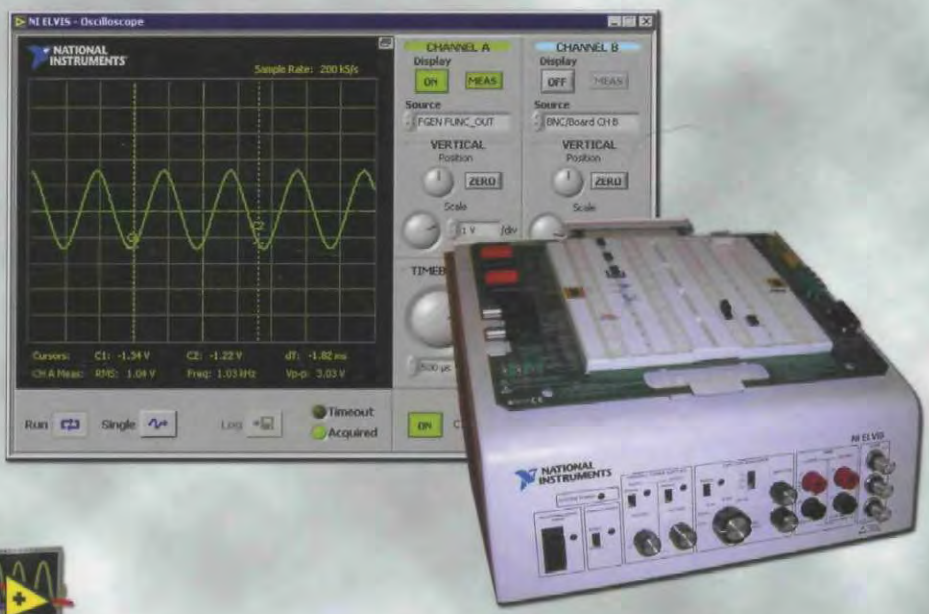
NATIONAL INSTRUMENTS™

ПриборКомплект

ДМК

Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)



П. А. Бутырин, Т. А. Васьковская, В. В. Каратаев, С. В. Материкин

К 75-ЛЕТИЮ МОСКОВСКОГО ЭНЕРГЕТИЧЕСКОГО ИНСТИТУТА

Бутырин П.А., Васьковская Т.А., Каратаева В.В., Материкин С.В.

Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)

Рекомендовано УМО
по университетскому политехническому образованию
в качестве учебного пособия для студентов
высших учебных заведений,
обучающихся по группе подготовки бакалавров
550000-«Технические науки»
дисциплине «Управление техническими системами»

ДМК
издательство
Москва, 2005

УДК 004.94
ББК 32.973.26-018.2

Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW 7/ Под. ред. Бутырина П. А. –М.: ДМК Пресс, 2005. 264 с.: ил.

ISBN 5-94074-084-7

Рецензенты: лауреат Государственной премии, д.т.н., проф. В. Г. Миронов, к.т.н., доц. А. И. Евсеев

Книга состоит из 30 глав, названных лекциями. Эти главы содержат как информацию о тех или иных возможностях LabVIEW, так и практические задания, выполнение которых необходимо для овладения этим прикладным инструментом исследования физических процессов и управления ими. Материал каждой главы рассчитан на одно занятие за компьютером и может быть использован как при обучении группы студентов преподавателем, так и при самообучении студента. Для большей доступности курса большинство практических заданий ограничивается исследованием чисто виртуальных объектов, что не требует приобретения специальной материальной части (аналогово-цифровых преобразователей и т. д.).

Издание предназначено для инженеров и студентов технических вузов.

ISBN 5-94074-084-7

Алексейчик Леонард Валентинович
Бутырин Павел Анфимович
Васьковская Татьяна Александровна
Герасименко Вадим Петрович
Каратаев Владимир Васильевич
Материкин Сергей Владимирович
Немов Юрий Николаевич
Рубцов Александр Андреевич
Шакирзянов Феликс Нигматзянович

Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW (30 лекций)

Главный редактор *Мовчан Д. А.*
dm@dmkpress.ru
Корректор *Синяева Г. И.*
Верстка *Мухамедьярова Л. Л.*
Дизайн обложки *Мовчан А. Г.*

Подписано в печать 22.10.2004. Формат 70x100 1/16.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 24,75. Тираж 3000 экз. Заказ № К-4936.
Отпечатано с готовых диапозитивов в ГУП «ИПК «Чувашия».
428019, г. Чебоксары, пр. И. Яковлева, 13.

Содержание

Введение

Автоматизация физических исследований и эксперимента

Компьютер для массового пользователя – это эффективный инструмент делопроизводства, выполнения математических расчетов и финансовых операций, средство обучения, получения и передачи информации, а также проведения досуга. Его возможности как инструмента управления и измерения менее известны, хотя история создания и развития вычислительной техники напрямую связана именно с этими возможностями.

Прообраз современной вычислительной машины был разработан в середине XIX века Чарльзом Бэббеджем для управления ткацким производством – наиболее технологичным производством того времени. Создание в середине XX века электронной вычислительной техники и последующее ее совершенствование во многом обусловливалось необходимостью автоматизации высокотехнологичных производств, космических и ядерных исследований, военной техники. Вопросы измерения параметров физических процессов и последующего управления последними были в центре внимания таких разработок. Появлялась специализированная вычислительная техника для решения этих вопросов, проводилась ее стандартизация и унификация. Так, в 70-х годах XX века для ядерной физики и атомной техники была разработана и получила затем более широкое распространение информационно-измерительная система КАМАК, позволяющая автоматизировать измерения параметров различных физических процессов и управления ими.

В начале XXI века проблемы автоматизации измерений параметров физических процессов становятся насущными не только для перечисленных высоких сфер, но практически для всех областей жизнедеятельности человека. Оставив в стороне инженерные и научные области, рассмотрим проблемы автоматизации измерений на примере быта. Современное жилье европейца и американца оснащено тепло-, электро-, водо- и газоснабжением и массой потребителей этих ресурсов. Номенклатура только используемых в жилищах передовых стран электроприемников

достигает полусотни. В этой связи возникает необходимость контроля за расходом перечисленных ресурсов, качеством их параметров, а также необходимость рационального управления их потреблением. Надвигающиеся ресурсный и энергетический кризисы сделают подобные задачи актуальными практически для каждой семьи, и решаться они должны наиболее эффективным образом. Привлекает внимание возможность решения этих задач с использованием персонального компьютера как инструмента измерения, анализа данных и управления, уже имеющегося в большинстве семей развитых стран. В этой связи актуальным становится распространение знаний о возможности использования компьютера как элемента автоматизации физических исследований и эксперимента, трактуя последние в самом широком смысле. Один из наиболее перспективных путей реализации этой возможности дает использование среды программирования LabVIEW.

LabVIEW

Среда разработки лабораторных виртуальных приборов LabVIEW (Laboratory Virtual Instrument Engineering Workbench) представляет собой среду прикладного графического программирования, используемую в качестве стандартного инструмента для проведения измерений, анализа их данных и последующего управления приборами и исследуемыми объектами. LabVIEW может использоваться на компьютерах с операционными системами Windows, MacOS, Linux, Solaris и HP-UX. Компьютер, оснащенный измерительно-управляющей аппаратной частью и LabVIEW, позволяет полностью автоматизировать процесс физических исследований. Создание любой программы для достижения этих целей (виртуального прибора) в графической среде LabVIEW отличается большой простотой, поскольку исключает множество синтаксических деталей.

Особо следует отметить динамику развития LabVIEW. Первая его версия была создана в 1986 году компанией National Instruments в результате поисков путей сокращения времени программирования измерительных приборов. Версии LabVIEW с второй по седьмую проявлялись в 1990, 1992, 1993, 1996, 2000 и 2003 годах. Каждая последующая существенно расширяла возможности предыдущей версии и прежде всего по обмену данных с измерительными приборами и работе с другими программными продуктами.

Сфера применимости LabVIEW также непрерывно расширяется. В образовании она включает лабораторные практикумы по электротехнике, механике, физике. В фундаментальной науке LabVIEW используют такие передовые центры как CERN (в Европе), Lawrence Livermore, Batelle, Sandia, Oak Ridge (США), в инженерной практике – объекты космические, воздушного, надводного и подводного флота, промышленные предприятия и т.д.

Сведения о коллективе авторов книги

Настоящее издание разработано коллективом сотрудников кафедры Теоретических основ электротехники (ТОЭ) Московского энергетического института (МЭИ).

Содержание пособия и форма изложения материала в значительной мере основываются на опыте преподавания LabVIEW, накопленном сотрудниками кафедры ТОЭ МЭИ. Работы по изучению и преподаванию LabVIEW на этой кафедре были развернуты академиком РАН Камо Сероповичем Демирчяном в середине 90-х годов и в настоящее время играют большую роль в жизни кафедры. Авторский коллектив выражает глубокую признательность академику К. С. Демирчяну за проявленную настойчивость в привлечении сотрудников кафедры к этой работе в прошлом и большое внимание к созданию данного пособия в настоящем.

На кафедре ТОЭ МЭИ с 2000 года работает учебно-научная лаборатория «Виртуальные приборы электротехники» с 18 рабочими местами, оснащенными аппаратно-программным комплексом LabVIEW. Ежегодно в этой лаборатории около 400 студентов энергетического и электротехнического институтов (факультетов) МЭИ под руководством сотрудников кафедры изучают основы LabVIEW. Сотрудниками кафедры выпущено несколько книг по LabVIEW и его использованию в практике преподавания электротехнических дисциплин.

Содержание книги

Книга состоит из 30 глав, названных лекциями. Эти главы содержат как информацию о тех или иных возможностях LabVIEW, так и практические задания, выполнение которых необходимо для овладения этим прикладным инструментом исследования физических процессов и управления ими. Материал каждой главы рассчитан на одно занятие за компьютером и может быть использован как при обучении группы студентов преподавателем, так и при самообучении студента. Для большей доступности курса большинство практических заданий ограничивается в издании исследованием чисто виртуальных объектов, что не требует приобретения специальной материальной части (аналогово-цифровых преобразователей и т.д.). Здесь следует заметить, что в LabVIEW при исследовании виртуальных и реальных объектов используется один и тот же подход, а богатые графические возможности создают иллюзию реальности при работе с чисто виртуальными объектами. Последнее обстоятельство делает эту систему особо привлекательной для создания разнообразных тренажеров, учебных лабораторий и т.д.

