПП сконфигурированным по умолчанию, то результаты его выполнения окажутся непредсказуемыми, поскольку при вызове каждого узла ВПП в его сдвиговом регистре окажутся данные, оставшиеся от другого узла (так как ВПП выполняются поочередно). При выборе опции **Выполнение с повторным вхождением** каждому из узлов виртуального подприбора выделяется независимая область памяти для хранения данных, как если бы они являлись двумя абсолютно разными ВПП.

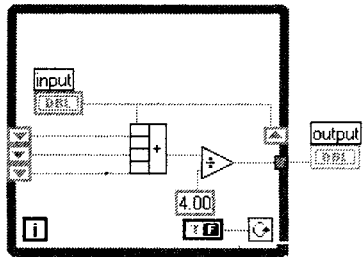


Рис. 13.12. Блок-диаграмма ВПП ***Бегущее усреднение***

Упражнение 13.2: обзор повторного выполнения программы

В качестве упражнения для иллюстрации вышесказанного запустите виртуальный прибор **Running Average.vi**, который находится на компакт-диске, и посмотрите, как он работает. Затем запустите ВП **Reentrant.vi** (с компакт-диска) вначале так, как он есть, затем установите **Выполнение с повторным вхождением** для ВП **Running Average.vi**, произведите повторный запуск и сравните полученные результаты.

Печать

Настройки раздела **Печать** (Print Options) устанавливают режимы распечатки и границы поля печати.

13.3. Сервер виртуальных приборов

Сервер ВП является очень эффективным средством LabVIEW, которое предоставляет программный доступ к таким функциям LabVIEW, как открытие и запуск виртуальных приборов, изменение цвета или данных объекта лицевой панели, управление печатью и т.д.

Эта тема действительно очень специальная, поэтому не смущайтесь, если большая часть введения в механизм сервера ВП покажется непонятной. Практически никто из тех, кто сталкивается с сервером ВП впервые, не в состоянии сразу освоиться с его функциями. Однако хорошая новость: при написании простых приложений сервер ВП вряд ли будет необходим. И только когда вы наберетесь опыта и начнете чувствовать себя в LabVIEW комфортно, вы неизбежно захотите поэкспериментировать с этим эффективным средством разработки – и довольно скоро вы не будете представлять себе, как можно не использовать сервер ВП! В данной книге нигде не требуется ни понимать, ни использовать сервер ВП, поэтому можете безбоязненно пропустить раздел, особенно если вам уже не терпится дочитать до конца.

Пусть название вас не смущает: «сервер виртуальных приборов» намного больше, чем какой-либо тип сетевого сервера, встроенного в LabVIEW (хотя и это тоже...). Функционально сервер ВП является одним из воплощений *объектно-ориентированного программирования* (ООП) в LabVIEW.

С помощью сервера ВП вы можете, например, программно выполнить следующие операции:

- загрузить ВП в память, запустить его, а затем закрыть без необходимости его постоянного подсоединения как подприбора на блок-диаграмме;
- динамически вызвать и запустить ВПП во время работы приложения, зная лишь имя ВПП и структуру соединительной панели;
- изменить свойства определенного виртуального прибора, такие как размер и местоположение окна лицевой панели, возможность его редактирования и т.д.;
- переместить окно LabVIEW на передний план экрана;
- на блок-диаграмме вызвать ВПП, но не ожидать нормального завершения его выполнения (один из немногих случаев, когда вы можете легко прервать выполнение, не подчиняясь парадигме обработки потока данных);
- динамически изменить свойства (атрибуты) объекта лицевой панели, например цвет и размер.

В дополнение ко всем этим приятным вещицам, еще один, не менее приятный сюрприз: все, что мы только что перечислили, *прозрачно работает в сети*. Прозрачность сети означает, что вы можете осуществить указанные действия с виртуальным прибором или со средой LabVIEW *на удаленном компьютере* через сеть (включая Internet) точно так же, как на своей локальной машине. Таким образом, допустимо организовать ввод/вывод сигналов на удаленном компьютере, тогда как компьютер на вашем рабочем месте будет заниматься анализом собранной информации без необходимости создания каких-либо специальных сетевых программ и программирования сложных функций протокола TCP/IP.

Пример, изображенный на рис. 13.13 и 13.14, показывает, насколько просто запустить виртуальный прибор дистанционно. (Не пугайтесь, если вы не видели этих функций ранее – все они из палитры **Управление приложением** (Application Control), мы скоро будем их рассматривать более детально.)

Возможности сервера ВП реализуются в LabVIEW посредством вызова функций блок-диаграммы, подобных приведенным на рис. 13.13 и 13.14. Но сервер ВП

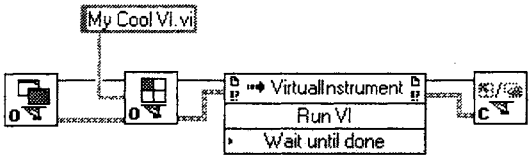


Рис. 13.13. Этот фрагмент запускает ВП **My Cool VI.vi** *на локальном компьютере*

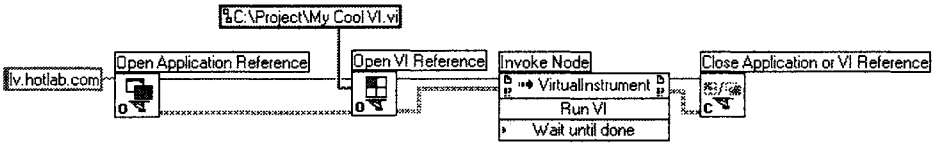


Рис. 13.14. Этот фрагмент запускает тот же ВП, но расположенный в Internet на компьютере с адресом lv.hotlab.com. В качестве пути к файлу ВП указан его локальный путь на диске удаленного компьютера

также предоставляет доступ к своим функциям из внешних приложений (например, программы на Visual Basic или макрос) в Windows через клиент автоматизации (automation client) ActiveX и из удаленного ВП LabVIEW через протокол TCP/IP. Схема, приведенная на рис. 13.15,.иллюстрирует эту архитектуру.

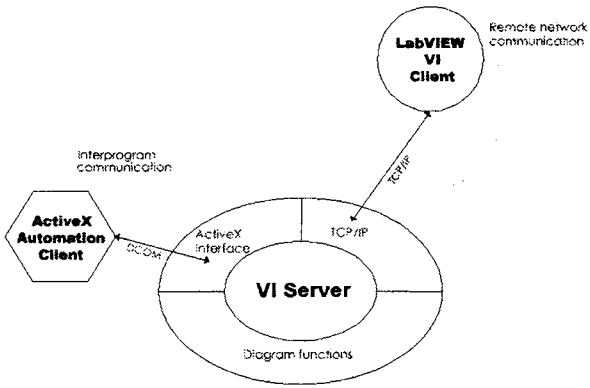


Рис. 13.15

13.3.1. Механизмы доступа к серверу ВП

Для активизации сервера ВП в LabVIEW войдите в меню **Инструменты** (Tools) и выберите **Опции ⇒ Сервер ВП: Конфигурация** (Options ⇒ VI Server: Configuration).

Функции для использования сервера ВП находятся в палитре **Управление приложением** (Application Control) – рис. 13.17. В LabVIEW определены три класса объектов, которыми можно манипулировать с помощью сервера ВП:

- класс Приложение (Application Class) – относится к самой среде LabVIEW и ее компонентам;
- класс Виртуальный прибор (VI Class) – определенный ВП в памяти или на диске;
- класс Управление (Control Class) – элемент управления или индикации на лицевой панели.

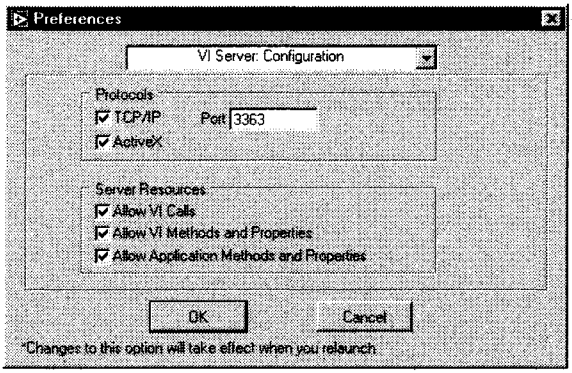


Рис. 13.16

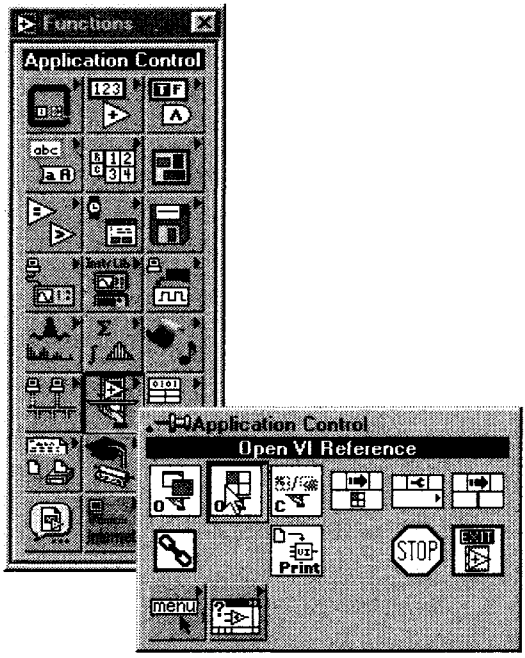


Рис. 13.17

Функции сервера ВП позволяют работать со всеми тремя классами; они кратко описываются ниже.

Рассмотрим простые примеры использования сервера ВП:

- **Ссылка на открытое приложение** (Open Application Reference) – рис. 13.18. На выходе функция возвращает ссылку на сервер ВП, запущенный на компьютере **имя компьютера** (machine name). Если на вход **имя компьютера**

Рис. 13.18. ***Ссылка на открытое приложение*** *(Open Application Reference)*

будет подана пустая строка (или вход не будет подключен), то функция возвратит ссылку на локальную среду LabVIEW, в которой она выполняется. Если вы явно задали **имя компьютера**, то функция попытается установить связь по TCP с сервером ВП этой удаленной машины по заданному порту. Она возвратит **ссылку на приложение** (application reference), которая может использоваться другими функциями сервера ВП;

- **Ссылка на открытый ВП** (Open VI Reference) – рис. 13.19. Возвращает **ссылку на виртуальный прибор** (vi reference), заданный через **путь к файлу ВП** (vi path) строкой или путем к файлу ВП на диске. Если вход **ссылка на приложение** (application reference) не подключен, то функция обращается к ВП на локальной машине; для обращения к виртуальному прибору на удаленном компьютере необходимо вначале определить **ссылку на приложение** на этом компьютере при помощи предыдущей функции;

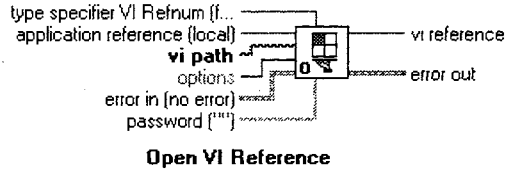


Рис. 13.19. ***Ссылка на открытый ВП*** *(Open VI Reference)*

- **Закрыть ссылку на приложение или ВП** (Close Application or VI Reference) – рис. 13.20. Закрывает открытую ссылку на объект класса Приложение или Виртуальный прибор;

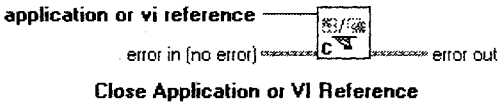


Рис. 13.20. ***Закрыть ссылку на приложение или ВП*** *(Close Application or VI Reference)*

- **Узел вызова по ссылке** (Call by Reference Node) – рис. 13.21. Узел вызова по ссылке очень похож на узел виртуального подприбора, который вызывает один ВП из другого. Входные и выходные терминалы этой

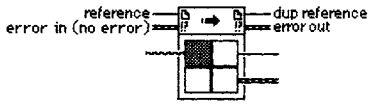


Рис. 13.21. ***Узел вызова по ссылке*** *(Call by Reference Node)*

функции подстраиваются под соединительную панель ВП, вызываемого по ссылке. Однако узел ВПП статически связан с определенным виртуальным прибором[1]. А вот функция **Узел вызова по ссылке** обращается к тому или иному ВП динамически, во время выполнения приложения, и имя вызываемого ВП определяется данными, подключенными ко входу **ссылка** (reference), который расположен в верхней части узла. Естественно, ВП, который вызывается при помощи функции **Узел вызова по ссылке**, может быть расположен и на удаленном компьютере;

- **Узел свойств** (Property Node) – рис. 13.22. Устанавливает (записывает) или считывает информацию о свойствах виртуального прибора или приложения. Чтобы выбрать класс Виртуальный прибор или Приложение, вызовите контекстное меню узла свойств и отметьте подменю **Выбрать класс LabVIEW** (Select LabVIEW Class). Для установки класса Приложение укажите опцию **Приложение** (Application), для класса ВП – опцию **Виртуальный прибор** (VI). Выбор класса осуществляется LabVIEW автоматически, если вы подключите к входу **ссылка** (reference) узла свойств ссылку (refnum) на объект данного класса;

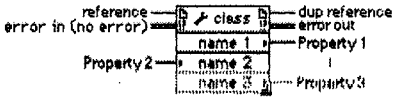


Рис. 13.22. ***Узел свойств*** *(Property Node)*

- **Узел вызова** (Invoke Node) – рис. 13.29. Вызывает *метод* или *действие* над объектом класса Приложение или Виртуальный прибор. Большинство методов имеет набор относящихся к ним параметров. Для задания метода щелкните правой кнопкой мыши в любой точке узла вызова и отметьте раздел **Методы** (Methods). После выбора метода под его именем появится поле, которое может содержать любое количество параметров, ассоциированных с указанным методом. Как и раньше, допустимо

[1] Его соединительная область неизменна до тех пор, пока вы не переопределите ее или не измените сам ВПП. – *Прим. перев*

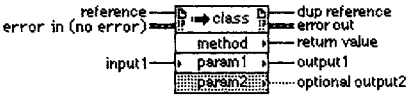


Рис. 13.23. ***Узел вызова*** *(Invoke Node)*

устанавливать (записывать) и считывать значения параметров. Параметры, размещенные в полях с белым фоном, являются *обязательными* входами, а параметры с серым фоном – *рекомендуемыми* (необязательными) входами.

Упражнение 13.3: использование свойств класса Приложение

В этом упражнении сервер ВП служит для определения операционной системы, на которой выполняется приложение, поиска пути установки среды LabVIEW и имени текущего пользователя системы.

1. Начав работу «с чистого листа», постройте блок-диаграмму, показанную на рис. 13.24. Для этого выберите функцию **Узел свойств** (Property Node) из палитры **Управление приложением** (Application Control), установите класс Приложение. С созданных для каждого из выбранных свойств индикаторов вы считаете имя операционной системы[1], каталог, в котором находится среда LabVIEW, и имя текущего пользователя LabVIEW.

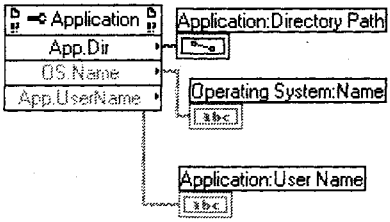


Рис. 13.24

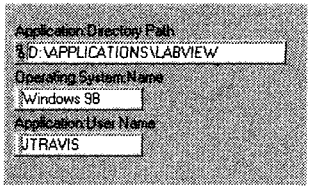


Рис. 13.25

При работе с классом Приложение на локальном компьютере вы можете опустить вызов функции Ссылка на открытое приложение, которая обычно должна предшествовать узлам свойств и вызовов.

2. Запустите созданный ВП и просмотрите результаты на индикаторах лицевой панели.
3. Сохраните ваш виртуальный прибор как **AppClassDemo.vi**.

[1] Следует различать имя и номер версии ОС: например, Windows 2000 является системой под именем Windows NT версии 5.0. – *Прим. перев.*

А теперь посмотрим, каким образом вы можете использовать сервер ВП для того, чтобы одному ВП изменить значение в элементе управления лицевой панели другого ВП без подключения их друг к другу.

Упражнение 13.4: использование свойств и методов класса Виртуальный прибор

В этом упражнении вы будете управлять ВП и его элементом управления из главного виртуального прибора через интерфейс сервера ВП.

1. Создайте простой ВП **Chart.vi**, как показано на рис. 13.26.
2. Откройте виртуальный прибор **Master.vi** из библиотеки `Ch13.llb` на компакт-диске. Убедитесь, что **Chart.vi** все еще открыт, но не запущен.
3. Запустите **Master.vi** – главный виртуальный прибор. Он выполнит следующие операции:
   - откроет лицевую панель **Chart.vi**;
   - отцентрирует лицевую панель на экране;
   - запустит ВП **Chart.vi**;
   - позволит изменить значение элемента управления (в данном случае переключить логический тумблер) **new data?** во время выполнения.

Изучите приведенную блок-диаграмму, чтобы понять, как реализованы поставленные задачи.

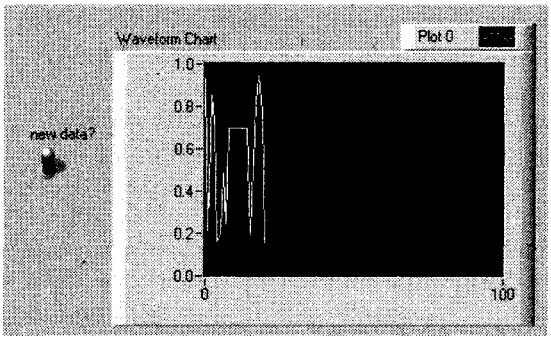


Рис. 13.26

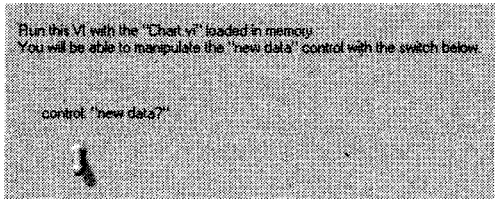


Рис. 13.27

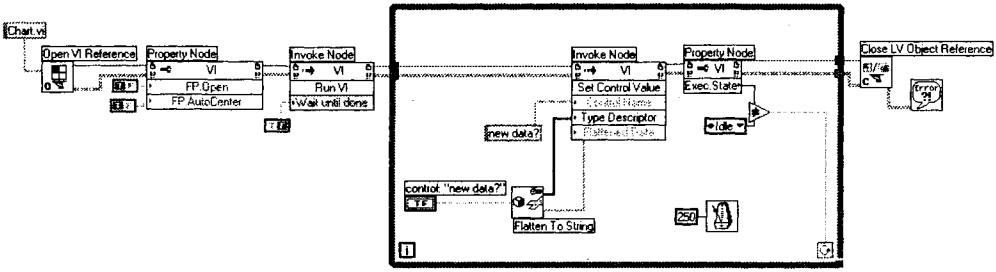


Рис. 13.28

В начале этого раздела мы уже предупреждали: не беспокойтесь, если сервер ВП кажется очень сложным и запутанным – просто он не похож на то, с чем вы сталкивались в LabVIEW ранее, он нарушает все привычные правила взаимодействия виртуальных приборов друг с другом. Но как только вы освоитесь с программированием в LabVIEW, обязательно вернитесь к этой теме. Вам помогут многочисленные примеры использования сервера ВП, которые встроены в LabVIEW.

13.3.2. Управление с клавиатуры

Если вы относитесь к тому типу людей, которые считают мышь недружелюбным созданием (или так думают пользователи разрабатываемого вами приложения[1]), то для вас есть хорошая новость! Разрешается настроить ВП таким образом, чтобы пользователи могли переключаться между элементами управления с помощью клавиши <tab>. Вообще говоря, допустимо использовать эту клавишу для выбора элемента управления, который будет принимать входные данные, и без какой-либо специальной настройки. Однако выбирать индикаторы таким же способом нельзя, поскольку индикаторы не допускают ввода данных с лицевой панели. Выбранный элемент управления – называемый еще **объект фокусировки** (Key Focus) – выделяется прямоугольником из ограничительной линии. Как только элемент управления становится объектом фокусировки, вы можете использовать клавиатуру, чтобы ввести его значение. Вот несколько полезных советов:

- если вы вводите значение в выбранный элемент управления с клавиатуры, то по завершении ввода нажмите клавишу <enter>, чтобы введенное значение вступило в силу;
- в числовых и циклических элементах управления вы также можете использовать клавиши управления курсором для пошагового увеличения/уменьшения числа. Нажатие клавиши <shift> совместно с клавишами-стрелками

[1] А если кроме шуток, то в ряде стандартов на программные средства промышленной автоматизации критических производств существует либо специальный запрет на использование графических указателей вообще, либо конкретизируется тип – чаще всего разрешены *трекболы* (trackball). – *Прим. перев.*

«вверх» или «вниз» ускоряет изменение числа. Разрешается установить минимальное приращение (инкремент) в опции **Диапазон значений** (Data Range) контекстного меню элемента управления;

- у логических элементов управления клавиша <enter> переключает логическое состояние;
- табуляция по элементам управления обычно происходит в той же последовательности, в какой вы создавали эти элементы управления.

Для виртуальных приборов с несколькими элементами управления вы можете установить иной порядок перехода от одного элемента к другому, то есть указать, какой элемент будет выбран объектом фокусировки при следующем нажатии клавиши <tab>. Эта последовательность известна в LabVIEW как *порядок обхода панели* (panel order). Чтобы его изменить, вызовите утилиту **Установить порядок табуляции** (Set Tabbing Order) из меню **Правка** (Edit). Она работает точно так же, как утилита **Порядок кластера** (Cluster Order), описанная в главе 7. На рис. 13.29 показано, как выглядит лицевая панель во время работы утилиты **Установить порядок табуляции**.

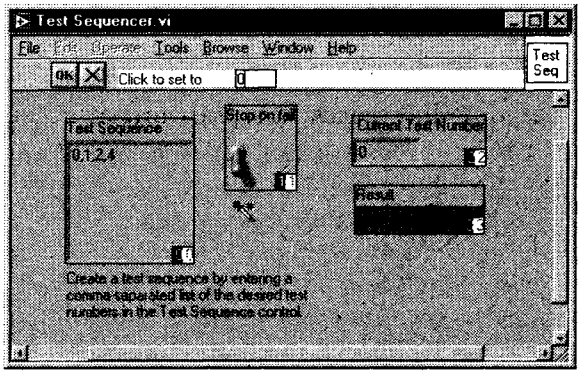


Рис. 13.29

На лицевой панели все элементы управления будут окружены окнами, содержащими два числа в нижнем правом углу. Число на белом фоне представляет собой предыдущий порядок обхода панели; число на черном фоне – новый, который вы устанавливаете. Чтобы создать новый порядок обхода панели, последовательно щелкните мышью по каждому элементу управления в желаемой последовательности либо наберите последовательность номеров в панели меню, а затем нажмите кнопку **OK**. Для отмены всех изменений щелкните по кнопке **X**.

Вы можете также назначить элементу управления клавиши быстрого вызова. Выбор опции **Расширенные ⇒ Управление с клавиатуры** (Advanced ⇒ Key Navigation) из контекстного меню элемента управления вызывает диалоговое окно, показанное на рис. 13.30.

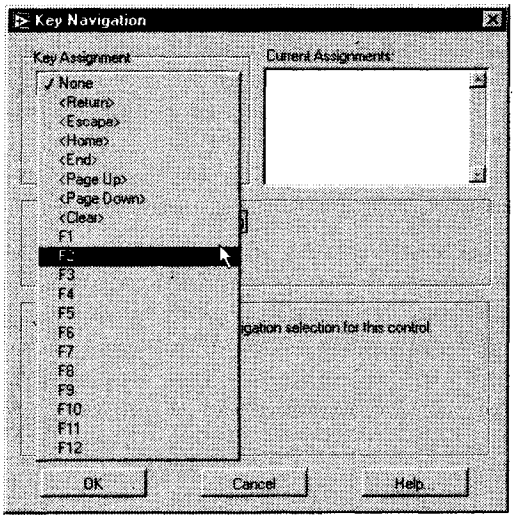


Рис. 13.30

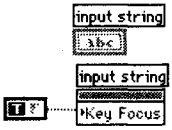


Рис. 13.31

Допустимо назначить элементу управления как функциональные клавиши (<F1>, <F2> и т.д.), так и сочетания функциональных клавиш с клавишами-модификаторами (modifiers), такими как <ctrl>. Нажатие на выбранную клавишу устанавливает объект фокусировки на указанном элементе управления. **Управление с клавиатуры** удобно применять, если у вашего ВП много элементов управления, но некоторые из них используются чаще.

Наконец, вы можете программно установить или запретить объект фокусировки для любого элемента управления. Элементы управления имеют свойство **объект фокусировки** (Key Focus), которое является логическим состоянием. Если это состояние установлено в ИСТИНУ, то элемент управления выбран и готов для ввода с клавиатуры.

Упражнение 13.5: виртуальный прибор авторизации

Создайте виртуальный прибор (A Login VI), реализующий процедуру авторизации в приложении (Login), лицевая панель которого показана на рис. 13.32. Этот ВП вначале устанавливает объектом фокусировки окно **Имя пользователя**. ВП должен распознать конец ввода имени (то есть нажатие клавиши <return> или <enter>) и переместить фокусировку на поле ввода **Пароль**. Символы, напечатанные в этом поле, не должны отображаться. И наконец, лицевая панель ВП должна появится в центре экрана, подобно стандартному диалоговому окну операционной системы.

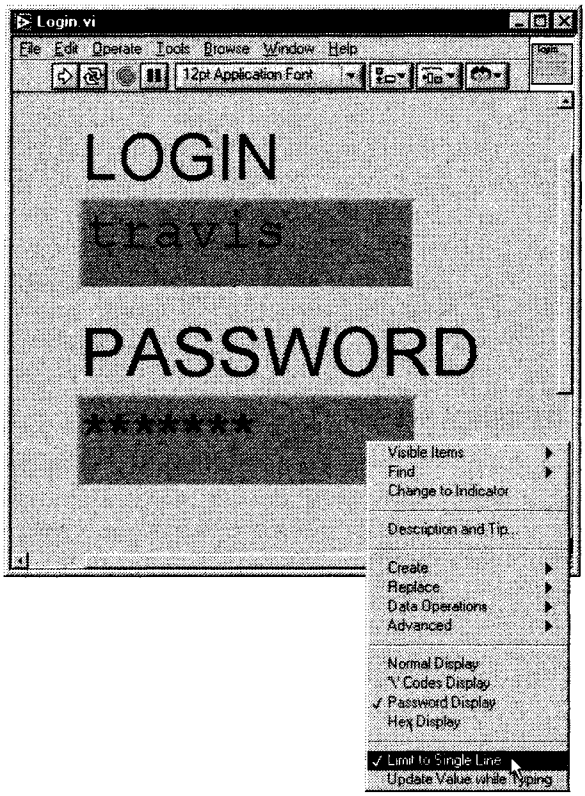


Рис. 13.32

1. Вам понадобится атрибут **Объект фокусировки**.
2. Контекстное меню строкового элемента ввода содержит опцию **Пароль** (Password), делающую введенные символы нечитаемыми.
3. В контекстных меню обоих строковых полей выберите режим **Ограничить до одной строки** (Limit to Single Line).
4. Как ВП узнает, завершил ли пользователь ввод регистрационного имени **имя пользователя**? Вам, вероятно, пригодится локальная переменная; периодически считывая ее, можно проверять, не содержит ли строка символ перевода строки/возврата каретки.

Решение этого упражнения приведено на компакт-диске.

13.4. Система счисления и единица размерности

Полезной особенностью числовых типов в LabVIEW является возможность манипулирования ими больше, чем просто числами. Числовые элементы управления и индикации LabVIEW могут иметь дополнительное представление, называемое *системой счисления* (radix), и *единицу размерности* (unit). Обычно мы представляем числа в десятичном формате, но для некоторых приложений нужны числа в двоичном, восьмеричном или шестнадцатеричном представлении. По аналогии может оказаться полезным связать определенные числовые переменные с некими единицами измерения (сантиметры, калории или градусы Цельсия) – особенно если в ходе вычислений планируются преобразования размерности.

13.4.1. Системы счисления

Числовые индикаторы LabVIEW довольно гибкие. Вы можете отображать числа в двоичном, восьмеричном, десятичном или шестнадцатеричном формате, а также в формате международной системы единиц СИ, выбрав пункт **Видимые элементы ⇒ Система счисления** (Visible Items ⇒ Radix) из контекстного меню любого числового объекта. Щелкните инструментом управления (на сей раз левой кнопкой мыши – это не контекстное меню!) по появившейся слева от числа крошечной букве, чтобы вызвать меню выбора системы счисления.

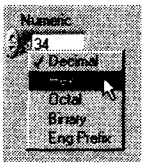


Рис. 13.33

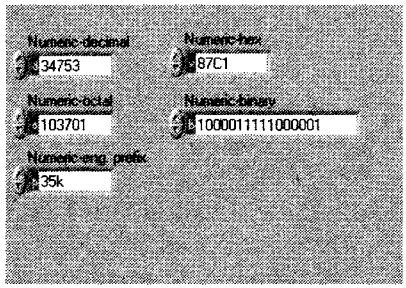


Рис. 13.34

Если тип вашей числовой величины не целочисленный, все варианты представления, кроме десятичного (Decimal) и формата СИ, будут запрещены.

Пользователь, который работает с цифровым вводом/выводом (digital I/O), найдет очень полезным представление числа в двоичном формате, поскольку это дает возможность увидеть точное соответствие каждой цифры числа состоянию каждой цифровой линии.

13.4.2. Единицы размерности

Любому числовому элементу управления (и константе) с плавающей запятой можно сопоставить физическую величину определенной размерности, например метры, километры в секунду и т.д. Метка размерности объекта доступна из контекстного меню объекта: **Видимые элементы** ⇒ **Метка размерности** (Visible Items ⇒ Unit Label).

Как только поле метки размерности появится на экране, вы сможете ввести в нем нужную единицу, используя стандартные обозначения, такие как m для метров, A для Ампер, kg для килограммов и т.д. Если вы введете единицу, незнакомую для LabVIEW, среда проинформирует вас об этом с помощью знака вопроса (?) в поле метки. Если вы хотите узнать, какие единицы размерности знакомы LabVIEW, то введите какую-нибудь простую метку, например m, а затем вызовите контекстное меню на этой метке и выберите пункт **Единица** (Unit). Появится диалоговое окно, содержащее описание размерных единиц, определенных в LabVIEW (рис. 13.36).

Рис. 13.35

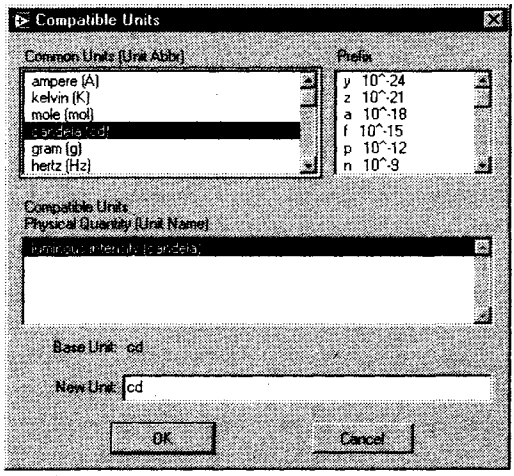


Рис. 13.36

Численная величина с сопоставленной единицей размерности должна иметь тип данных с плавающей запятой.

LabVIEW сохраняет связь величины с единицей размерности в ходе всех операций, проводимых с данными. Более того, источник «размерных» данных может подключаться проводниками лишь к приемникам данных с совместимой размерностью, как это показано на рис. 13.37.

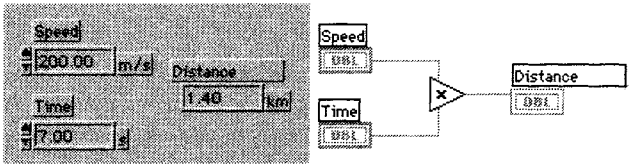


Рис. 13.37

Когда мы говорим «совместимый», то имеем в виду, что конечная единица размерности должна иметь смысл с физической или математической точки зрения. Конечная единица не обязана принадлежать к той же самой классификационной системе единиц (например, единицы СИ и традиционные англо-американские единицы) – рис. 13.38. Таким образом, в LabVIEW вы можете «прозрачно» преобразовывать совместимые единицы из различных систем.

Рис. 13.38

Нельзя соединить сигналы с несовместимыми размерностями, как это показано на следующей иллюстрации. Если вы сделаете подобную попытку, то в окне ошибки (которое вскоре будет обсуждаться) появится сообщение «Проводник данных: конфликт размерностей» (Wire: unit conflict).

Некоторые функции допускают неопределенность при работе с размерными данными, поэтому их использование не разрешено. Если вы, например, хотите выполнить арифметическую функцию **Увеличить на один** (Add One) над величиной с размерностью длины, функция не сумеет догадаться, следует ли ей добавить один метр, километр или, скажем, фут...

Наконец, следует проявить осторожность при выполнении арифметических действий, которые объединяют размерные и безразмерные числа. Например, можно *умножить* безразмерную константу на размерное число, но не *сложить* их. Однако LabVIEW позволяет «присваивать» размерную единицу безразмерному числу при помощи функции **Преобразовать размерность** (Convert Unit) из подпалитры **Преобразование** (Conversion) в палитре **Числовые** (Numeric), как показано на рис. 13.39 и 13.40.

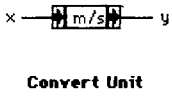


Рис. 13.39. Функция ***Преобразовать Размерность***

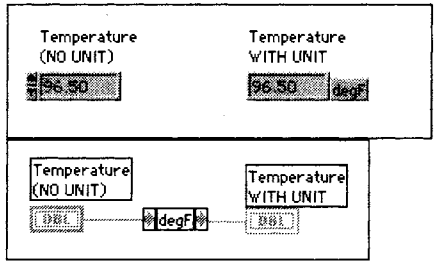


Рис. 13.40

Эта функция преобразует физическое число (размерное) в безразмерное и наоборот. Если на входе обычное число, на выходе будет число с размерностью, заданной в поле функции. Если на входе размерное число, на выход передается входное число, измеренное в заданных единицах

Для задания размерной единицы введите ее обозначение в поле функции (среднее окно терминала) либо, как и в случае с числовыми величинами, щелкните правой кнопкой мыши по терминалу и выберите пункт **Единица вывода** (Unit) для перечня допустимых единиц. Функция **Преобразовать размерность** также позволяет «изъять» из числовой величины признак размерности.

13.5. Автоматическое создание виртуального подприбора из фрагмента блок-диаграммы

Одним из ключей к правильному написанию приложений с наименьшим количеством ошибок является *модульность* (modularity). Модулями в LabVIEW являются, конечно, виртуальные подприборы (ВПП). Если вы пишете большое приложение, очень важно разделить его на отдельные задачи и выделить их в ВПП. Опция меню **Правка ⇒ Создать ВПП** (Edit ⇒ Create SubVI) легко преобразует фрагмент блок-диаграммы в ВПП. Предположим, вы работаете над каким-то приложением и хотите превратить определенную его часть в подприбор. Нет проблем! Выделите фрагмент блок-диаграммы ВП и вызовите опцию **Правка ⇒ Создать ВПП**. LabVIEW преобразует этот фрагмент в ВПП и автоматически создает для него элементы управления и индикаторы. Новый ВПП замещает выбранный участок блок-диаграммы ВП, и LabVIEW автоматически подключает ВПП к имеющимся проводникам. Описанная опция также отлично подходит при необходимости повторения фрагмента блок-диаграммы в другом виртуальном приборе.