Тот факт, что авторы данной книги – электротехники, наложил некоторый отпечаток на ее содержание в прикладной части, где большая часть примеров связана с исследованием электромагнитных процессов, что в целом, надеемся, не сузило круг возможных пользователей книги.

Материал книги условно можно разделить на три части. В первой из них (лекции 1–17) даются основные сведения о среде LabVIEW и ее возможностях, а также об исследовании виртуальных объектов при помощи математического моделирования. Вторая часть (лекции 18–23) посвящена построению виртуальных приборов для проведения измерений в реальных физических устройствах, в частности, дано описание лабораторной установки ELVIS, разработанной корпорацией National Instruments. В третьей части (лекции 24–30) описывается техника и методика составления больших проектов в среде LabVIEW.

Благодарности

Для написания данной книги корпорация National Instruments (США) любезно предоставила авторам лабораторную установку ELVIS и дала разрешение использовать фрагменты книги «LabVIEW™ 7 Express. Базовый курс 1», за что коллектив авторов выражает глубокую благодарность корпорации. Авторы благодарят регионального менеджера корпорации А. Салатяна (США) и менеджера по развитию NI в РФ А.В. Спиридонова за многолетнее плодотворное сотрудничество, внимание к данной работе и помощь в ее проведении. Авторы признательны менеджеру образовательных программ корпорации NI П.М. Михееву за оперативное разрешение организационных проблем, возникавших в ходе работы над книгой. Мы благодарны рецензентам проф. В. Г. Миронову и доц. А. И. Евсееву за ценные замечания и рекомендации. Мы благодарим также генерального директора издательства «Приборкомплект» А. И. Ушакова, финансового директора Д. А. Мовчана и их сотрудников за усилия по скорейшему выходу книги в свет.

Работа поддерживалась грантом Президента РФ НШ-1511.2003.8.

Лекция 1

Общие сведения о программно-инструментальной среде LabVIEW

В первой лекции рассматриваются основные элементы LabVIEW, дается краткое представление о приборах, инструментах и функциональных возможностях программы.

Введение

LabVIEW – среда разработки прикладных программ, в которой используется язык графического программирования G и не требуется написания текстов программ. Среда LabVIEW дает огромные возможности как для вычислительных работ, так и – главным образом – для построения приборов, позволяющих проводить измерения физических величин в реальных установках, лабораторных или промышленных, и осуществлять управление этими установками.

Программа, написанная в среде LabVIEW, называется виртуальным прибором (ВП) (VI – virtual instrument). Внешнее графическое представление и функции ВП имитируют работу реальных физических приборов. LabVIEW содержит полный набор приборов для сбора, анализа, представления и хранения данных. Источником кода виртуального инструмента служит блок-схема программируемой задачи.

Программная реализация виртуальных приборов использует в своей работе принципы иерархичности и модульности. Виртуальный прибор, содержащийся в составе другого виртуального прибора, называется прибором-подпрограммой (SubVI).

Вход в среду LabVIEW

При запуске LabVIEW появляется диалоговое окно (рис. 1.1). В верхней части окна находится панель меню со стандартными пунктами: **File**, **Edit** (редактирование), **Tools** (инструменты), **Help** (помощь). В правой части – набор кнопок:

- Кнопка **New** – создание нового ВП. Стрелка рядом с кнопкой используется для открытия пустого ВП или открытия диалогового окна.

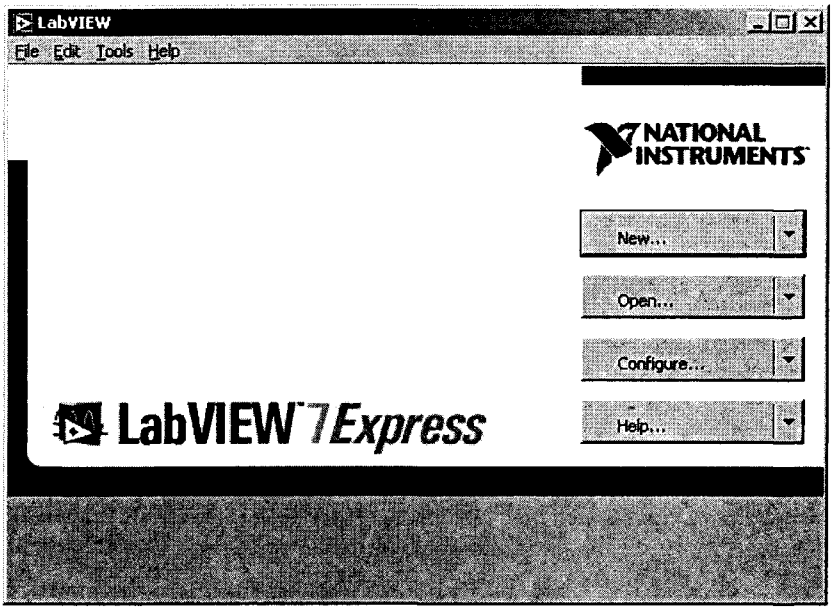


Рис. 1.1

- Кнопка **Open** – открытие созданного ранее ВП. Стрелка рядом с кнопкой предназначена для открытия недавно использовавшегося ВП.
- Кнопка **Configure** — настройка устройств DAQ. Стрелка рядом с кнопкой — конфигурация LabVIEW.
- Кнопка **Help** — запуск *LabVIEW Help* (встроенной помощи). Стрелка рядом с кнопкой — для выбора опций помощи.

Создание нового виртуального прибора

При нажатии кнопки New открывается окно **Create New** (создать новый ВП), где расположено меню, из которого можно выбрать либо пустые окна ВП (Blanc VI), либо окна с различными шаблонами (VI from Templates). Выберем пустые окна ВП (Blanc VI), и нажмем кнопку OK. На экране появляются две совмещенные панели, расположенные каскадом. Одна из них – лицевая панель (Front Panel) – имеет серый цвет рабочего пространства, другая – панель блок-диаграмм (Block Diagram) – белый цвет. Для развертывания панелей на левую и правую половины экрана нужно нажать на клавиатуре одновременно **Ctrl+T**. Панели можно развернуть также нажатием **Windows** в верхней части панели и затем **The Left and Right**. (Выбрав **The Up and Down**, можно развернуть панели на верхнюю и нижнюю половины экрана). Каждая из этих панелей может быть развернута на весь экран нажатием кнопки с изображением прямоугольника в верхнем правом углу панели. Возврат к двум панелям осуществляется нажатием той же кнопки с изображением сдвоенных прямоугольников.

Главное меню

Главное меню в верхней части окна ВП содержит пункты общие с другими приложениями, такие как **Open**, **Save**, **Copy**, **Paste**, а также специфические пункты меню LabVIEW. Некоторые из них содержат сведения о «горячих» клавишах вызова этих пунктов. **(MacOS)** Меню появляется в верхней части экрана.

Внимание. Во время выполнения ВП некоторые пункты главного меню недоступны.

- Пункт меню **File** используется для открытия, закрытия, сохранения и печати ВП.
- Пункт меню **Edit** используется для поиска и внесения изменений в компоненты ВП.
- Пункт меню **Operate** используется для запуска, прерывания выполнения и изменения других опций ВП.
- Пункт меню **Tools** используется для связи с приборами и **DAQ** устройствами, сравнения ВП, формирования приложений и конфигурации LabVIEW.
- Пункт меню **Browse** используется для перемещения по ВП и его иерархии.
- Пункт меню **Window** используется для отображения окон LabVIEW и палитр.
- Пункт меню **Help** используется для получения информации о палитрах, меню, инструментах, ВП и функциях, для получения пошаговой инструкции использования LabVIEW и информации о компьютерной памяти.

Палитра инструментов

Создавать, редактировать и отлаживать ВП можно с помощью **Tools Palette** (Палитры инструментов). Термин инструмент подразумевает специальный операционный режим курсора мыши. При выборе определенного инструмента значок курсора изменяется на значок данного инструмента. Палитра инструментов вызывается через пункт главного меню **Window** ⇒ **Show Tools Palette**. Палитру инструментов можно размещать в любой области рабочего пространства блок-диаграммы и лицевой панели. Вид палитры инструментов показан на рис. 1.2.

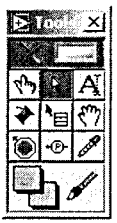


Рис. 1.2

Примечание. Удерживая нажатой клавишу **Shift** и щелкнув правой клавишей мыши, можно вывести на экран временную версию **Tools Palette** (Палитры Инструментов).

Если включен автоматический выбор инструмента, то при наведении курсора на объект лицевой панели или блок-диаграммы автоматически выбирается соответствующий инструмент из палитры **Tools** (Инструментов). Автоматический выбор инструментов включается нажатием на кнопку **Automatic Tool Selection** палитры **Tools** (Инструментов) или нажатием клавиш **Shift+Tab**.

Инструмент УПРАВЛЕНИЕ используется для изменения значения элементов управления или ввода текста. При наведении курсора на такой элемент как строковый элемент управления, значок инструмента меняется: 

Инструмент ПЕРЕМЕЩЕНИЕ используется для выбора, перемещения или изменения размеров объектов. При наведении инструмента на объект изменяемого размера значок инструмента меняется:

Инструмент ВВОД ТЕКСТА используется для редактирования текста и создания свободных меток. При создании свободных меток значок инструмента меняется:

Инструмент СОЕДИНЕНИЕ создает проводники данных, соединяя объекты на блок-диаграмме.

Инструмент ВЫЗОВ КОНТЕКСТНОГО МЕНЮ вызывает контекстное меню соответствующего объекта по щелчку левой кнопки мыши.

Инструмент БЫСТРАЯ ПРОКРУТКА ЭКРАНА используется для просмотра окна без использования полосы прокрутки.