Несмотря на легкость в применении, существуют и некоторые ограничения на использование опции **Создать ВПП**:

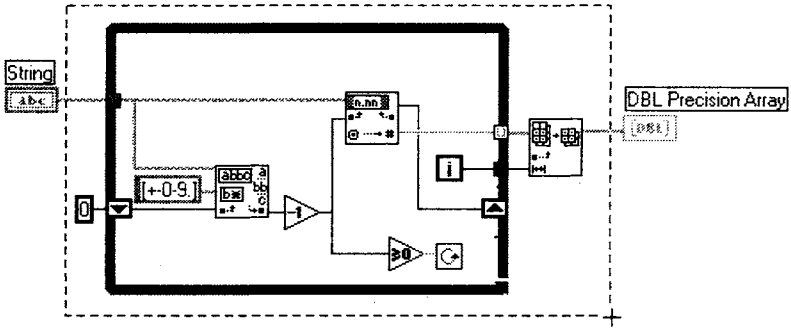


Рис. 13.41

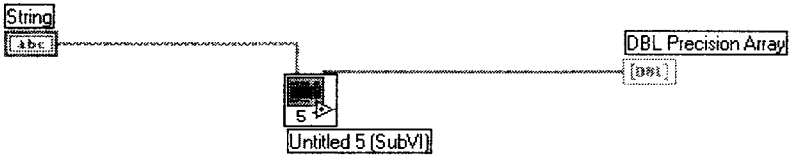


Рис. 13.42

Поскольку преобразование фрагмента блок-диаграммы в ВПП меняет поведение приложения в целом, встречаются ситуации, когда вы не сможете воспользоваться этим решением. Потенциально проблемные случаи обсуждаются далее.

- нельзя преобразовать участки блок-диаграммы, которые создадут ВПП с более чем 28 вводами/выводами, так как это максимально допустимое количество вводов и выводов на соединительной панели. Нам будет жаль разработчика, которому придется подключить даже 15 вводов! В подобном случае выберите меньший фрагмент блок-диаграммы или, что лучше, сгруппируйте данные в массивы или кластеры перед выбором преобразуемой области блок-диаграммы;
- нельзя преобразовать фрагменты блок-диаграммы, в которой выбраны элементы внутри и вне структуры, а сама структура не выбрана;
- если отмеченный фрагмент содержит локальные переменные, но не включает соответствующий элемент управления, то некая локальная переменная к элементу управления останется в вызывающем ВП и будет передавать данные в ВПП или из ВПП. Внутри ВПП первая локальная переменная для элемента управления становится инструментом

считывания или записи, а последующие локальные переменные будут ссылаться на элемент управления в ВПП. Вот это да! Что бы это значило? Итак, подчеркнем еще раз: будьте внимательны при создании ВПП из фрагмента блок-диаграммы, если он содержит локальные переменные;

- если выбранный фрагмент блок-диаграммы содержит локальные переменные или терминалы лицевой панели в теле цикла, то измеряемая ими величина может быть изменена в другом месте блок-диаграммы, пока цикл выполняется. Таким образом, когда вы преобразовываете цикл в ВПП, существует возможность изменения функциональности отмеченного фрагмента. Если вы выбрали некоторые, но не все локальные переменные или терминалы лицевой панели, то LabVIEW выдаст предупреждающее сообщение, предлагая продолжить или прекратить операцию.

Опция **Создать ВПП** удобна в случаях, когда вы начали работать с маленьким проектом (ха!), но вскоре осознали, что приложение становится все более сложным. Однако старайтесь не «запихивать» громоздкие фрагменты блок-диаграммы в виртуальные подприборы с целью просто увеличить свободное рабочее пространство – ВПП должны всегда иметь четко определенные задачи. Принимая решение о превращении фрагмента диаграммы в ВПП, спросите себя: «А смогу ли я задействовать этот участок кода или функцию где-нибудь еще?»

Мы вернемся к виртуальным подприборам и модульному программированию в главе 16.

13.6. Вспомогательные средства LabVIEW

13.6.1. Окно иерархии

Вообще **Окно иерархии** (The Hierarchy Window) становится полезным, когда вы работаете над виртуальным прибором высокой сложности, содержащим, например, 10 или более подприборов. Окно иерархии помогает отслеживать, откуда вызываются те или иные ВПП и кто их вызывает.

При открытой лицевой панели ВП выберите опцию **Обозреватель ⇒ Показать иерархию ВП** (Browser ⇒ Show VI Hierarchy) для вызова окна иерархии данного виртуального прибора. Окно иерархии в графическом виде отображает иерархию вызовов для всех виртуальных приборов в памяти, включая определения типов и глобальные переменные.

Кнопки панели инструментов в этом окне позволяют настроить несколько вариантов отображения. Например, графическое дерево вызовов может отображаться вертикально или горизонтально; разрешается спрятать или открыть столько уровней иерархии, сколько вам хочется и включить или исключить из рассмотрения глобальные переменные и определения типов. Удобная особенность этого окна: чтобы открыть лицевую панель любого ВП на диаграмме, достаточно дважды щелкнуть мышью по его иконке.

13.6.2. Поиск объектов в «виртуальном стоге сена»

LabVIEW имеет хорошую поисковую систему, поэтому вы можете быстро найти текст, объекты или виртуальные приборы в вашем проекте. Функция **Найти** (Find) в меню **Правка** (Edit) поможет обнаружить любую функцию, подпрограмму, глобальную переменную, атрибутивный узел данных, терминал лицевой панели или текст на блок-диаграмме. Допустимо ограничить диапазон поиска одним виртуальным прибором или набором ВПП, либо искать во всех виртуальных приборах в памяти.

Вы можете выбрать поиск объектов или текста, щелкнув мышью по соответствующему флажку **Что искать:** (Search for:). Если вы выбрали поиск объекта, щелкните по кнопке **Выбрать объект** (Select Object), чтобы войти в контекстное меню, которое поможет задать тип объекта для поиска. Если вы ищете текст, введите этот текст и щелкните по кнопке **Дополнительно** (More Options) для ограничения диапазона поиска фрагментами ВП и метками объектов.

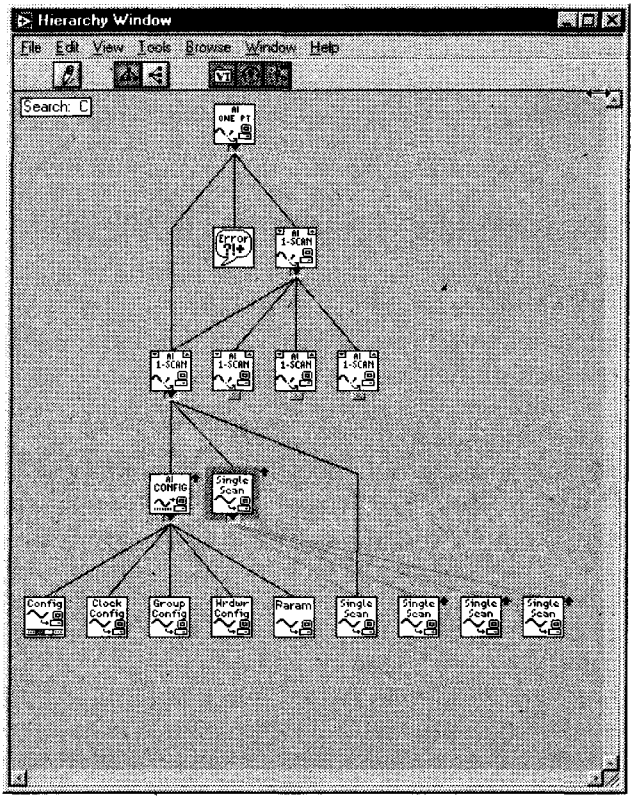


Рис. 13.43

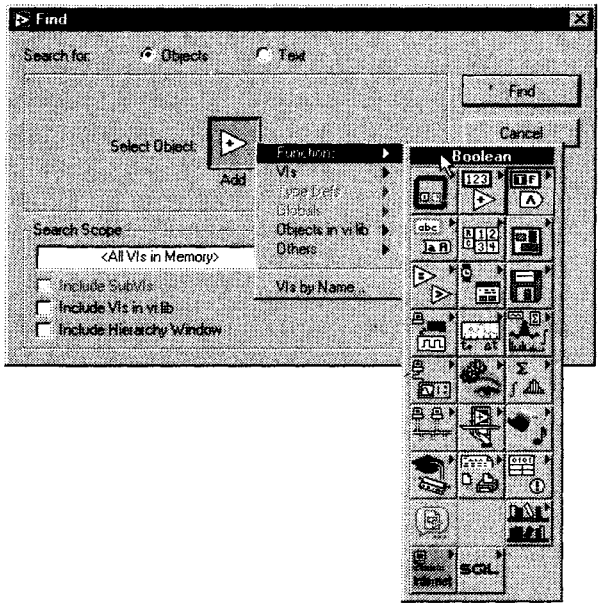


Рис. 13.44

Если LabVIEW отыскивает единственное совпадение, он открывает соответствующий ВП и выделяет найденный объект. Если же совпадений несколько, появится диалоговое окно **Результаты поиска** (Search Results), показанное на рис. 13.45.

Из окна **Результаты поиска** можно с помощью двойного щелчка мышью перейти к любому из найденных объектов.

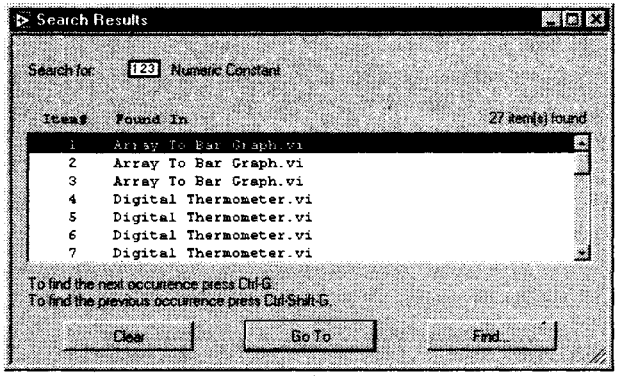


Рис. 13.45

Дополнительные инструментальные средства

В меню **Инструменты** (Tools) представлен целый набор других инструментальных средств LabVIEW, таких как Мастер создания Internet-страницы, профайлер виртуального прибора (VI Profiler), который измеряет время и память, занимаемые виртуальным прибором, и т.д. В зависимости от установленной версии LabVIEW (демонстрационная, базовая, полная или профессиональная среда разработки), а также наличия дополнительных библиотек (add-on toolkits) для LabVIEW – например, библиотека Internet-функций (Internet toolkit), библиотека для работы с базами данных (Database Connectivity toolset) или специальный набор средств для промышленной автоматизации LabVIEW Datalogging and Supervisory Control Module[1] – в этом разделе появятся соответствующие наборы функций и инструментальных средств.

13.7. Итоги

В этой главе вы познакомились с некоторыми интересными и полезными функциями LabVIEW. Мы рассмотрели такие разделы, как **Опции**, **Свойства ВП**, **Установка узла ВПП**, изучили сервер виртуального прибора, объект фокусировки, системы счисления и размерные единицы числовых данных. Вы научились создавать ВПП из выделенных фрагментов блок-диаграммы, узнали, как пользоваться иерархией виртуального прибора и средствами поиска.

Раздел **Опции** позволяет настраивать множество опций в нескольких категориях. Эти опции помогают настроить среду LabVIEW и дают возможность подогнать под свои потребности шрифты и цвета отображения, типы печати, пути к файлам и т.д.

Опции **Свойства ВП** и **Установка узла ВПП** содержат полезные настройки организации и функционирования виртуальных приборов как самостоятельных приложений или как подприборов. С помощью установок **Свойства ВП** вы можете прятать или открывать кнопки панели инструментов, управлять размерами и положением окна ВП, настраивать опции выполнения ВП и т.д. **Установка узла ВПП** позволяет создать всплывающее окно ВПП при его вызове и закрывать его после выполнения.

Сервер ВП является эффективным объектно-ориентированным механизмом управления виртуальными приборами LabVIEW, в том числе в сети.

Объект фокусировки представляет собой логический атрибут элементов управления, который позволяет выделить элемент управления с целью ввести в него данные с клавиатуры. Вы можете установить фокусировку программно или просто переключать элементы управления по очереди с помощью клавиши <tab>, а также назначить специальные клавиши (например, функциональные), чтобы переключать логические элементы управления с клавиатуры.

[1] Полный перечень и функциональное описание всех дополнительных библиотек LabVIEW, предлагаемых National Instruments в настоящее время, представлен на сайте http://ni.com. – *Прим. перев.*

Система счисления и единицы размерности являются особыми встроенными атрибутами числовых типов в LabVIEW. Система счисления позволяет представить числовые данные в двоичном, восьмеричном, десятичном и шестнадцатеричном формате. Дополнительные встроенные единицы размерности числовых типов LabVIEW позволяют выполнять математические действия над размерными числами и обеспечивают автоматическое преобразование для различных систем единиц.

Модульность LabVIEW ярко проявляется в возможности преобразовать выбранный участок блок-диаграммы в виртуальный подприбор с соответствующими входами и выходами (при соблюдении определенных условий).

LabVIEW также предлагает некоторые вспомогательные инструментальные средства программирования, такие как **Окно иерархии**, функция **Поиск** и др.

Обзор

В этой главе вы изучите различные методы соединения виртуальных приборов LabVIEW с Internet, локальной сетью и другими программами и увидите, как можно использовать встроенный Web-сервер LabVIEW для размещения лицевой панели ВП в Internet. Вы познакомитесь с протоколом DataSocket и его возможностями по обеспечению совместного доступа к данным среди виртуальных приборов в локальной сети. Используя сервер ВП, вы начнете программно контролировать многие настройки виртуальных приборов и LabVIEW. ActiveX (Windows) и AppleEvents (MacOS) позволяют LabVIEW взаимодействовать с внешними программами. Наконец, вы увидите, как LabVIEW органично встраивается в картину большого предприятия.

ЗАДАЧИ

- Познакомиться с основными протоколами LAN и Internet (URL, TCP/IP, HTTP, и т.д.)
- Научиться использовать Web-сервер LabVIEW для размещения ВП в Internet
- Использовать DataSocket для публикации и подписки на источники данных
- Изучить возможности ActiveX в LabVIEW
- Познакомиться с возможностями низкоуровневых сетевых функций LabVIEW: ВП TCP и UDP
- Понять роль взаимодействия баз данных, отчетов, публикации в Internet и виртуальных приборов

ОСНОВНЫЕ ТЕРМИНЫ

- Internet
- Клиент-сервер
- URL
- HTTP
- Web-сервер
- DataSocket
- dstp
- ActiveX
- AppleEvents
- TCP/IP
- UDP
- Предприятие
- Базы данных

КОММУНИКАЦИОННЫЕ ВОЗМОЖНОСТИ LabVIEW

14

14.1. LabVIEW, работа в сети и Internet

Одной из главных целей применения *виртуального прибора*, которую мы пытаемся показать в этой книге, является создание мощных, гибких и недорогих измерительных систем, построенных на основе персональных компьютеров с использованием программного обеспечения в качестве механизма и интерфейса. Виртуальный прибор может легко экспортировать данные и обмениваться информацией с другими видами программного обеспечения, поскольку они часто находятся на одном и том же компьютере.

LabVIEW, разработанный более 15 лет назад компанией National Instruments, – это уникальная программа для создания виртуальных приборов. Привлекательность LabVIEW в том, что это графический язык программирования, предназначенный специально для инженеров. В результате легко создать прототип и разработать программу за короткий промежуток времени по сравнению с написанием программы на языке типа C++. Существует интересная параллель между популярностью LabVIEW и популярностью сети Internet: своим успехом они обязаны не лежащей в их основе передовой технологии, а хорошо продуманному дизайну графического интерфейса, который делает эти приложения доступными. В конце концов почти все, что позволяет делать LabVIEW, можно было написать на C или ассемблере задолго до разработки LabVIEW. Появление Internet датируется 60-ми годами. В основе создания этих революционных инструментов лежит разработка интуитивного, дружественного интерфейса. Для LabVIEW сказанное означает программирование путем соединения графических объектов наподобие построения электрических схем. Для Internet – появление Web-браузера, который, помимо всего прочего, использует выделение и щелчки мышью по изображениям или словам, представляющим интерес и связанным с другими местами во Всемирной паутине.

Итак, можно сказать, что применение LabVIEW для создания приложений, доступных в Internet, соединяет лучшие черты интерфейсов этих двух программ. Существующие возможности для создания простых в использовании и доступных через сеть программ поднимают виртуальные приборы на совершенно другой уровень.

В этой главе вы познакомитесь с рядом инструментов LabVIEW, которые позволяют публиковать виртуальные приборы в Internet, обмениваться данными через сеть, взаимодействовать с другими программами и т.д. Если вы хотите поближе познакомиться с тематикой виртуального оборудования, доступного через Internet, прочтите мою книгу «Internet Application in LabVIEW» (Prentice Hall, 2000). Много дополнительной информации представлено также на сайте http://jeffreytravis.com/books.

14.1.1. Internet и модель клиент/сервер

Прежде чем разбирать практические примеры, давайте кратко рассмотрим некоторые базовые технологии, доступные для использования в виртуальных системах, связанных с Internet.

Сам термин *Internet* является сокращением от *interconnected network* (взаимосвязанная сеть). За основное определение сети можно взять следующее: сеть – это два или более электронных устройства, связанные таким образом, чтобы обеспечить обмен информацией между собой. В информационной технологии сеть рассматривается как последовательность точек или узлов, соединенных каналами связи. Internet состоит из тысяч соединенных сетей. Он представляет собой систему клиент/сервер, где ваш Web-браузер является клиентом, а любой сервер узла Internet (Website) – сервером.

В модели клиент/сервер, общей для сетевых приложений, один набор процессов (клиенты) требует обслуживания другим набором процессов (серверы). Если вы хотите контролировать программу LabVIEW с портативного компьютера, запросите с сервера информацию, которая должна быть доступна потребителю – в данном случае вашему компьютеру. По этому сценарию сервер обычно ожидает, чтобы клиент начал процесс соединения. Во многих случаях сервер может одновременно обслуживать большое количество клиентов.

Итак, вы познакомились с тем, как функционирует Internet в модели клиент/сервер. Вы наверняка знакомы также с электронной почтой (e-mail) и FTP (протокол передачи файлов), которые используются в модели клиент/сервер для передачи информации.

Каким образом информация передается от сервера к клиенту? Если говорить просто, сначала к информации открывается доступ, она разбивается на сегменты, затем отсылается в электронном виде по различным сетевым маршрутам, переупорядочивается и заново собирается. На самом деле все это выполняется посредством несколько усложненной последовательности *протоколов обмена*. Они

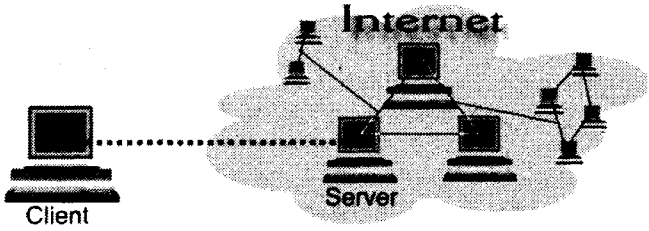


Рис. 14.1

представляют собой наборы правил или «язык», который позволяет двум компьютерам или приборам общаться друг с другом. Протоколы существуют на нескольких *уровнях*, начиная от высокоуровневых протоколов (таких, как протокол HTTP), построенных на основе низкоуровневых протоколов (HTTP зависит от TCP/IP), которые в свою очередь могут зависеть от низших протоколов (TCP/IP нуждается в протоколе физического уровня, таком как Ethernet). Наиболее общими протоколами обмена, с которыми вы можете столкнуться в процессе создания ВП с сетевым доступом, являются TCP/IP, UDP, HTTP, DSTP (Data Socket Transport Protocol) и WAP.

14.1.2. Выбор технического решения с помощью LabVIEW

Если вы используете LabVIEW как основное программное обеспечение для сбора данных и их анализа, то при создании системы контрольно-измерительного оборудования с доступом в Internet у вас появится широкий выбор инструментов. LabVIEW обладает рядом встроенных возможностей для организации связи через Internet, в том числе:

- функции TCP/IP и UDP;
- встроенный Web-сервер, позволяющий создать изображения лицевой панели ВП «на лету»;
- сервер виртуальных приборов LabVIEW – мощная структура для виртуальных приборов и программ с технологией ActiveX, взаимодействующих посредством сети;
- протокол DataSocket для обмена данными через LAN или Internet;

  Кроме того, при использовании дополнительных инструментов и других пакетов разработки LabVIEW может быть оснащен дополнительными технологиями, такими как:

  – Java-приложения для дистанционного управления ВП;
  – элементы управления ActiveX;
  – поддержка CGI на G Web-сервере;
  – e-mail, FTP и Telnet.

14.2. Общее представление о работе Internet

Наиболее распространено использование ВП с сетевым доступом для дистанционного мониторинга и управления через Internet. Прежде чем узнать, как это делается, познакомимся с работой Всемирной сети.

14.2.1. Анатомия URL

Работая в сети, нужно прежде всего представлять, что такое формат URL (Uniform Resource Locator – унифицированный указатель информационного ресурса). URL дает возможность определить, где документ находится в сети, задавая его адрес в Internet и маршрут. URL можно видеть всегда, когда вы вводите адрес в Web-браузер, например `http://www.ni.com/labview/index.html`.

Основным синтаксисом для URL большинства документов Internet является

```
http://host/path,
```

где `http` обозначает использование протокола передачи гипертекстовых файлов, `host` является Internet-адресом узла WWW (то есть www.ni.com), а `path` (то есть labview/index.html) указывает на документ, запрашиваемый на сервере.

Большая часть Web-браузеров позволяет опустить часть URL, используя значения по умолчанию. Например, если вы пропустите «http://» в начале URL, браузер все же задействует HTTP для поиска документа. Кроме этого, если `path` указывает на директорию вместо файла, то сайт покажет документ по умолчанию внутри заданной директории. Именно сервер, а не браузер управляет тем, какое имя документа указано по умолчанию. Обычно это что-то вроде `index.html` или `default.htm`.

Рис. 14.2. Протоколы, используемые в URL

Хотя HTTP наиболее часто применяется для запроса документов посредством Internet, существуют другие схемы для получения доступа к данным. Более обобщенным синтаксисом для URL является

```
scheme://host[:port]/path[extra-path-info][?query-info],
```

где `scheme` является протоколом, используемым для подключения к `host`. Для сайтов Internet протоколом является `http`; для сайтов FTP – `ftp`.

В табл. 14.1 представлены наиболее часто используемые протоколы.

Не гипертекстовый URL может выглядеть следующим образом:

```
news://comp.lang.labview
```

Таблица 14.1

Протокол	Значение
http://	Протокол передачи гипертекстовых файлов, для узлов WWW
ftp://	Протокол передачи файлов, для сайтов FTP
gopher://	Гоферовские серверы, устаревший тип сервера, индексирующего документы
news://	Серверы новостей
file://	Протокол, не использующий TCP/IP, но указывающий документ посредством локальной файловой системы
telnet://	На самом деле это не URL (поскольку не указывает на документ), но может применяться для организации сессии Telnet с определенным хостом
dstp://	DataSocket Transport Protocol, запатентованный протокол компании National Instruments для пересылки данных посредством сети

Чтобы посмотреть содержимое этого URL, необходимо запустить клиент чтения новостей и отправится по данному адресу группы новостей.

- `host` – Internet-адрес хоста;
- `port` – необязательный номер порта для запрашиваемого сервиса. Обычно порт можно опустить, если сервис работает через порт по умолчанию. Например, Web-сервер обычно использует порт 80, так что URL. `http://www.travis.to:80` аналогичен `http://www.travis.to`. Однако, если Web-сервер работает через порт 8172 (например), URL необходимо определить в виде `http://www.travis.to:8172`;
- `path` – путь к документу или источнику данных;
- `extra-path-info` – необязательная информация, иногда употребляемая программами CGI;
- `query-info` – необязательные параметры, используемые программами CGI; всегда в начале имеют символ «знак вопроса» (?).

Большинство браузеров, таких как Netscape Communicator или Internet Explorer, позволяют получить документы, используя протоколы HTTP, FTP, Gopher или File. Кроме этого, часто вы можете так настроить браузер, чтобы он запускал дополнительные приложения для работы с Telnet или группами новостей.

14.2.2. Кодирование URL

Когда вы используете приложения, создающие URL, важно понимать синтаксические правила написания URL. Весь набор правил называется *кодированием URL*. Кроме синтаксиса URL, который мы уже видели, существуют специальные символы, которые нельзя ввести в URL как обычные символы ASCII. Пробелы, вопросительные знаки, символ @ и т.д. имеют особое значение при использовании

в URL. Эти символы необходимо представлять как «%», за которым следует двухцифровое шестнадцатеричное ASCII-значение символа.

Например, если вы запрашиваете файл с названием, имеющим пробел, такой как `Come&Get It.html`, то кодированный URL будет выглядеть следующим образом:

```
http://myserver/Come%26Get%20It.html
```

где `%26` заменяет символ &, а `%20` заменяет пробел (`0x26` является шестнадцатеричным значением &, а `0x20` – шестнадцатеричным значением пробела).

В отдельных случаях символ пробела вдобавок к шестнадцатеричному представлению может быть закодирован в виде знака «+». Таким образом, документ `Hello World.html` можно закодировать как `http://myserver/Hello%20World.html` или `http://myserver/Hello+World.html`.

14.2.3. Web-браузеры, сетевые серверы и протокол передачи гипертекстовых файлов

Всемирная паутина (World Wide Web) в основном работает с архитектурой клиент/сервер, как показано на рис. 14.3.

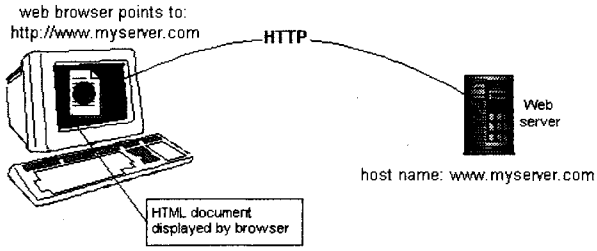


Рис. 14.3

Web-браузеры – это клиенты, а WWW-серверы – серверы. HTTP – это протокол связи между ними. Кратко рассмотрим каждый из этих элементов, из которых и состоит Всемирная паутина.

Web-браузеры

Web-браузер представляет собой приложение-клиент, реализующее протокол HTTP. Он может присоединяться к WWW-серверам и находить, доставлять HTML-документы и изображения. Современные Web-браузеры выполняют большее количество функций. Среди них как составная часть – клиенты электронной почты. Они могут обрабатывать различные документа, например

аудио, видео, приложение Java и элементы ActiveX. Двумя наиболее известными Web-браузерами являются *Netscape Navigator* и *Microsoft Internet Explorer*. Практически все персональные компьютеры имеют заранее установленный Web-браузер.

14.3. Публикация и управление виртуальными приборами в Internet

Часто требуется, чтобы другие пользователи могли получить доступ к вашим данным или ВП посредством Web-браузера. Вам необходимо лишь знать, будут ли эти люди просто наблюдать (считывать) или управлять (отсылать данные назад в виртуальный прибор) посредством браузера.

Для наблюдения (не управления) работы ВП вы можете использовать встроенный Web-сервер LabVIEW, о котором речь пойдет ниже.

Для управления ВП через сеть имеются различные способы. Одним из наиболее простых является применение набора инструментов LabVNC. LabVNC – это бесплатное приложение, которое содержится на компакт-диске, сопровождающем данную книгу. Последнюю версию этой программы можно найти по адресу http://jeffreytravis.com[1].

Рассмотрим Web-сервер LabVIEW подробнее. С помощью этого сервера вы можете *динамично* создавать страницы Internet с изображением лицевой панели ВП без какого-либо дополнительного кода на блок-диаграмме.

14.3.1. Настройка встроенного Web-сервера LabVIEW

Не путайте сервер виртуальных приборов (VI Server) и Web-сервер в LabVIEW. Они никак между собой не связаны и не зависят друг от друга. Web-сервер позволяет удаленному Web-браузеру просматривать изображение ВП. Сервер виртуальных приборов позволяет локальным и удаленным виртуальным приборам вызывать функции и свойства вашего ВП.

Чтобы запустить Web-сервер LabVIEW, проделайте следующее:

1. Прежде чем запустить сервер LabVIEW, убедитесь, что никакой другой сервер (например, Microsoft Personal Web Server или Internet Toolkit's

[1] Возможность контролировать LabVIEW через Internet будет доступна в будущих версиях (на момент написания этой книги LabVIEW 6.0 поддерживает только удаленное наблюдение).

server) не работает через тот же самый порт. Это необходимо для предотвращения конфликтов в системе.

2. Перейдите к опции **Инструменты ⇒ Опции ⇒ Web-сервер: Настройка** (Web Server: Configuration), как показано на рис. 14.4.

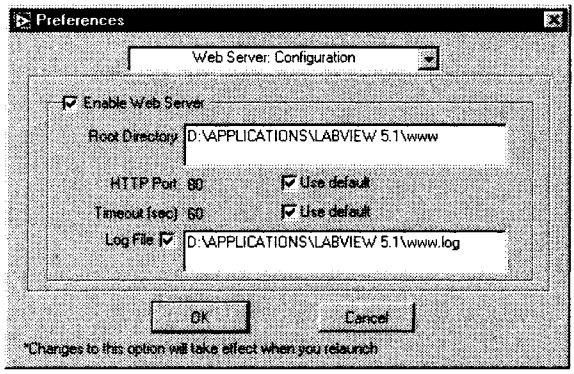


Рис. 14.4

3. Поставьте флажок **Enable Web Server**.
4. При желании измените корневой каталог. В этом каталоге вашего жесткого диска будут размещаться необходимые файлы.
5. Щелкните по кнопке **ОК**.
6. Закройте LabVIEW, затем вновь запустите (это необходимо для запуска Web-сервера).

Допустимо управлять свойствами Web-сервера LabVIEW посредством интерфейса сервера ВП. При этом необходимо использовать узел свойств (Property Node) для класса Приложение. Таким образом, вы можете динамически программно включать или выключать сервер, устанавливать разрешения в список контроля доступа и т.д. При работе этого интерфейса перезапуск LabVIEW не требуется.

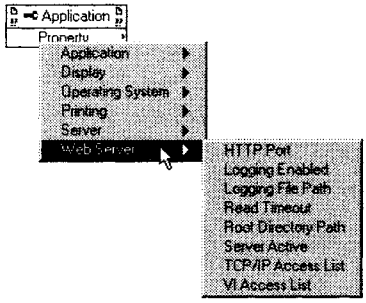


Рис. 14.5

14.3.2. Публикация в HTML с помощью Web-сервера LabVIEW

При использовании Web-сервера LabVIEW нет необходимости что-либо добавлять на блок-диаграмму. Он динамически создает изображения лицевой панели в ответ на запрос Web-браузера. Перед публикацией ВП следует загрузить в память в LabVIEW для обслуживания его Web-сервером. Последний может выдать либо статическое изображение лицевой панели ВП («моментальный снимок»), либо динамическое анимированное («дисплей»).

Следующее упражнение показывает, как легко наблюдать ВП **Temperature System Demo.vi** через Web-браузер.

14.3.3. Упражнение 14.1: использование встроенного Web-сервера LabVIEW

1. Убедитесь, что вы включили встроенный Web-сервер LabVIEW, как описано выше.
2. Откройте виртуальный прибор **Temperature System Demo.vi** (из библиотеки примеров LabVIEW `/examples/apps/tempsys.llb`).
3. Откройте Web-браузер.
4. Чтобы увидеть статическое изображение виртуального прибора, введите `http://127.0.0.1/.snap?Temperature+System+Demo.vi` в окне адреса браузера.
5. Чтобы увидеть анимированное изображение, воспользуйтесь Web-браузером, поддерживающим технологию «push» в изображениях. В браузере Netscape введите следующий URL:

   ```
   http://127.0.0.1/.monitor?Temperature+System+Demo.vi
   ```

   Вы увидите анимированные части изображения, например график. «Push» работает только в Netscape, но не в Internet Explorer. В Internet Explorer вместо отображения пиксельной анимации вы добьетесь постоянного обновления страницы.

Рассмотрим URL из предыдущего упражнения более подробно:

- `127.0.0.1` – специальный IP-адрес для локальной машины (если бы вы следили за виртуальным прибором через сеть с удаленного компьютера, вам бы потребовалось ввести реальный IP-адрес).
- `.snap?` – специальная команда, заставляющая сервер LabVIEW сделать снимок ВП, название которого будет следовать за символом «?».
- `.monitor?` – специальная команда, заставляющая сервер LabVIEW отследить и создать анимированное изображение ВП, название которого будет следовать за символом «?».
- `Temperature+System+Demo.vi` – просто закодированный URL для **Temperature System Demo.vi**. В любых указателях ресурсов пробелы должны быть заменены на знак «+» или «%20»; любые специальные

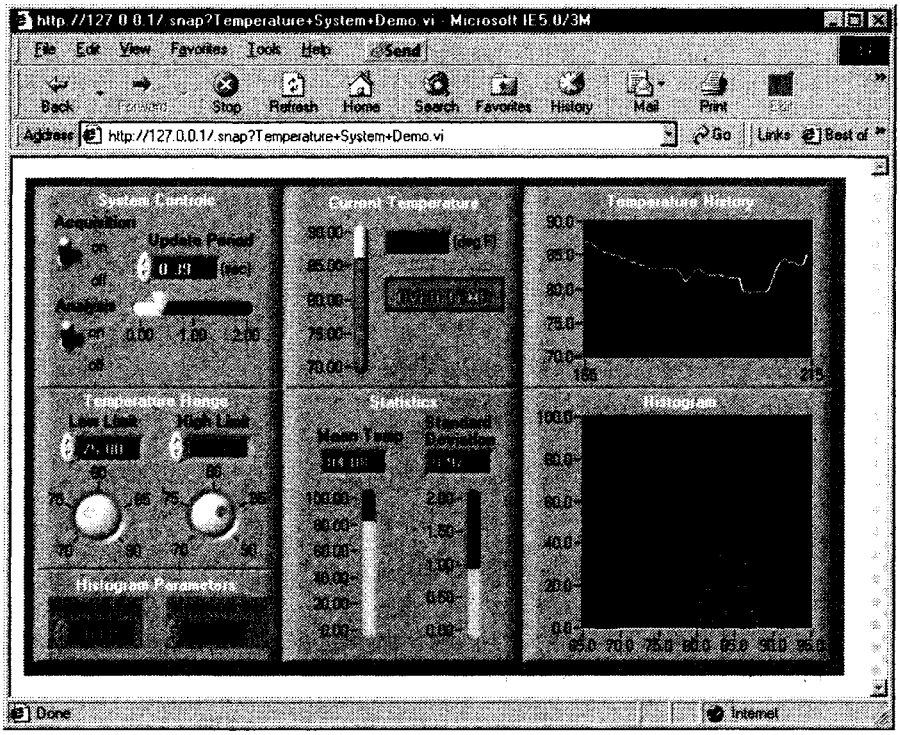


Рис. 14.6

символы также должны быть заменены (например, вместо слэша необходимо использовать соответствующий код).

Обратите внимание, что динамические документы, возвращенные Web-сервером LabVIEW, представляют собой обычные изображения лицевой панели. Они не являются документами HTML. Однако очень просто создать собственные документы HTML, которые отображали бы динамические снимки лицевой панели. Это можно сделать путем задания URL в качестве источника изображения в тэге <IMG>.

Web-сервер LabVIEW позволяет только наблюдать за работой ВП, но не управлять ею. Если это все же необходимо, используйте программное обеспечение сторонних производителей, например LabVNC с компакт-диска.

Вместе с Web-сервером LabVIEW вы можете использовать **Средства публикации в Интернет** (Web Publishing Tool) из меню **Инструменты** – инструменты Web-публикации. Эта небольшая программа создает простой файл HTML, где легко ввести вести текст вокруг изображения ВП.

Дополнительно к встроенному Web-серверу LabVIEW разрешается приобрести Internet Toolkit (набор инструментов для работы с Internet) компании National Instruments. Этот набор инструментов снабдит вас усовершенствованным Web-сервером, добавляя поддержку безопасности каталогов (пользователь/пароль/группы), e-mail, FTP и виртуальные приборы для работы с Telnet (табл. 14.2).

Таблица 14.2

LabVIEW 6i	+Internet Toolkit
http (Web) сервер	ftp
dstp datasacket	smpt (e-mail)
	telnet
	http с дополнительными возможностями, такими как CGI и работа с директориями, имеющими пароль

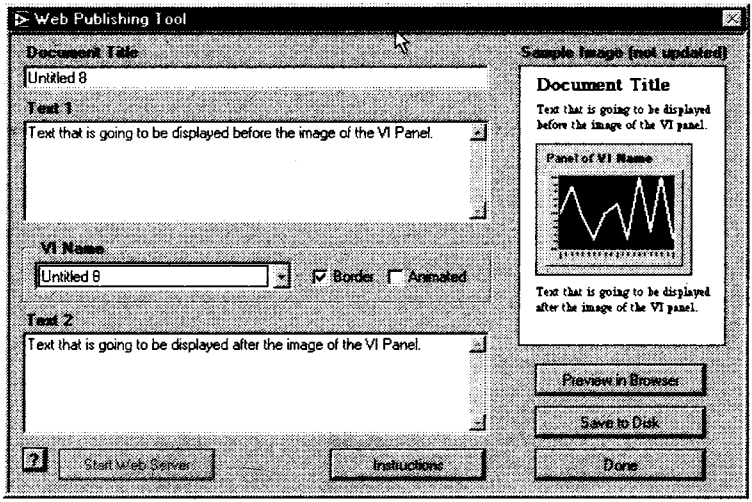


Рис. 14.7

14.4. Обмен данными в сети: DataSocket

Иногда возникает необходимость в обмене данными между несколькими компьютерами посредством сети. Предполагается, что все эти компьютеры используют LabVIEW. Как мы только что видели, Web-публикация полезна для быстрого просмотра состояния ВП, однако она не позволяет эффективно обмениваться данными.

DataSocket является интерфейсом LabVIEW, который дает возможность посредством сети опубликовать (вписать) данные и подписаться (считать) на данные, форматированные LabVIEW. Кроме LabVIEW, протокол DataSocket поддерживается другими средами программирования, такими как C++, Visual Basic и Java. Подробнее об этом вы можете узнать на сайте http://ni.com/datasocket.

В сущности, DataSocket представляет собой технологию, которая позволяет посылать и получать данные через сеть с большого количества программных платформ (включая LabVIEW), не беспокоясь о низкоуровневых элементах исполнения. С помощью DataSocket вы можете, например, посылать отформатированные данные между двумя компьютерами с LabVIEW, или между LabVIEW и LabWindows, или даже между несколькими Web-браузерами и LabVIEW.

Поскольку одним из ключевых компонентов DataSocket является *сервер DataSocket* – маленькая самостоятельная программа, – то вашей программе не нужно ни управлять подключениями TCP/IP, ни следить за скоростью ее выполнения в зависимости от числа подключенных клиентов. Кроме того, DataSocket может манипулировать несколькими типами данных, включая целые числа, числа с плавающей запятой, строки и логические значения, а также массивы всех перечисленных типов. Позволив DataSocket осуществлять управление и преобразование внутри себя при передаче данных через сеть, вы не будете заботиться об отправке заголовка или о форматировании данных.

DataSocket имеет два основных элемента, работающих вместе:

1. Сервер DataSocket;
2. DataSocket API (программный интерфейс приложения).

Сервер DataSocket является отдельной программой, которая выполняется на компьютере и управляет подключением клиентов. Клиентские подключения могут записывать данные на сервер (*источники* (*publishers*) DataSocket) или считывать

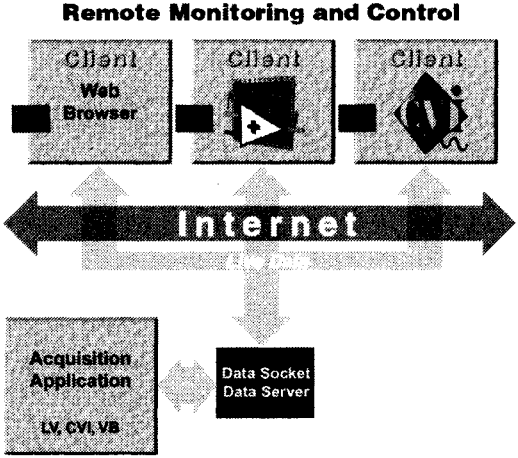


Рис. 14.8

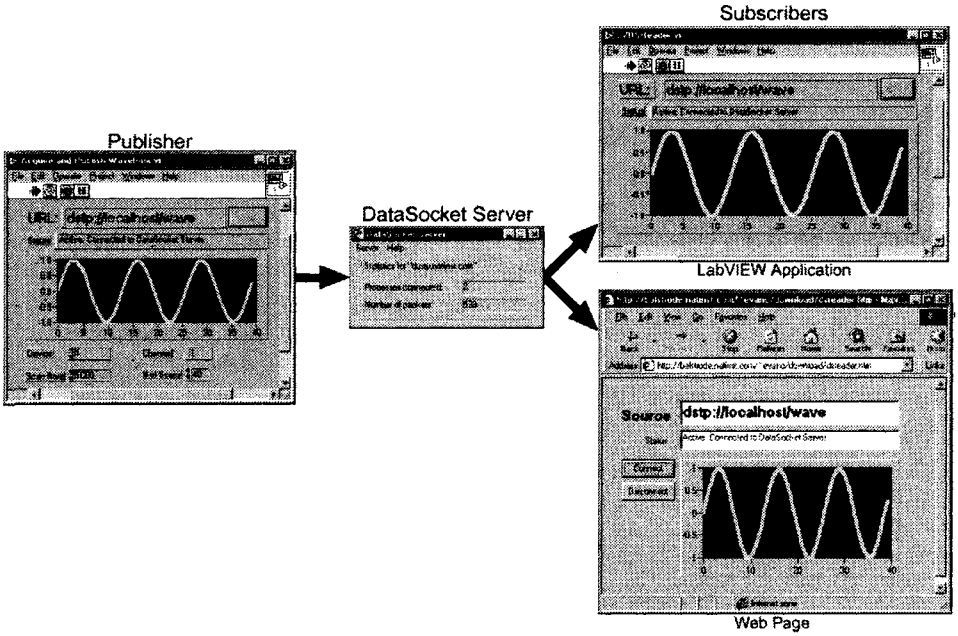


Рис. 14.9

данные (*приемники* (*subscribers*) DataSocket) любого источника. Сервер DataSocket автоматически управляет базовыми сетевыми подключениями и передачей пакетов данных. Ниже мы кратко рассмотрим работу сервера DataSocket.

Сервер DataSocket автоматически устанавливается вместе с LabVIEW для Windows. Поскольку сервер DataSocket снабжен интерфейсом ActiveX, он может существовать только на платформе Windows (к большому сожалению, пользователи MacOS и UNIX не могут задействовать *сервер* DataSocket, однако *клиент* DataSocket запустится на любых платформах).

Для запуска сервера DataSocket просто перейдите в меню **Пуск ⇒ Программы ⇒ National Instruments ⇒ DataSocket ⇒ DataSocket Server**.

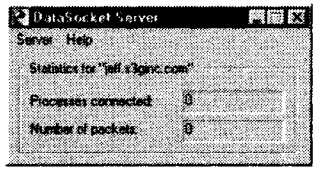


Рис. 14.10

DataSocket не представляет трудностей в работе. Вы можете использовать подключения DataSocket непосредственно с лицевой панели без какого-либо программирования блок-диаграммы.

Каждый элемент управления или отображения способен записывать или считывать данные посредством собственного DataSocket-подключения. Поскольку подключения лицевой панели публикуют лишь данные, но не рисунок элемента управления лицевой панели, то ВП, считывающий данные посредством DataSocket-подключения, может выполнять собственные операции с данными.

Применяйте DataSocket-подключения лицевой панели для публикации (записи) или считывания реальных данных объекта лицевой панели. Когда пользователи получают опубликованные данные и видят их на лицевой панели, они подписываются на эти данные.

Для создания DataSocket-подключения элемента лицевой панели вызовите его контекстное меню и выберите опцию **Действия с данными ⇒ DataSocket-подключения** (Data Operation ⇒ DataSocket Connection). Появится диалоговое окно, изображенное на рис. 14.11. Здесь вы можете указать, будут данные этого элемента опубликованы или данные для этого элемента будут считываться. Необходимо также определить действующий DataSocket-сервер (который может выполняться на вашем или другом компьютере в сети).

Следующее упражнение показывает, насколько легко выполняется обмен данными между виртуальными приборами LabVIEW.

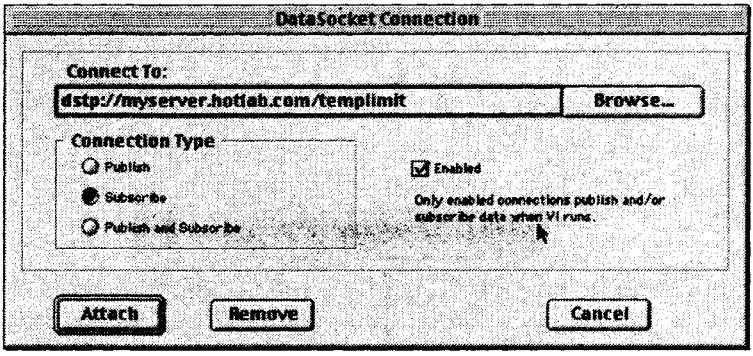


Рис. 14.11

14.4.1. Упражнение 14.2: передача данных лицевой панели

В этом упражнении при обмене данными через сеть с помощью DataSocket вам даже не придется менять блок-диаграмму.

Упражнение лучше выполнять, имея доступ к двум подключенным к сети компьютерам, на которых запущен LabVIEW. Если такой возможности нет, рассматривайте компьютер A и компьютер B как один и тот же компьютер. Вы должны знать IP-адрес или имена ваших компьютеров (в Windows вы получите эту информацию, введя `ipconfig` в командной строке).

Если у вас только один компьютер, используйте локальный IP-адрес – «localhost». Этот специальный IP-адрес всегда указывает на локальный компьютер.

1. На компьютере А должен быть запущен сервер DataSocket. Активируйте сервер из меню **Пуск** ⇒ **Программы** ⇒ **National Instruments** ⇒ **DataSocket** ⇒ **DataSocket Server**.
2. На компьютере А создайте ВП с элементом управления на лицевой панели и ничего более. Щелкните правой кнопкой мыши по элементу управления и выберите опцию **Действия с данными** ⇒ **DataSocket-подключения**. Введите `dstp://<computer-ip-address>/knob` в URL и выберите **Publish**. Затем щелкните мышью по **Attach**.
3. На компьютере В создайте ВП с разверткой осциллограммы на лицевой панели и ничего более. Щелкните правой кнопкой мыши по развертке и выберите опцию **Действия с данными** ⇒ **DataSocket-подключения**. Введите `dstp://<computerA-ip-address>/knob` в URL и выберите **Subscribe**. Затем щелкните мышью по **Attach**.

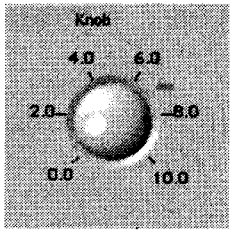


Рис. 14.12 Лицевая панель на компьютере А

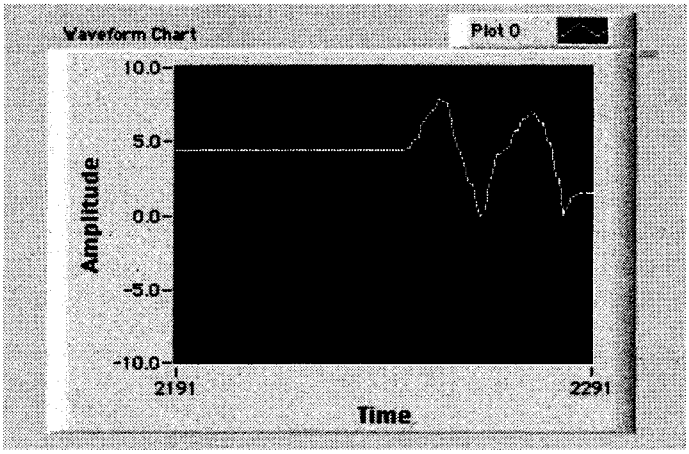


Рис. 14.13. Лицевая панель на компьютере В

4. Теперь на обоих компьютерах А и В нажмите на кнопку непрерывного запуска вашего ВП. Если все работает нормально, то маленький прямоугольный светодиод на элементах лицевых панелей окрасится в зеленый цвет (если он красного цвета, проверьте URL).
5. На компьютере А покрутите элемент управления и посмотрите, как соответственно изменяется график на развертке. Обратите внимание на *отсутствие* дополнительного кода блок-диаграммы.

14.4.2. Виртуальные приборы DataSocket

Мы только что рассмотрели, как работает DataSocket с лицевой панели. Однако иногда нужно ясно задать любые сетевые функции на блок-диаграмме. Для этого вы можете использовать простые виртуальные приборы **DataSocket Чтение** (DataSocket Read) и **DataSocket Запись** (DataSocket Write). Палитра функции **DataSocket** является частью LabVIEW. Вы можете найти ее в меню **Функции** ⇒ **Коммуникация** (Communication) ⇒ **DataSocket** (рис. 14.14).

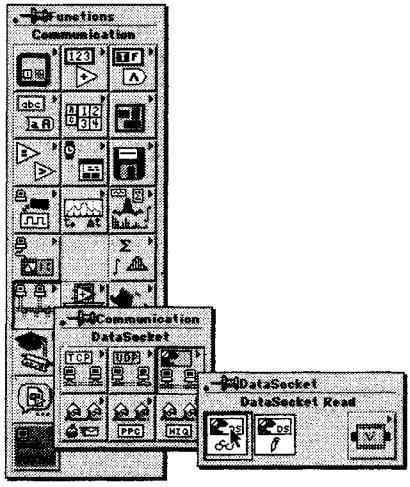


Рис. 14.14

Палитра **DataSocket** содержит подпалитру функций **DataSocket Чтение**, **DataSocket Запись** и **Вариант** (Variant).

URL идентифицирует источник данных для считывания.

Тип (Вариант) – Type (Variant) – определяет тип данных для считывания и тип выходных данных. По умолчанию выбран вариант, который подразумевает любой тип данных.

Ожидание мс (Ms timeout) – определяет время ожидания обновления значения. Это время игнорируется, если значение **ожидание обновленного значения** (wait for updated value) равно ЛОЖЬ, а первоначальное значение уже поступило.

URL идентифицирует источник данных для записи (обычно DSTP-сервер).

Данные (Data) является полиморфным входом данных для записи.

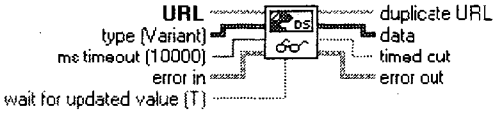


Рис. 14.15. ВП DataSocket Чтение

Рис. 14.16. ВП DataSocket Запись

Подпалитра **Вариант** палитры **DataSocket** содержит функции, которые вы наверняка не будете часто применять, если не попытаетесь наладить связь с другой системой программного обеспечения, которая использует тип данных **Вариант** (например, Visual Basic с Componentworks).

В следующем упражнении некоторые из этих функций применяются для создания источника или приемника данных.

14.4.3. Упражнение 14.3: создание простого источника и приемника данных с помощью DataSocket

Как и в предыдущем примере, наилучшие результаты можно получить, если использовать два различных компьютера. Однако нет ничего страшного в том, что оба ВП будут на одном и том же компьютере.

Используя ВП **DataSocket**, вам нужно будет создать: виртуальный прибор для публикации данных, который бы записал осциллограмму в DataSocket-сервер, и виртуальный прибор приемника, выбирающий, какой тип осциллограммы он хочет считать, а затем отображающий осциллограмму, считанную с DataSocket-сервера.

1. Для создания источника данных постройте ВП, как показано на рис. 14.17–14.19.

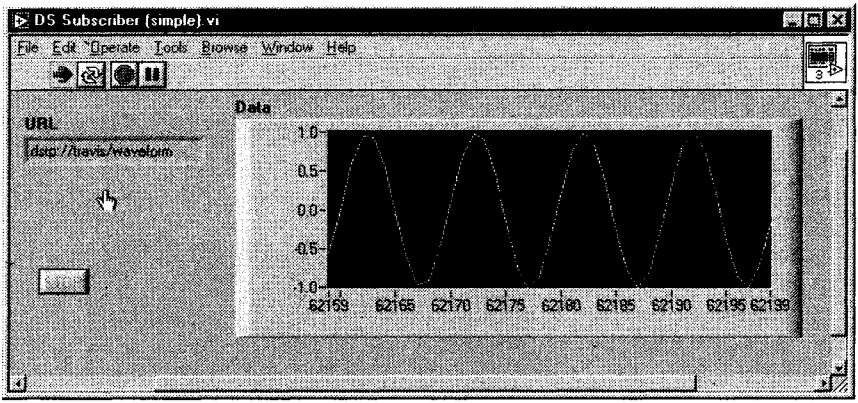


Рис. 14.17

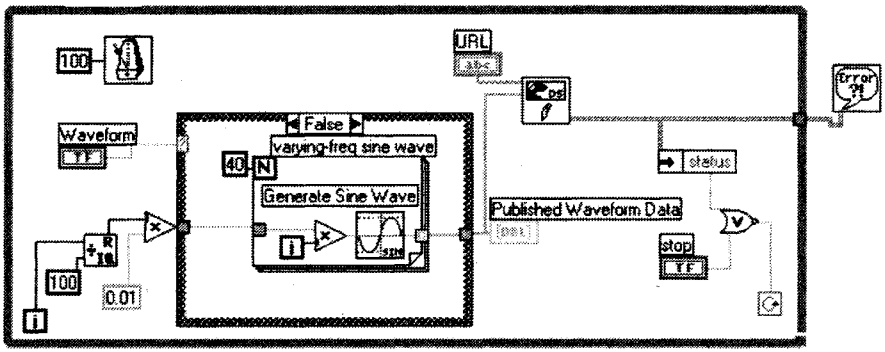


Рис. 14.18

2. Сохраните этот виртуальный прибор как **DS Publisher (simple).vi**.
3. Создайте считывающий виртуальный прибор, как изображено на рис. 14.20 и 14.21. Обратите внимание на необходимость создания константы на блок-диаграмме для информирования функции **DataSocket Чтение** о типе ожидаемых данных.
4. Сохраните этот виртуальный прибор как **DS Sybscriber (simple).vi**.

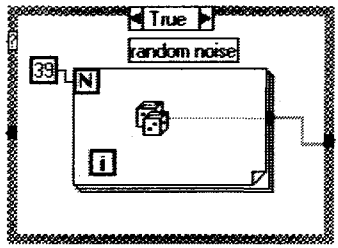


Рис. 14.19

5. На машине, где находится виртуальный прибор **DS Publisher (simple).vi**, запустите DataSocket-сервер из меню **Программы** ⇒ **National Instruments** ⇒ **DataSocket** ⇒ **DataSocket-сервер**.
6. В текстовом окне URL виртуального прибора **DS Publisher (simple).vi** введите `dstp://localhost/waveform`.

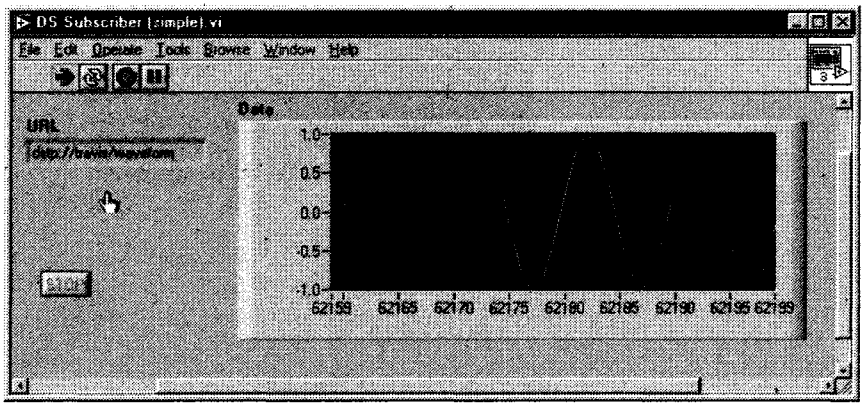


Рис. 14.20

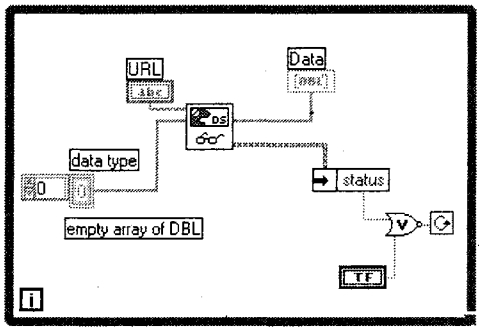


Рис. 14.21

7. В результате на локальном DataSocket-сервере появится элемент **waveform**.
8. Запустите виртуальный прибор **DS Publisher (simple).vi**. Вы увидите, как сервер DataSocket покажет один подключенный процесс передачи в него данных от источника.

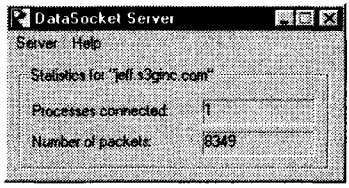


Рис. 14 22

9. Если вы собираетесь запустить ВП **DS Subscriber (simple).vi** на том же самом компьютере, что и ВП издателя, введите строковый элемент управления URL `dstp://localhost/waveform`.
   Если же вы собираетесь запустить ВП **DS Subscriber (simple).vi** на другой, подключенной к сети машине, то укажите в URL компьютер, на котором выполняется ВП источника, например:
   `dstp://remote.server.machine/waveform`.
10. Запустите ВП **DS Subscriber (simple).vi** и посмотрите, как осциллограмма издателя появляется в виртуальном приборе приемника. (Если вы используете две машины, то сервер DataSocket покажет два присоединенных процесса.)
11. Измените осциллограмму в виртуальном приборе источника. Произведите несколько запусков и остановок ВП либо источника, либо приемника. Обратите внимание на устойчивость системы. Ни источник, ни приемник не испытывают какого-либо воздействия друг на друга при выполнении вышеуказанных действий. Даже нет необходимости выключать программу, если другая отсоединилась.

14.5. Возможность взаимодействия с другими программами и приборами

Если вы хотите использовать LabVIEW для взаимодействия с низкоуровневыми протоколами, то вполне можете это сделать – но помните, что работа с низкоуровневыми программами требует много времени и часто затруднительна. Применяйте функции TCP/IP, например, для связи с каким-либо другим компьютером, устройством или программой, которые также поддерживают TCP/IP. В Windows имеется автоматизация ActiveX для взаимодействия LabVIEW с другими программами, использующими ActiveX (например, Microsoft Excel, Interner Explorer).

LabVIEW поддерживает следующие дополнительные протоколы связи (доступ к ним осуществляется через палитру **Коммуникация**):

- TCP/IP;
- UDP;
- автоматизация ActiveX – только для Windows;
- AppleEvents и PPC – только для MacOS.

14.5.1. TCP/IP

TCP/IP (Протокол управления передачей данных/протокол Internet) является основным протоколом для работы в сети Internet и большинстве внутренних сетей.

В данном случае TCP – сокращение от Transmission Control Protocol (Протокол управления передачей данных), а IP – аббревиатура для Internet Protocol (Протокол Internet). Протокол Internet разделяет ваши данные на управляемые пакеты, называемые *датаграммами*, и решает, каким образом передать их из A в B. Проблема заключается в том, что протокол Internet не является «вежливым» и не будет сотрудничать с удаленным компьютером, который может создать проблемы. И как в случае с большим объемом почты в Почтовой службе США, протокол Internet не гарантирует доставку датаграмм. Поэтому к протоколу Internet добавлен протокол управления передачей, который обеспечивает взаимодействие с другими компьютерами и гарантирует доставку датаграмм в установленном порядке (что больше похоже на частного экспресс-курьера).

TCP представляет собой протокол с установлением соединения. Это означает, что перед передачей данных вы должны установить связь, используя определенный протокол. При подключении к сайту следует задать его IP-адрес и порт по этому адресу. IP-адресом является 32-битовое число, которое часто представляется в виде строки из четырех чисел, разделенных точками, например 128.39.0.119. Портом служит число в диапазоне от 0 до 65535. Одновременно вы можете установить несколько соединений. Если вы уже знакомы с UNIX или использовали программы, работающие с Internet, сказанное не будет новостью.

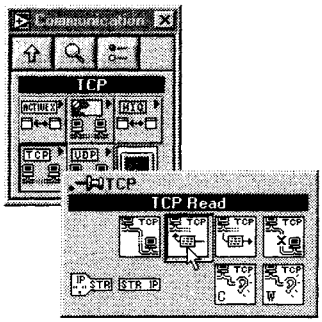


Рис. 14.23. Палитра ***TCP***

LabVIEW имеет набор виртуальных приборов, которые находятся в подпалитре **TCP** палитры **Коммуникация**. Они позволяют выполнять операции, связанные с TCP, такие как открытие связи по заданному IP-адресу, просмотр текущих TCP-подключений, считывание и запись данных и т.д. Они легки в использовании при условии, что ваша сеть сконфигурирована правильно.

Хорошим началом работы по созданию сетевого ВП будет знакомство с примерами, имеющимися в полной версии LabVIEW: **Simple Data Client.vi** и **Simple Data Server.vi**. Их диаграммы показаны на рис. 14.24 и 14.25.

Немного познакомиться с ВП клиент/сервер можно, изучая эти диаграммы. Процесс создания ВП клиента в основном состоит из трех этапов:

1. Запрос TCP-соединения. Вы можете установить время ожидания (timeout), чтобы избежать «зависания» ВП, если сервер не отвечает.
2. Считывание (или запись) данных. Данные всегда передаются в виде строки.
3. Закрытие TCP-соединения.

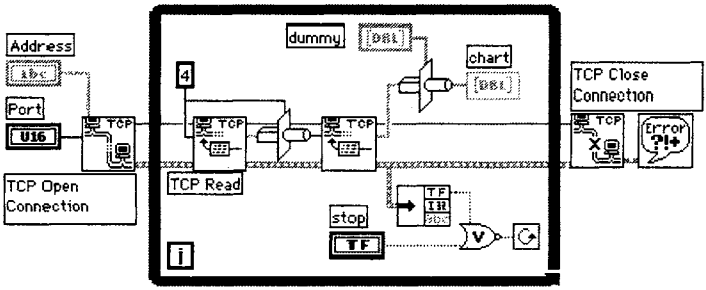


Рис. 14.24 Простейший клиент

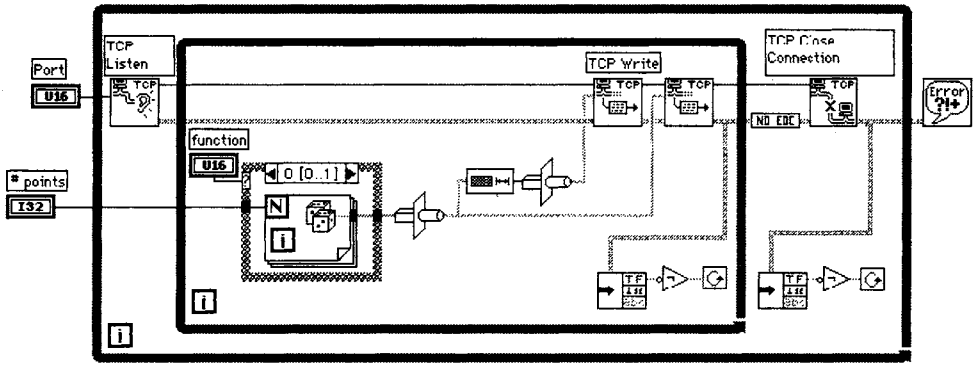


Рис. 14.25. Простейший сервер данных

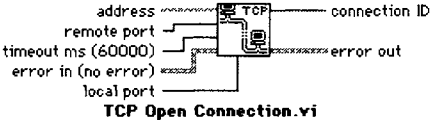


Рис. 14.26. ВП
TCP Установить соединение

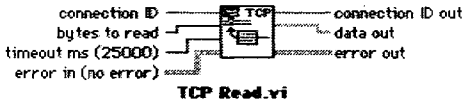


*Рис. 14.27. ВП **TCP Чтение***

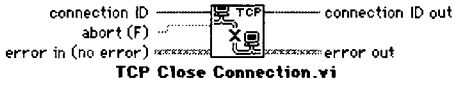


*Рис. 14.28. ВП **TCP Закрыть Соединение***

Процесс создания ВП сервера также состоит из трех этапов:

1. Ожидание соединения.
2. Запись (или считывание) данных. Данные всегда передаются в виде строки.
3. Закрытие TCP-соединения.

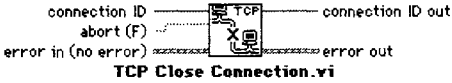


*Рис. 14.29. ВП **TCP Прослушивание***

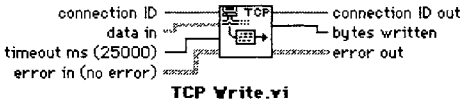


*Рис. 14.30. ВП **TCP Запись***

Поскольку все данные в сети TCP/IP должны передаваться в виде строки, возникает необходимость в преобразовании ваших данных в строковый тип LabVIEW. Самым легким способом для этого является использование функции **Приведение типа** (Type Cast). Удостоверьтесь лишь в том, что и клиент, и сервер знают точно, какой тип данных они передают. Если, например, сервер переводит числа повышенной точности с плавающей запятой в строку, а клиент пытается преобразовать строку в число удвоенной точности, то результат будет неправильным.

Сетевые приложения очень удобны в тех случаях, когда требуется написать программу для управления большой распределенной системой. Более подробно о TCP/IP в LabVIEW вы можете узнать в книге «Internet Applications in LabVIEW».

14.5.2. Протокол UDP

UDP (Universal Datagram Protocol – универсальный протокол передачи датаграмм) похож на TCP и базируется на IP-адресах. Однако он представляет собой «бессоединительный» протокол: сервер может передавать данные большому количеству клиентов, не управляя ими и не беспокоясь о наличии связи между клиентами. Именно поэтому для UDP не существует понятия «слушателя», как в случае TCP.

LabVIEW поддерживает UDP посредством приборов из палитры **UDP** (рис. 14.31).

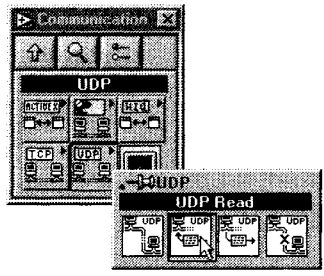


*Рис. 14.31. Палитра **UDP***

14.5.3. ActiveX

Структура ActiveX была разработана Microsoft для обеспечения приложениям Windows возможности взаимодействовать, а также внедрять одно приложение в другое. Например, с помощью ActiveX вы можете внедрить таблицу Microsoft Excel в документ Microsoft Word (эта возможность чаще именуется OLE).

ActiveX представляет собой обширное поле для рассмотрения, поэтому мы ограничимся лишь некоторыми аспектами взаимодействия LabVIEW с этой структурой. Нет нужды говорить, что лишь версия LabVIEW для Windows поддерживает ActiveX, поскольку он является протоколом Windows. Технологии ActiveX, поддерживаемые LabVIEW, сгруппированы в две категории: *Автоматизация* (Automation) и *Контейнер* (Container).

- *Автоматизация ActiveX*. Например, LabVIEW, действующий как клиент, способен запустить приложение Excel, открыть его книгу и т.д. Вы можете использовать скрипт Visual Basic для управления LabVIEW, действующим как *сервер автоматизации* (automation server).
- *Контейнер ActiveX*. Технология позволяет программе внедрять (монтировать) компоненты других программных средств. Например, на лицевую панель ВП LabVIEW можно поместить таблицу Microsoft Excel.

Функция **Автоматизация ActiveX** позволяет LabVIEW быть либо сервером, либо клиентом; функция **Контейнер ActiveX** всегда воспринимает LabVIEW как ActiveX-клиент.

LabVIEW как сервер автоматизации ActiveX

LabVIEW может предоставить доступ к свойствам и методам самой среды LabVIEW или определенного ВП другим приложениям ActiveX (например, Microsoft Excel, Visual Basic и т.д.) посредством интерфейса сервера ВП. Как уже говорилось в главе 13, доступ к функциям сервера ВП осуществляется через TCP/IP (только из других приложений LabVIEW), через функции блок-диаграммы и посредством ActiveX.

Разрешить доступ к серверу ВП через ActiveX можно, поставив соответствующую галочку в меню: **Инструменты ⇒ Опции ⇒ Сервер ВП: Конфигурация** (Tools ⇒ Options ⇒ VI Server: Configuration), как показано на рис. 14.32.

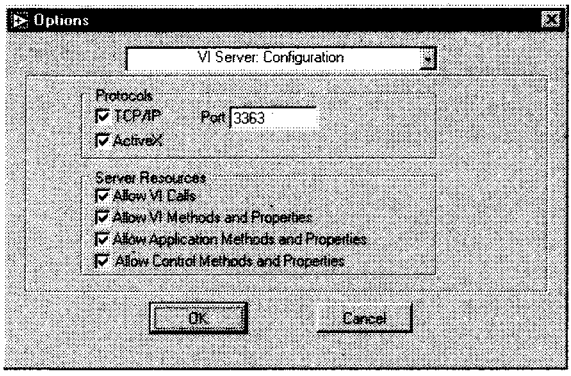


Рис. 14.32

Разрешив работу ActiveX-сервера, вы можете создавать внешние программы на других языках, например Visual Basic или C++, которые будут взаимодействовать с LabVIEW. В частности, вы сумеете написать на Visual Basic программу, которая вызовет определенный виртуальный прибор, откроет его лицевую панель, выровняет ее по центру экрана, запустит ВП, а затем закроет его – осуществляя все эти операции программно через интерфейс ActiveX/сервер ВП.

LabVIEW как клиент автоматизации ActiveX

LabVIEW может выступать и в качестве клиента автоматизации ActiveX: так он получает доступ к свойствам и методам других ActiveX-приложений и к средствам управления ActiveX.

На лицевой панели LabVIEW есть палитра **ActiveX**, которая позволяет создать **Контейнер ActiveX** (Container), **Вариант OLE** (Variant) или **Ссылку автоматизации** (Automation Refnum) – рис. 14.33.

На блок-диаграмме для работы с ActiveX представлены специальные функции.