Инструмент ВВОД КОНТРОЛЬНОЙ ТОЧКИ позволяет расставлять контрольные точки на ВП, функциях, узлах, проводниках данных, структурах и приостанавливать в них выполнение программы.

Инструмент УСТАНОВКА ОТЛАДОЧНЫХ ИНДИКАТОРОВ дает возможность исследовать поток данных в проводниках блок-диаграммы. Используется для просмотра промежуточных значений при наличии сомнительных или неожиданных результатов работы ВП.

Инструмент КОПИРОВАНИЕ ЦВЕТА предназначен для копирования цвета с последующей вставкой с помощью инструмента РАСКРАШИВАНИЕ.

Инструмент РАСКРАШИВАНИЕ позволяет изменить цвет объекта. Он также отображает текущий передний план и параметры настройки цвета фона.

Если автоматический выбор инструмента выключен, можно менять инструменты палитры **Tools** (Инструментов) с помощью клавиши **Tab**. Для переключения между инструментом ПЕРЕМЕЩЕНИЕ и СОЕДИНЕНИЕ на блок-диаграмме или между инструментом ПЕРЕМЕЩЕНИЕ и УПРАВЛЕНИЕ на лицевой панели – достаточно нажать пробел.

Лицевая панель

Лицевая (передняя) панель имитирует панель реального физического прибора. На ней располагаются управляющие и измерительные элементы виртуального прибора.

Пример лицевой панели представлен на рис. 1.3.

Палитра элементов лицевой панели

Лицевая панель создается с использованием палитры элементов под общим названием **Controls**, которая вызывается нажатием правой клавиши мыши на свободное поле лицевой панели (либо можно выбрать в пункте главного меню **Window** ⇒ ⇒ **Show Controls Palette**). Эти элементы могут быть либо средствами ввода данных – элементами собственно управления (**Controls**), либо средствами отображения данных – элементами отображения (**Indicators**).

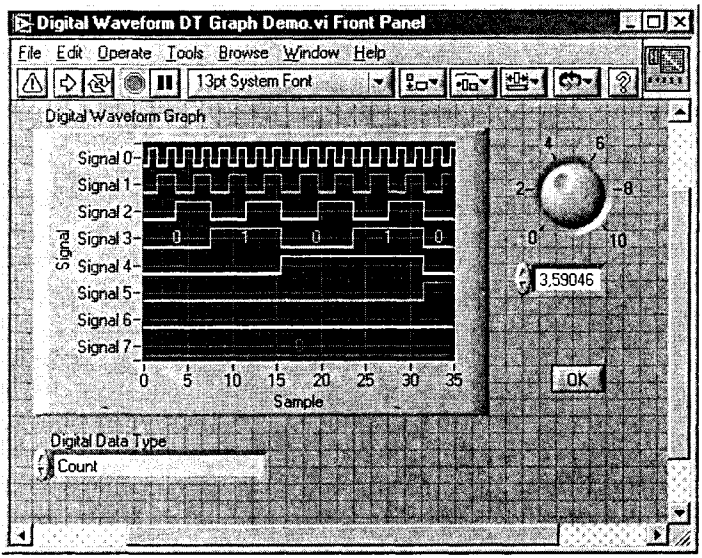


Рис. 1.3

По умолчанию палитра элементов появляется в экспресс-виде (рис. 1.4) и содержит лишь наиболее часто используемые элементы.

Выбранный элемент выделяется инструментом «перемещение» («стрелка») и выводится на лицевую панель.

Для получения полной палитры используется кнопка **All Controls**, находящаяся в правом нижнем углу. Такая палитра показана на рис. 1.5.

Данные, вводимые на лицевой панели ВП, поступают на блок-диаграмму, где ВП производит с ними необходимые операции. Результат вычислений передается на элементы отображения информации на лицевой панели ВП.

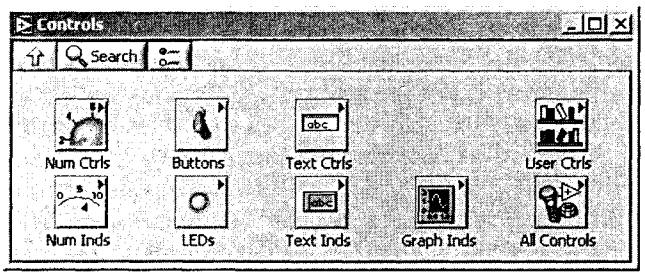


Рис. 1.4

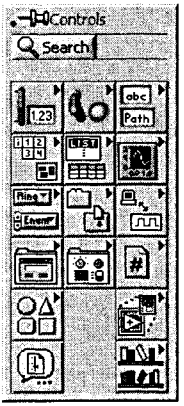


Рис. 1.5

Инструментальная панель лицевой панели

Инструментальная панель (рис. 1.6) используется для запуска и редактирования ВП.

Рис. 1.6

Кнопка запуска **Run** – запускает ВП

Во время работы ВП кнопка **Run** меняет свой вид, как показано слева, если этот виртуальный прибор высокого уровня.

Если ВП работает в качестве подпрограммы, то кнопка **Run** выглядит, как показано слева.

Кнопка **Run** выглядит в виде «сломанной» стрелки, как показано слева, во время создания или редактирования ВП. В таком виде кнопка показывает, что ВП не может быть запущен на выполнение. После нажатия этой кнопки появляется окно **Error list,** в котором перечислены допущенные ошибки.

Кнопка непрерывного запуска **Run Continuously** – ВП выполняется до момента принудительной остановки.

Во время выполнения ВП появляется кнопка **Abort Execution.** Эта кнопка используется для немедленной остановки выполнения ВП.

Примечание. По возможности следует избегать использования кнопки **Abort Execution** для остановки ВП. Следует позволить ВП закончить передачу данных или выполнить остановку программным способом, гарантируя остановку ВП в определенном состоянии. Например, можно установить на лицевой панели кнопку, по нажатию которой ВП останавливается.

Кнопка **Pause** приостанавливает выполнение ВП. После нажатия кнопки **Pause** LabVIEW подсвечивает на блок-диаграмме место остановки выполнения. Повторное нажатие – продолжение работы ВП.

Text Settings – выпадающее меню установок текста, включая размер, стиль и цвет.

В меню **Align Objects** производится выравнивание объектов по осям (по вертикали, по осям и т.д.).

В меню **Distribute Objects** производится выравнивание объектов в пространстве (промежутки, сжатие и т.д.).

В меню **Resize Objects** производится приведение к одному размеру многократно используемых объектов лицевой панели.

Меню **Reorder** используется при работе с несколькими объектами, которые накладываются друг на друга. Выделив один из объектов с помощью инструмента ПЕРЕМЕЩЕНИЕ, в меню **Reorder** следует выбрать его порядок отображения на лицевой панели.

Кнопка **Context Help** выводит на экран окно **Context Help** (контекстной справки)

Блок-диаграмма

После помещения элементов **Управления** или **Отображения** данных на Лицевую панель, они получают свое графическое отображение на блок-диаграмме. Объекты блок-диаграммы включают графическое отображение элементов лицевой панели, операторов, функций, подпрограмм ВП, констант, структур и проводников данных, по которым производится передача данных между объектами блок-диаграммы.

Палитра функций блок-диаграммы

Палитра **функций** (рис. 1.7) используется для создания блок-диаграммы. Она доступна только в окне блок-диаграмм. Чтобы отобразить палитру **функций**, следует либо выбрать в пункте главного меню **Window** Ю **Show Functions Palette,** либо щелкнуть правой кнопкой мыши в рабочем пространстве блок-диаграммы. Используя кнопку в верхнем левом углу палитры, можно зафиксировать ее на экране. По умолчанию палитра **функций** появляется в экспресс-виде и отображает экспресс-ВП. Экспресс-ВП — узлы функций, которые можно настраивать с помощью диалогового окна. Они используются для выполнения стандартных измерений при минимальных соединениях.

Для получения полной палитры используется кнопка **All Functions**, находящаяся **в** правом нижнем углу. Такая палитра показана на рис. 1.8.

Полную палитру функций можно получить также при нажатии кнопки **Options** (опции), показанной слева. При этом отображается стра-

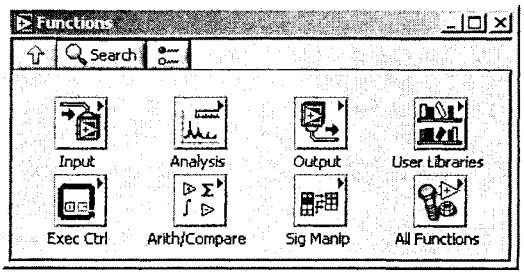


Рис. 1.7

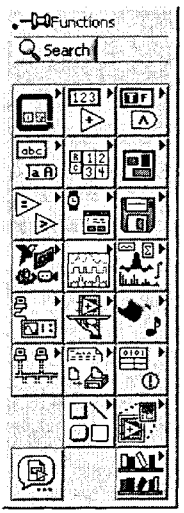


Рис. 1.8

ница **Controls/Functions Palettes** диалогового окна **Options.** Следует заменить **Palette View** на **Advanced**.

Инструментальная панель блок-диаграммы

При запуске ВП на блок-диаграмме появляется показанная на рис. 1.9 инструментальная панель:

Кнопка **Highlight Execution** предназначена для просмотра потока данных через блок-диаграмму (режим отладки). Повторное нажатие кнопки отключает этот режим.

Кнопка **Step Into** используется при пошаговом выполнении цикла от узла к узлу, подпрограммы ВП и т.д. При этом узел мигает, обозначая готовность к выполнению.

Кнопка **Step Over** позволяет пропустить в пошаговом режиме цикл, подпрограмму и т.д.

Кнопка **Step Out** позволяет выйти из цикла, подпрограммы и т.д.

Рис. 1.9

Выход из узла предполагает завершение выполнения этого узла в пошаговом режиме и переход в следующий.

Кнопка **Warning** появляется, когда есть потенциальная проблема с блок-диаграммой, но она не запрещает выполнение ВП. Кнопку **Warning** можно активизировать, войдя в пункт главного меню **Инструменты**, далее – **Опции, Отладка (Tools ⇒ Options ⇒ Debugging)**.

Пример 1.1

Требуется построить виртуальный прибор для выполнения операции сложения и вычитания двух чисел a и b.

Для решения этой задачи нужно построить блок-диаграмму и соответствующую ей лицевую панель, представленные на рис. 1.10. Построение ведется следующим образом.

1. Вызываются четыре цифровых элемента на лицевой панели, из которых два управляющих (обозначены a и b) и два индикатора ($a+b$, $a-b$). Их изображения одновременно появляются на блок-диаграмме. Из палитры функций вызываются элементы «сумма» (+) и «разность» (-) по пути **Arithmetic & Compare ⇒ Numeric**.
2. Соединение элементов осуществляется инструментом «соединение» (катушка). Катушка подводится к элементу; когда элемент начнет мигать, нажимается левая клавиша мыши и появившийся провод подводят к нужному элементу. Когда последний начинает мигать – клавишу нужно отпустить.

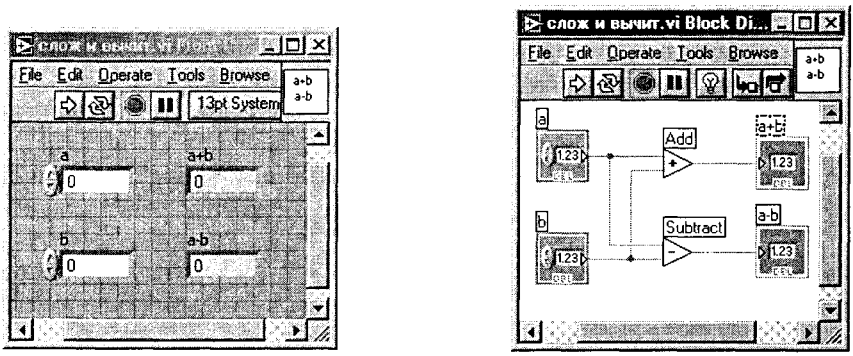


Рис. 1.10

3. В окна управляющих элементов вводятся заданные значения *a* и *b* (посредством инструмента «ввод текста»).
4. Задача запускается на выполнение кнопкой запуска с панели блок-диаграмм или с лицевой панели. В окнах индикаторов появляются результаты вычислений.

Поиск объектов на палитрах Controls и Functions

Для быстрого перемещения по разделам палитры **Controls** (Элементы) и палитры **Functions** (Функции) предназначены кнопки, показанные ниже:

Up – перемещает на один уровень вверх в иерархии палитры.

Search – вызывает окно поиска (рис. 1.11). В этом режиме в палитрах производится поиск узлов, функций и ВП по названию. Например,

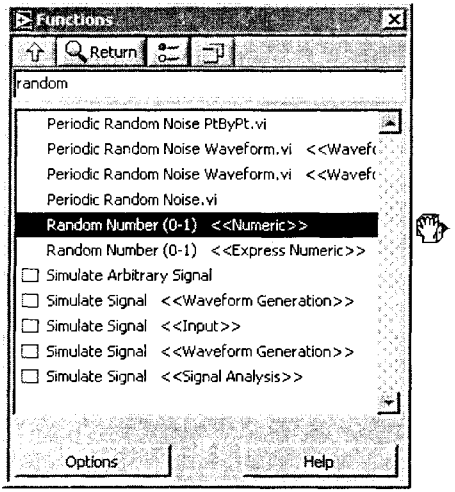


Рис. 1.11

чтобы найти функцию **Random Number** (Генератор случайных чисел), следует нажать кнопку **Search** на палитре **Functions** (Функций) и ввести в поле ввода текста **«Random Number»**. LabVIEW выводит на экран список узлов и функций, в названии которых встречается введенный текст. Выбрав в результатах поиска искомую функцию, можно перенести ее на блок-диаграмму с помощью мыши Двойной щелчок кнопкой мыши на искомой функции покажет ее местоположение на палитре.

Options – после нажатия этой кнопки открывается страница **Controls/Functions Palettes** диалогового окна **Options**, в которой производится настройка внешнего вида палитры.

Контекстное меню

Контекстное меню используется наиболее часто. Все объекты LabVIEW, свободное рабочее пространство лицевой панели и блок-диаграммы имеют свои контекстные меню. Контекстное меню используется для изменения поведения объектов блок-диаграммы и лицевой панели. Контекстное меню вызывается щелчком правой кнопкой мыши на объекте, лицевой панели или блок-диаграмме. Пример контекстного меню показан на рис. 1.12.

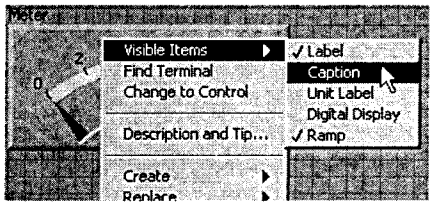


Рис. 1.12

Выводы

В лекции мы получили первое представление о среде LabVIEW, кратко ознакомились с применяемыми здесь инструментами; получили понятие о построении виртуального прибора и об основных его частях: о лицевой панели с элементами управления и измерения и о панели блок-схем с некоторыми функциональными элементами; ознакомились также с применением контекстного меню для поиска нужных элементов. Практическое построение несложного виртуального прибора мы проведем в течение следующей лекции.

Лекция 2

Выполнение арифметических действий в среде LabVIEW

Лекция посвящена изучению начальных приемов и методов работы в программной среде LabVIEW. На примере простой электрической цепи рассматривается проведение арифметических вычислений в среде LabVIEW. Самостоятельно составляется программа расчета токов ветвей с использованием арифметических элементов. Изучаются способы исправления ошибок и редактирования программ.

Большая часть поставленных задач выполняется слушателями самостоятельно.

Пример 2.1

Требуется создать виртуальный прибор, состоящий из источника и приемника электрической энергии.

Для этой цели нужно:

1. Выбрать в качестве источника элемент управления на лицевой панели **Controls** ⇒ **Numeric** ⇒ **Digital Control**. В качестве приемника – индикатор **Controls** ⇒ **Numeric** ⇒ **Digital Indicator**.
2. На панели блок-схем появляются изображения (иконки) этих элементов.
3. Для соединения источников и индикаторов на панели блок-схем выбирается инструмент (курсор) «соединение» («катушка»). Его нужно поместить на изображение источника. Когда конец провода катушки попадает в область терминала (об этом свидетельствует «мигание» объекта), нажатием левой клавиши мыши фиксируется соединение. После подвода курсора к изображению индикатора, аналогичным образом фиксируется другой конец соединительного проводника. При корректном соединении линия окрашивается в красный цвет, иначе она остается пунктирной. В окно источника вводится числовое значение соответствующей величины.
4. Программа запускается на исполнение кнопкой запуска **Run**. В окне приемника появляется числовое значение, введенное в окно источника.

Пример 2.2

В цепи рис. 2.1 изображена схема электрической цепи. Значения R_1, R_2, R_3 сопротивлений резисторов и значение E электродвижущей силы источника энергии будем считать известными (их можно задать произвольно). Рассмотрим определение токов всех ветвей.

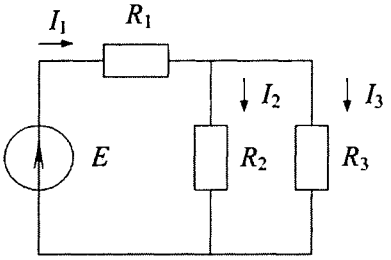


Рис. 2.1

Расчет ведется по уравнениям, составленным по законам Ома и Кирхгофа:

$$I_1 = E/(R_1 + \frac{R_2 R_3}{R_2 + R_3});$$

$$I_2 = R_3 I_1/(R_2 + R_3); \qquad (1)$$

$$I_3 = R_2 I_1/(R_2 + R_3).$$

Для решения задачи нужно:

1. Вывести на лицевую панель элементы управления (источники), представляющие значения сопротивлений и ЭДС а также индикаторы для записи искомых токов.
2. Затем следует собрать блок-схему для расчета. Для этого вызываются арифметические операторы по пути **Functions** ⇒ **Numeric** и соединяются источники с приемниками в соответствии с записанными уравнениями. После сборки блок-схемы программа запускается на исполнение.

Задача 2.1

Рассчитать токи в цепи рис. 2.2 при условии, что сопротивления в цепи комплексные (величины их можно задать произвольно).

Поскольку структура схемы рис. 2.2 такая же, как и у схемы 2.1, расчет можно вести по той же блок-схеме, что и в предыдущем примере, с тем отличием, что все управляющие и индикаторные элементы должны быть комплексными. С этой целью необходимо изменить тип данных этих элементов. Устанавливаемый по умолчанию тип DBL (действительные числа с двойной

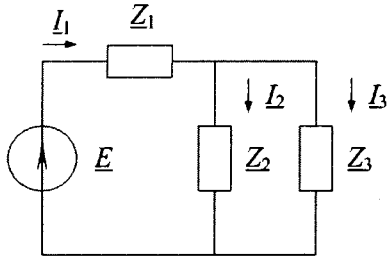


Рис. 2.2

точностью) следует заменить на **CDB** (комплексные числа с двойной точностью). Для этого нужно нажать правой клавишей мыши на элемент и из всплывающего меню выбрать **Representation ⇒ CDB**. Результат выводится в декартовой системе координат (действительная и мнимая части). Для перевода в полярную систему используется элемент **Complex To Polar**, вызываемый из палитры функций: **All Functions ⇒ Complex**.

Редактирование ВП

После построения нескольких несложных схем следует более подробно ознакомиться с методикой редактирования схем и исправления ошибок.

Существует несколько методов редактирования объектов лицевой панели и блок-диаграммы.

Создание объектов

В дополнение к созданию объектов лицевой панели с помощью палитры **Controls** предусмотрена возможность создания элементов управления и отображения данных, констант по щелчку правой копкой мыши на узле. Для этого в контекстном меню следует выбрать пункт **Create**.