С помощью опции **Контейнер ActiveX** вы можете внедрить элементы управления ActiveX непосредственно на лицевую панель виртуального прибора.

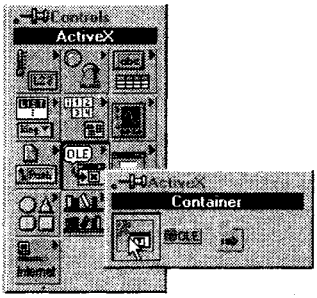


Рис. 14.33. Палитра элементов управления ActiveX

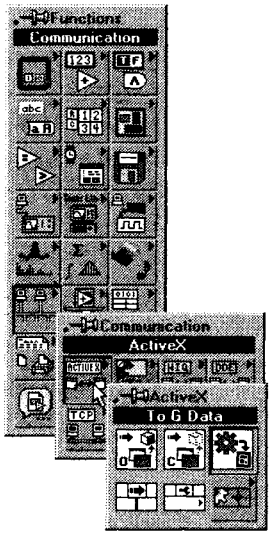


Рис. 14.34. Палитра функций ActiveX

Ссылка автоматизации является особым типом данных LabVIEW, который указывает на внешнее приложение. При помощи этой ссылки легко использовать функции автоматизации для чтения и записи свойств, вызова методов и управления

событиями, относящимися к внешнему приложению. Например, открыть из LabVIEW таблицу Excel на вашем локальном или удаленном компьютере и провести вычисления над данными таблицы.

Вариант OLE представляет собой специальный тип данных, используемый приложениями ActiveX (если вы работали в Visual Basic, то знаете, что такое *вариант*). Варианты OLE служат для передачи данных в элемент управления или приложение ActiveX. Чтобы задействовать данные ActiveX в LabVIEW, выполните преобразование типа данных Вариант-в-LabVIEW при помощи функции **К данным G-типа** (To G data).

Рассмотрим главные функции блок-диаграммы для работы с ActiveX:

- **Автоматизация: Открыть** (Automation Open) открывает ссылку автоматизации, которая указывает на определенный объект ActiveX. Задать класс объекта можно, щелкнув правой кнопкой мыши по функции и отметив опцию **Выбрать класс ActiveX** (Select ActiveX Class). После открытия объекта ActiveX ссылка на него может быть передана другим функциям работы с ActiveX. Если средства DCOM сконфигурированы правильно, вы также вправе открыть указатель на удаленный объект, передав его IP-адрес на вход **имя машины** (machine name);
- **Автоматизация: Закрыть** (Automation Close) закрывает ссылку автоматизации;
- **К данным G-типа** (To G data) преобразует данные типа Вариант ActiveX в данные типа G. Тип данных определяется входом **type**. Эта функция работает так же, как и функция приведения типа данных (type cast);
- **Вызов узла** (Invoke Node) вызывает узел объекта, определенного функцией **ссылка автоматизации**. Появляющиеся в узле методы относятся к объекту ActiveX, не являясь методами LabVIEW;

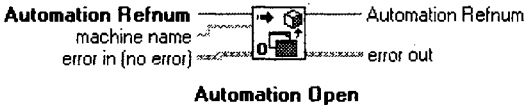


Рис. 14.35. Функция ***Открыть***

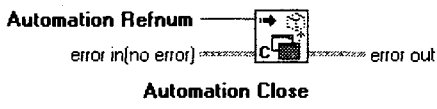


Рис. 14.36. Функция ***Закрыть***

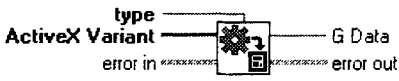


Рис. 14.37. Функция ***К данным G-типа***

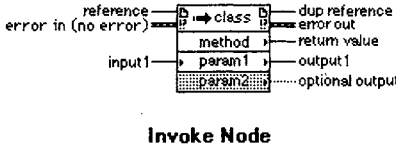


Рис. 14.38. Функция ***Вызов узла***

- **Узел свойств** (Property Node) считывает или устанавливает свойства, относящиеся к объекту ActiveX, указанному функцией **ссылка автоматизации**.

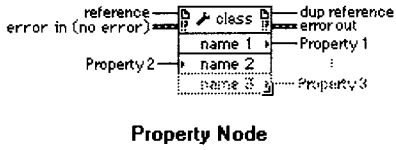


Рис. 14.39. Функция ***Узел свойств***

В LabVIEW также присутствуют функции для обработки событий ActiveX – в палитре **Коммуникации ⇒ ActiveX ⇒ События ActiveX** (Communication ⇒ ActiveX ⇒ ActiveX Event). Если вы хотите узнать об этих функциях больше, обращайтесь к документации LabVIEW.

Возможно, наиболее эффективным средством использования ActiveX совместно с LabVIEW являются контейнеры ActiveX (ActiveX Containers). Контейнеры позволяют внедрять объекты или документы одного приложения в другое. Например, вы можете добавить элемент управления **календарь**, проигрыватель мультимедиа или, как в нашем случае, Internet-браузер в контейнер ActiveX на лицевой панели LabVIEW. Выполните следующее упражнение, чтобы понять, как добавить Internet-браузер на лицевую панель ВП.

14.5.4. Упражнение 14.4: добавление Internet-браузера как ActiveX-компонента в ВП (только в ОС Windows)

1. Создайте новый виртуальный прибор в LabVIEW. Из палитры **Элементы управления ⇒ ActiveX** (Controls ⇒ ActiveX) выберите объект **Контейнер** (Container). Сделайте его размер на лицевой панели максимальным.
2. Вызовите контекстное меню контейнера и выберите опцию **Вставить объект ActiveX** (Insert ActiveX Object). В диалоговом окне **Выбрать объект ActiveX** (Select ActiveX Object) укажите **Создать элемент управления** (Create Control). Появится перечень всех доступных объектов. Пролистайте этот список и выберите компонент **Интернет-браузер Microsoft** (Microsoft Web Browser).

   Переключитесь на блок-диаграмму. К появившемуся терминалу ссылки автоматизации следует подключить функции автоматизации. Выберите функцию **Вызов узла** (Invoke Node) из палитры **Коммуникация ⇒**

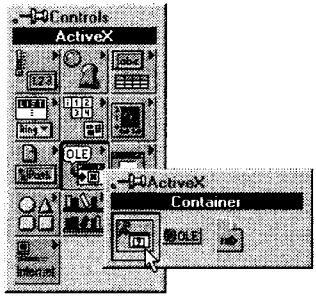


Рис. 14.40

ActiveX. Как только вы соедините ссылку контейнера с функцией **Вызов узла**, в заголовке функции появится имя объекта (IWebBrowser2), доступ к которому она предоставляет. Теперь выберите метод действия из списка, появившегося в меню терминала **Метод** (Method) – в данном случае **Навигация** (Navigate).

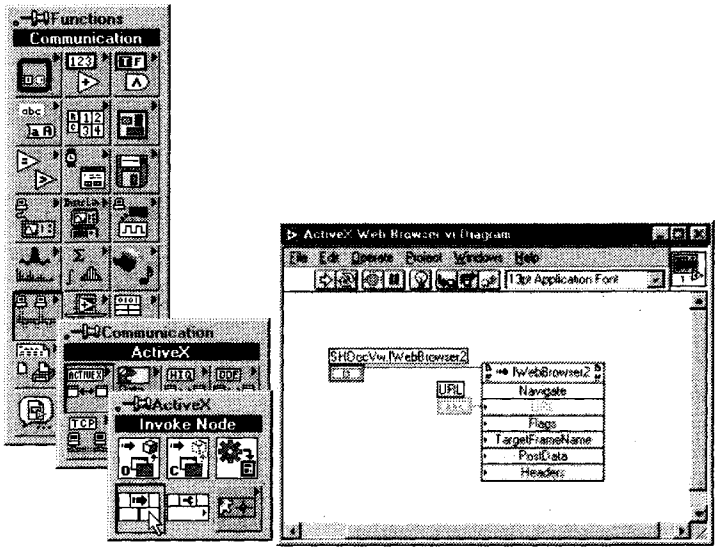


Рис. 14.41

3. Щелкните правой кнопкой мыши по полю **URL** и выберите опцию **Создать элемент управления**. При этом создается поле для текстового ввода Internet-адреса (URL), к которому вы хотите подключиться.
4. Вернитесь к лицевой панели, введите с клавиатуры любой адрес Internet и однократно запустите созданный ВП. Вы увидите соответствующую страницу Internet, появившуюся в контейнере. На рис. 14.42 показан пример загрузки российского образовательного сайта http://www.labview.ru[1].

Если вы не знакомы с технологией ActiveX, этот пример покажется нереальным. Неужели возможно использование Internet-браузера в качестве элемента управления лицевой панели LabVIEW? На самом деле браузер не является частью LabVIEW – реально LabVIEW тут почти ни при чем! Главное, что LabVIEW и Microsoft Internet Explorer могут взаимодействовать через ActiveX, поэтому браузер легко внедрить в виртуальный прибор.

Следует сделать несколько замечаний относительно использования этого ActiveX Internet-браузера:

[1] В оригинальном авторском тексте использовалась ссылка на сайт www.rayodyne.com. – *Прим. перев.*

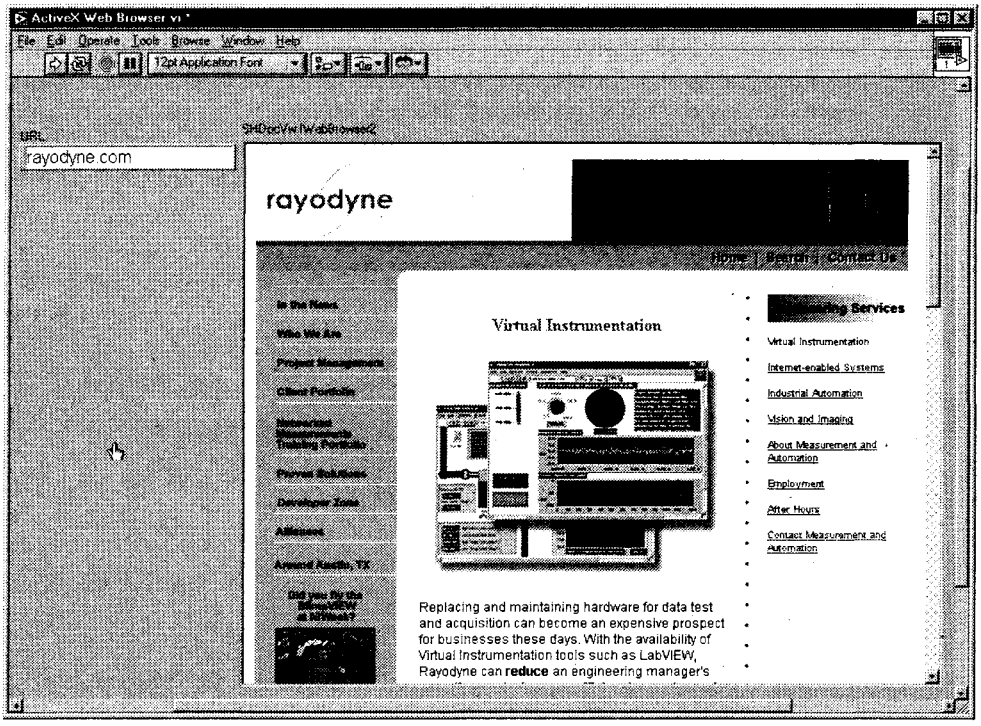


Рис. 14.42

- он очень прост в применении и создается минимальным программированием на блок-диаграмме;
- столь же легко вы можете использовать множество других свойств и методов браузера Microsoft Internet Explorer, таких как изменение размера его окна, перемещение вперед и назад по Internet-страницам, обновление содержимого окна (refreshing) и т.д.;
- вы можете одинаково легко загружать из LabVIEW как страницы Internet (через http:// URL), так и локальные статические документы HTML (через file:// URL);
- все это работает лишь в LabVIEW для ОС Windows; для других платформ описанная методика непригодна.

Несмотря на простоту, приведенный пример может быть исключительно полезен в реальном приложении – например, для создания справочной системы, получающей информацию из Internet.

В заключение отметим, что существует два способа указать класс ActiveX при создании ссылки автоматизации (Automation Refnum) в LabVIEW:

1. Помещение контейнера ActiveX или ссылки автоматизации на лицевую панель, затем вызов контекстного меню и выбор опции **Вставить объект ActiveX** (Insert ActiveX Object).

2. Использование функции **Автоматизация: открыть** (Automation Open) на блок-диаграмме или создание константы **ссылка автоматизации** на лицевой панели, затем вызов контекстного меню и указание опции **Выбрать класс ActiveX** (Select ActiveX Class).

Первый метод удобен, если вы хотите внедрить элемент управления ActiveX на лицевую панель ВП LabVIEW. Второй метод служит больше для взаимодействия между приложениями.

Как вы могли заметить, функции ActiveX очень похожи на функции сервера ВП; поэтому, если вы освоили сервер ВП, сложностей при работе с ActiveX не возникнет. Основным различием является то, что вам понадобится знать методы и свойства внешнего ActiveX-объекта (поскольку LabVIEW не может знать, как он функционирует). Многие компоненты ActiveX включают в себя файл описания «своих» объектов, который вы можете использовать.

14.5.5. AppleEvents и связь программы с программой

В LabVIEW на MacOS Classic (версия 9.1 и более ранние) есть ряд ВП, которые обеспечивают взаимодействие между приложениями через **События Apple** (AppleEvents).

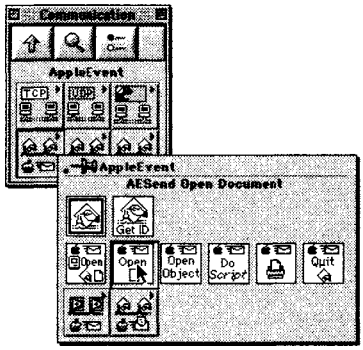


Рис. 14.43

События Apple являются особым протоколом MacOS, который используется приложениями для взаимодействия друг с другом. События Apple отправляют сообщения в другие приложения или в саму операционную систему для открытия документа, запроса данных, печати и т.д. Приложение может отправить сообщение самому себе, другому приложению на этом же компьютере или приложению на удаленном компьютере.

Вы не сможете применять AppleEvents для взаимодействия с компьютерами, на которых не установлена MacOS, например работающими под управлением ОС Windows. Если тем не менее взаимодействие необходимо, пользуйтесь протоколами общего назначения, такими как TCP/IP.

Низкоуровневые сообщения Событий Apple являются довольно сложными: вы сможете успешно использовать их в LabVIEW только при хорошем знании самой системы MacOS и наличии хорошего справочного руководства по ее функциям. Чтобы упростить ситуацию, LabVIEW содержит ряд высокоуровневых ВП для отправки некоторых широко распространенных команд MacOS в приложения, например инструкцию «открыть документ» системной утилите *Finder*. Виртуальные приборы Событий Apple

Рис. 14.44

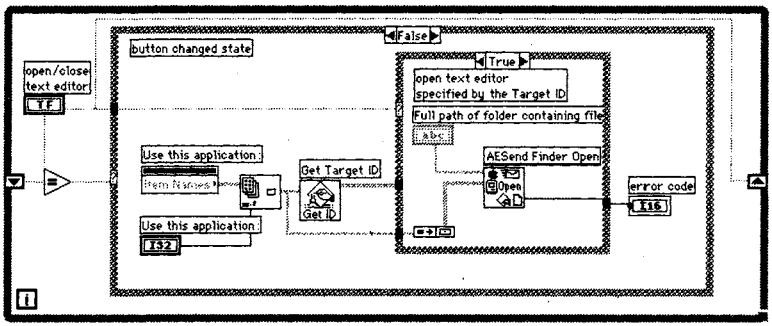


Рис. 14.45

находятся в подпалитре **События Apple** (AppleEvents) палитры **Коммуникации** (Communications).

С помощью AppleEvents допустимо посылать системные команды. Например, вам может пригодиться ВП, который будет открывать и закрывать выбираемое пользователем приложение для просмотра текстового файла данных, который вы только что записали.

Хороший пример взаимодействия LabVIEW с Microsoft Excel под MacOS имеется в полной версии LabVIEW в каталоге `>examples:comm:AEexamples.llb`.

Другой протокол в Macintosh, **Коммуникация: Приложение-Приложение** (КПП PPC – Program-to-Program Communication), является низкоуровневым средством взаимосвязи между приложениями Apple, которое дает возможность передавать данные блоками. LabVIEW предлагает виртуальные приборы для реализации КПП, которые расположены в соответствующей подпалитре **Коммуникации**. КПП еще более сложный и продвинутый протокол по сравнению с AppleEvents, поэтому здесь мы не станем его рассматривать. Если вы знакомы с протоколом КПП, то легко научитесь использовать его ВП в LabVIEW, особенно изучив приведенные примеры и ознакомившись с руководством.

14.6. Промышленные телекоммуникации – полная картина

При разработке приложений, требующих хранения и обработки большого объема данных, или при потребности в их быстрой сортировке и поиске становится очевидной необходимость привлечения к программированию экспертов, которые знают принципы организации различных уровней промышленной информационной сети масштаба предприятия (enterprise-level network). Эта работа часто предусматривает использование баз данных (БД) SQL, таких как Oracle или Sybase, связывая их вместе с измерительно-управляющими алгоритмами LabVIEW в общую

сеть и осуществляя доступ к ней удаленных клиентов. На рис. 14.46 показана подобная архитектура.

Предположим, например, что у вас есть несколько испытательных лабораторий, использующих LabVIEW для функционального тестирования определенных изделий. Подключение к ресурсам Internet позволит отслеживать текущее состояние испытаний в любой лаборатории с помощью обычного Internet-браузера. Далее, объединяя лабораторные компьютеры в сеть с базой данных, вы можете получить не только полную информацию об испытаниях через тот же самый браузер, но и исследовать различные виды интересующих вас взаимосвязей – например, сравнить результаты «тест прошел / не прошел» в разных лабораториях либо запросить сравнение историй текущих испытаний в реальном времени.

Таким образом, особо важным элементом промышленной информационной системы является накопительная база данных. Во многих случаях необходимо хранить собранные экспериментальные данные в БД, а не на локальном компьютере. Базы данных обеспечивают значительно более эффективное хранение и быстрый доступ к большому объему информации, особенно при наличии нескольких конкурирующих процессов записи/чтения данных. Кроме того, БД позволяют программистам генерировать все типы запросов для определения взаимосвязей между элементами.

Хотя существует большое разнообразие баз данных – от несложной Microsoft Access до высокоуровневых Oracle-систем – большинство из них поддерживает работу со стандартным алгоритмическим языком *SQL* (Structured Query Language – язык структурированных запросов). Этот язык используется в основном для создания запросов, которые посылаются в БД, чтобы извлечь требуемые данные. Функции SQL могут применяться в различных БД благодаря интерфейсу *ODBC* (Open DataBase Connectivity – открытый интерфейс доступа к базам данных) – «склеивающему» уровню, который позволяет абстрагировать SQL-запросы от специфики конкретного приложения и тем самым дает им возможность работать с разнообразными драйверами БД.

Базы данных часто поддерживают удаленный доступ через Internet. Создав соответствующие скрипты Internet-сервера (Web-server scripts) используя CGI или ASP, можно обеспечить удаленному пользователю возможность отправлять запросы в БД и извлекать данные или диаграммы, которые будут отображены непосредственно в Internet-браузере. Если приложение на LabVIEW предназначено для работы с большим объемом данных и вам хочется организовать доступ из Internet не только к процессам реального времени, но также и к истории процесса, целесообразно подключить базу данных к LabVIEW-приложению, выступающему как источник данных (publisher of data) и к Internet-серверу – приемнику данных (subscriber of data). Вы можете даже написать скрипты для Internet в виде виртуальных приборов LabVIEW!

National Instruments предлагает Библиотеку интерфейса доступа к базам данных (DataBase Connectivity Toolkit) для LabVIEW, которая поддерживает любую

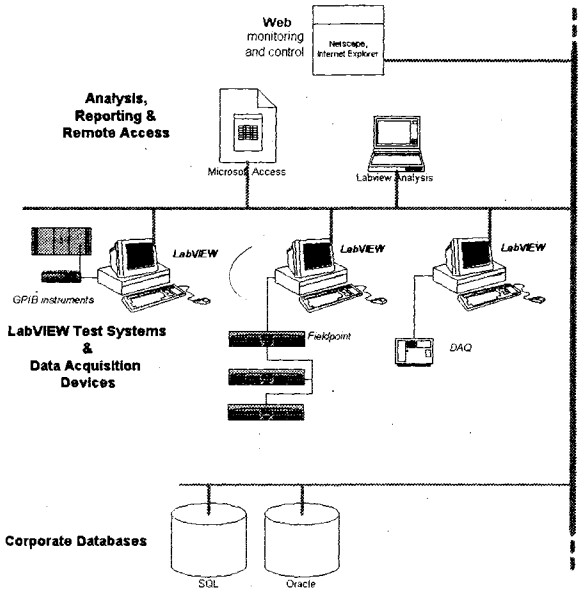


Рис. 14.46. Схема взаимодействия на предприятии

ODBC-совместимую БД, что дает возможность LabVIEW считывать и записывать данные в практически любую SQL-базу данных. Также допустимо использовать бесплатную библиотеку с открытым кодом LabSQL для подключения к практически любой БД. LabSQL находится на сопровождающем книгу компакт-диске, ее также можно загрузить с Internet-сайта «LabVIEW – ресурсы с открытым кодом» (LabVIEW Open Source Tools – LOST) по адресу http://jeffreytravis.com/lost.

14.7. Итоги

В этой главе вы познакомились с ролью Internet и сетей, а также с различными коммуникационными ресурсами LabVIEW: виртуальными приборами работы с Internet, DataSocket, TCP/IP, UDP, ActiveX, AppleEvents, подключением к базам данных.

Встроенный в LabVIEW Internet-сервер легко позволяет управлять виртуальными приборами через Internet. Запустите Internet-сервер LabVIEW, напечатайте адрес URL вашего компьютера в Internet-браузере – и вы тотчас получите доступ к лицевой панели виртуального прибора! Для дистанционного управления ВП вы можете также использовать бесплатные утилиты типа LabVNC или создать собственный интерфейс при помощи технологий Java, ActiveX или CGI.

DataSocket представляет собой протокол National Instruments для обмена данными между приложениями. При помощи DataSocket легко подключить любой элемент управления или индикатор на лицевой панели к серверу DataSocket так, чтобы другие ВП или приложения National Instruments могли считывать показания или записывать данные в этот элемент. Использовать функции DataSocket для обмена данными разрешается также при помощи соответствующих ВП на блок-диаграмме.

Низкоуровневые коммуникационные протоколы, в частности TCP/IP, UDP и ActiveX в ОС Windows и AppleEvents в MacOS, также поддерживаются LabVIEW. Если вы знакомы с этими протоколами, то можете применять их для создания виртуальных приборов, обеспечивающих обмен данными в сети или между приложениями.

И наконец, вы оценили важность баз данных для промышленных измерительно-управляющих приложений. На практике убедившись в удобстве работы с базами для накопления и поиска данных, полученных в LabVIEW, вы наверняка часто будете обращаться к специальным средствам LabVIEW для работы с БД, таким как DataBase Connectivity Toolset или LabSQL.

Обзор

В этой главе вы узнаете о некоторых дополнительных возможностях LabVIEW по работе с файлами. Зная, как использовать различные типы файлов (текстовые, файлы протокола, двоичные, осциллограммы), вы сможете выбрать наилучший тип для своего приложения. Когда настанет время сделать печатный отчет, мы расскажем об опциях печати LabVIEW и функциях генерации отчетов. В этой главе вы научитесь также программно управлять процессом печати из ВП.

ЗАДАЧИ

- Познакомиться с различными типами файлов, применяемых в LabVIEW
- Использовать низкоуровневые функции ввода/вывода файлов протокола, текстовых и двоичных файлов
- Понять, как можно сохранять и считывать данные осциллограмм из файлов различных типов
- Изучить различные способы печати из LabVIEW
- Познакомиться с функциями генерации отчетов для создания распечатанных отчетов или отчетов в формате HTML из LabVIEW
- Узнать, как программно управлять процессом печати

ОСНОВНЫЕ ТЕРМИНЫ

- Двоичный файл
- Файл протокола
- Файл ASCII
- Осциллограммы и файлы
- Ссылка файла
- Прроцесс печати
- HTML
- Отчеты

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ВВОДА/ ВЫВОДА ФАЙЛОВ, ПЕЧАТИ И СОЗДАНИЯ ОТЧЕТОВ 15

15.1. Дополнительные возможности ввода/вывода файлов

В главе 9 вы узнали, как, используя ВП из палитры **Ввод/вывод файла**, сохранить данные в текстовом файле независимо от формата – простого текстового или табличного. Файлы, сохраненные в ASCII или текстовом формате наиболее удобны: практически любой компьютер, работающий под управлением любой ОС, может считывать или записывать текстовый файл. Однако текстовые файлы имеют ряд недостатков: они наименее эффективны в плане использования ресурсов (много байтов на единицу информации) и требуют много времени на преобразования и обработку, если данные, которые вы сохраняете, не являются текстом (например, график). В LabVIEW есть возможность хранения и считывания двух других типов файлов: *файлов протокола* (datalog files) и *двоичных файлов* (binary files).

Файлы протокола являются особым видом двоичного файла, используемого для хранения информации лицевой панели или любых данных LabVIEW. При сохранении всей информации лицевой панели файлы протокола можно рассматривать как «снимок» панели вашего ВП. При создании файла протокола в нем записываются все значения всех элементов управления и отображения на момент сохранения файла. Позднее вы можете загрузить этот файл в ВП, чтобы увидеть сохраненные значения элементов лицевой панели. Допустимо сохранить несколько «наборов» значений с той же самой лицевой панели в одном файле протокола. Данный тип файлов может создаваться и считываться только в LabVIEW. Они весьма легки в использовании: манипулировать файлами протокола из меню LabVIEW можно без написания какого-либо кода. Также разрешается создавать файлы протокола, которые записывают специфические типы данных LabVIEW, такие как кластер или строка.

Двоичные файлы обычно содержат побайтовое изображение данных в том виде, в каком они сохранены в памяти. Вы не можете просто читать двоичный файл с помощью текстового редактора или какой-либо другой программы. Для этого нужно точно знать, как отформатирован файл, – аналогично тому, как нужно знать тип данных при использовании двоичных строк (см. главу 12). Преимуществами двоичного файла являются малая ресурсоемкость, поскольку не требуется никаких преобразований и занято небольшое дисковое пространство по сравнению с файлами ASCII. Например, сохранение массива из 100 чисел в 8-битовом двоичном файле занимает около 100 байт, тогда как текстовый файл потребует около 400 байт. Это происходит потому, что каждое 8-битовое целое число занимает только 1 байт (в двоичном формате), но то же самое число в текстовом формате может занять от 3 до 4 байт (1 байт для каждой цифры ASCII).

Таблица 15.1. Краткий обзор типов файлов

Файл ASCII	Файл протокола	Двоичный файл
• очень легок в использовании, • совместим с другими программами, легко читаем и управляем; • требует большого количества преобразований и много места на диске; • удобен для небольшого и среднего количества данных, которые могут применяться в других программах (например, таблицах символов)	• удобен в интерактивном режиме, требует более сложного программирования для использования в приложениях; • может работать только в LabVIEW; • очень удобен для хранения объекта LabVIEW или всей лицевой панели	• требует определенных протоколов программирования, • наиболее эффективен в загрузке диска и времени процессора; • характеризуется быстрой записью на диск/считыванием с диска; • может считываться другими программами при соблюдении осторожности; • удобен в случаях, когда требуется запись больших файлов в реальном времени

15.1.1. Задание путей размещения файла

Для определения местонахождения файла в файловой системе LabVIEW использует специальный тип данных, называемый *путем* (path). Элементы управления или отображения пути, доступ к которым осуществляется из палитры **Строки и пути** (String&Path), выглядят и действуют как строковые элементы управления и отображения. Вы можете задать абсолютный или относительный путь к файлу.

Маленькая иконка в виде папки справа от элемента управления путем к файлу (рис. 15.1) представляет собой кнопку просмотра (Browse Button). Щелчок мышью по этой кнопке приводит в действие системную программу просмотра файлов (browser), которая позволяет легко определить местоположение файла или директории, вместо того чтобы вводить его с клавиатуры.

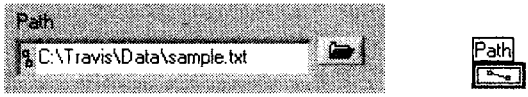


Рис. 15.1

Отличие от строкового типа заключается в том, что в элементе управления путем разрешается ввести лишь путь или имя файла в соответствии с правилами операционной системы. Полное имя маршрута для файла `sample.txt` может выглядеть следующим образом:

- Windows: `C:\TRAVIS\DATA\SAMPLE.TXT`
- MacOS: `PowerHD:Travis:Data:Sample Text file`
- UNIX: `usr/travis/data/sample_text_file`

При открытии или создании файла для него необходимо задать путь. Если вы не подключите путь к соответствующей функции, работающей с файлами, то при обращении к функции **Открыть файл** (Open File) или **Создать новый файл** (New File) LabVIEW вызовет диалоговое окно, подсказывающее, как его найти. Можно заставить LabVIEW сообщить пользователю имя файла с помощью собственной подсказки, используя функцию **Файловый диалог** (File Dialogue). Мы рассмотрим эти функции через некоторое время.

15.1.2. Трехступенчатый процесс

Когда вы используете операции ввода/вывода файла в ВП, придерживайтесь процедуры, состоящей их трех ступеней: *открытие*, *считывание* или *запись* и *закрытие*. Здесь уместна аналогия с папкой для бумаг. Извлекаете вы из папки или помещаете в нее какие-то материалы, вы должны в первую очередь открыть папку. Затем вы вправе делать с бумагами все, что угодно. Наконец, нетрудно оставить папку открытой – но зачем? Кто-нибудь еще может воспользоваться бумагами или перепутать их. Поэтому закрытие файла в компьютере обеспечивает целостность ваших данных.

Диаграмма на рис. 15.2 иллюстрирует этот процесс. Доступ к функциям **Открыть/создать/заменить файл** (Open/Create/Replace File), **Записать файл** (Write File) и **Закрыть файл** (Close File) осуществляется с палитры **Ввод/вывод файла** (File I/O). Обратите внимание на зависимость данных от каждой функции при переходе к следующей. После завершения операций **Открыть/... файл** LabVIEW генерирует *refnum* – ссылку файла (file reference number), которая передается в функцию **Записать файл**. Как только строка заносится в файл, функция **Записать файл** в свою очередь передает ссылку в функцию **Закрыть файл**. Наконец, ВП **Обработчик ошибок** (Error Handler) проинформирует вас, если что-то не в порядке.

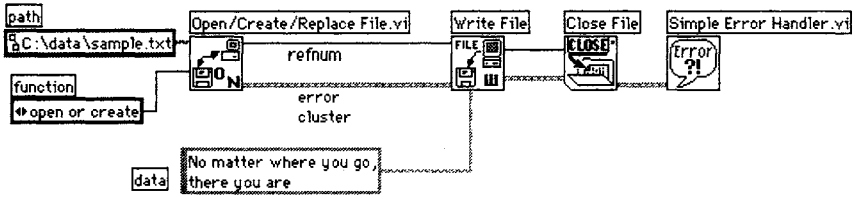


Рис. 15.2

LabVIEW использует ссылку файла для отслеживания файла, на который ссылается ВП. Вас, собственно, не должно интересовать, что собой представляют ссылки файлов и как они работают, – достаточно соединить соответствующие входы функций ввода/вывода файла. При соединении входов ссылки ВП усиливается зависимость данных, что автоматически определяет выполнение операций в установленном порядке.

Существует несколько функций ввода/вывода файла, которыми вам придется пользоваться наиболее часто:

- **Файловый диалог** (File Dialog) из палитры **Ввод/вывод файлов ⇒ Дополнительные функции** (Advanced Functions) отображает диалоговое окно для выбора нужного файла. Это диалоговое окно используется для указания новых или уже существующих файлов и директорий. Вы можете задать подсказку (prompt), которая появится в диалоговом окне. Использование ввода типа протокола (datalog type) мы обсудим позже;
- **Открыть файл** (Open File) из той же палитры открывает существующий файл. Если хотите, можете подключить реальный путь к входу **путь файла** (file path). Эта функция не может создавать или заменять файлы. Она открывает только *существующие* файлы. Терминал **тип протокола** (datalog type) используется только при открытии файлов протокола LabVIEW;

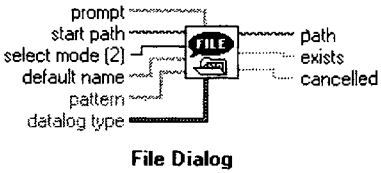


*Рис. 15.3. Функция **Файловый диалог***

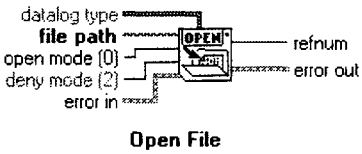


*Рис. 15.4. Функция **Открыть файл***

- **Создать файл** (New File) создает и открывает новый файл для считывания или записи. Вы должны подключить путь к вводу **путь файла** (file path) данной функции, и это должен быть путь к *несуществующему* файлу. Терминал **тип протокола** (datalog type) используется только при создании файлов протокола LabVIEW;

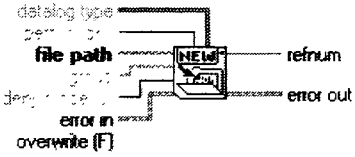


*Рис. 15.5. Функция **Создать файл***

- **Записать файл** (Write File) записывает данные в открытый файл. Действие этой функции слегка изменяется в зависимости от того, записываете вы данные в двоичный файл или в файл протокола LabVIEW. Ввод заголовка (header) используется в двоичных файлах и игнорируется при работе с файлами ASCII;
- **Считать файл** (Read File) считывает данные из открытого файла. При считывании двоичных файлов вы можете использовать ввод двоичного типа (byte stream type) данных для того, чтобы показать, как LabVIEW должен интерпретировать данные. Двоичные файлы детально рассматриваются ниже;

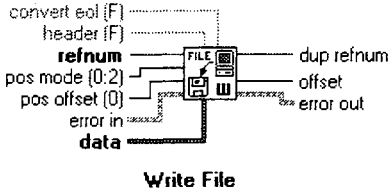


*Рис. 15.6 Функция **Записать файл***

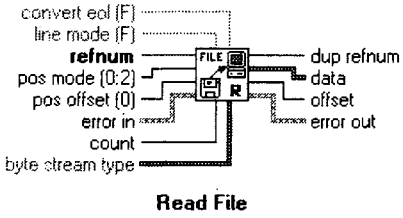


*Рис. 15.7. Функция **Считать файл***

- **Закрыть файл** (Close File) закрывает файл, ассоциируемый со ссылкой файла (refnum);
- **Открыть/создать/заменить файл** (Open/Create/Replace File) – полезная функция, позволяющая программно открыть файл, создать новый файл или заменить старый с тем же именем. Вы можете дополнительно задать такие опции, как диалоговая строка подсказки, начальный путь и т.д. Этот ВП вызывает несколько упомянутых выше функций ввода/вывода файла.

*Рис. 15.8. Функция **Закрыть файл***

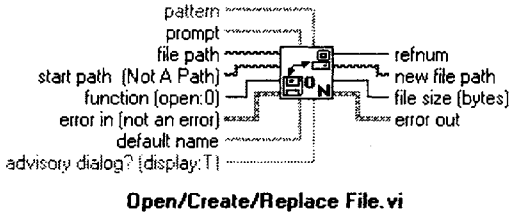


*Рис. 15.9. Функция **Открыть/создать/заменить файл***

15.1.3. Запись и считывание текстовых файлов

Если вы собираетесь записать текстовые файлы для сохранения данных и вам не требуется высокоскоростная потоковая запись, используйте одну из высокоуровневых функций ввода/вывода файла (рис. 15.10). Эти виртуальные приборы открывают, закрывают файлы и проверяют ошибки.

Однако иногда возникает необходимость написания собственной программы по вводу/выводу файла. И без использования функции **Записать файл** вам никак не обойтись (рис. 15.11).

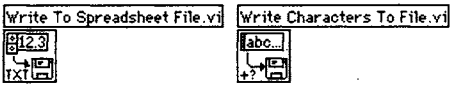


Рис. 15.10

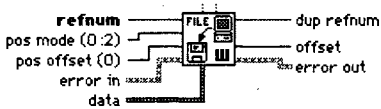


*Рис. 15.11. Функция **Записать файл**. Записывает данные в файл, заданный ссылкой файла*

Режим записи (pos mode):

0: с начала файла
1: с конца файла
2: с текущего положения указателя файла

Если ввод **Режим считывания** подключен, то по умолчанию выставляется 0, в противном случае выставляется 2.

Ввод данных (data) функции **Записать файл** является полиморфным, то есть вы можете подавать числовые, строковые или кластерные типы данных. Тип данных на входе этой функции определяет формат записываемого файла. Очевидно, для текстовых файлов следует подавать строковые данные или никаких данных вообще.

Основные положения

Управление текстовыми файлами, как вы убедились, является несложным процессом, но возникает два вопроса:

1. Каким образом вы получите путь и имя файла? Создаст ли LabVIEW их автоматически? Будет ли пользователь вводить их с лицевой панели или в диалоговом окне?
2. Каким образом программа станет обрабатывать ошибку при работе с файлом (переполнение диска, нерабочая ссылка и т.д.)? Нужно ли остановить выполнение программы? Или ее не следует останавливать, а лишь сообщить пользователю о наличии ошибки?

Естественно, нужно знать, как преобразовывать данные в строку. К счастью, LabVIEW предлагает все виды строковых преобразований и функций управления (пролистайте палитру **Строки** (String) или посмотрите главу 9).

Предположим, что вы создали простой виртуальный прибор **Acquire Data**, который считывает одну точку данных и возвращает ее в виде числа с плавающей запятой. Вам необходимо периодически считывать значение и передавать его в файл до тех пор, пока не будет нажата кнопка **Стоп**. Пользователь должен иметь возможность выбрать имя файла во время выполнения программы.

Начните с построения блок-диаграммы, похожей на изображенную на рис. 15.12.

Пользователь получает подсказку для выбора имени файла, создается новый файл, и затем начинает выполняться цикл по условию. Каждый раз, когда цикл завершается, значение преобразуется в строковый формат с помощью функции **Преобразовать в строку** и записывается в файл. При нажатии кнопки **Стоп** цикл заканчивает работу и файл закрывается. Просто, не правда ли? Этот виртуальный прибор будет работать так, как ему и положено. Но, откровенно говоря, он не совсем подходит для управления файлами. Его можно немного усовершенствовать. Для начала следует отметить, что в ВП строки, которые представляют точку данных, не отделяются друг от друга. Файл превращается в одну длинную беспорядочную строку. А что произойдет, если пользователь нажмет кнопку **Отмена** на блок-диаграмме?

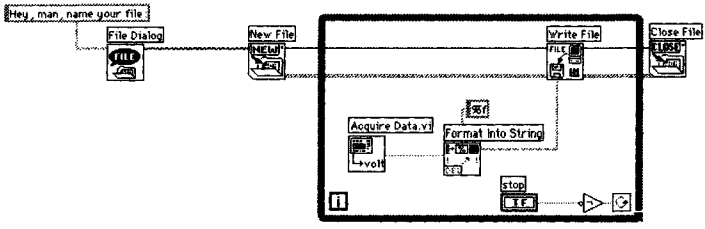


Рис. 15.12

Следующая блок-диаграмма реализует более управляемую программу. Обратите внимание, что каждое значение отделяется возвратом каретки и переходом на новую строку (символы `\r\n`). Теперь каждое число будет находиться на новой строке в тексте. Мы приняли во внимание возможность нажатия кнопки **Cancel** путем введения структуры варианта, основанного на выходном значении функции **Файловый диалог**. Мы также использовали кластеры ошибок для выявления ошибки при вводе/выводе файлов. Если появится хоть одна ошибка, цикл остановится.

Однако существует еще одна проблема этого кода, которую непросто заметить. Она относится к функции **Записать файл** внутри цикла по условию. Пока в ВПП **Acquire Data** работает функция **Задержка**, программа будет записывать данные в файл столько раз, сколько раз она будет подавать команду микропроцессору – может быть несколько тысяч раз в секунду. Необходимо ввести функцию **Задержка** (Wait) внутрь цикла для ограничения количества записей в файл, скажем до одного раза в секунду.

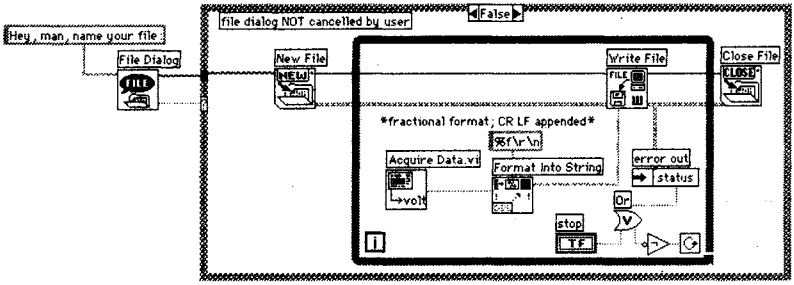


Рис. 15.13

При программировании процедуры ввода/вывода в файл не забудьте учесть наихудший сценарий развития событий, иначе у вас могут возникнуть сложности во время выполнения ВП. Попробуйте запрограммировать выход из ситуации, связанной с ошибкой ввода/вывода файла или каким-либо аномальным событием.

Увы, многое может пойти не так, как хотелось бы, когда мы имеем дело с вводом/выводом файлов. Не падайте духом! Большую часть времени вы будете пользоваться высокоуровневыми виртуальными приборами LabVIEW для считывания и записи текстовых файлов. Когда потребуются специализированные операции ввода/вывода файла, будьте внимательны и всегда отдавайте себе отчет в том, что вы понимаете, что делаете.

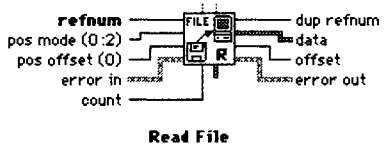


Рис. 15.14. Функция ***Считать файл***

Считывание текстового файла является аналогом операции записи файла. Вы можете использовать функцию **Считать файл** (Read File) в качестве дополнительной к функции **Записать файл** (Write File) – рис. 15.14.

Если нужно всего лишь записать новые данные в конец файла, то нет необходимости что-либо подключать к терминалам **режим считывания** (pos mode) и **относительное смещение** (pos offset). Вы также можете применить некоторые более простые файловые функции, упомянутые в главе 9.

Объяснение некоторых опций ВП ввода/вывода файла

Опции **режим считывания/записи** (pos mode) и **относительное смещение** (pos offset) связаны с файловой системой компьютера и могут оказаться сложными для понимания. Для отслеживания ввода/вывода файлов операционной системой используется невидимая переменная, называемая *маркер файла* (file mark). Маркер файла обычно указывает на текущее местоположение внутри файла, где были в последний раз сохранены данные. Местоположение измеряется в байтах относительно некоторой точки. Всякий раз, когда вы сохраняете данные в существующий файл, LabVIEW записывает их, начиная с **относительного смещения** (в байтах), которое определяется **режимом записи**.

Такой относительной точкой может быть начало файла, конец файла или текущее местоположение последней записи. С помощью функции **Записать файл** легко манипулировать точкой введения новых данных и перемещать ее, используя **режим записи** и **относительное смещение**. Названная особенность делает возможным произвольный доступ к данным в файле. Если вы оставите неподсоединенными эти два ввода, то по умолчанию новые данные будут записываться последовательно друг за другом – как раз тот случай, который наиболее часто употребляется.

Для лучшего понимания представленных концепций посмотрите на рис. 15.15. Маркер файла указывает на место, где была сделана последняя запись данных. Точки (a), (b) и (c) отображают три возможных места в файле, куда разрешается записать данные. Ниже представлены способы, позволяющие получить доступ к каждому из этих мест:

(a) подключение к терминалам **режим записи** числа 0 (относительно начала файла) и **относительное смещение** – положительного числа;

(b) неподключение терминалов **режим считывания** и **относительное смещение**;

(c) подключение к терминалам **режим записи** числа 2 (которое определяет запись относительно текущего положения маркера) и **относительное смещение** – положительного числа.

Естественно, существует много других возможностей, кроме перечисленных выше. Они нужны только для получения быстрого доступа к произвольному месту

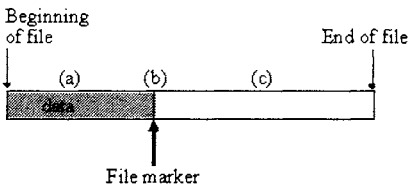


Рис. 15.15

внутри файла. Если вы оставите упомянутые терминалы неподключенными, это даст вам возможность считывать данные с начала файла и записывать в место, следующее за маркером файла, что является обычным способом обработки файлов.

15.1.4. Упражнение 15.1: считывание и запись текстовых файлов

Создайте ВП, который мог бы считывать и записывать файлы ASCII. Входные данные должны быть представлены двумерным массивом чисел. Определение режима работы ВП (будет он читать файл или записывать новый) осуществляется логическим переключателем. Лицевая панель данного прибора изображена на рис. 15.16.

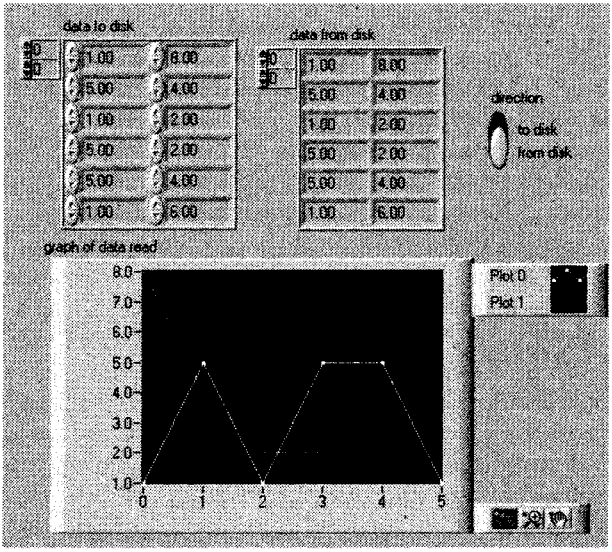


Рис. 15.16

Сохраните этот виртуальный прибор как **ASCII Read/Write.vi**. Одно простое решение находится на компакт-диске под этим же именем.

1. Для считывания файла используйте простую функцию ввода/вывода файлов, например **Считывание символов из файла** (Read Characters from File).
2. Применяйте функцию **Подмножество массива** (Array Subset) для преобразования содержимого массива в величины, которые можно преобразовать в строки.

15.1.5. Запись и считывание файлов протокола

Файлы протокола хранят информацию объектов LabVIEW (кластер, строка, логический массив и т.д.) в специальном двоичном формате. Всякий раз, когда вы

записываете в файл протокола, LabVIEW обычно прилагает *регистрацию* (record) данных. Регистрация представляет собой что-то вроде файла внутри файла: операционная система «видит» только один файл, но из-под LabVIEW вы заметите несколько отдельных регистраций в файле протокола. Важной особенностью этой структуры является возможность произвольного доступа к любой регистрации в файле. Кроме того вам не обязательно знать, через сколько байтов необходимо «перепрыгнуть», чтобы добраться до определенного набора данных, – достаточно помнить номер регистрации. Файлы протокола особенно полезны для сохранения смешанных типов данных, таких как логические данные и массивы. Существует два способа создания файла протокола:

- путем использования встроенных возможностей протоколирования лицевой панели. Соответствующие опции находятся в меню **Управление** (Operate). Эти команды позволяют записать всю лицевую панель в файл без помощи какого-либо ВПП ввода/вывода файла. Данные заносятся либо по завершении работы ВП, либо по команде пользователя из меню **Управление**;
- путем чтения и записи файлов протокола посредством функций **Считать файл** и **Записать файл**. В этом случае вы можете придать сохраненным данным тот вид, какой хотите (вместо целой лицевой панели), и сохранять данные тогда, когда хотите.

Особенности протоколирования лицевой панели

Протоколирование лицевой панели достаточно простой процесс, который не требует никого программирования блок-диаграммы. При ее активизации LabVIEW сохраняет данные, находящиеся во всех элементах управления лицевой панели, и метку даты/времени в файл протокола. Можно иметь несколько отдельных файлов, каждый из которых будет заполнен данными различных тестов. Позднее эти данные восстанавливаются посредством того же или другого ВП при помощи функции ввода/вывода файла.

Для того чтобы заставить ВП регистрировать данные лицевой панели, выберите опцию **Протоколирование данных ⇒ Протокол** (Data Logging ⇒ Log) из меню **Управление**. В первый раз при регистрации данных вас спросят о «переплете» (binding) файла протокола, который и будет содержать все регистрации. Это всего лишь имя вашего файла, поэтому назовите его так, как вам хочется.

Можно настроить ВП на автоматическое протоколирование данных по завершении выполнения ВП, выбрав функцию **Управление протоколом по завершении** (Log at Completion). Всякий раз, когда вы заносите данные в тот же самый файл, вы создаете в нем новую регистрацию.

Для интерактивного обзора протоколированных данных используйте функцию **Управление ⇒ Протоколирование данных ⇒ Восстановить** (Retrieve). Инструментальная панель (Toolbar) превратится в инструментальную панель восстановленных данных, как это показано на рис. 15.17. Все элементы управления и отображения лицевой панели тотчас изменят свои значения для показа сохраненных данных.

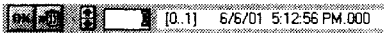


Рис. 15.17

Выделенный номер информирует о регистрации, которую вы в данный момент наблюдаете. Диапазон чисел в скобках справа указывает на количество существующих регистраций. Вы можете последовательно переходить от одной регистрации к другой, используя кнопки со стрелками. Во время такого перехода объекты лицевой панели будут отображать данные, соответствующие той или иной регистрации. Справа от номера регистрации метка даты/времени показывает время, когда запись была сделана. Допустимо удалить отдельную регистрацию, выбрав ее номер и щелкнув по иконке Корзины. Щелкните по **ОК** для выхода из режима восстановления данных.

Наиболее простым способом программного протоколирования информации в файле протокола является использование ВП, который регистрировал бы данные как ВПП. Вызвав контекстное меню этого ВПП, вы можете выбрать опцию **Разрешить доступ к базе данных** (Enable Database Access). После этого вокруг ВП появится желтая рамка (halo). Если вы запустите такой ВПП с доступом в базу данных, он не будет выполняться. Наоборот, он возвращает сохраненные данные его лицевой панели в виде кластера в соответствии с номером регистрации, подключенным к входу. Кластер содержит все данные лицевой панели. Каждый элемент в кластере расположен в том же порядке, что и объекты лицевой панели.

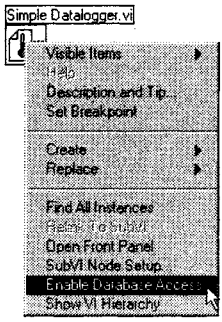


Рис. 15.18

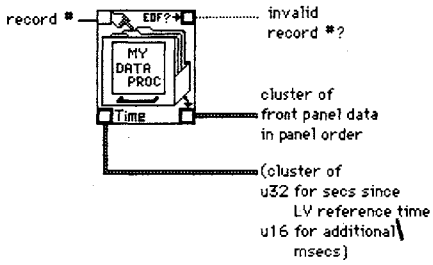


Рис. 15.19

Программный ввод/вывод файла протокола

Кроме создания и считывания файлов протокола посредством протоколирования LabVIEW, вы можете задействовать немного более функциональные программы ввода/вывода файлов протокола, применяя ВП **Считать файл** и **Записать файл**.

При использовании *программируемых* возможностей работы с файлами протокола вам не нужно сохранять *все* данные лицевой панели, достаточно сохранить лишь часть. Ведь на самом деле данные не обязательно находятся только на лицевой

панели, они могут создаваться и на блок-диаграмме. Задачей файлов протокола является сохранение в одном файле нескольких записей одного типа данных LabVIEW.

Несмотря на то что разрешается сохранить лишь один тип данных в файле протокола, существует возможность объединить несколько различных переменных в один кластер, который является естественным типом данных. Тип данных может быть также представлен массивом, строкой или логическими значениями. Важность файлов протокола заключается в том, что вы вправе считывать и записывать данные напрямую в переменные LabVIEW без преобразования в текст, заголовки и т.д.

Хотя использование функций **Записать файл** и **Считать файл** для сохранения файлов протокола аналогично применению этих файлов для сохранения текстовых файлов, смысл терминалов **режим считывания** и **относительное смещение** меняется. Теперь значения этих терминалов определяют не маркер файла, а номер регистрации. Таким образом, для считывания определенной регистрации вы можете подключить номер регистрации к терминалу **относительное смещения** ВПП **Считать файл**. Терминал **счетчик** (count) определяет количество регистраций, которые вы хотите считать, но не количество байтов.

Ввод данных или типа данных ВПП ввода/вывода файлов называется **данные** (data), или **двоичный тип данных** (byte stream type), или **тип данных протокола** (datalog type) – в зависимости от особенностей функции. В окне помощи ввод этих терминалов всегда показан в виде жирной коричневой линии, что свидетельствует о полиморфизме ввода. Для простоты мы будем называть его **тип данных** (data type).

При использовании файлов протокола следует подключать ввод типа данных ко всем вспомогательным функциям (таким, как **Создать файл** или **Открыть файл** – но не к **Считать файл**). Это необходимо для информирования виртуальных приборов считывания и записи о том, что вы работаете с файлами протокола (в противоположность двоичным файлам), что весьма важно, поскольку вводы **режима считывания** и **относительного смещения** меняют свое поведение. Этот аспект является, возможно, одним из самых сложных при работе с LabVIEW. Следующее правило поможет вам избежать возникновения затруднений:

При работе с файлами протокола *всегда* подключайте ввод **тип данных (данные, тип данных протокола)** во всех важных ВПП ввода/вывода файлов за исключением функции **Считать файл**. При работе с ней оставьте ввод **тип данных** неподключенным, чтобы задать файл протокола.

При работе с двоичными файлами подключите ввод **тип данных (данные, двоичный тип данных)** только в функциях **Считать файл** и **Записать файл**. Для установки двоичного файла в функциях **Создать файл**, **Файловый диалог** и т.д. необходимо оставить ввод **тип данных** неподключенным.

На рис. 15.20 изображена блок-диаграмма ВП **Simple Datalogger.vi**, который можно найти на компакт-диск. Этот ВП сохраняет файлы протокола, которые

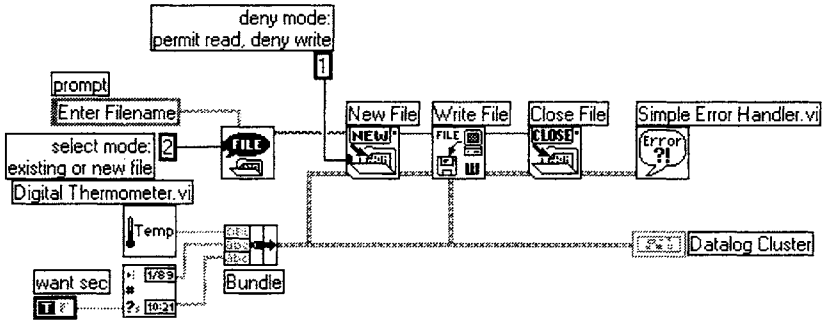


Рис. 15.20

включают в себя отметку времени/даты и числовые значения температуры. ВП содержит также обработчик ошибки, который желательно применять в случае использования операций ввода/вывода файла. Обратите внимание, что тип данных протокола соединен со всеми функциями ввода/вывода файла (за исключением ВПП **Закрыть файл**, для которого задание типа данных несущественно).

ВП **Simple Temp Datalog Reader** предназначен для извлечения данных из файлов протокола (он также есть на компакт-диске). Его блок-диаграмма показана на рис. 15.21.

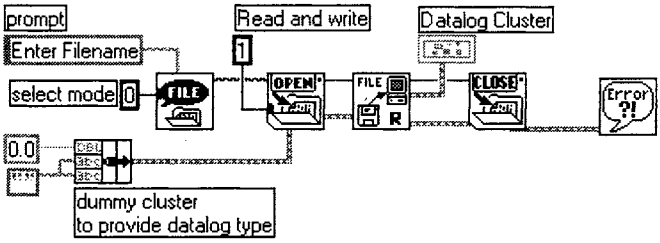


Рис. 15.21

Обратите внимание, каким образом на блок-диаграмме создается константа соответствующего типа кластера для подключения к вводу **тип данных**. Здесь нужно быть осторожным, поскольку необходимо задать точный тип данных, в формате которого была сохранена информация. Например, в случае числового типа **I32** всякая попытка считать данные посредством вышеупомянутого ВП обречена на неудачу. Чтобы удостовериться, что вы считываете нужную переменную, очень важно знать порядковые номера элементов кластера.

15.1.6. Запись и считывание двоичных файлов

Двоичные файлы (binary, byte stream) по сравнению с текстовыми – как легковой автомобиль Porsche по сравнению с пятиосным грузовиком: они намного меньше

и намного быстрее. Недостатком является то, что при считывании двоичного файла вам следует знать все подробности его написания, иначе ничего не получится. В файле могут находиться необработанные данные, поэтому вашей задачей, как программиста, будет создание способа интерпретации этих данных перед их считыванием. Поскольку в двоичном файле отсутствует подробная информация о типе данных или о таких вещах, как заголовки, то вы должны знать, что в нем находится.

Коротко говоря, двоичные файлы не так уж просты в использовании по сравнению с другими типами файлов. Однако LabVIEW содержит некоторые высокоуровневые ВП, которые дают возможность считывать и записывать числовые данные. Эти функции находятся в подпалитре **ВП Двоичных файлов** (Binary File Vis) палитры **Ввод/вывод файлов** (рис. 15.22).

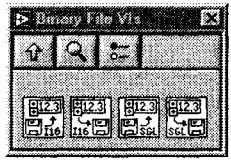


Рис. 15.22

Два первых ВП дают возможность считывать и записывать числовые данные формата I16; вторая пара позволяет считывать и записывать числовые данные типа SGL. Все ВП основаны на предположении, что данные представлены в формате массива (одномерного или двумерного). Ниже приводится описание выполняемых ими функций:

- записывает одно- или двумерный массив целых чисел без знака (I16) в двоичный файл, заданный путем к нему (file path). Вы можете создать новый файл либо добавить к старому с помощью логического ввода **добавить к файлу?** (append to file?);
- считывает двоичный файл, типом данных которого является массив целых чисел (I16). Если вы хотите получить двумерный массив заданного размера на выходе **2D-массив** (2D array), то подключите к входам **количество строк** (2D number of rows) и **количество столбцов** (2D number of columns) определенные значения. Для считывания всех данных файла в виде одномерного массива оставьте эти вводы неподключенными. Вы можете дополнительно задать смещение (в байтах) через вход **смещение начала считывания** (start of read offset) для доступа в любое место файла.

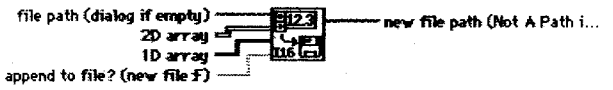


Рис. 15.23. Запись в файл I16 (Write To I16 File.vi)

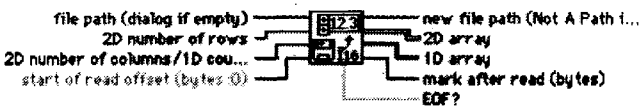


Рис. 15.24. Считывание из файла I16 (Read From I16 file.vi)

Два других ВП в данной палитре: **Запись в файл SGL** (Write To SGL File) и **Чтение из файла SGL** (Read From SGL File) – выполняют те же функции с другим типом данных.

В качестве простого примера рассмотрите ВП **Binary Read/Write.vi**, который получает осциллограмму, сохраняет ее в файле или дает вам возможность считать эти данные из файла. В любом случае данные поступают на развертку осциллограммы.

Обратите внимание на функцию преобразования данных в тип SGL. При работе с такими двоичными файлами очень важно подключать правильный тип данных, иначе вы не сможете их правильно считать.

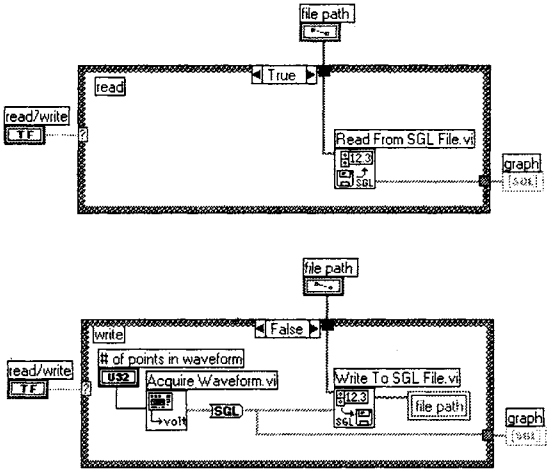


Рис. 15.25

Информация для любопытных

Если вы хотите подробнее узнать, как использовать двоичные файлы с ВПП низкого уровня, то продолжайте чтение этой книги.

Функция **Записать файл** «перескакивает» через информацию о заголовке, которую LabVIEW использует для хранения строки в памяти и просто записывает содержимое строковых данных в файл. В действительности эта функция не отличает строку ASCII от двоичной строки – она просто помещает данные в файл. Другими словами, между текстовыми и двоичными текстовыми файлами не имеется *функционального* отличия: будет строка пониматься как текст или как двоичная информация зависит от того, как вы ее интерпретируете.

Однако помните, что вход **данные** функции **Записать файл** является *полиморфным*: он может адаптироваться к любому типу данных, который вы на него подаете.

Например, допустимо подать двумерный массив чисел на вход **данные** так же, как вы это делали с файлами протокола.

Однако файлы протокола и двоичные файлы сильно отличаются друг от друга по структуре и поведению, несмотря на то что вы используете те же самые функции LabVIEW при работе с ними. Поскольку данный аспект является довольно сложным для понимания, мы повторим правило о том, как работать с этими двумя типами файлов.

Функция **Записать файл** на блок-диаграмме на рис. 15.26 просто байт за байтом записывает входные данные в файл. Она не преобразует числа в ASCII и не помещает в файл никакой информации о количестве строк и столбцов в массиве. Если массив входа **данные** имеет три строки и два столбца, то размер файла будет равен 24 байтам (3 строки × 2 столбца × 4 = 24 байта). (Вспомните, что число одинарной точности с плавающей запятой занимает 4 байта памяти.)

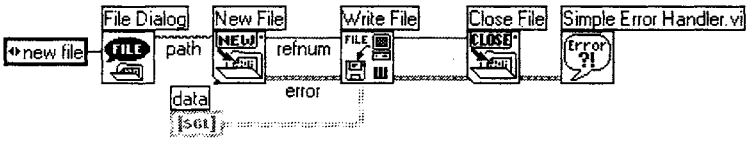


Рис. 15.26

Данные, сохраненные функцией **Записать файл**, совпадают с данными, полученными посредством удаления заголовка с выхода функции **Перевести в строку** и записи результирующей строки в файл. Эта функция все данные входного терминала размещает в двоичной строке. Перед данными выходной строки она располагает заголовок, необходимый для декодирования строки в исходный тип данных.

Приведенный пример показывает очень важный аспект при работе с двоичными файлами. Если вы не сохраните в файле какой-либо информации заголовка, то вам не удастся успешно интерпретировать файл. В нашем примере ВП сохранил в файл 24 байта данных. Даже если вы знаете, что первоначальными данными были числа одинарной точности с плавающей запятой, вы не сможете успешно реконструировать двумерный массив. Как же так? – спросите вы. Вы знаете, что имеется шесть значений в массиве (24/4 = 6), но откуда вам знать, были ли первоначальные данные сохранены в одномерном массиве из шести элементов, в массиве с одной строкой и шестью столбцами или в таблице с тремя столбцами и двумя строками?

Когда вы будете работать с двоичными файлами, вы поймете, что сохранение информации заголовка является очень важной процедурой при работе с файлом. К счастью, функция **Записать файл** имеет очень простой способ создания заголовков, с тем чтобы в дальнейшем записать их в файл. Если вы подадите логическое значение ИСТИНА на вход **заголовок** (header) функции **Записать файл**, то она

запишет в файл точно такие же данные, как если бы вы создали двоичную строку с помощью функции **Перевести в строку** и записали ее в файл.

Записи двоичных данных в файл с использованием опции **Заголовок** (header) показаны на рис. 15.27.

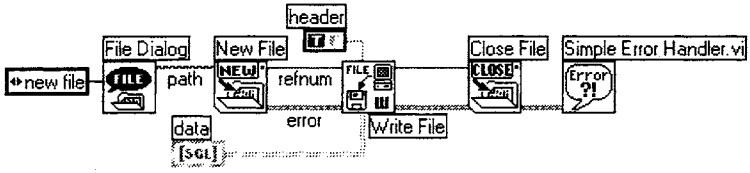


Рис. 15.27

Для считывания файла вы можете использовать диаграмму на рис. 15.28. Обратите внимание, что на вход **двоичный тип данных** (byte stream type) функции **Считать файл** поступает пустой двумерный массив чисел одинарной точности. Функция **Считать файл** применяет лишь тот *тип данных*, который поступает на названный вход. При использовании этой функции вышеуказанным способом она оперирует данными файла аналогично тому, как функция **Восстановить из строки** (Unflatten From String) оперирует входной строкой. Вспомните, что, если на входы **строковые данные** (data string) и **тип данных** (data type) функции **Восстановить из строки** подать некоторые значения, то она преобразует информацию, сохраненную в строке, в заданный тип данных при условии, что информация заголовка двоичных данных является правильной.

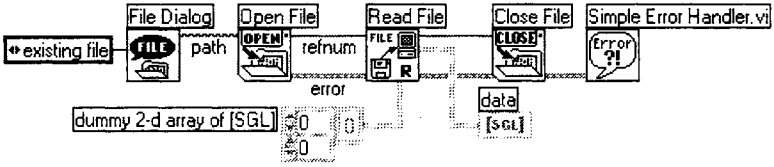


Рис. 15.28. Считывание файла, созданного в предыдущих упражнениях

Когда вы создаете двоичные файлы, то конструирование соответствующего заголовка для этого файла является зачастую наиболее важной задачей. Кроме того, вы должны четко документировать всю информацию, необходимую для интерпретации заголовка, чтобы иметь возможность работать с этим файлом в будущем.

Преимущество двоичных файлов – возможность доступа в произвольное место файла. Например, если вы сохраняете массивы числовых данных, то вам может

потребоваться доступ к данным, находящимся в разных местах файла. В файлах ASCII произвольный доступ затруднен из-за наличия отрицательных знаков, меняющегося количества цифр в разных значениях данных и других факторов. Эти препятствия отсутствуют при работе с двоичными файлами.

В LabVIEW двоичное отображение самого числа является сжатым форматом числа в двоичных файлах. Следовательно, каждое число в массиве использует фиксированное количество байтов на диске. Если известно, что в файле сохранены числа одинарной точности, которые используют 4 байта на число, вы легко можете извлечь произвольную группу элементов из массива, как это показано на рис. 15.29.

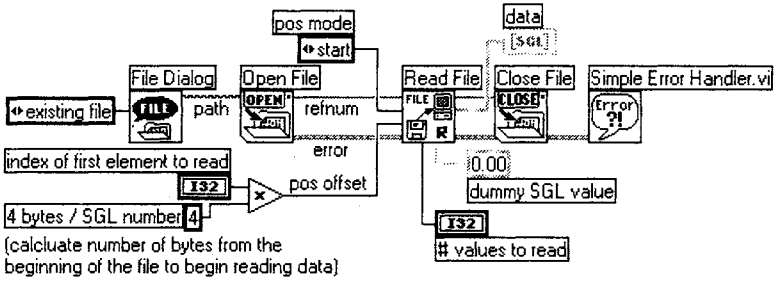


Рис. 15.29

Рассмотрим пример.

ВП **Записать двоичный файл** (Write Binary File) на рис. 15.30 создает массив случайных числовых (DBL) данных и записывает их в двоичный файл. Обратите внимание, что ввод **тип данных** подключен только к ВП **Записать файл**.

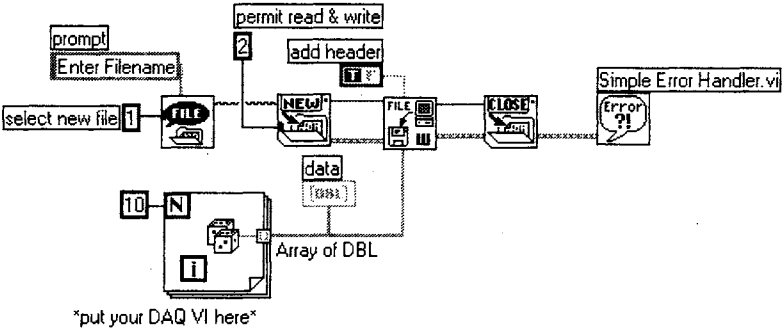


Рис. 15.30

Для считывания двоичных данных мы можем использовать ВП **Считать двоичный файл** (рис. 15.31).

Считывание файла происходит не так просто. Чтобы восстановить числовой массив, мы должны подключить ввод **счетчик** (count) в ВПП **Считать файл**. ВП **EOF** используется для подсчета общего количества байтов в файле, которое делится на 8 для подсчета количества элементов массива.

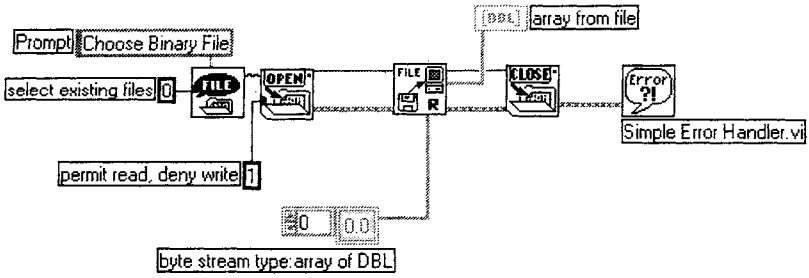


Рис. 15.31

Если некоторые операции ввода/вывода файла были затруднительны для понимания, не переживайте: даже бывалые программисты на LabVIEW испытывают трудности с метками, заголовками и т.д. Лучший совет в этом случае – использовать хорошие примеры, вроде тех, которые представлены в данной книге или содержатся в LabVIEW.