- **Constant** — создание констант, отображающихся только на блок-диаграмме.
- **Control** — создание элемента управления на лицевой панели ВП.
- **Indicator** — создание элемента отображения данных на лицевой панели.

Выделение объектов

Выделение объектов на лицевой панели и блок-диаграмме производится с помощью инструмента ПЕРЕМЕЩЕНИЕ.

Когда объект выделен, его окружает пунктирная линия. Для выбора нескольких объектов, следует во время их выделения нажать и удерживать клавишу **Shift**.

Можно также выделить несколько объектов, щелкнув мышью в свободном пространстве и обведя их курсором.

Перемещение объектов

Перемещение объектов осуществляется при помощи инструмента ПЕРЕМЕЩЕНИЕ. Перемещать объекты можно также при помощи стрелок на клавиатуре. Для перемещения объекта с шагом в несколько пикселей в момент перемещения следует нажать и удерживать клавишу **Shift.**

Можно ограничить направление движения выбранного объекта только по горизонтали или только по вертикали, если в момент его перемещения удерживать клавишу **Shift**. Первоначально выбранное направление движения (горизонтальное или вертикальное) определяет направление перемещение объекта.

Удаление объектов

Чтобы удалить объект, следует выделить его с помощью инструмента ПЕРЕМЕЩЕНИЕ, после чего нажать на клавиатуре клавишу **Delete** или выбрать пункты главного меню **Edit** ⇒ **Clear**.

Отмена и восстановление действий

Если в процессе редактирования ВП была допущена ошибка, можно отменить или восстановить действия, выбрав **Undo** (Отменить) или **Redo** (Восстановить) в пункте главного меню **Edit** (Редактирование). Установка количества действий, подлежащих отмене или восстановлению, производится в пункте главного меню **Tools** ⇒ **Options**. Для этого из выпадающего меню следует выбрать раздел **Block Diagram**. Установка небольшого числа повторений сохраняет ресурсы памяти компьютера.

Копирование объектов

Большинство объектов можно копировать, перемещая выделенный объект и одновременно удерживая клавишу **Ctrl**.

(MacOS) Нажать кнопку **Option. (Sun)** Нажать кнопку **Meta. (Linux)** Нажать кнопку **Alt.**

После переноса выбранного объекта на новое место, отпускается сначала кнопка мыши, а затем клавиша **Ctrl**. В этом месте появляется копия объекта, а первоначальный объект остается на старом месте. Этот процесс называется копированием либо клонированием.

Можно копировать объекты и стандартным способом, выбирая пункты главного меню **Edit** ⇒ **Copy** и затем **Edit** ⇒ **Paste**.

Метки объектов

Метки используются для идентификации объектов. Среда LabVIEW имеет два вида меток: свободные и собственные. Собственные метки принадлежат объекту, описывают только его и двигаются вместе с ним. Собственную метку можно перемещать независимо от объекта, но при перемещении объекта метка

перемещается вместе с ним. Свободные метки не принадлежат объектам, их можно создавать, перемещать, вращать или удалять независимо. Они используются для описания объектов, ввода комментариев на лицевой панели и блок-диаграмме.

Для создания свободной метки используется инструмент ВВОД ТЕКСТА. Выбрав этот инструмент, необходимо щелкнуть в свободном пространстве одной из панелей и ввести текст. После ввода текста метки поместить курсор в пространство вне метки или нажать кнопку **Enter** на инструментальной панели.

Совет. По умолчанию нажатие на клавиатуре клавиши **Enter** добавляет новую строку. Чтобы закончить ввод текста с клавиатуры, следует нажать **Shift+Enter**. Можно закончить ввод текста с клавиатуры нажатием клавиши **Enter**, для этого в пункте главного меню следует выбрать **Tools ⇒ Options**, далее, в выпадающем меню найти **Front Panel** и отметить пункт **End text entry with Return key**.

Выделение и удаление проводников данных

Сегмент проводника данных – это отдельная горизонтальная или вертикальная его часть. Место соединения двух сегментов – излом проводника данных. Точка, в которой встречаются два, три или четыре проводника данных называется точкой соединения.

Проводник данных содержит все сегменты между точками соединения, между терминалом данных и точкой соединения, между терминалами данных, если нет точек соединений. Для выделения сегмента используется инструмент ПЕРЕМЕЩЕНИЕ. Двойной щелчок мыши выделяет проводник данных, тройной щелчок – выделяет множество проводников данных (рис. 2.3).

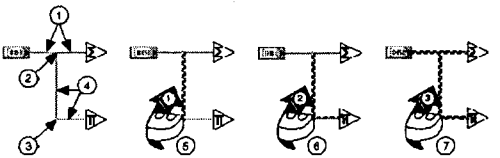


Рис. 2.3

1. *Сегмент*
2. *Точка соединения*
3. *Излом проводника данных*
4. *Проводник данных*
5. *Выделенный сегмент*
6. *Выделенный проводник данных*
7. *Выделенное множество проводников данных*

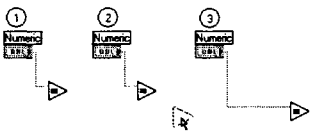


Рис. 2.4

Автомасштабирование проводников данных

Как показано на рис. 2.4, перемещение объектов не приводит к нарушению проводника данных.

Разорванные проводники данных

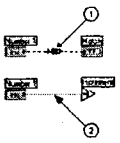


Рис. 2.5

1. Нарушенный проводник
2. Правильный проводник

Разорванный проводник данных выглядит, как черная штриховая линия с красным крестом посередине, как показано ниже. Разрыв проводников данных происходит по причинам разного рода. Например, при попытке соединения объектов с несовместимыми типами данных (рис. 2.5):

Описание причины разрыва проводника данных появляется в окне всплывающей подсказки после наведения на проводник инструмента СОЕДИНЕНИЕ. Тройной щелчок инструментом ПЕРЕМЕЩЕНИЕ на проводнике и последующее нажатие клавиши **Delete** удаляет выделенный проводник. Удаление всех разорванных проводников производится через пункт главного меню **Edit** ⇒ **Remove Broken Wires**.

Внимание. Использование пункта главного меню **Remove Broken Wires** требует определенной осторожности. Иногда проводник является разорванным потому, что еще не закончено создание блок-диаграммы.

Редактирование текста (изменение шрифта, стиля и размера)

Выбрав пункт меню **Text Settings** на инструментальной панели, можно изменить шрифт, стиль, размер и провести выравнивание любого текста внутри меток или на дисплеях элементов управления и отображения.

На некоторых элементах управления и отображения данных, текст может быть помещен более чем в одном месте, например оси графиков. В этом случае текст в каждом поле можно изменять независимо. Текст выделяется инструментом ВВОД ТЕКСТА, как показано на рис. 2.6, и на инструментальной панели выбирается пункт меню **Text Settings**.

Изменение размеров объектов

Большинство объектов лицевой панели допускают изменение размеров. Чтобы подготовить объект к изменению размера, необходимо навести на него инструмент ПЕРЕМЕЩЕНИЕ. По углам объекта появляются маркеры, показанные слева. Затем курсор следует установить на

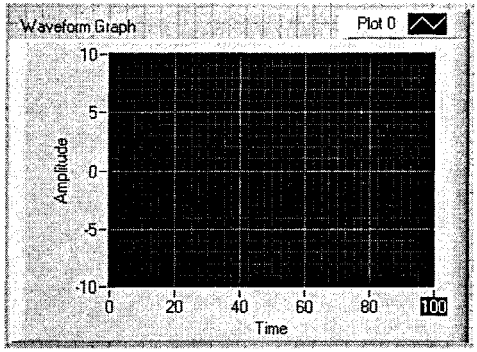


Рис. 2.6

один из маркеров и, удерживая нажатой левую кнопку мыши, переместить маркер, размер шрифта при этом не меняется. Промежуточные границы изменяемого размера обозначаются штриховой линией. Когда нужный размер элемента достигнут, кнопку мыши следует отпустить. Удержание клавиши **Shift** во время перемещения маркеров сохраняет пропорции объекта.

Можно изменять размеры и объектов блок-диаграммы, таких как структуры и константы.

Выравнивание и распределение объектов в пространстве

Выравнивание группы объектов по оси производится с помощью опций в пункте инструментальной панели **Align Objects**. Для равномерного распределения объектов в пространстве следует воспользоваться пунктом **Distribute Objects**.

Установка порядка размещения объектов, объединение объектов в группу и закрепление местоположения объектов на рабочем пространстве лицевой панели

В случае, когда объекты перекрывают друг друга, можно установить порядок размещения объектов – один впереди другого. Для этого объект следует выделить с помощью инструмента ПЕРЕМЕЩЕНИЕ и в пункте меню **Reorder** инструментальной панели выбрать необходимые установки: **Move Forward** (Поместить на передний план), **Move Backward** (Поместить на задний план), **Move To Front** (Передвинуть вперед), **Move To Back** (Передвинуть назад).

Для объединения объектов в группу и закрепления их местоположения на рабочем пространстве лицевой панели следует выбрать необходимые установки в пункте меню **Reorder** инструментальной панели: **Group** (Группировать), **Ungroup** (Разгруппировать), **Lock** (Блокировать), **Unlock** (Разблокировать).

Приведение нескольких объектов к одному размеру

Приведение нескольких объектов к одному виду производится с помощью выпадающего меню **Resize Objects** (Изменение размеров объектов). Предусмотрена возможность изменения размера всех выбранных объектов по ширине или высоте до ширины/высоты наименьшего или наибольшего объекта, также имеется возможность задать размер всех выбранных объектов в пикселях.

Отдельные объекты допускают изменения размера лишь по вертикали или горизонтали, например, числовые элементы управления и отображения; некоторые объекты сохраняют пропорции при изменении размера. Например, если среди объектов, выбранных для изменения размера по высоте, присутствует числовая константа, LabVIEW не изменит ее размер, изменив размер остальных объектов, допускающих изменение размера.