15.1.7. Упражнение 15.2: использование текстовых, двоичных файлов и файлов протокола

Применяя ваши собственные ВП сбора данных, создайте ВП, который выполнял бы три функции: считывание/запись в двоичный файл, считывание/запись в текстовый файл (ASCII) и работа в качестве регистратора данных (сохранение в файл протокола). Затем с помощью ВП **Информация о файле/директории** (File/Directory Info) из палитры **Ввод/вывод файлов** ⇒ **Дополнительно** (Advanced) измерьте размер каждого из трех созданных файлов. Сравните их размер – результат будет интересным. В качестве дополнительной возможности включите способ измерения времени, которое тратится на чтение и запись файла.

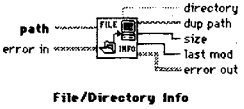


*Рис. 15.32. ВП **Информация о файле/директории***

Сохраните ваш ВП с именем **File Type Comparison**.

15.1.8. Работа с данными осциллограммы в файлах

Если вашей главной задачей является сохранение и считывание данных в формате осциллограммы LabVIEW (см. главу 8), воспользуйтесь функциями **Ввод/вывод осциллограмм** (Waveform I/O) в палитре **Осциллограмма** (рис. 15.33). Это актуально, например, когда вы получаете данные в виде осциллограммы с платы ввода/вывода и хотите сохранить их на диске. На самом деле эти функции работают лишь с файлами протокола. Единственным отличием является то, что они могут сами «позаботиться» о задании формата.

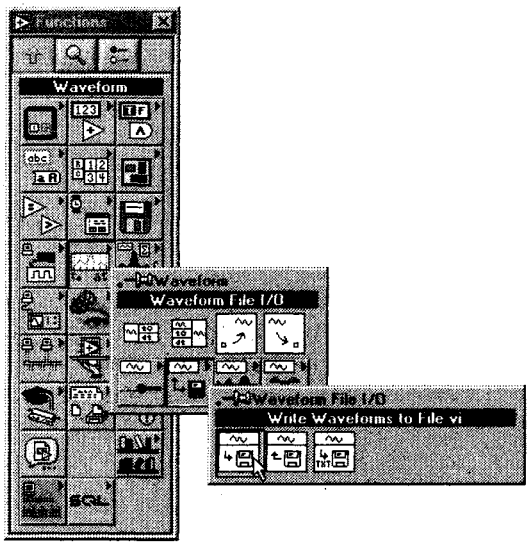


Рис. 15.33

ВП создает новый файл протокола, записывает определенное количество регистраций в файл, а затем закрывает его и проверяет ошибки. Каждая регистрация представляет собой кластер, который содержит строку и массив чисел одинарной точности.

ВП открывает файл протокола, созданный ВП **Запись осциллограмм в файл**, считывает одну за другой регистрации, пока все не будут считаны. Затем закрывает файл и проверяет ошибки. Регистрация представляет собой кластер, который содержит строку и массив чисел одинарной точности.

ВП преобразует двумерный или одномерный массив чисел одинарной точности в текстовую строку и записывает строку в двоичный файл или присоединяет

строку к существующему файлу. Дополнительно вы можете транспонировать данные. Этот ВП открывает или создает файл заранее и затем закрывает его. Допустимо использовать его для создания текстового файла, удобочитаемого текстовыми редакторами.

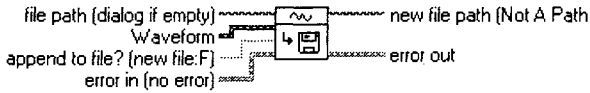


Рис. 15.34. ВП ***Запись осциллограмм в файл***

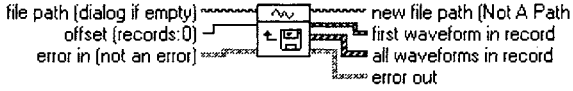


Рис. 15.35. ВП ***Считывание осциллограмм из файла***

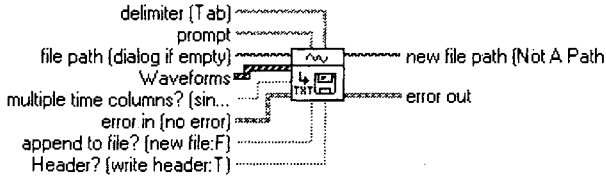


Рис. 15.36. ВП ***Экспорт осциллограмм в файл табличного формата***

15.2. Печать в LabVIEW

Как уже говорилось в главе 5, вы можете распечатать активное окно в LabVIEW, выбрав функцию **Печатать окно** (Print Window) в меню **Файл**. Этот метод не является исчерпывающим, если нужно, чтобы программа автоматически создавала большое количество распечаток. Вам может понадобиться программа, которая имела бы кнопку **Печать** (Print) на лицевой панели или распечатывала бы график при появлении определенной структуры данных. К счастью, LabVIEW позволяет автоматизировать печать.

LabVIEW предоставляет возможность автоматизированной распечатки лицевой панели, когда ВП заканчивает выполнение. Для этого следует выбрать функцию **Печатать по завершении** (Print at Completion) в меню **Управление**. Это довольно простое дело, если ваша задача заключается лишь в получении распечатки лицевой панели при каждом запуске ВП. Но что делать, если надо, например, распечатать лишь часть лицевой панели ВП? Или вы хотели бы печатать лишь в определенных случаях, а не тогда, когда ВП заканчивает выполнение? Это тоже легко сделать, если выбрать функцию **Печатать панель ВП** (Print Panel VI) из палитры **Управление приложением** (Application Control).

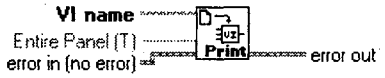


Рис. 15.37. ВП ***Печатать панель***

ВП **Печатать панель** совершает такую же распечатку ВП, как и **Печатать по завершении**, если только лицевая панель ВП была открыта перед его выполнением. Функция **Печатать панель** может быть вызвана в любое время из другого ВП, а не только по завершении выполнения. По умолчанию она распечатывает всю панель, а не только то, что видно в окне. Этот ВПП предполагает, что ВП загружен, но не требует, чтобы окно было открыто.

Давайте выполним простое упражнение, чтобы увидеть, как программно делать распечатку.

15.2.1. Упражнение 15.3: запрограммированная печать

В этом упражнении необходимо распечатать лицевую панель, используя функцию **Печатать панель**. Вы должны дать пользователям возможность нажимать кнопку запуска печати, когда им угодно.

1. Откройте **Temp Limit (Max/Min).vi**, который вы создали в главе 8. Если у вас его нет, обратитесь к директории `CH8.LLB` на компакт-диске.
2. Сохраните его копию под именем **Temp Limit with Print.vi**.
3. Введите на лицевую панель кнопку **Печать** с механическим действием «Срабатывать при отпускании» (Latch When Released).
4. Измените блок-диаграмму, как показано на рис. 15.38. Не ошибитесь при вводе имени вашего ВП.
5. Запустите ВП.

Также разрешается программно управлять функциями печати, используя методы и свойства сервера ВП. Мы уже говорили о сервере ВП в главе 13. Например, при работе с классом ВП вы можете получить доступ к таким методам, как **Печатать на принтере** (Print to Printer), **Печатать в HTML** (Print to HTML) и **Печатать в RTF** (Print to RTF). С помощью класса приложения легко управлять такими свойствами, как печать границы вокруг ВП и т.д. За дополнительной информацией обращайтесь к документации LabVIEW.

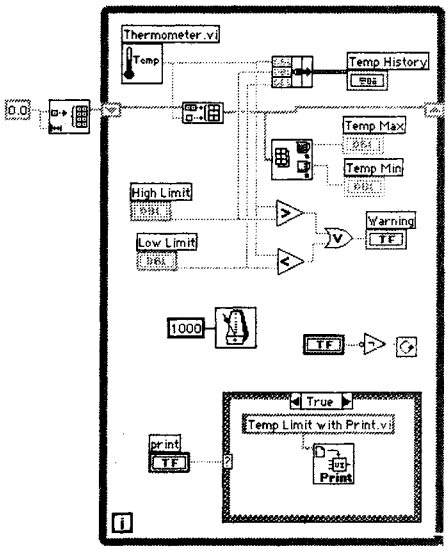


Рис. 15.38

15.3. Отчеты в LabVIEW

Поскольку распечатывание лицевых панелей ВП часто полезно в качестве документированного результата, иногда возникает необходимость создать текстовые отчеты, которые требуют точного размещения на странице и использования различных видов шрифта (жирный, курсив) или таблицы данных. В этом случае вы можете воспользоваться функциями **Отчет** (Report) из палитры **Создание отчета** (Report Generation).

Функции **Создание отчета** дают возможность создавать текстовые отчеты (включая RTF) или отчеты HTML, которые легко сохранить в файл или отправить прямо на принтер.

15.4. Итоги

В данной главе вы детально познакомились с ВПП ввода/вывода файлов и печати, узнали, что LabVIEW поддерживает различные типы файлов: ASCII, протокола (Datalog), двоичные (Binary).

Файлы ASCII (или текстовые файлы) являются наиболее простыми в применении, но занимают много места на диске.

Двоичные файлы достаточно эффективны, но требуют некоторой дополнительной работы при форматировании, чтобы в дальнейшем их можно было использовать.

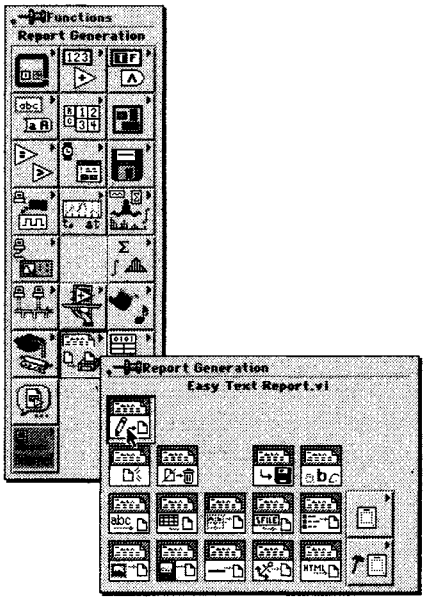


Рис. 15.39

Файлы протокола – специфический тип файлов LabVIEW, который дает возможность сохранять регистрации типов данных LabVIEW.

Каждый из перечисленных типов файлов служит для выполнения соответствующих задач, но в большинстве случаев вы можете использовать для работы с ними одни и те же функции из палитры **Ввод/вывод файлов**. Вы проверили, как действуют функции открытия, чтения, записи и закрытия, а также как создаются новые файлы.

Наконец, вы познакомились с функциями программной печати в LabVIEW.

Обзор

В этой главе основное внимание уделяется технике, советам и предложениям по улучшению виртуальных приборов LabVIEW. Мы начнем с лицевой панели прибора и поговорим о том, как сделать ее более приятной для глаз и более функциональной. В LabVIEW используется концепция интуитивного графического интерфейса пользователя, и мы покажем, как сделать лицевую панель более привлекательной. Будут рассмотрены расположение элементов панели, оформление, настройка элементов управления, окна оперативной помощи и т.д. Мы обсудим некоторые общие проблемы программирования и их решение. Будут затронуты такие темы, как быстродействие, управление памятью и межплатформенная совместимость для создания наиболее совершенных ВП. В завершение мы испытаем некоторые общие подходы стиля программирования, который вам особенно пригодится при разработке крупных проектов в LabVIEW.

ЗАДАЧИ

- Познакомиться с основными принципами и рекомендациями для создания эстетически приятного и профессионально выглядящего графического интерфейса
- Научиться импортировать внешнее изображение на лицевую панель и внутрь кольцевых списков
- Индивидуально настраивать элементы управления, используя редактор
- Научиться динамически открывать и закрывать окно помощи
- Познакомиться с некоторыми остроумными решениями вопросов повышенной сложности программирования в LabVIEW
- Научиться увеличивоть производительность, использовать меньше памяти и, при необходимости, делать ВП независимыми от платформы
- Узнать некоторые тонкости создания удивительных приложений
- Сделаться самым «крутым» программистом на LabVIEW в городе

ОСНОВНЫЕ ТЕРМИНЫ

- Оформление
- Настройка элементов управления
- Редактор элементов управления
- Импортирование изображений
- Настройка помощи
- Эстетика
- Наиболее распространенные вопросы
- Использование памяти
- Производительность
- Зависимость от платформы
- Хороший стиль

ИСКУССТВО ПРОГРАММИРОВАНИЯ В LabVIEW

16

16.1. Почему так важен графический интерфейс

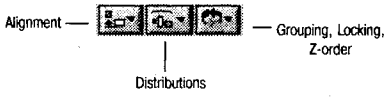
В наше время изображение значит очень многое – даже в программном обеспечении. На людей чаще всего производит впечатление сам вид программы на экране, а не то, как она в действительности работает. По крайней мере, мне известна пара причин, побуждающих навести лоск и улучшить графический интерфейс пользователя:

- он производит впечатление на других людей (вашего руководителя, посетителей, вашу супругу (супруга)) и убеждает их в высоком качестве вашего программного обеспечения;
- он делает ваше программное обеспечение более легким в использовании для конечного пользователя;
- его создание проще, чем выполнение конкретной работы.

Именно в этом и заключается привлекательность LabVIEW – даже новичок способен собрать впечатляющий графический интерфейс на лицевой панели (ничего страшного в том, что он пока ничего не может делать) раньше, чем программист (на C) высокого класса успеет моргнуть глазом.

Объекты лицевой панели LabVIEW уже имеют достаточно привлекательный вид. В этой главе мы научим вас нескольким трюкам, которые помогут создать еще более совершенный интерфейс. Вы также узнаете, как ввести оперативную подсказку, чтобы пользователь узнавал функции каждого элемента управления, не обращаясь к руководству. Создание привлекательного интерфейса преследует не только эстетические цели, но и экономит время и усилия пользователя.

Взгляните на ВП, изображенный на рис. 16.1, – системный мониторинг температуры. Он выглядит достаточно привлекательно. Однако почти все согласятся с тем, что этот самый ВП с интерфейсом, показанным на рис. 16.2, был бы намного удобнее в работе и, несомненно, имел бы большую визуальную привлекательность.



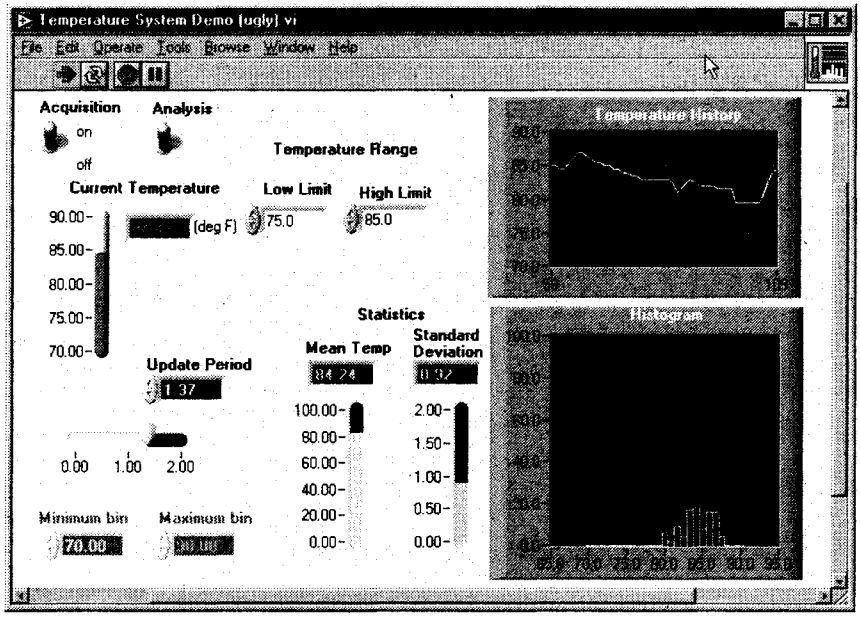


Рис. 16.1

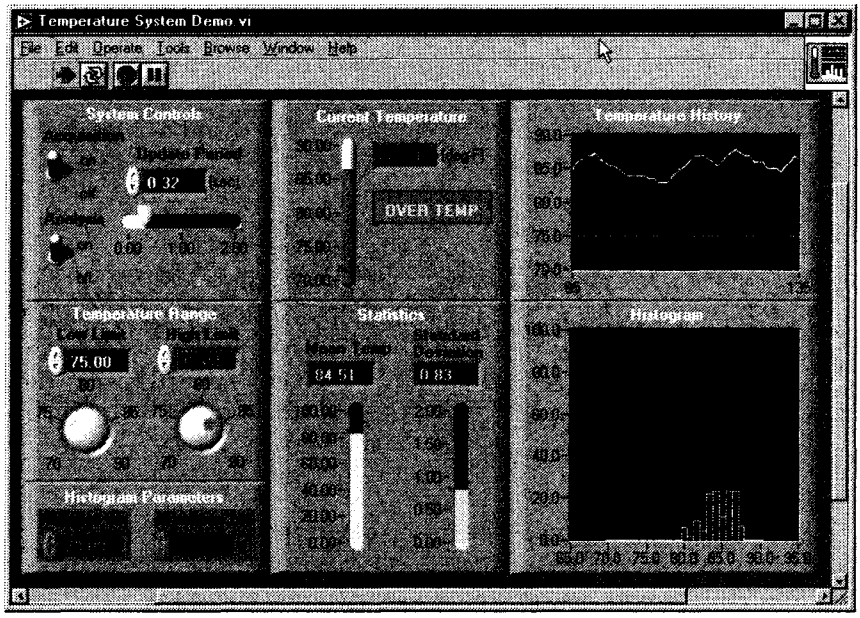


Рис. 16.2

16.2. Размещение, оформление, группировка и блокирование

Одним из способов улучшения внешнего вида лицевой панели ВП является правильное физическое размещение объектов на ней. Вы можете сделать это посредством выравнивания и равномерного распределения объектов, а также группировкой наборов из них так, чтобы они логически соответствовали своему назначению.

LabVIEW дает возможность *выравнивания, распределения, группировки и блокирования* объектов. Имеется широкий выбор способов выравнивания объектов. Распределение объектов предполагает их организацию в пространстве среди других объектов. Группировка позволяет легко перемещать целый набор объектов, не нарушая их взаимного расположения. Блокирование объектов предотвращает случайный сдвиг или изменение их размера. Вы можете также изменить глубину размещения объектов (то есть указать, что находится на переднем и на заднем планах). Управление этими графическими опциями осуществляется с инструментальной панели в окне ВП.

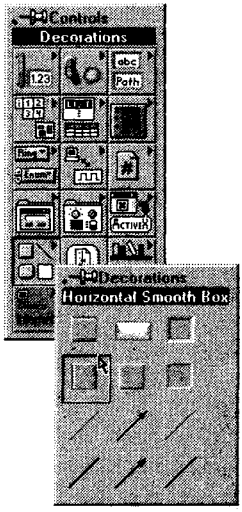


Рис. 16.3

Помните, что ярлык объекта может размещаться в любом месте, но он все равно остается привязанным к объекту и может служить эталонной точкой при выравнивании и распределении объектов.

В палитре **Элементы управления** вы найдете палитру набора **Оформление** (Decorations) – рис. 16.3.

Эти элементы лицевой панели ничего не выполняют и не имеют соответствующего терминала на блок-диаграмме. Вы можете использовать окна, линии, кадры и т.п. для группировки элементов управления и индикаторов лицевой панели в логические блоки. Также разрешается применять их для создания большего сходства лицевой панели ВП с панелью реального прибора.

Вам часто придется обращаться к функциям глубины размещения, таким как **Сдвинуть на задний план** (Move to Back), для того чтобы разместить декорации позади элементов управления. Причина использования этих команд становится очевидной при создании декорации, содержащей ряд элементов управления. Декорация обычно не является прозрачной, поэтому при размещении ее на лицевой панели *после* создания элементов управления она затемнит их, поскольку находится впереди.

Щелчок мышью по декорации отнюдь не равнозначен щелчку по свободному месту лицевой панели. Используя декорацию позади элементов управления, очень просто неумышленно выделить ее при работе курсора в режиме редактирования.

16.3. Да здравствует искусство: импортирование рисунков

Можно вставлять рисунки прямо в LabVIEW и включать их в лицевую панель. Например, вам понадобилось сделать клише, которое содержало бы логотип вашей компании. Или возникла идея вставить рисунки системы труб и клапанов, которые бы представляли цикл управления каким-либо процессом (легко сделать рисунки клапанов логическими элементами управления, как это показано в разделе 16.4).

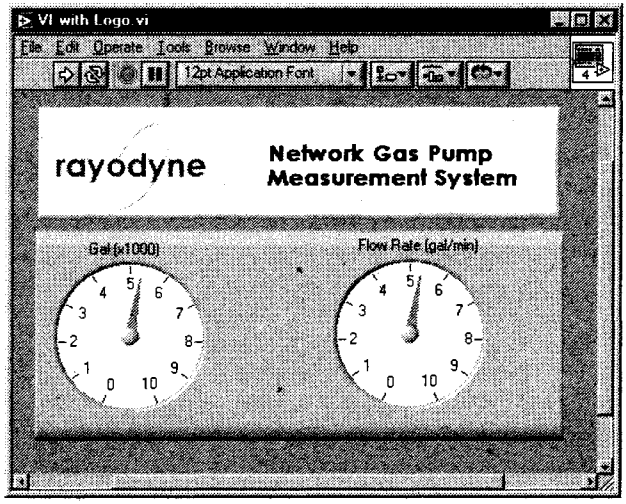


Рис. 16.4

Для импортирования рисунка в LabVIEW в Windows или MacOS просто скопируйте его из программы, в которой он открыт, и вставьте на лицевую панель ВП. LabVIEW будет рассматривать этот рисунок как декорацию: вы можете изменить его размер, поместить позади или впереди других объектов и т.д, но не редактировать.

При работе в Mac LabVIEW преобразует графику в формат PICT, который имеет достаточно высокую разрешающую способность.

В системе Windows LabVIEW обычно преобразует рисунок в растровое изображение (BMP). Недостатком такого изображения по сравнению с другими популярным графическими форматами в Windows является потеря некоторой разрешающей способности при изменении размера рисунка. Однако LabVIEW поддерживает формат усовершенствованного метафайла (EMF) – более новый графический формат для Windows. Рисунок в таком формате может быть растянут без искажений и способен иметь прозрачные участки, что избавляет вас от необходимости подбирать цвет заднего плана лицевой панели ВП. Если вы импортируете рисунок такого формата в LabVIEW, то последний принимает его без преобразования в растровое изображение.

Другой областью, где используются рисунки, являются функции **Кольцевой список рисунков** (Picture Ring) или **Кольцевой список рисунков и текста** (Picture & Text Ring) в палитре **Кольцевые списки** (List & Ring) – два примера представлены на рис. 16.5. Применение кольцевых списков рисунков позволяет творчески подойти к созданию собственного графического набора опций, которые можно индексировать и отслеживать.

Чтобы вставить рисунок в кольцевой список, сначала скопируйте его, а затем выберите функцию **Импортировать рисунок** (Import Picture) в контекстном меню элемента управления кольцевого списка. Для введения большего количества рисунков воспользуйтесь функциями **Импортировать рисунок после** (Import Picture After) или **Импортировать рисунок до** (Import Picture Before). Функция **Импортировать рисунок** накладывает вставляемый рисунок на уже имеющийся.

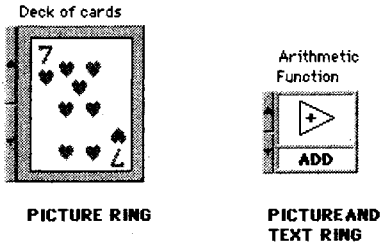


Рис. 16.5

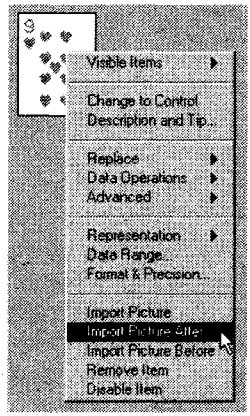


Рис. 16.6

16.4. Настройка внешнего вида элементов управления и индикаторов

Вопрос: Что общего у объектов, представленных на рис. 16.7?
Ответ: Все они являются элементами управления LabVIEW!

Вы можете переделывать элементы управления и отображения по своему желанию с целью лучшего соответствия программе и более впечатляющего графического интерфейса. В предыдущих упражнениях для представления одного из этапов изготовления какого-либо изделия вы могли бы, например, отобразить положение ящика на ленте конвейера. Или допустимо было бы включать и выключать печь, щелкая мышью по рисунку элемента управления печью (рис. 16.7).

Вы можете сохранить созданный элемент управления или отображения в директории или библиотеке ВП, как это делалось с виртуальными приборами и глобальными

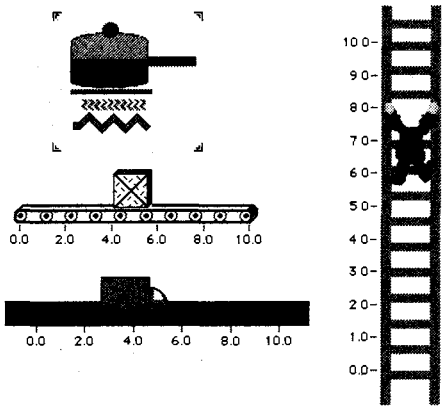


Рис. 16.7

переменными. Общепринято использовать при наименовании файлов элементов управления расширение .ctl. Сохраненный элемент управления разрешается применять в других ВП. Вы даже можете создать эталон элемента управления, если хотите задействовать его во многих местах одного и того же ВП, сохранив этот элемент управления в качестве *определителя типа*. При внесении изменений в определитель типа LabVIEW автоматически обновляет все ВП, в которых он присутствует.

Часто используемый созданный элемент управления можно разместить в палитре **Элементы управления**, щелкнув по кнопке **Опции** в палитре.

Как же создать собственный элемент управления?

16.4.1. Упражнение 16.1: создание собственных элементов управления

Обычно при осуществлении этой операции требуется вставить рисунки в создаваемые элементы управления. Поэтому вначале следует обзавестись файлами рисунков, возможно вместе с графической программой. В LabVIEW нет какого-либо редактора рисунков. При создании элемента управления используется форма существующего элемента управления, такого, например, как логический светодиод или числовой ползунковый элемент управления. В этом упражнении вы создадите логический элемент управления типа клапана, который открывается и закрывается.

1. Поместите логический элемент управления в виде светодиода на лицевую панель и выделите его.
2. Запустите редактор, выбрав опцию **Настроить элемент** (Customize Control) из меню **Правка**.
3. Окно редактора элемента управления покажет логический элемент.
4. В режиме редактирования окно редактора работает как лицевая панель. Вы можете вызвать контекстное меню элемента управления для изменения его характеристик (масштаб, точность и т.п.).

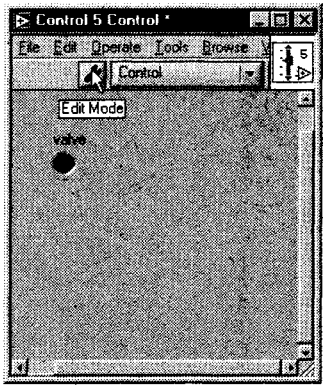


Рис. 16.8

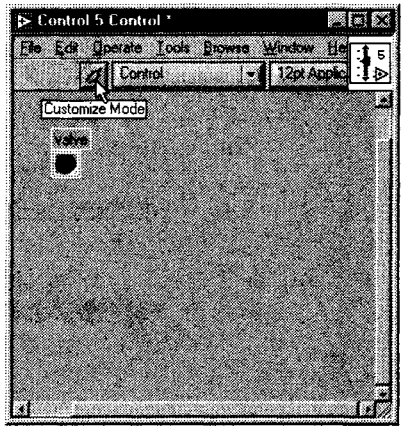


Рис. 16.9

5. В режиме настройки (Customize mode), в который вы перейдете щелчком мыши по кнопке с изображением инструмента, можно изменить размер, цвет и поменять различные компоненты рисунка элемента управления.
6. Скопируйте файл рисунка закрытого клапана (он есть на компакт-диск под именем `closed.bmp`). В режиме настройки вызовите контекстное меню логического элемента управления и выберите функцию **Импортировать рисунок**.
7. Повторите шаг 6 для варианта ИСТИНА логического элемента управления, используя другой рисунок клапана. Вариант ИСТИНА, или открытый клапан, также находится на компакт-диске (файл `open.bmp`).
8. Сохраните созданный вами элемент управления, выбрав функцию **Сохранить** из меню **Файл**. По принятому соглашению элементы управления имеют расширение .ctl.
9. Лицевую панель ВП, изображенную на рис. 16.12, можно найти в примерах полной версии LabVIEW (`examples\apps\demos.llb\Control Mixer Process.vi`). На этой лицевой панели наряду с некоторыми другими используются те же логические элементы управления, которые вы только что создавали.

Рис. 16.10

Рис. 16.11

Важно понимать, что вы можете изменить не поведение элемента управления, а лишь его внешний вид. Также нельзя изменить способ отображения этим элементом своих данных (например, измененный элемент управления, основанный на ползунковом переключателе, всегда будет содержать нечто, перемещающееся вдоль линии). За исключением этих ограничений, допустимо создавать причудливые графические интерфейсы, особенно если вы любите экспериментировать.

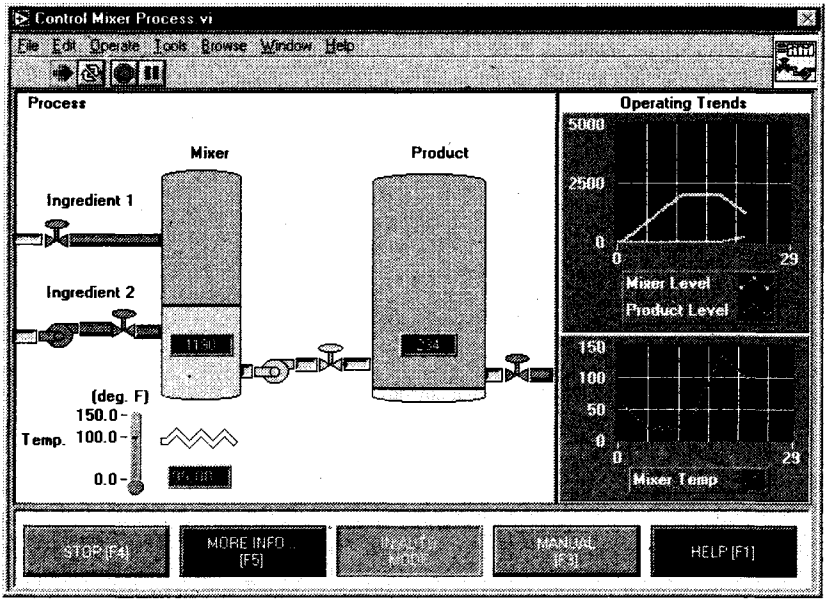


Рис. 16.12

Если вы хотите создать эталон (master copy) элемента управления с тем, чтобы все ВП, в которых он используется, автоматически обновлялись при изменении эталона, выберите функцию **Опр. Типа** (Type Def.) в кольцевом списке панели инструментов редактора элемента управления.

Определитель типа (type definition) заставляет тип данных элементов управления оставаться одинаковым везде, где он применяется. Различные копии определителя типа могут иметь собственное имя, цвет, размер и т.п. И это важно, поскольку лишь у эталона разрешается менять поведение, предотвращая таким

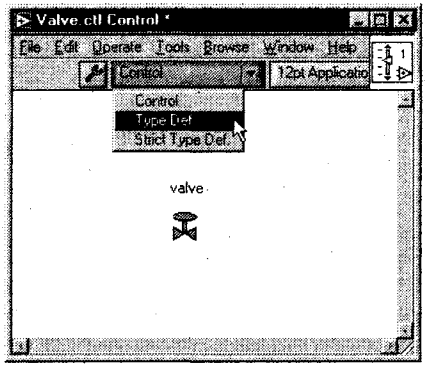


Рис. 16.13

образом появление случайных изменений. *Абсолютный определитель типа* (strict type definition) заставляет все элементы, связанные с элементом управления, быть идентичными везде, где они используются.

16.5. Добавление оперативной подсказки

В тех случаях, когда вам необходимо быстро получить информацию о чем-либо, окно помощи (Help) в LabVIEW будет весьма полезным. Большинство людей просто зависят от нее (помощи) при создании блок-диаграмм. И это хорошая привычка. Но вы можете создать программу, с которой было бы легко работать, путем введения в окно помощи вашей собственной информации об этой программе или контекстной связи с гипертекстовым документом помощи. Доступны следующие три вида справки:

- окно всплывающей подсказки (Tooltip Help): текст находится в окне желтого цвета, которое появляется во время остановки курсора над элементом управления;
- окно контекстной помощи: комментарии, которые появляются в этом окне, описывают элементы управления и отображения во время перемещения над ними курсора;
- помощь в режиме online (гипертекст): текст, в котором вы можете создать контекстную ссылку для вызова файла помощи извне.

Создать первые два вида помощи достаточно легко. Мы уже говорили об этом в главе 5. Нужно выбрать функцию **Описание и подсказка** (Description and Tip) в контекстном меню элемента управления или индикатора. Затем ввести совет и описание в диалоговое окно и щелкнуть мышью по кнопке **OK** для их сохранения. LabVIEW отображает это описание в любой момент, когда вы открываете окно контекстной помощи и перемещаете курсор над объектом лицевой панели.

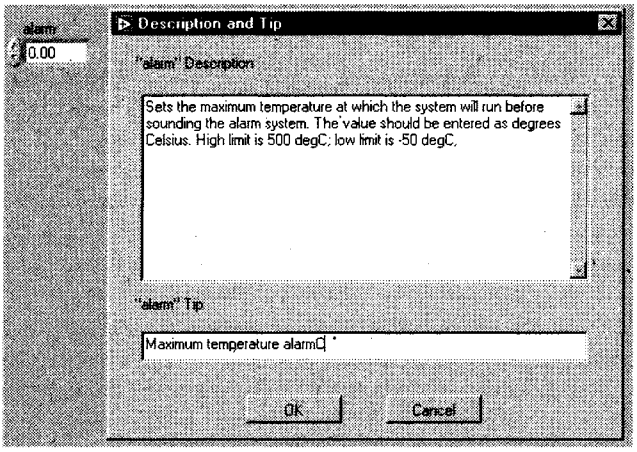


Рис. 16.14

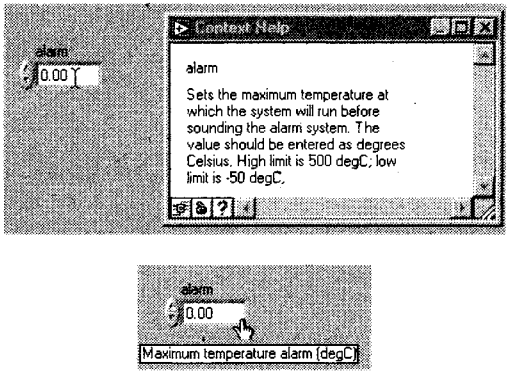


Рис. 16.15

Если вы задокументируете все элементы управления и отображения лицевой панели путем введения их описаний, то конечный пользователь сможет открыть окно контекстной помощи и быстро выяснить смысл объектов лицевой панели.

Для создания описаний ВП введите информацию в текстовое окно, которое появляется при выборе функции **Документация** (Documentation) из меню **Свойства ВП** (VI Properties). При остановке курсора над иконкой ВП на блок-диаграмме другого ВП эти комментарии появятся в окне контекстной помощи вместе со схемой подключений.

Вы можете программно ввести текст, разместить его и закрыть окно контекстной помощи. Для этого воспользуйтесь функциями **Управление окном помощи** (Control Help Window) и **Получение статуса окна помощи** (Get Help Window Status) из подпалитры **Помощь** (Help) палитры **Управление приложением**.