Копирование объектов между ВП или между другими приложениями

Копировать и вставлять объекты из одного ВП в другой можно выбором пунктов главного меню **Edit** ⇒ **Copy** и затем **Edit** ⇒ **Paste**. Возможно копирование изображения или текста из других приложений и их использование на лицевой панели или блок-диаграмме. Если оба ВП открыты, можно копировать выбранные объекты, перемещая их с одного ВП на другой.

Окрашивание объектов

Можно изменять цвет большинства объектов ВП, но не всех. Например, терминалы данных и проводники данных блок-диаграммы используют только определенные цвета, соответствующие типу представленных данных.

Изменение цвета объекта или фона рабочего пространства производится с помощью инструмента РАСКРАШИВАНИЕ. Для этого следует щелкнуть правой кнопкой мыши на выбранном элементе или рабочем пространстве любой из панелей. Можно изменить заданные по умолчанию цвета большинства объектов, выбирая пункты меню **Tools** ⇒ **Options** и затем **Colors**.

Можно также сделать объект прозрачным, выбрав **T** в меню **Colors**.

Выводы

На этой лекции слушатели получили начальные навыки практического построения виртуальных приборов в среде LabVIEW путем графического программирования и ознакомились с приемами исправления ошибок и редактирования собранных блок-схем.

Лекция 3

Решение линейных алгебраических уравнений в среде LabVIEW

Лекция является продолжением предыдущей, в ней рассматриваются способы построения виртуальных приборов для решения алгебраических задач. Дается представление об использовании формульного узла и о применении матричных методов.

Обратимся к ранее (в главе 2) введенной электрической схеме (рис. 2.1). Задача: составить уравнения для токов этой схемы и решить их различными способами: с использованием формульного узла (структуры LabVIEW, предназначенной для расчетов по формулам) и с применением матричного метода.

Пример 3.1. Определение токов в цепи с использованием формульного узла

1. Токи в цепи рис. 3.1 можно рассчитать по законам Ома и Кирхгофа

$$I_1 = E/(R_1 + \frac{R_2 R_3}{R_2 + R_3});$$

$$I_2 = R_3 I_1/(R_2 + R_3); \qquad (1)$$

$$I_3 = R_2 I_1/(R_2 + R_3).$$

   Значения сопротивлений R_1, R_2, R_3 и электродвижущей силы E выбираются самостоятельно.

2. Расчет по этим формулам можно выполнить при помощи формульного узла **Formula Node**, который относится к элементам «Структуры» и вызывается правой клавишей мыши на панели блок-диаграмм по пути: **All Functions** ⇒ **Structures** ⇒ **Formula Node**. Появившаяся рамка формульного узла растягивается до нужного размера и в нее вписываются расчетные формулы (1). Неизвестные записываются в левой части формул. Каждая формула пишется на отдельной строке и заканчивается точкой с запятой.

3. Затем в формулы нужно внести исходные данные и вывести результаты расчета. Для этого курсор устанавливается правой клавишей мыши на рамке формульного узла и из всплывающего меню левой клавишей вызывается **Add Input** (добавить вход) для входных величин и **Add Output** (добавить выход) для выходных величин. В появившиеся рамки вписываются наименования этих величин.
4. К входным рамкам подключаются цифровые управляющие элементы, к выходным – индикаторы. Входы и выходы можно устанавливать в любом месте рамки. Наименования в рамках должны быть точно такими же, как в формульном узле. Допускается применение одного и того же наименования для входной и выходной величины.
5. При помощи управляющих элементов задаются исходные данные, после чего схема запускается на решение.

Вид формульного узла показан на рис. 3.2.

Примечание. Формульный узел не применяется для работы с комплексными числами.

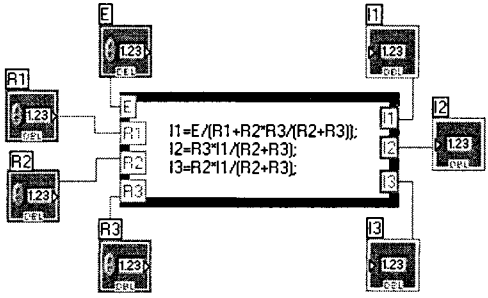


Рис. 3.2

Пример 3.2. Решение алгебраических уравнений в матричной форме

Расчет токов в цепи рис. 3.1 можно провести по линейным алгебраическим уравнениям, составленным по законам Кирхгофа:

$$\begin{aligned} -I_1 + I_2 + I_3 &= 0; \\ R_1 I_1 + R_2 I_2 &= E; \\ R_2 I_2 - R_3 I_3 &= 0 \end{aligned} \qquad (2)$$

Эти уравнения можно записать в матричной форме

$$\begin{bmatrix} -1 & 1 & 1 \\ R_1 & R_2 & 0 \\ 0 & R_2 & -R_3 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 0 \\ E \\ 0 \end{bmatrix}. \qquad (3)$$

Рис. 3.3

Для решения системы линейных алгебраических уравнений в среде LabVIEW существует элемент «Решение линейных уравнений», вызываемый по пути **All Functions ⇒ Analyze ⇒ Mathematics ⇒ Linear Algebra ⇒ Solve Linear Equatons**. Терминалы его можно раскрыть нажатием на иконку правой клавишей мыши и далее из всплывающего меню **Visible Items ⇒ Terminals**. Вид элемента, исходный и с открытыми терминалами, изображен на рис. 3.3.

Для определения назначения терминалов можно снова нажать на иконку правой клавишей мыши и открыть переднюю панель **Open Front Panel**. Передняя панель имеет вид, представленный на рис. 3.4. Уравнение (3) вводится таким образом: матрица коэффициентов **Input Matrix** (первая матрица уравнения) – подается на левый верхний терминал, вектор заданных воздействий **Known Vector** (правая часть) – подводится к левому среднему терминалу, а результирующий вектор **Solution Vector** (вектор искомых токов) – снимается с правого верхнего терминала.

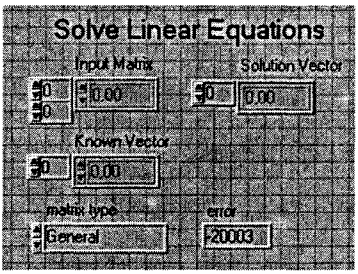


Рис. 3.4

Для того чтобы ввести матрицу, вызывается элемент **Array** (построение массива). Элемент вызывается на лицевой панели по пути **All Controls ⇒ Array&Cluster ⇒ Array**. Появляется пустая ячейка, в которую вносится цифровой управляющий элемент (для матрицы коэффициентов и вектора управляющих воздействий) или цифровой индикатор (для вектора искомых токов). Затем элемент растягивается до нужной размерности матрицы инструментом «перемещение» (стрелка). Окна управляющих элементов и индикаторов имеют серый цвет, после внесения туда цифровых данных они становятся белыми. Одновременно с вызовом элемента Array на лицевой панели появляется его иконка на панели блок-диаграмм. Вид лицевой панели и панели блок-диаграмм, где выполнены необходимые соединения, представлен на рис. 3.5.

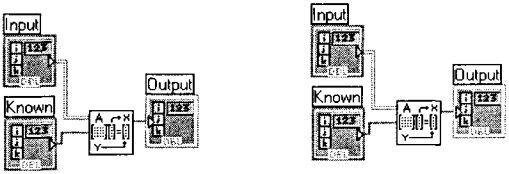


Рис. 3.5

В результате выполнения операции получается вектор решений **Output**. В этом столбце представлены значения токов $I_1 = 3\text{A}$, $I_2 = 2\text{A}$, $I_3 = 1\text{A}$.

Можно из этого столбца выделить значения отдельных неизвестных. Для этого используется функция извлечения элемента массива **All Functions ⇒ Array ⇒ Index Array**. Выходы функции подключаются к элементам индикации (рис. 3.6). После запуска программы индикаторы покажут величины, соответствующие значениям строк выходного вектора.

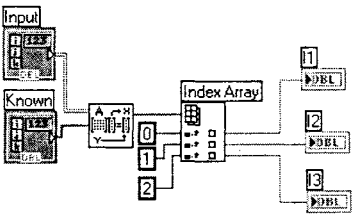


Рис. 3.6

Следует отметить, что программный инструмент **Solve Linear Equation**, кроме всего, обладает в некотором роде универсальными свойствами, так как может решать также и переопределенные и недоопределенные системы уравнений. В первом случае находится решение, наиболее удовлетворяющее уравнениям (по наименьшему небалансу), а во втором отыскивается одно из возможных решений.

Дополнение. Матричные операции в среде LabVIEW

Ввиду того, что вычислительные операции в матричной форме имеют исключительно важное значение, рассмотрим их подробнее.

LabVIEW поддерживает все основные матричные операции. По своей сути матрица является двумерным массивом, а значит, к ней применимы все операции по работе с многомерными массивами (подробнее работа с массивами будет рассмотрена в лекции 6).

Использование матриц и матричных вычислений обычно упрощает внешний вид и структуру программы, однако следует помнить, что массивы могут занимать в памяти значительное пространство, а операции над матрицами требуют для своей реализации большого числа алгебраических вычислений.

ВП для работы с матрицами находятся на панели **All Functions ⇒ Analyze ⇒ Mathematics ⇒ Linear Algebra**. Список функций по работе с матрицами приведен в табл. 3.1

Таблица 3.1

ВП	Название	Описание
	Solve Linear Equations	*Решение системы линейных алгебраических уравнений*
	Inverse Matrix	*Обращение матрицы*

Таблица 3.1 (окончание)

ВП	Название	Описание
	*Determinant*	*Вычисление определителя*
	EigenValues and Vectors	*Вычисление собственных чисел и векторов*
	*A x B*	*Перемножение матриц*
	*A x Vector*	*Умножение матрицы на вектор*
	*Dot Product*	*Скалярное произведение векторов*
	*Outer Product*	*Внешнее произведение векторов*

Различные матричные функции в LabVIEW имеют похожий набор входных и выходных параметров. Один из входов, **matrix type**, позволяет уточнить структуру исходной матрицы. Значения свойства matrix type приведены в табл. 3.2.