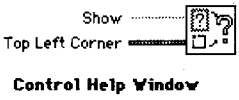


*Рис. 16.16. Функция **Управление окном помощи***

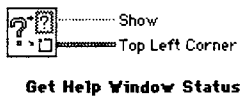


*Рис. 16.17. Функция **Получение статуса окна помощи***

Сигнал на логическом входе **Показать** (Show) закрывает или открывает окно контекстной помощи; ввод кластера состоит из двух числовых индикаторов, которые определяют положение верхнего левого угла окна.

Функция **Получение статуса окна помощи** возвращает состояние и положение окна контекстной помощи.

Более совершенный вид помощи заключается в вызове контекстной связи с внешним файлом помощи. Для создания исходного документа вам понадобится компилятор файлов помощи операционной системы. Компилятор файлов помощи для Windows можно найти на сайте компании Microsoft или у сторонних

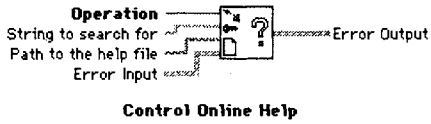


Рис. 16.18 Функция ***Управление online-помощью***

производителей – *RoboHelp* компании Blue Sky Software или *Doc-to-Help* компании WexTech Systems. Для Macintosh используется *QuickHelp* компании Altura Software, для UNIX – *HyperHelp* компании Bristol Technologies. Все компиляторы файлов помощи имеют инструменты для создания соответствующих документов. Для программного вызова этих файлов служит оставшаяся функция в подпалитре **Помощь**.

Функция **Управление online-помощью** манипулирует внешним файлом помощи. Вы можете отобразить содержание, указатель или перейти в определенную часть файла помощи.

16.6. Дополнительные указания и рекомендации

Графический интерфейс

Для достижения впечатляющего эффекта от использования ВП мы приготовили «мешок с трюками», которые собирали в течение длительного времени работы с LabVIEW. Создание многих ВП начинается с набросков без какого-либо порядка или эстетики, которые позднее никогда не подчищаются. Поверьте, улучшение стоит усилий. Даже если вы и не собираетесь кого-либо удивить, вам и другим людям будет легче работать с ВП. Что, если вам потребуется модифицировать те графические спагетти, которые вы составили наспех в прошлом году?

Ниже приводится набор рекомендаций для создания «крутого» графического интерфейса пользователя. Они, естественно, не претендуют на повсеместность использования, а могут служить лишь отправной точкой.

Расположение панели

- если возможно, сделайте размер лицевой панели и окна такими, чтобы был заполнен весь экран (ведь вам известен размер монитора и его разрешающая способность);
- старайтесь пользоваться выровненными прямоугольными декорациями в качестве «модулей» на лицевой панели, которые в свою очередь аккуратно объедините в группы;
- если есть свободное пространство, заполните его декоративным «модулем», изображающим, что вы воспользуетесь этим местом позднее, либо заполните его логотипом вашей компании или каким-нибудь рисунком;
- дайте имена декоративным «модулям».

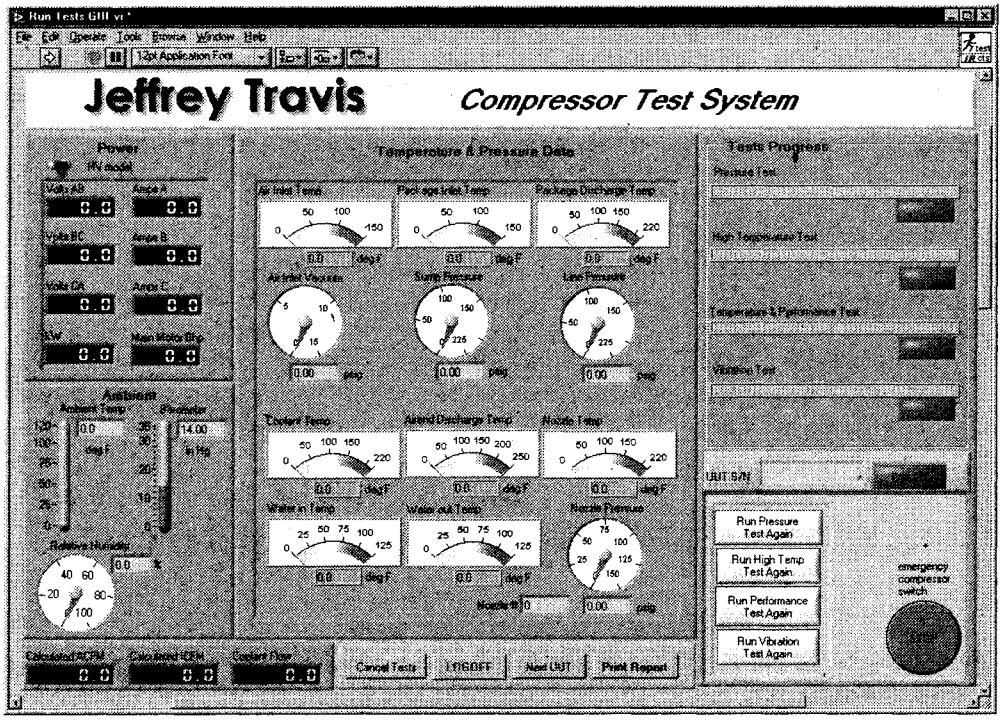


Рис. 16.19

Текст, шрифт и цвет

- используйте цвет, но не переусердствуйте и будьте *последовательны* в выборе цвета. Если вы применяете красный для сигнала тревоги на одной панели, то и на других панелях этот сигнал должен быть красного цвета (и, пожалуйста, никакого зеленого!);
- цвет заднего фона должен отличаться от раскраски окон декораций. Зачастую темно-серый или черный цвет наиболее удачен для заднего фона. И наоборот, если вы не пользуетесь «модулями» декораций, то белый или мягкий светлый цвет будет весьма подходящим для заднего фона;
- выберите опцию **прозрачный** (transparent) для ярлыков из меню **Инструменты** ⇒ **Опции** ⇒ **Лицевая панель**. Ярлыки выглядят намного лучше без их серого окна;

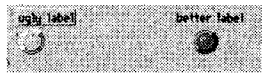


Рис. 16.21

- для многих ярлыков и особенно для числовых элементов управления и индикаторов окрашивание заднего фона текста в черный цвет, а самого текста в зеленый или желтый, делает их хорошо заметными. Применение жирного шрифта также делает текст более четким;

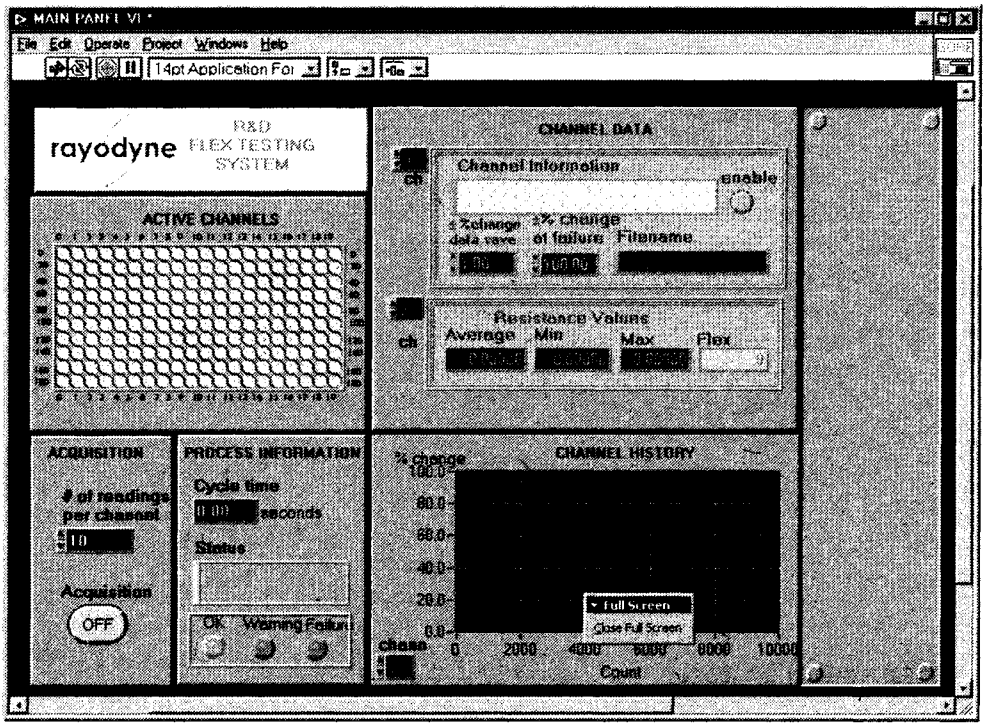


Рис. 16.20

- используйте яркие цвета только для тех элементов, которые являются очень важными, например для сигнала тревоги или сигнала появления ошибки;
- в LabVIEW имеется три стандартных шрифта: *программный* (Application), *системный* (System) и *диалоговый* (Dialog). Их следует придерживаться при написании кросс-платформенной программы. Выбрав свой стиль или размер шрифта на компьютере, работающем под Windows, вы рискуете получить совсем другой (обычно слишком большой или ма́ленький) шрифт в MacOS.

Графики и импортированные рисунки

- сделайте так, чтобы все знали: лицевая панель ВП действительно создана вами – введите в нее логотип вашей компании;
- помните, что рисунки и графические элементы отображения могут в значительной степени замедлить процесс обновления ВП, особенно если вы разместили какие-либо элементы управления и отображения в верхней части рисунка или графического элемента отображения;
- чтобы избежать раздражающего мелькания, которое возникает при построении графиков и других затрагивающих изображения операциях,

выберите опцию **Использовать плавное обновление** (Use Smooth Updates) из меню **Опции**.

Другие рекомендации

- при необходимости создавайте ярлыки единиц измерения соответствующих числовых элементов управления. Если вы захотите ими воспользоваться, помните, что LabVIEW имеет встроенные единицы. Сделать видимым ярлык единиц числового элемента управления или индикатора можно, щелкнув по нему правой кнопкой мыши и выбрав опцию **Видимые элементы ⇒ Ярлык единиц измерения** (Unit Label) – см. главу 13;
- когда возможно, изменяйте элементы управления для создания лучшего визуального эффекта;
- если вы хотите подойти к делу творчески, то для создания аудиовизуальных эффектов воспользуйтесь узлами звуковых данных и узлами свойств, таких как расположение;
- используйте узлы свойств для сокрытия элементов управления, если вы не хотите, чтобы ими кто-то пользовался;
- опции **Свойства ВП**, рассмотренные в главе 13, дают возможность по умолчанию развернуть лицевую панель на весь экран, центрировать ее и т.п.;
- не забывайте работать с Редактором иконок;
- если увидите хороший дизайн лицевой панели, попытайтесь его имитировать.

Еще несколько слов о графическом интерфейсе пользователя

Большинству из вас когда-либо приходилось выступать перед руководством или другой аудиторией с докладом о работе вашего программного обеспечения. Успех такого мероприятия обычно зависит от суеты вокруг интерфейса, а не от эффективности и работоспособности программного обеспечения. Как правило, именно таким образом принимаются решения о запуске проекта. Рассказывают, что в одной компании по производству полупроводников несколько инженеров решили провести ревизию части программного обеспечения, написанного на C, на создание которого ушло два года. Другой инженер сделал набросок лицевой панели в LabVIEW, которая показывала, как должен выглядеть интерфейс нового программного продукта. Один из ведущих руководителей компании был настолько поражен интерфейсом, что приказал внедрить его во все принадлежащие компании предприятия – хотя это был всего лишь схематический набросок.

Мораль этой истории такова: потратьте некоторое время на создание лучшего, «продвинутого» графического интерфейса. Никогда не знаешь, как далеко это может завести.

16.7. Как что-либо сделать в LabVIEW?

В этом разделе речь пойдет больше о решениях в области программирования, а не о характерных особенностях LabVIEW. Во время создания программ возникают общеизвестные проблемы, и мы покажем некоторые из них вместе с решениями. Одни решения являются очень простыми, другие же относятся к разряду оригинальных. Мы не будем говорить о проблемах, связанных с программированием; все примеры собраны из различных источников для того, чтобы подтолкнуть вас к созданию некоторых программ или, по крайней мере, дать пищу для размышлений.

Вы можете отнестись к любой из этих проблем как к вызову: попытайтесь сами найти решение задачи, прежде чем заглянуть в ответ!

Как получить данные из каналов с различными частотами выборок?

Все идет хорошо, если вы делаете выборки данных из всех каналов с одной частотой. И совсем другое дело, когда сбор данных из одного канала осуществляется на частоте 10 кГц, а из другого – на частоте 50 Гц. Не существует какого-либо магического решения данной проблемы, но мы можем предложить пару вариантов:

- если у вас имеется несколько плат ввода/вывода, попытайтесь распределить каналы таким образом, чтобы каждая плата обрабатывала каналы на одной и той же частоте выборки;
- делайте сбор данных на самой высокой частоте, а затем выбросьте лишние точки данных низкочастотных каналов. Наиболее эффективно эта операция осуществляется с помощью функции **Опустошение массива** (Decimate Array) из палитры **Массив**.

Как создать набор логических элементов управления, чтобы только один имел значение ИСТИНА в любое время?

Это известная всем задача о «кнопках радиоприемника». Решение проблемы весьма важно для многих программ. Формулировка задачи звучит примерно так: вы хотите предоставить пользователю список опций (логических элементов управления), каждая из которых будет соответствовать определенной подпрограмме. Например, есть лицевая панель тестовой программы, которая позволяет пользователю провести тест, отредактировать параметры теста, посмотреть результаты

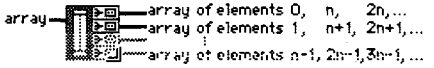


*Рис. 16.22. Функция **Опустошение массива**. Разделяет входные элементы на выходные массивы, как крупье раздает карты. Вход должен быть одномерным. Функция может менять размер*

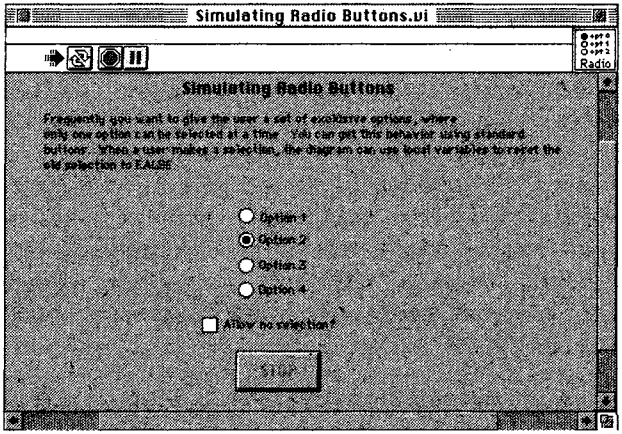


Рис. 16.23

и выйти из программы. Единовременно пользователь может выбирать лишь одну из этих опций. Как же наиболее эффективно сделать это? Естественно, нежелательны четыре параллельных цикла по условию или какая-либо сложная логическая блок-диаграмма. К счастью, в LabVIEW содержится одно решение данной проблемы. Его можно найти в библиотеке `examples\general\controls\booleans.llb` под именем **Simulating Radio Buttons.vi**.

Основой этого ВП является ВПП **Manage Radio Buttons**, который находится в той же самой библиотеке полной версии LabVIEW. Чтобы вы могли понять, как он работает, на рис. 16.25 приведена его блок-диаграмма. Мы рекомендуем использовать данный виртуальный прибор вместо того, чтобы изобретать колесо.

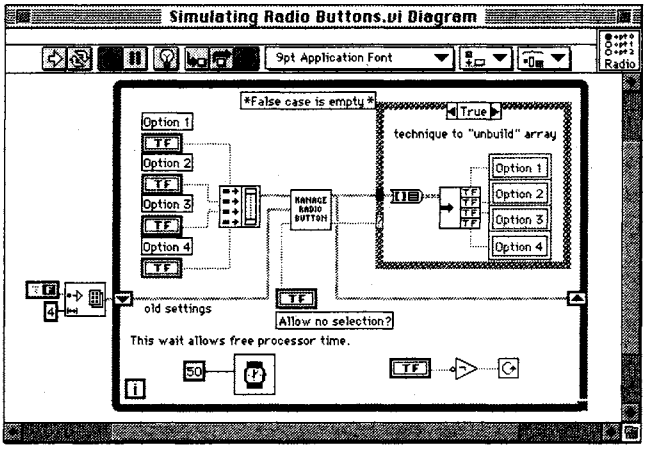


Рис. 16.24

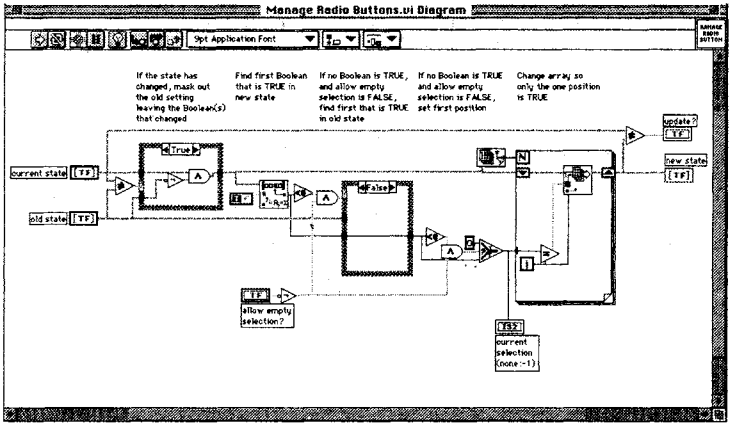


Рис. 16.25

Как программно очистить развертку осциллограммы?

Подключите пустой массив к атрибуту развертки **История данных** (History Data). Тип данных этого пустого массива точно такой же, что и тип данных, поданных на развертку. Существует прекрасный пример, который иллюстрирует эту методику. Вы

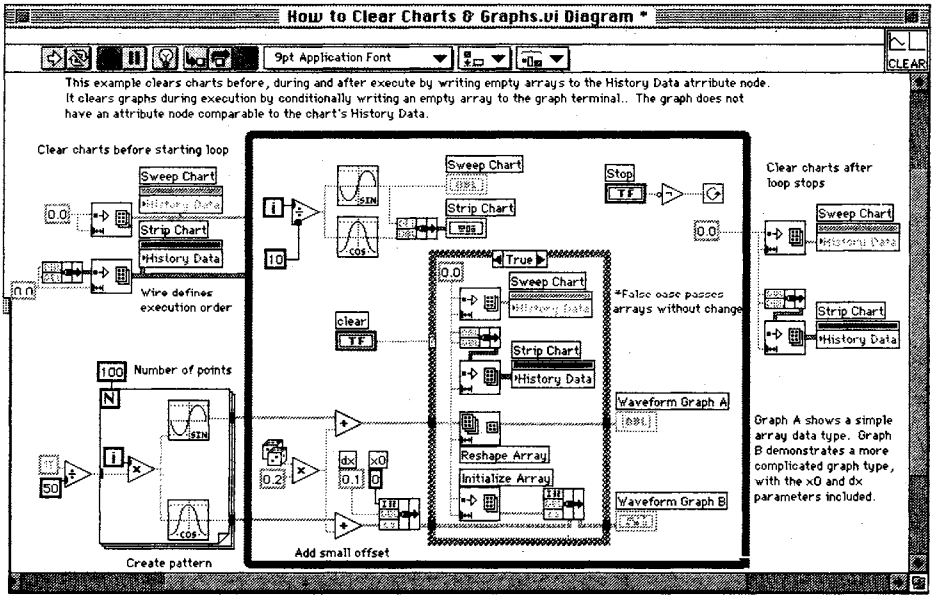


Рис. 16.26

найдете его в директории `Examples\General\Graphs\Charts.llb\How to Clear Charts and Graphs.vi`.

Вы также можете вызвать контекстное меню развертки на лицевой панели и выбрать опцию **Очистить развертку**.

Как отобразить кривую водопада?

Центральной идеей, лежащей в основе создания кривой водопада или последовательности кривых, располагающихся одна над другой во времени, является построение многолучевого графика, где распределение вдоль оси Z создается путем прибавления константы смещения к каждой кривой. Поищите примеры на сайте http://zone.ni.com.

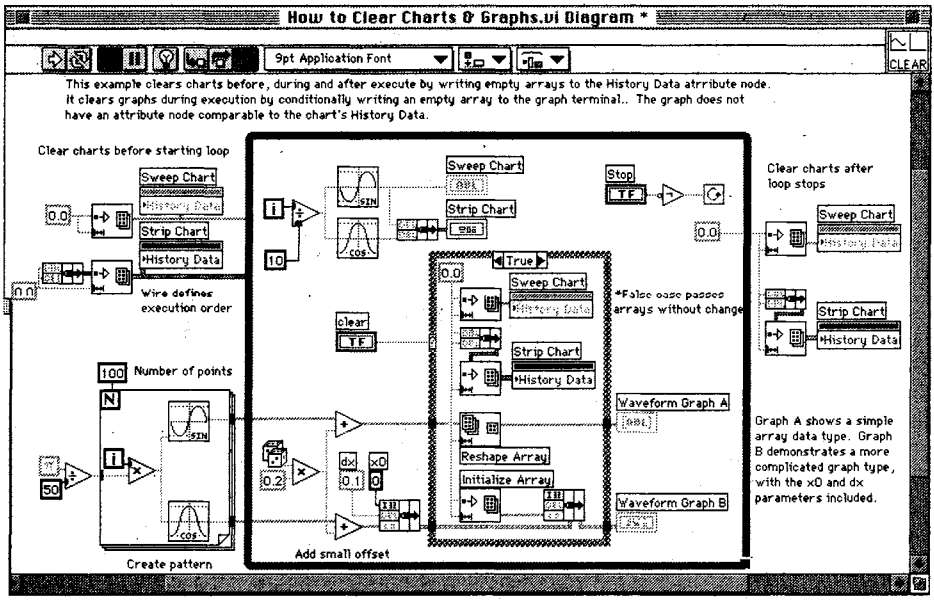


Рис. 16.27

Существует ли способ ввести несколько осей Y на график для двух и более кривых?

Да. Вызовите контекстное меню графика и выберите опцию **Видимые элементы** ⇒ **Панель редактирования шкалы**. Затем раздвиньте панель редактирования, чтобы добавить больше осей Y.

Мои графики мерцают при обновлении. Это раздражает. Существует ли способ устранить мерцание?

К счастью, да. В меню **Инструменты** выберите **Опции**. Затем укажите элемент **Лицевая панель**. Поставьте галочку напротив надписи **Осуществлять плавное**

обновление во время прорисовки (Use smooth updates during drawing). После этого мерцание пропадет, однако потребуется больше памяти.

Существует ли какой-либо способ войти в любой заданный кадр структуры последовательности?

Это довольно интересная и сложная задача, но ее решение дарит нам мощную структуру программирования.

Проблема может быть изложена следующим образом. Предположим, у вас есть структура последовательности, которая выполняет некоторые операции в заданной последовательности. Предположим также, что нужно «перескочить» через кадр, если определенное логическое значение становится ИСТИНА. «Нет проблем, – скажете вы. – Введите оператор варианта». Но давайте представим, что этот процесс становится более сложным. Например, вам необходимо снова перейти из кадра 4 в кадр 2 при заданных условиях. То есть нужна структура, которая могла бы находиться в определенном количестве *состояний*, установленных алгоритмом. Всякий раз, когда у вас имеется цепь событий, в которой каждая операция зависит от предыдущей, такая структура наилучшим образом выполнит эту работу.

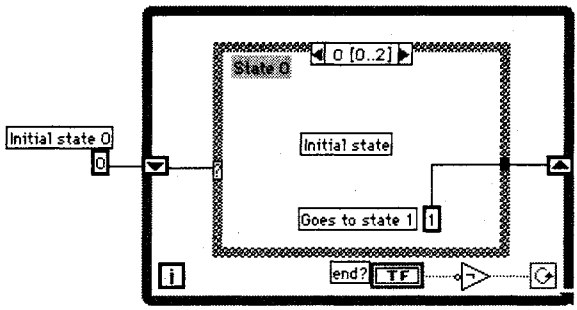


Рис. 16.28

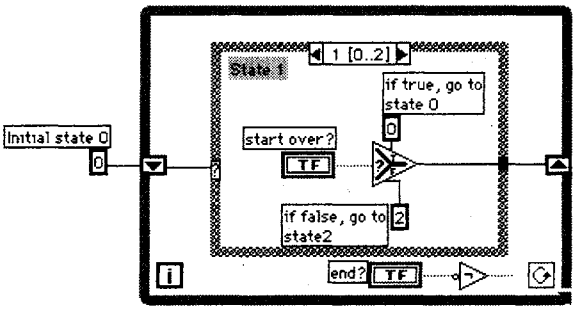


Рис. 16.29

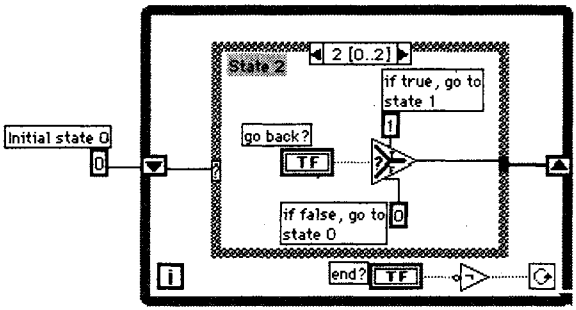


Рис. 16.30

Несмотря на то что в LabVIEW нет встроенной структуры механизма состояний, вы можете легко создать ее сами, используя структуру варианта внутри цикла по условию и введя несколько сдвиговых регистров.

С каждой итерацией кадр структуры варианта может переносить управление в любой другой кадр в соответствии с алгоритмом или заставить цикл по условию прекратить выполнение. Размещая каждое состояние в отдельном варианте, вы будете «перескакивать» из одного состояния в другое в произвольном порядке.

Как создать что-то вроде инструментальной панели, которая вызывала бы различные наборы элементов управления и отображения в одном и том же окне?

Это довольно искусный трюк. Идея заключается в создании кластеров элементов управления и индикаторов, по одному для каждой кнопки инструментальной панели. Кластеры должны иметь один и тот же размер, а вы в буквальном смысле слова накладываете их друг на друга. Используя узлы свойств, вы превращаете все кластеры в невидимые за исключением одного, выбранного вами. Кнопки инструментальной панели должны быть настроены таким образом, чтобы единовременно срабатывала только одна.

Конечно, неплохо, что лицевая панель требует много элементов управления и отображения, но вам будет затруднительно показать их все одновременно. Кнопки же инструментальной панели действуют в качестве меню, которое содержит логически сгруппированный набор объектов.

Можно ли получить доступ к параллельному порту из LabVIEW?

Да. Используйте ВП, работающие с последовательными портами (это ничего, что параллельный порт является противоположным последовательному). В LabVIEW под Windows порт 10 означает LPT1, порт 11 – LPT2 и т.д., что аналогично использованию порта 0 для COM1, порта 1 для COM2 и т.д. Вы даже можете вывести данные на принтер, подключенный к параллельному порту, с помощью функции **Записать в последовательный порт** (Serial Port Write). При этом вам

понадобятся специальные символы управления принтером. Другим отличным способом применения параллельного порта является цифровой ввод/вывод без съемной платы: вы получите бесплатно восемь цифровых линий (при этом не забудьте о буферной схеме для защиты компьютера).

Можно ли осуществлять объектно-ориентированное программирование в LabVIEW?

Да, в некоторой степени. Если вы знакомы с концепциями объектно-ориентированного программирования, попробуйте отыскать соответствующие ВП на сайте http://ni.com/goop.

Есть ли способ превратить ВП в отдельно исполняемые программы так чтобы люди могли запускать их без LabVIEW?

Да. В этом случае вам понадобится либо профессиональная версия LabVIEW, либо набор инструментов Application Builder Toolkit, приобретаемый отдельно.

Как управлять ВП через Internet или отслеживать их?

Если вам необходимо отслеживать ВП, воспользуйтесь встроенным Web-сервером. Для управления ВП служит программа LavVNC, находящаяся на компакт-диске. Подробнее об этом рассказывалось в главе 14.

Может ли LabVIEW приготовить мне обед?

Пока еще нет.

16.8. Память, производительность и тому подобное

Ваш профессиональный рост в качестве программиста LabVIEW означает, что вы знаете, как создавать программы, использующие мало ресурсов памяти и процессора. Конечно, при работе на компьютере с двумя процессорами и огромным количеством оперативной памяти вам нет нужды беспокоиться о такого рода проблемах. Но на компьютерах среднего класса программы реального времени будут работать лучше, если вы возьмете на вооружение некоторые рекомендации, позволяющие увеличить производительность и снизить потребление памяти. Программисты, которые никогда не принимают в расчет использование памяти и загрузку процессора, не являются профессионалами. Никому не нравится отлаживать небрежно созданные программы.

16.8.1. Лечение амнезии и лености

Посмотрим на вещи прямо: LabVIEW имеет тенденцию заставлять программы «пожирать» память, но вы можете изменить ситуацию к лучшему, воспользовавшись некоторыми советами. Управление памятью – это, конечно, сложная тема,

но она становится актуальной при использовании больших массивов и/или проблемах синхронизации. Ниже приводится несколько советов:

- вы уверены, что применяете подходящие типы данных? Числа с плавающей запятой повышенной точности хороши в тех случаях, когда требуется высокая точность. Но они занимают лишнюю память, если используется низший тип данных. Этот фактор становится важным при работе с большими массивами;
- глобальные переменные занимают много памяти. Уменьшите не только их количество, но и число раз их считывания и записи;
- не используйте сложные иерархические типы данных (такие, как массив кластеров массивов);

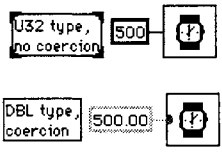


Рис. 16.31

- избегайте ненужного приведения типов данных (серые точки на вводах). Приведение показывает, что ожидаемый тип данных отличается от типа данных, поданных на ввод. Несмотря на то что LabVIEW может принимать практически любые типы данных (полиморфизм), результатом всегда является потеря скорости и увеличение затрат памяти, поскольку необходимо делать копии данных. Это особенно актуально для больших массивов;
- как вы обрабатываете массивы и строки? Используете ли вы циклы там, где не нужно? Иногда имеется встроенная функция, которая выполняет эту работу, или объединяются несколько циклов. Избегайте введения ненужных элементов в цикл, как это показано на рис. 16.32;

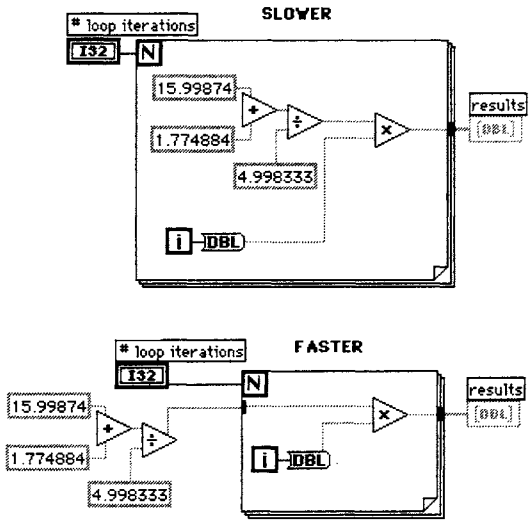


Рис. 16.32

- там, где возможно избегайте использования функции **Создать массив** (Build Array) внутри циклов, предотвращая, таким образом, повторные вызовы менеджера памяти. Всякий раз, когда вы вызываете функцию **Создать массив**, в памяти резервируется новое пространство для нового массива. Используйте вместо этого автоиндексирование или функцию **Заменить элемент массива** (Replace Array Element) с заранее установленным размером массива. Те же самые проблемы возникают с функцией **Объединить строки**;
  По всей видимости, это не выход из положения, если вы хотите отобразить данные массива в реальном времени в момент его создания. В таком случае сначала установите максимальный размер массива, а затем используйте функцию **Заменить подмножество массива** (Replace Array Subset) вместо функции **Создать массив**;
- сведите к минимуму применение графики (особенно графиков и разверток осциллограмм), когда скорость выполнения является самым важным фактором;
- когда возможно, обновляйте элементы управления и индикаторы за пределами циклов, то есть тогда, когда не столь важно видеть значение объекта до момента окончания выполнения цикла.

16.8.2. Декларация Независимости

LabVIEW выполняет удивительную работу по переносу ВП между различными платформами. В большинстве случаев вы можете взять ВП, созданный в Sun, и запустить его в системах MacOS или Windows 95 (если они используют ту же самую версию LabVIEW). Однако не все части блок-диаграммы являются совместимыми. Если нужно создать ВП, не зависящие от платформы, обратите внимание на несколько советов относительно совместимости:

- помните, что *сама программа* LabVIEW и все, что в нее входит (например, все ВП из директории `vi.lib`), не являются совместимыми. Все, что делает LabVIEW под каждой ОС, заключается в *конвертации* ВП

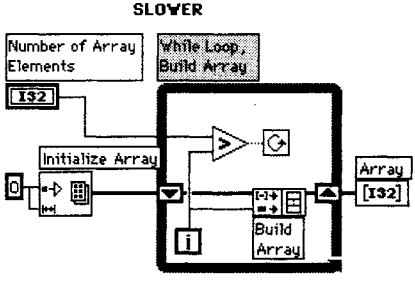


Рис. 16.33

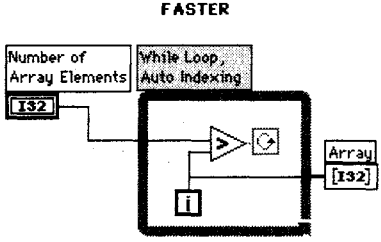


Рис. 16.34

из других платформ в ВП для своей платформы при помощи внутреннего кода;

- некоторые ВП из палитры **Дополнительно** (такие, как **System Exec** в LabVIEW для Windows), а также ВП из палитры **Коммуникация** (типа AppleEvents или ActiveX) являются специфическими для данных систем и поэтому не обладают совместимостью;
- ВП, содержащие узел взаимодействия с программным кодом, не являются совместимыми до тех пор, пока вы не сделаете код источника независимым от платформы и не перекомпилируете его под новую операционную систему;
- помните об именах файлов, которые имеют свои специфические правила для каждой ОС. Не используйте такие символы, как скобки, двоеточие и т.п., являющиеся разграничителями путей в разных ОС;
- символ конца строки (EOL) неодинаков на разных платформах (`\r` для MacOS, `\r\n` для Windows и `\n` для Sun). Наиболее легким решением этой проблемы является использование константы **Конец строки** (End of Line) из палитры **Строки**;
- шрифты могут внести серьезную путаницу при работе под разными системами. По возможности придерживайтесь трех стандартных шрифтов LabVIEW (программного, системного и диалогового);
- разрешающая способность экрана также является немаловажным фактором при работе с разными системами. Некоторые программисты рекомендуют, чтобы она была 800×600.

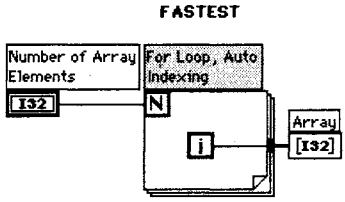


Рис. 16.35

Если вы хотите знать, какой объем памяти использует ваш ВП, воспользуйтесь инструментом **Сведения о ВП** (Profile VIs) из меню **Инструменты ⇒ Дополнительно**.

16.9. Искусство программирования

Программирование действительно является искусством, и оно может приносить удовольствие в LabVIEW. Этот раздел состоит из набора рекомендаций для создания действительно хороших программ. Найти некоторые из этих концепций можно в книге «The LabVIEW Guide: A Guide to Better LabVIEW Applications», написанной Гэри Джонсон и Мег Кей, а также на страницах Internet (http://www.ni.com).

16.9.1. Модулирование и испытание ваших ВП

Хотя теоретически существует возможность использования модульности при создании программ, на деле это происходит довольно редко. Старайтесь создавать простые ВПП: так вам удастся протестировать каждую отдельную часть кода, прежде чем обращаться с ним как с целым. Кроме того, вы будете работать более

организованно, не испытывая трудностей при изменении размера блок-диаграммы. Не забудьте протестировать каждый ВПП как высокоуровневый ВП: проверяйте все виды входных комбинаций. Если вы уверены, что все ВПП работают, вам легко будет устранить любые проблемы, связанные с высокоуровневым ВП.

Часто программисты LabVIEW не тестируют определенные ВП, требующие оборудования ввода/вывода или каких-нибудь внешних устройств. Не ждите, пока у вас появится такое оборудование. Напишите простейший генератор данных, чтобы проверить, по крайней мере, функциональность части вашего ВП.

16.9.2. Документирование в процессе работы

Пожалуйста, всегда документируйте вашу работу. Многие считают, что это можно сделать позднее или вообще не делать. Но спустя некоторое время потребитель или вы сами попытаетесь вспомнить, как функционирует этот ВП. LabVIEW имеет некоторые встроенные функции для документирования работы.

- **Документация ВП**. Составьте краткое описание каждого создаваемого вами виртуального прибора. Это тем более важно для человека, который смотрит на иконки ВПП на блок-диаграмме и хочет узнать их назначение;
- **Описание**. В идеале вам следует написать подсказку для каждого элемента управления и отображения, используя контекстную команду **Описание** (Description). Тогда в окне контекстной помощи появится необходимая информация, которой интересуется пользователь;
- **История создания ВП**. Эта опция из меню **Окно** (Windows) является более сложным инструментом, используемым в больших проектах. Она дает возможность ввести комментарии о сделанных изменениях в ВП во время создания. Опция нужна главным образом тогда, когда над проектом работает несколько человек, поскольку в ней отслеживается имя пользователя, время и дата;
- **Текст лицевой панели**. Для сообщения важной информации просто напишите текст (жирным шрифтом или большими буквами) на самой лицевой панели. Пользователи всегда его увидят.

16.9.3. Еще раз о потоке данных

Так как вы достаточно хорошо познакомились с LabVIEW, вы начнете пользоваться всеми его преимуществами в плане программирования потока данных. Запомните несколько советов:

- поток данных означает перенос данных по проводникам. Когда данные достигнут *всех* входов ВПП или функции, этот ВПП или функция начнет выполняться;

- два или более объекта, или группа объектов на блок-диаграмме, которые не соединены между собой, не имеют определенной последовательности выполнения. Люди, не знакомые с LabVIEW, считают, что выполнение должно осуществляться слева направо или сверху вниз. Это неверно. Нельзя предсказать, в каком порядке выполнятся два или более сегмента блок-диаграммы, если это не установлено потоком данных;
- при необходимости «заставить» ВПП выполняться в определенной последовательности воспользуйтесь структурой последовательности, которая может быть неуместной для больших блок-диаграмм, или искусственной структурой зависимости данных, как показано на рис. 16.36;

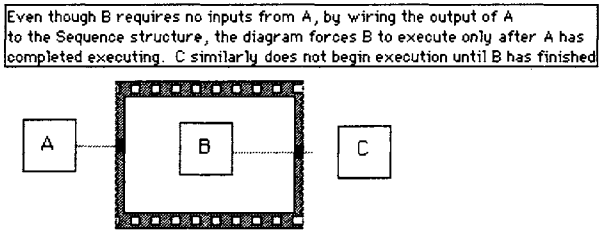


Рис. 16.36

- вы наверняка заметили, что многие функции LabVIEW имеют одинаковые выводы: ссылки файлов, идентификаторы задач, кластеры ошибок и т.д. Они созданы для связывания нескольких общеупотребительных ВП в естественную последовательность потока данных. Вы также можете ими пользоваться в своих виртуальных приборах – это снизит необходимость в структурах последовательности, которые имеют неуклюжие локальные переменные.

16.10. Итоги

В этой главе вы получили некоторые навыки графического программирования. Вначале мы рассматривали вид лицевой панели: причины, предложения и указания по созданию удобного графического интерфейса пользователя (GUI). Затем мы сконцентрировали внимание на блок-диаграмме: тонкости программирования, производительность, память и стиль.

Создание хорошего графического интерфейса пользователя является важным фактором для «продажи» вашей программы покупателю или руководителю. Для этого в LabVIEW имеются декоративные модули, команды выравнивания и рассредоточения объектов и их наложения друг на друга.

Собственноручно созданные элементы управления и отображения добавляют цену интерфейсу вашего ВП, обеспечивая инструментами моделирования и анимации. Редактор элементов управления дает возможность модифицировать стандартные элементы управления и индикаторы LabVIEW и импортировать файлы рисунков для изображения новых объектов.

Окно помощи предназначено не для вас, программиста, а для конечного потребителя, который будет использовать это окно для изучения описаний объектов лицевой панели. Вы можете программно открывать и закрывать данное окно.

В процессе программирования в LabVIEW всегда возникает много вопросов и проблем. Некоторые их решения были предложены выше.

При необходимости увеличения скорости выполнения ВП и оптимизации памяти вам следует руководствоваться указаниями, приведенными в данной главе. Избегайте ненужных операций внутри циклов, особенно при работе с массивами.

Несмотря на то что виртуальные приборы LabVIEW не зависят от платформы, вы можете столкнуться с некоторыми проблемами в плане совместимости, например при использовании функций, ориентированных на определенную ОС.

Наконец, чтобы быть хорошим программистом в LabVIEW, вы должны быть последовательны при создании модульных ВП и тщательно их тестировать. Необходима хорошая документация для создания качественного и управляемого программного обеспечения. Научитесь пользоваться потоком данных – отличительным признаком LabVIEW, который неизвестен в других языках программирования.

16.11. Заключительные замечания

Ну вот мы и подошли к концу книги! Теперь вы хорошо понимаете, как работает LabVIEW и как он может работать на вас, будь вы преподавателем электротехники или программистом систем управления для большого предприятия.

Как же обучаться дальше? Прежде всего – на собственном опыте. Это главный учитель. Экспериментируйте с ВП. Создавайте прототипы. Изучайте примеры. Будьте сообразительны. Ну и, конечно, вам понадобится удача. Если вы решите приобрести полную версию LabVIEW, изучите сопутствующие учебники и примеры. Они расскажут вам обо всех тонкостях создания программ. Ну и, наконец, не забудьте посетить сайт автора http://jeffreytravis.com, а также сайты: http://ni.com/russia, www.labview.ru и www.dmk.ru.

Всего доброго!

ПРИЛОЖЕНИЕ: РЕСУРСЫ LabVIEW

Имеется много вариантов получения помощи и дополнительной информации о LabVIEW и виртуальных приборах. Ниже приводится перечень сайтов Internet, организаций, публикаций и других источников для пользователей LabVIEW.

Документация LabVIEW и помощь в режиме online

Компьютерные программы и печатные материалы содержат большое количество информации о LabVIEW. В них вы можете отыскать ответы практически на все вопросы. Помощь в режиме online необходима во время создания программ.

Мастерская Джеффри Трэвиса

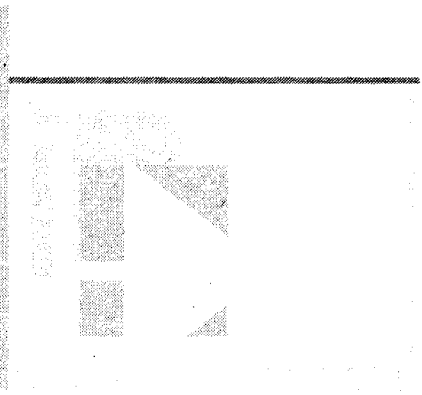
Если у вас имеются вопросы, на которые вы не можете найти ответы, обратитесь к автору этой книги. Я могу проконсультировать в области создания программного обеспечения, обучения и т.д. Пишите мне по электронной почте: consulting@jeffreytravis.com.

National Instruments

Это компания, которая создает и продает LabVIEW. Также она предлагает широкую техническую поддержку по телефону, электронной почте и посредством Internet. Вы можете связаться с представительством компании National Instruments следующим образом:

Тел. (095) 783-68-51, 783-68-52
Internet: http://www.labview.ru,
http://www.ni.com/russia
Электронная почта: ni.russia@ni.com.

ГЛОССАРИЙ



∞
Бесконечность
π
Число «пи»
Δ
Дельта; разница. **Δ**x обозначает величину, на которую меняется значение x при изменении индекса на единицу.

A

Absolute Path (Абсолютный путь)
Путь к файлу или директории, описывающий положение относительно верхнего уровня файловой системы.

Active Window (Активное окно)
Окно ОС Windows, которое в текущий момент готово для принятия данных пользователя. Обычно это окно на переднем плане Рабочего стола. Строка заголовка активного окна выделена. Окно можно сделать активным, выделив его кнопкой мыши или выбрав из меню Windows.

A/D
Аналого-цифровое преобразование. Понятие восходит к работе электронных схем. Обозначает преобразование реального аналогового сигнала в цифровую форму (набор битов) для дальнейшей обработки на компьютере.

ADC (АЦП)
См. *A/D*.

ANSI
Национальный институт стандартизации США.

Array (Массив)

Упорядоченный, проиндексированный набор элементов данных одного типа.

Array Shell (Шаблон массива)

Объект лицевой панели, обозначающий место, которое отведено под массив. Он состоит из элемента управления/отображения индексом массива, окна объекта данных и – реже – ярлыка. Может принимать различные типы данных.

Artificial Data Dependency (Искусственная зависимость данных)

Относится к языкам программирования потока данных. Это условие, при котором само событие прихода данных в большей степени, чем их величина, обусловливает выполнение кода какого-либо узла.

ASCII

Американский стандартный код обмена информацией. Термин относится к 7-битной схеме кодировки букв и чисел.

Asynchronous Execution (Асинхронное выполнение)

Режим работы, при котором время работы процессора распределяется между несколькими процессами.

Auto-Indexing (Автоматическая индексация)

Способность циклических структур формировать и расформировывать массивы на их границах. При входе массива в цикл с включенной автоиндексацией цикл автоматически расформировывает одномерный массив на скалярные величины, двумерный массив на одномерные массивы и т.д. Циклы формируют данные в массивы на своих границах в соответствии с обратной процедурой.

Autoscaling (Автоматическая регулировка масштабов)

Способность осей графика подстраивать свои масштабы для отображения всего диапазона данных. На осях графика при этом отображаются максимальное и минимальное значения величины.

Autosizing (Автоматическая регулировка размеров)

Автоматическое изменение размеров ярлыков для отображения всего введенного вами текста.

B

Block Diagram (Блок-диаграмма)

Графическое описание или представление программы или алгоритма. В LabVIEW блок-диаграмма, состоящая из исполняемых иконок (узлов) и проводников данных (они передают данные между узлами), содержит исходный код ВП. Блок-диаграмма находится в окне блок-диаграмм ВП.

Boolean Controls (Логические элементы управления)

Объекты лицевой панели, используемые для управления или индикации входных и выходных логических (ИСТИНА или ЛОЖЬ) данных. Существует целый ряд стилей для этих элементов: переключатели, кнопки и светодиоды.

Breakpoint (Точка останова)

Пауза в выполнении программы. Установить точку останова можно на ВП, узле или проводнике с помощью соответствующего инструмента из палитры инструментов.

Breakpoint Tool (Инструмент установки точки останова)

Инструмент, применяемый для установки точки останова на ВП, узле или проводнике.

Broken VI (Неисправный ВП)

ВП, который не может быть скомпилирован или запущен; опознается по сломанной стрелке кнопки запуска ВП.

Bundle Node (Узел объединения)

Функция, создающая кластер из элементов различного типа.

Byte Stream File (Двоичные файлы)

Файл, хранящий данные в виде последовательности символов ASCII или байтов.

C

Case (Вариант)

Одна из поддиаграмм структуры варианта.

Case Structure (Структура варианта)

Условная структура управления с ветвлением, которая выполняет один и только один из ее вариантов в зависимости от состояния входа. Ее можно описать, как комбинацию операторов `IF`, `THEN`, `ELSE` и `CASE` в обычных языках программирования.

Channel (Канал)

Вывод или контакт, с которого (или на который) считывается (или поступает) аналоговый сигнал.

Chart (Развертка)

См. *Панорамная развертка* (strip chart), *Временная развертка* (scope chart) и *Временная развертка с маркером* (sweep chart).

CIN (УКИ)

См. *Узел кодового интерфейса* (Code Interface Node).

Cloning (Копирование)

Для копирования элемента управления или некоторых других объектов LabVIEW щелкните по нему левой кнопкой мыши, одновременно удерживая нажатой клавишу <ctrl> (в Windows, или <option> в Macintosh, <meta> в Sun, <alt> в Linux и перенесите копию в любое место.
Работая под ОС Sun или Linux, вы можете скопировать объект, нажимая среднюю кнопку мыши, а затем перетаскивая копию в новое место.

Cluster (Кластер)

Упорядоченный набор неиндексированных элементов данных любого типа, включая числовые, логические, строковые, массивы или кластеры. Все элементы кластера должны быть либо элементами управления, либо индикаторами.

Cluster Shell (Шаблон кластера)

Объект лицевой панели, содержащий элементы кластера.

Code Interface Node (CIN, Узел кодового интерфейса)

Специальный узел блок-диаграммы, посредством которого можно передать традиционный текстовый код в ВП.

Coercion (Приведение типов)

Автоматическое приведение типов LabVIEW изменяет числовое представление элементов данных.

Coercion Dot (Точка приведения типов)

Специальный знак на узле или терминале, свидетельствующий о том, что в этой точке произошло изменение числового представления данных.

Color Tool (Инструмент раскрашивания)

Инструмент, необходимый для выбора цвета переднего и заднего плана.

Color Copy Tool (Инструмент копирования цвета)

Копирует цвета для дальнейшего использования инструментом раскрашивания.

Compile (Компиляция)

Процесс преобразования высокоуровневого кода в машинный код. LabVIEW автоматически компилирует ВП перед их первым запуском или запуском после изменения.

Conditional Terminal (Терминал условия выхода)

Терминал цикла по условию, работающий с логическими значениями и определяющий, будет ли ВП выполнять следующую итерацию.

Connector (Поле ввода/вывода)

Часть ВП или функционального узла, содержащая все входные и выходные терминалы, посредством которых данные поступают на узел либо выходят из узла.

Connector Pane (Соединительная панель)

Область в правом верхнем углу окна лицевой панели, отображающая схему подключений к ВП. Находится за иконкой.

Constant (Постоянная)

См. *Универсальная постоянная* и *Константа, заданная пользователем.*

Continuous Run (Непрерывный запуск)

Режим работы, при котором ВП выполняется непрерывно, повторяясь до тех пор, пока пользователь не остановит его. Режим включается нажатием кнопки непрерывного запуска.

Control (Элемент управления)

Объект лицевой панели для ввода данных интерактивно в ВП или автоматически в ВПП.

Control Flow (Управление потоком)

Система программирования, в которой последовательный порядок инструкций определяет порядок выполнения программы. Наиболее распространенные текстовые языки программирования, такие как C, Pascal и Basic, являются языками подобного типа.

Control Palette (Палитра элементов управления/индикаторов)

Палитра, содержащая элементы управления и индикаторы.

Conversion (Преобразование)

Изменение типа элемента данных.

Count Terminal (Терминал количества итераций)

Терминал цикла с фиксированным числом итераций, чье значение определяет число выполнений содержимого цикла.

CPU (ЦПУ)

Центральный процессор

Current VI (Текущий ВП)

ВП, лицевая панель (или блок-диаграмма, или редактор иконки) которого является активным окном.

Custom PICT Controls (Настраиваемые элементы лицевой панели)

Элементы управления и индикаторы, чьи составляющие части можно заменить по желанию пользователя.

D

D/A (ЦАП)

Цифро-аналоговое преобразование. Противоположно термину АЦП (A/D).

Data Acquisition (DAQ, Сбор данных)

Процесс сбора данных, который обычно состоит из аналого-цифрового преобразования. Это понятие иногда расширяют, включая генерацию данных (ЦАП).

Data Dependency (Зависимость данных)

Отличительная черта языков программирования потока данных, когда узел не может выполняться до тех пор, пока не получит данные от другого узла. См. также *Искусственная зависимость данных.*

Data Logging (Протоколирование данных)

Чаще всего обозначает сбор данных и одновременное сохранение их в файл на диск. Функция LabVIEW **Ввод/вывод файла** может протоколировать данные.

Data Storage Formats (Форматы хранения данных)

Расположе́ние и представление данных, сохраненных в памяти.

Data Type Descriptor (Дескриптор типа данных)

Код, идентифицирующий тип данных. Используется при сохранении и представлении информации.

Dataflow (Поток данных)

Система программирования, состоящая из исполняемых узлов. Узлы выполняются только тогда, когда все необходимые данные поступят на их входы. После выполнения автоматически возвращаются выходные данные. LabVIEW – это среда программирования потока данных.

Datalog File (Файл протокола)

Файл, хранящий данные в виде последовательности записей (регистраций) одного произвольного типа данных, который вы определяете при создании файла. Хотя все регистрации в файле протокола должны принадлежать к одному типу, сам тип может быть сложным. Например, каждая запись способна быть кластером, состоящим из строки, числа и массива.

DataSocket

Протокол обмена, поддерживаемый LabVIEW, для совместного использования динамически меняющихся данных в сети.

DC

Постоянный ток. Противоположен термину AC (переменный ток). Применяется в описании низкочастотных сигналов, например меняющихся реже одного раза в секунду.

Device (Устройство)

Встраиваемая многофункциональная плата ввода/вывода.

Device Number (Номер устройства)

Номер, назначенный устройству (плате) в программе конфигурации NI-DAQ.

Description Box (Окно описания)

Гипертекстовая документация объекта LabVIEW.

Destination Terminal

См. *Терминал-приемник* (sink terminal).

Dialog Box (Диалоговое окно)

Интерактивное окно с подсказками, в котором вы определяете дополнительную информацию, необходимую для завершения операции.

Differential Measurement (Дифференциальная схема измерений)

Способ конфигурации устройства для считывания сигналов, когда нет необходимости заземлять входы. Измерение производится между двумя входными каналами.

Dimension (Размерность)

Свойство, описывающее размер и структуру массива.

DMA

Прямой доступ к памяти. Метод, с помощью которого можно передать данные от устройства в ОЗУ (и наоборот), минуя процессор. DMA – наиболее быстрый метод передачи данных в память компьютера и обратно.

Drag (Перетаскивание)

Действие, состоящее в перемещении указателя мыши по экрану с целью выбора, перемещения, копирования или удаления объекта.

E

Empty Array (Пустой массив)

Массив, содержащий ноль элементов, но имеющий определенный тип данных. Например, пустой числовой массив содержит числовой элемент управления, в который не ввели какого-либо значения для всех элементов массива.

EOF

Конец файла. Символ смещения конца файла относительно его начала (то есть размер файла).

Execution Highlighting (Подсветка выполнения)

Функция, позволяющая анимировать выполнение ВП для иллюстрации потока данных в ВП.

F

FFT (БПФ)

Быстрое преобразование Фурье.

File Refnum (Ссылка файла)

Идентификатор, который LabVIEW ассоциирует с файлом при его открытии. Вы используете ссылку файла для того, чтобы указать функции или ВП совершать операцию с открытым файлом.

Flattened Data (Приведенные данные)

Данные любого типа, которые были преобразованы в строку, обычно для последующей записи в файл.

For Loop (Цикл с фиксированным числом итераций)

Циклически повторяющаяся структура, выполняющая свою поддиаграмму заданное количество раз. Эквивалентна обычному коду:

```
For I = 0 to n-1, do ...
```

Formula Node (Узел Формула)

Узел, который вычисляет формулу, введенную в виде текста. Особенно полезен при использовании длинных формул, которые затруднительно собрать в виде блок-диаграмм.

Frame (Кадр)

Поддиаграмма структуры последовательности.

Free Label (Свободный ярлык)

Ярлык на лицевой панели или блок-диаграмме, который не принадлежит какому-либо объекту.

Front Panel (Лицевая панель)

Интерактивный интерфейс пользователя ВП. Смоделирован на основе лицевой панели физических приборов и инструментов и содержит простые и ползунковые переключатели, счетчики, графики, развертки, шкалы, светодиодные индикаторы и другие элементы управления и индикаторы.

Function (Функция)

Встроенный выполняющий элемент, похожий на оператор или функцию в обычных языках программирования.

Function palette (Палитра функций)

Палитра, содержащая структуры, константы, элементы взаимодействия и ВП блок-диаграммы.

G

G

Графический язык программирования LabVIEW.

Global Variable (Глобальная переменная)

ВПП однократного входа с отведенной областью памяти, который использует неинициализированный сдвиговый регистр для хранения данных после одного выполнения для следующего. Область памяти копий этих ВПП открыта для совместного доступа и, следовательно, может применяться для передачи между ними глобальных данных.

Glyph (Глиф)

Маленькая картинка или иконка.

GPIB (General purpose interface bus – канал общего пользования, КОП)

Также известен как HP-IB (Hewlett Packard interface bus) и IEEE 488.2 (Стандарт Института инженеров по электротехнике и электронике 488.2). Стал мировым стандартом для связи практически любого прибора с компьютером. Был разработан компанией Hewlett Packard в 60-х годах для программирования их приборов с компьютера на языке Basic. В настоящее время при участии Института инженеров по электротехнике и электронике (IEEE) определен строгий протокол для этого канала, обеспечивающий согласованность между приборами.

Graph Control (График)

Объект лицевой панели, отображающий данные в декартовой системе координат.

Ground (Заземление, «земля»)

Опорное напряжение в системе. «Земля» находится под напряжением 0 В.

H

Help Window (Окно справки)

Специальное окно, отображающее имена и расположение терминалов функции или ВПП, описание элементов управления и индикаторов, значения универсальных констант, описания и типы данных управляющих атрибутов.

Hertz, Hz (Герц, Гц)

Количество циклов в секунду.

Hex

Шестнадцатеричная система счисления.

Hierarchical Palette (Иерархическая палитра)

Меню, содержащее палитры и подпалитры.

Hierarchy Window (Окно иерархии)

Окно, графически отображающее иерархию ВП и ВПП.

Housing (Корпус)

Неподвижная часть элементов управления и индикаторов лицевой панели, содержащая ползунки и шкалы.

I

Icon (Иконка)

Графическое представление узла на блок-диаграмме.

Icon Editor (Редактор иконки)

Программа-редактор для создания иконки ВП.

Icon Pane (Область иконки)

Область в правом верхнем углу лицевой панели и блок-диаграммы, где отображена иконка.

IEEE

Институт инженеров по электротехнике и электронике.

Indicator (Индикатор)

Объект лицевой панели для отображения сгенерированных данных.

Inf

Значение числового индикатора для отображения бесконечности в представлении чисел с плавающей запятой.

Instrument Driver (Драйвер прибора)

ВП, управляющий программируемым инструментом (прибором).

I/O

Ввод/вывод. Перенос данных из компьютера или в него, включая каналы связи, устройства ввода и/или сбор данных и интерфейсы управления.

Iteration Terminal (Терминал счетчика числа итераций)

Терминал цикла с фиксированным числом итераций и цикла по условию, содержащий текущее число выполненных итераций.

L

Label (Ярлык)

Текстовый объект, используемый для описания других объектов или их групп на лицевой панели и блок-диаграмме.

Labeling Tool (Инструмент ввода текста)

Инструмент, применяемый для создания ярлыков и ввода текста в текстовые окна.

LabVIEW

Среда разработки лабораторных виртуальных приборов.

LED

Светоизлучающий диод.

Legend (Панель редактирования)

Объект, принадлежащий графику или развертке осциллограмм, который отображает имена и стили графиков, вычерчиваемых на этих индикаторах.

Line (Линия)

Эквивалент понятия «аналоговый канал» – цепь, в которой устанавливается или считывается цифровой сигнал.

M

Marquee (Область выделения)

Движущаяся пунктирная граница, окружающая выбранный объект.

Matrix (Матрица)

Двумерный массив.

Menu Bar (Панель меню)

Горизонтальная панель, содержащая имена главных меню.

Modular Programming (Модульное программирование)

Программирование, использующее чередующиеся вычислительные процедуры.

N

NaN

Значение числового индикатора для объекта, не являющегося числом в представлении чисел с плавающей запятой. Обычно появляется при выполнении неопределенной операции, такой как `Log(-1)`.

NI-DAQ

Набор драйверов для плат сбора данных компании National Instruments и модулей SCXI. Это программное обеспечение работает в качестве интерфейса между LabVIEW и устройствами.

NI-MAX (National Instruments Measurement and Automation Explorer – программа анализа измерений и автоматизации)

Программа настройки, взаимодействующая с NI-DAQ и позволяющая конфигурировать оборудование, устанавливать виртуальные каналы и проверять ввод/вывод прямо с Рабочего стола.

Nodes (Узлы)

Исполняемые элементы блок-диаграммы, состоящие из функций, структур и ВПП.

Nondisplayable Characters (Неотображаемые символы)

ASCII символы, которые не могут быть отображены, такие как «новая строка», «табуляция» и т.д.

Not-a-Path (Не путь)

Заранее определенное значение элемента управления путем, означающее, что путь недействителен.

Not-a-Refnum (Не ссылка файла)

Заранее определенная величина, означающая, что ссылка файла недействительна.

Numeric Controls and Indicators (Числовые элементы управления и индикаторы)

Объекты лицевой панели, используемые для ввода и отображения входных и выходных числовых данных.

NRSE

Общий незаземленный провод

NRSE Measurement

Схема измерения с общим незаземленным проводом, потенциал которого может меняться относительно «земли».

Nyquist Frequency (Частота Найквиста)

Половина частоты выборки. Если сигнал содержит частоты, превышающие частоту Найквиста, то результирующий сигнал после дискретизации будет содержать ложные частоты или искажаться.

O

Object (Объект)

Общий термин для элемента на лицевой панели или блок-диаграмме, включая элементы управления/индикаторы, узлы, проводники и импортированные картинки.

Object Pop-Up Menu Tool (Инструмент вызова контекстного меню)

Инструмент, используемый для вызова контекстного меню объекта.

Octal

Восьмеричная система счисления.

Operating Tool (Инструмент управления, «палец»)

Инструмент, применяемый для ввода данных в элементы управления и манипуляций с этими элементами.

P

Palette (Палитра)

Меню, представляющее возможные опции.

Platform (Платформа)

Компьютер и операционная система.

Plot (График)

Графическое представление массива данных на графике или развертке осциллограмм.

Polymorphism (Полиморфизм)

Способность узла автоматически подстраиваться к данным различного представления, типа или структуры.

Pop Up

Механизм вызова контекстного меню щелчком правой кнопки мыши по объекту (в Windows, Sun, Linux) или щелчком левой кнопки мыши при нажатой клавише <command> (в MacOS).

Pop-up Menus (Контекстное меню)

Меню объекта, вызываемое способом, описанным выше. Меню опций, специфичных для данного объекта.

Port (Порт)

Набор цифровых линий, сконфигурированных для работы в одном направлении, которые можно использовать одновременно.

Positioning Tool (Инструмент перемещения)

Инструмент, применяемый для передвижения, выделения и изменения размера объекта.

Probe (Пробник)

Инструмент отладки для проверки промежуточных значений в ВП.

Probe Tool (Инструмент установки пробников)

Инструмент, служащий для установки пробников на проводники.

Programmatic Printing (Программно управляемая печать)

Автоматическая распечатка лицевой панели ВП по окончании выполнения.

Pseudocode (Псевдокод)

Упрощенное, не зависящее от языка программирования представление программного кода.

Pull-down Menus (Выпадающее меню)

Меню, доступные из панели меню. Опции выпадающих меню обычно являются общими для всех программ.

R

Reentrant Execution (Выполнение с повторным входом)

Режим работы, при котором вызовы нескольких копий ВПП могут обрабатываться параллельно с различными и разделенными местами хранения данных.

Representation (Представление)

Подтипы цифрового типа данных: целочисленные со знаком и без знака с разной разрядностью – 8, 16 и 32, с плавающей запятой различной точности – одинарной, двойной и расширенной, действительные и комплексные.

Resizing Handles (Метка-манипулятор)

Скобки-метки по углам объектов, обозначающие точки изменения размера.

Ring Control (Кольцевой элемент управления)

Особый числовой элемент управления, ставящий в соответствие 32-битное целое число, начиная с 0, элементу набора текстовых ярлыков или картинок.

RS-232

Рекомендуемый стандарт № 232, предложенный Instrument Society of America для последовательной передачи данных. Термин аналогичен понятию последовательной передачи данных, хотя под последней обычно понимают передачу одного бита в единицу времени. Другие стандарты, с которыми вы можете столкнуться, – это RS-485, RS-422 и RS-423.

RSE

Общий заземленный провод.

RSE Measurement

Схема измерения с общим заземленным проводом.

S

Sample (Выборка)

Отдельная точка данных при аналоговом вводе/выводе.

Scalar (Скаляр)

Объект, представляемый точкой на оси. Единичное значение как противоположность массиву. Скалярные логические значения и кластеры – особые примеры соответствующих типов данных.

Scale (Шкала)

Часть механически действующих или графических элементов управления и индикаторов, содержащая набор отметок или точек через известные интервалы для обозначения единиц измерения.

Scope Mode (Временная развертка)

Режим развертки осциллограммы, моделирующий работу осциллографа.

Scroll Tool (Инструмент прокрутки)

Инструмент, используемый для прокрутки окна.

SCXI

Модули расширения, преобразующие сигнал для работы с оборудованием. Совершенная система преобразования сигналов, разработанная компанией National Instruments, в которой используется внешнее шасси, содержащее модули ввода/вывода для согласования, переключения сигналов и т.д. Шасси соединяется с компьютером посредством многофункциональной платы ввода/вывода.

Sequence Local (Локальная переменная структуры последовательности)

Терминал, передающий данные между кадрами структуры последовательности.

Sequence Structure (Структура последовательности)

Структура управления программой, выполняющая поддиаграммы в числовом порядке. Обычно служит для того, чтобы заставить не зависящие друг от друга узлы выполняться в необходимом порядке.

Shift Register (Сдвиговый регистр)

Механизм (включаемый по выбору) циклических структур, используемый для передачи значения переменной из одной итерации в следующую.

Sink Terminal (Терминал-приемник)

Терминал – приемник данных.

Slider (Ползунок)

Подвижная часть ползунковых элементов управления/индикаторов.

Source Terminal (Терминал-источник)

Терминал – отправитель данных.

State Machine (Конечный автомат)

Метод выполнения, при котором частные задания являются отдельными вариантами структуры варианта, вложенной в цикл по условию. Последовательности определяются как массивы строк вариантов.

String Controls and Indicators (Строковые элементы управления и индикаторы)

Объекты лицевой панели, используемые для управления и отображения вводимого и генерируемого текста.

Strip Mode (Панорамная развертка)

Режим работы развертки осциллограммы, моделирующий бумажно-ленточный самописец, который поворачивает барабан по мере поступления данных.

Structure (Структура)

Элемент управления выполнением программы, такой как структура последовательности, варианта, циклы по условию и с фиксированным числом итераций.

Subdiagram (Поддиаграмма)

Блок-диаграмма внутри границ структуры.

SubVI (ВПП)

ВП, используемый на блок-диаграмме другого ВП. Аналог подпрограммы.

Sweep Mode (Временная развертка с маркером)

Похожа на временную развертку за исключением наличия перемещающейся линии, отделяющей старые данные от новых.

T

Terminal (Терминал)

Объект или область узла, через который поступают данные.

Tool (Инструмент)

Специальный режим работы курсора в LabVIEW, который позволяет совершать определенные действия.

Toolbar (Линейка инструментов)

Панель, содержащая кнопки управления, которые можно использовать для запуска и отладки ВП.

Tools Palette (Палитра инструментов)

Палитра, содержащая инструменты редактирования и отладки объектов лицевой и диаграммной панелей.

Top-level VI (ВП верхнего уровня)

ВП, находящийся на вершине в иерархии виртуальных приборов. Этот термин служит для различения ВП и его ВПП.

Trigger (Запуск, триггер)

Условие запуска или остановки операции сбора данных.

Tunnel (Точка ввода в цикл/вывода из цикла)

Терминал входа/выхода данных структуры.

Typecast (Подгонка типов)

Изменение дескриптора типа данных без изменения образа данных в памяти.

Type Descriptor (Дескриптор типа)

См. *Дескриптор типа данных* (Data Type Descriptor).

U

Universal Constant (Универсальная постоянная)

Нередактируемый объект диаграммной панели, генерирующий особый символ ASCII или стандартную числовую константу, например π.

User-defined Constant (Константа, определенная пользователем)

Объект блок-диаграммы, генерирующий величину, введенную пользователем.

V

VI

См. *Виртуальный прибор* (Virtual Instrument).

VI Library (Библиотека ВП)

Файл, содержащий набор взаимосвязанных ВП для специального использования.

VI Server (Сервер ВП)

Особенность LabVIEW, позволяющая удаленно программно задавать поведение ВП и элементов управления.

Virtual Instrument (Виртуальный прибор)

Программа LabVIEW, моделирующая внешний вид и функции физического инструмента или прибора.

W

Waveform (Осциллограмма)

Тип данных LabVIEW, который обычно представляет сигнал. Y – массив актуальных величин регистрируемого сигнала, X_0 – время начала дискретизации и ΔX – временной промежуток между соседними выборками (определяет частоту дискретизации).

While Loop (Цикл по условию)

Циклическая структура, повторяющая поддиаграмму до тех пор, пока не будет выполнено определенное условие. Термин эквивалентен циклу `Do` или `Repeat-Until` в обычных языках программирования.

Wire (Проводник)

Путь прохождения данных между узлами.

Wiring Tool (Инструмент соединения, «катушка»)

Инструмент, используемый для задания пути передачи данных от терминала-источника к терминалу-приемнику.

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС-КНИГА» наложенным платежом, выслав открытку или письмо по почтовому адресу: **123242, Москва, а/я 20** или по электронному адресу: **post@abook.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в Internet-магазине: **www.abook.ru**.

Оптовые закупки: тел. **(095) 258-91-94, 258-91-95**; электронный адрес **abook@abook.ru**.

Джеффри Тревис

LabVIEW для всех

Главный редактор *Захаров И. М.*
Выпускающий редактор *Левицкая Т. В.*
Верстка *Захарова Е. П.*
Графика *Салимонов Р. В.*
Дизайн обложки *Дудатий А. М.*

Подписано в печать 24.05.2005. Формат 70×100 1/16.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 44,2. Тираж 2000 экз. Зак. № 1462

Электронный адрес издательства «ДМК Пресс»: www.dmk.ru.
Internet-магазин: www.abook.ru.

ООО «ПриборКомплект», 113208, Москва, ул. Чертановская, д. 2/11.

Отпечатано в ОАО «Типография «Новости»
105005, Москва, ул. Фр. Энгельса, 46