Таблица 3.2

Значение matrix type	Тип матрицы	
0	*General*	*Общего вида*
1	*Positive definite*	*Положительно определенная*
2	*Lower triangular*	*Нижняя треугольная*
3	*Upper triangular*	*Верхняя треугольная*

Если этот вход оставлен неподключенным, считается, что матрица имеет общий вид. Параметр matrix type относится к перечислимому типу, поэтому если нажать правой кнопкой мыши на этом входе любого ВП для работы с матрицами и выбрать из контекстного меню пункт **Create Constant**, значение этого параметра можно будет выбирать по текстовым названиям.

Указав тип матрицы, можно существенно повысить скорость выполнения программы.

На панели **Linear Algebra** имеются еще две функции: **Dot Product** и **Outer Product. Dot Product** (скалярное произведение) считает первый вектор строкой, а второй столбцом и вычисляет сумму произведений элементов векторов. **Outer Product** (внешнее произведение) формирует матрицу из произведений взаимно ортогональных элементов.

В полной версии LabVIEW на панели **Linear Algebra** имеется еще функция, предназначенная для вычисления собственных чисел и собственных векторов матриц. К ее входу помимо самой матрицы подключаются два специальных признака. Первый из них **matrix type** указывает тип матрицы, причем здесь, в отличие от других матричных операций, предусмотрено всего два типа матриц: общего вида и симметричная. Если на вход подается симметричная матрица, в качестве **matrix type** следует указать единицу. Второй признак output option определяет, нужно ли вычислять собственные векторы: если к нему подключить 0, будут вычислены только собственные числа, в противном случае и собственные числа и собственные векторы.

Полная версия LabVIEW содержит на панели **Linear Algebra** еще две вспомогательные панели: **Complex Linear Algebra** и **Advanced Linear Algebra**. Первая панель содержит те же инструменты, что и панель **Linear Algebra**, но предназначенные для работы с комплексными числами.

Вторая содержит более сложные функции, список которых приведен в табл. 3.3.

Таблица 3.3

VI	Название	Описание
	*LU Factorization*	*LU-(LH-)разложение матрицы*
	*QR Factorization*	*QR-разложение матрицы*
	*SVD Factorization*	*Сингулярное разложение*
	*Cholesky Factorization*	*Разложение Холецкого*
	*Trace*	*Вычисление следа матрицы*
	*Matrix Rank*	*Определение ранга матрицы*
	*Matrix Norm*	*Вычисление нормы матрицы*
	*Matrix Condition Number*	*Вычисления числа обусловленности матрицы*
	*PseudoInverse Matrix*	*Вычисление псевдообратной матрицы*
	*Create Special Matrix*	*Создание специальной матрицы*
	*Test Positive Definite*	*Определение типа матрицы*

Выводы

В этой лекции продолжено изучение способов построения виртуальных приборов для решения алгебраических уравнений с применением формульного узла и матричной схемы «Решение линейных уравнений». Дан обзор матричных операций в среде LabVIEW.

Лекция 4

Моделирование и измерение переменных напряжений и токов в среде LabVIEW

Среда LabVIEW предназначена в основном для измерений в реальных устройствах, и прежде всего в электротехнических установках переменного тока. Цель настоящей лекции – получить предварительные сведения о виртуальном моделировании напряжений и токов, изменяющихся во времени по синусоидальному закону, и об измерениях в цепях переменного тока на основе виртуальных приборов LabVIEW.

Моделирование синусоидальных токов и напряжений

В работе моделируются напряжения и токи, представляющие собой синусоидальные функции времени

$$u = U_m \sin \omega t\ ,\ \ i = I_m \sin(\omega t - \varphi)\ , \qquad (1)$$

а также мгновенная мощность

$$p = ui \qquad (2)$$

и активная мощность P.

Моделирование в среде LabVIEW можно выполнить различными способами: вызвать генератор синусоидальных колебаний, или записать выражение синусоидальной функции в формульном узле, либо вызвать синусоидальную функцию. В данной работе предусмотрены эти три способа задания напряжений и токов, изменяющихся по синусоидальному закону. Напряжения, токи и мощности наблюдаются на виртуальных осциллографах.

Переменные токи и напряжения характеризуются их действующими значениями. Как известно, действующее значение (среднеквадратичное) связано с амплитудным соотношением $U = U_m / \sqrt{2}$. Активная мощность (среднее за период значение мгновенной мощности) определяется выражением $P = UI \cos\varphi$.

В системе LabVIEW существуют специальные виртуальные приборы для определения действующих и средних за период значений измеряемых величин. Для измерения действующих значений (среднеквадратичных) применяется прибор

RMS (Root mean square). Средняя за период величина измеряется прибором Mean. Показания этих приборов считываются при помощи цифровых индикаторов.

Пример 4.1

Требуется смоделировать синусоидальное напряжение при помощи виртуального генератора синусоидальных колебаний.

1. С этой целью нужно вызвать генератор на панели блок-схем по пути **Functions ⇒ Analyse ⇒ Waveform Generation ⇒ Sine Waveform**. Для задания частоты, амплитуды и начальной фазы напряжения следует создать три цифровых источника напряжения (**Controls ⇒ Numeric ⇒ Digital Control**).
2. Наблюдение полученной кривой осуществляется при помощи виртуального осциллографа, который вызывается с лицевой панели (**Controls ⇒ Graph ⇒ Waveform Graph**).

   Цикл по заданию (с фиксированным числом итераций) **For Loop** выполняет повторяющиеся операции над потоком данных определенное количество раз.

   N Цикл **For** расположен в палитре функций в разделе **Functions ⇒ Structures**. Значение, присвоенное терминалу N цикла, показанному слева, определяет максимальное количество повторений операций над потоком данных.

   i Терминал счетчика показанный слева, содержит значение количества выполненных операций. Начальное значение счетчика итераций всегда равно 0.

   Цикл **For** завершает работу, выполнив заданное максимальное число итераций *N*.
3. Для подключения генератора следует открыть его терминалы, для чего нужно нажать правой клавишей мыши на его иконку и вызвать **Visible Items ⇒ Terminals**, после чего снова нажать на иконку и вызвать Help, откуда определить точки подключения источников. После сборки блок-схемы следует задать амплитуду напряжения и фазу (в градусах) и установить частоту (по умолчанию частота 10 Гц, на экране осциллографа один период колебаний; задание $f = 2$ означает двойную частоту, на экране будет изображено два периода).
4. Задавая различные (по собственному выбору) амплитуды, фазы и частоты, построить несколько осциллограмм синусоидального напряжения.

Пример 4.2

Собрать блок-схему для моделирования двух синусоидальных величин – напряжения и тока с применением синусоидальных функций и цикла по заданию **For Loop** (рис. 4.1).

Рассмотрим вычисление синусоидальных функций на протяжении периода с интервалом 1 градус, при этом $N = 360$.

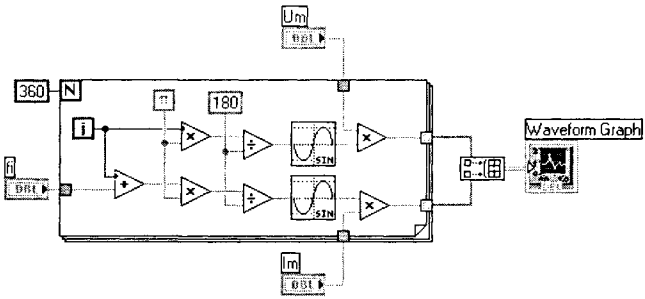


Рис. 4.1

1. Цикл вызывается на панели блок-схем: **Functions ⇒ Structures ⇒ For Loop**. Рамку цикла нужно растянуть за уголок до нужного размера инструментом «перемещение» («стрелка»).
2. Внутрь цикла вносятся иконки синусоидальных функций, одна из которых изображает напряжение, а другая ток (**Functions ⇒ Numeric ⇒ Trigonometric ⇒ Sine**). На вход их подаются значения углов в радианах, поэтому при градусной мере задания углов их следует перевести в радианы по известному соотношению $\varphi_{рад} = \pi\varphi_{град}/180$.
3. Количество вычислений за один период в нашем случае $N = 360$ (к терминалу N нужно подключить источник с числом 360). Текущий параметр цикла i (синяя буква в рамке) соответствует количеству градусов $i = \varphi_{град}$ и изменяется от 0 до 360.
4. Начальная фаза задается прибавлением нужного количества градусов к параметру соответствующей синусоиды.
5. Выходные величины синусоидальных функций умножаются соответственно на амплитудные значения напряжения и тока и наблюдаются на виртуальном осциллографе.
6. В работе предлагается создать прибор для одновременного наблюдения двух синусоидальных функций – виртуальный двухлучевой осциллограф. Для этого на лицевой панели вызывается осциллограф **Waveform Graph** и на нем устанавливается второй график – для этого нужно инструментом «перемещение» растянуть окошко **Plot** по вертикали. Затем следует нажать правой клавишей мыши на иконку осциллографа на панели блок-схем и вызвать построитель массива **Array Tools ⇒ Build Array**, который нужен для объединения двух сигналов на входе осциллографа. Появившийся построитель массива растянуть по вертикали так, чтобы он имел два входа.
7. Запустить программу на выполнение. Построить кривые тока и напряжения при различных фазовых сдвигах (положительных и отрицательных).

Примечание. Двухлучевой осциллограф имеет общую шкалу ординат, поэтому его целесообразно применять в тех случаях, когда амплитудные значения измеряемых величин соизмеримы по абсолютной величине.

Пример 4.3

Собрать блок-схему для моделирования напряжения, тока и мощности в цепи синусоидального тока с применением формульного узла и цикла по заданию.

1. Для решения задачи следует вызвать цикл по заданию (**For Loop**) и поместить в него формульный узел (**Functions ⇒ Structures ⇒ Formula Node**), в который вписать формулу $u = U_m \sin \omega t$, где аргумент ωt в радианах записывается как $\omega t = k\pi / 180$. Здесь параметр счетчика операций обозначен буквой k для того, чтобы отличить его от обозначения тока. Для замены достаточно установить инструмент «ввод текста» (курсор А) на место синей буквы i и нажать k. Параметр k соответствует числу градусов $k = \varphi_{град}$. В нашем случае целесообразно провести расчет на протяжении одного – двух периодов. Поэтому нужно задать число отсчетов N = 360 или 720.
2. Нажав правой клавишей мыши на рамку формульного узла, вводим входные и выходные величины (**Add Input, Add Output**).
3. Полученные значения напряжения, тока и мгновенной мощности нужно вывести на осциллографы **Waveform Graph**.
4. Действующие (среднеквадратичные) значения синусоидального напряжения и тока измеряются виртуальными приборами RMS (Root Mean Square), вызываемыми на панели блок-схем по пути **Functions ⇒ Analyze ⇒ Mathematics ⇒ Probability and Statistics ⇒ RMS**, к выходам которых подключаются цифровые индикаторы. Активная мощность, представляющая собой среднее за период значение мгновенной мощности, измеряется виртуальным прибором Mean, вызываемым аналогично: **Functions ⇒ Analyze ⇒ Mathematics ⇒ Probability and Statistics ⇒ Mean**; на выходе также требуется индикатор.

Блок-диаграмма изображена на рис. 4.2.

Нужно построить кривые напряжения, тока и мгновенной мощности при различных сдвигах фаз между напряжением и током. Результаты измерений можно сравнить с результатами расчетов по формулам (1), (2).

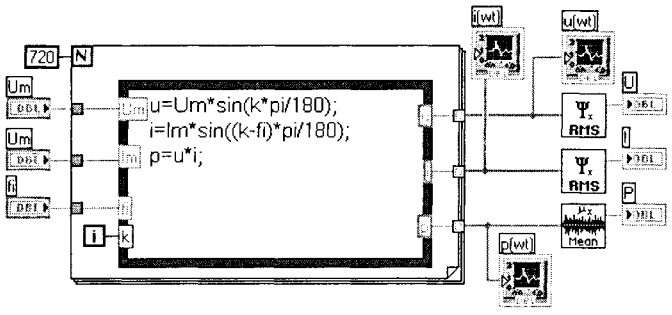


Рис. 4.2.

Пример 4.4

При многих исследованиях требуется не только измерить значение данной величины, но и построить ее характеристику при изменении какого-либо параметра в широких пределах. Такого типа задачи при виртуальном моделировании также решаются с помощью цикла.

Рассматривается последовательное соединение катушки, индуктивность которой L, конденсатора емкостью C и резистора с сопротивлением R. Цепь подключена к источнику синусоидальной ЭДС E с переменной частотой ω. Требуется построить зависимости тока в цепи и напряжений на катушке и конденсаторе от частоты при неизменном напряжении U.

1. Ток и напряжения определяются по закону Ома

$$I = U/z;\ \ U_L = \omega L I;\ \ U_C = I/\omega C, \tag{3}$$

где $z = \sqrt{R^2 + (\omega L - 1/\omega C)^2}$. (4)

Пусть заданы числовые значения: U = 100 В, L = 0,025 Гн, C = 0,004 Ф, R = 1 Ом.

Для решения задачи используется формульный узел, работающий в цикле по заданию. В формульный узел записываются уравнения (3), (4). В качестве изменяющейся частоты ω можно применить счетчик операций i. Здесь нужно заметить, что в формулах (3), (4) частота содержится в знаменателе, поэтому для того чтобы избежать появления в знаменателе нуля, целесообразно полагать ω = i+1. В качестве индикатора рекомендуется взять трехлучевой осциллограф, который подключается по аналогии с тем, как это было сделано в примере 4.2. Количество операций N выбирается таким образом, чтобы наблюдались резонансные пики тока и напряжений. В нашем случае принято N = 200.

Блок-схема изображена на рис. 4.3, наблюдаемые резонансные характеристики – на рис. 4.4.

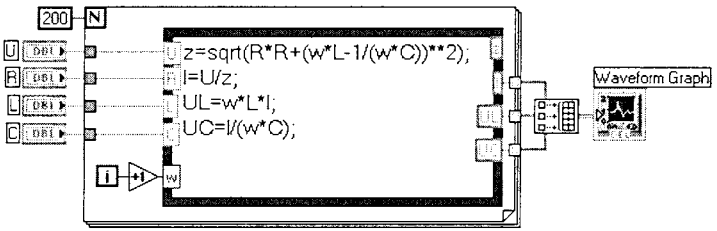


Рис. 4.3

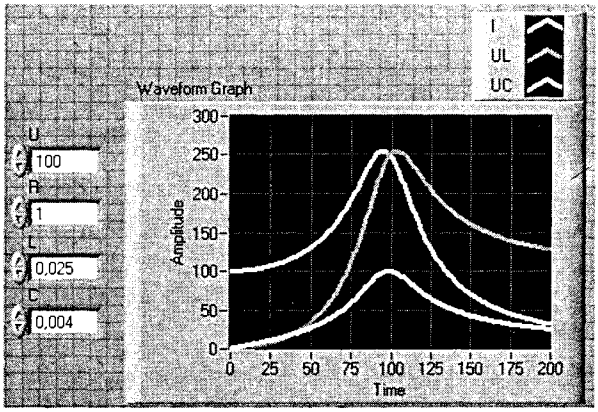


Рис. 4.4

Выводы

Слушатели ознакомились с моделированием переменных напряжений и токов, получили представление о самых простых виртуальных измерительных приборах, о графических индикаторах (виртуальных осциллографах) – однолучевых и многолучевых. В процессе работы изучен цикл по заданию **For Loop**. Показан способ построения характеристик.

Лекция 5

Численное решение обыкновенных дифференциальных уравнений в среде LabVIEW

Цель лекции – ознакомление слушателей с простейшими методами численного интегрирования дифференциальных уравнений, описывающих переходные процессы в электрических цепях, и с моделированием этих процессов в среде LabVIEW. При моделировании используется регистр сдвига. Проводится построение временных характеристик переходного процесса. Рассматривается понятие устойчивости вычислительного процесса. Материал излагается на примере переходного процесса в цепи RC.

Расчетные алгоритмы

Рассматривается переходный процесс в электрической цепи, состоящей из источника постоянной электродвижущей силы E, резистора с сопротивлением R и конденсатора емкостью C (рис. 5.1). Величины E, R, C можно задать самостоятельно. Начальное значение напряжения на конденсаторе $u(0)$ принято равным 0. Требуется определить напряжение $u(t)$ и ток в цепи $i(t)$ при $t \geq 0$, полагая, что в момент $t = 0$ замыкается ключ K.

На основании второго закона Кирхгофа ток в цепи

$$i = (E - u)/R, \qquad (1)$$

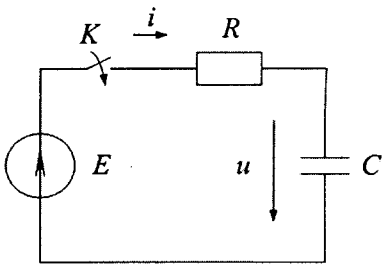


Рис. 5.1

откуда после подстановки значения тока $i = Cdu/dt$ получается дифференциальное уравнение для определения переходного процесса

$$du/dt = (E-u)/RC. \qquad (2)$$

Решение данного дифференциального уравнения проводится с помощью программы LabVIEW, составленной с использованием явного либо неявного метода Эйлера интегрирования дифференциальных уравнений.

Для этого перейдем к конечным приращениям $dt = \Delta t$, $du = \Delta u$ и определим приращение напряжения на последующей итерации как

$u_{k+1} = u_k + \Delta u_k$ для явного метода Эйлера,
$u_{k+1} = u_k + \Delta u_{k+1}$ для неявного метода Эйлера,
где k – номер шага интегрирования, $u_{k+1} = u(t_{k+1})$, $u_k = u(t_k)$.
Здесь значения приращений

$$\Delta u_k = ((E-u_k)/(RC))\Delta t,$$

$$\Delta u_{k+1} = ((E-u_{k+1})/(RC))\Delta t;$$

шаг интегрирования $\Delta t = h$.

Таким образом, алгоритм явного метода Эйлера для расчета напряжения в переходном режиме имеет вид

$$u_{k+1} = u_k + ((E-u_k)/(RC))\Delta t, \qquad (3)$$

в случае неявного метода

$$u_{k+1} = u_k + ((E-u_{k+1})/(RC))\Delta t.$$

Выражая отсюда u_{k+1}, получаем алгоритм неявного метода Эйлера

$$u_{k+1} = (u_k + E\Delta t/(RC))/(1+\Delta t/(RC)). \qquad (4)$$

После определения напряжения ток вычисляется по уравнению (1).

Известно, что явный метод Эйлера неустойчив. Действительно, нетрудно видеть, что при больших шагах расчета $\Delta t > 2rC$ уравнение (3) дает расходящееся решение – это легко просчитать, задав, например, $\Delta t = 3rC$. Что касается неявного метода Эйлера, то он устойчив, то есть при любом значении Δt даст решение, стремящееся к установившемуся режиму. Так, из уравнения (4) следует, что даже при Δt, стремящемся к бесконечности, u_{k+1} стремится к E. Однако, при больших шагах Δt погрешность вычисления характеристик переходного процесса $u(t)$ и $i(t)$ очень велика. Поэтому в нашей работе шаг расчета Δt должен быть значительно меньше постоянной времени переходного процесса τ (в нашем случае $\tau = rC$). Целесообразно его ориентировочное значение взять примерно на 2 порядка меньше, чем постоянная времени переходного процесса τ. Количество итераций N примерно должно соответствовать 3τ.

Пример 5.1

Построить кривые напряжения $u(t)$ и тока $i(t)$ в цепи рис. 5.1 с использованием алгоритма явного метода Эйлера. Параметры цепи можно выбрать самостоятельно. Напряжение определяется по уравнению (3), ток – по уравнению (1).

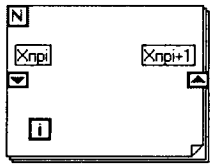


Рис. 5.2.

1. Для выполнения работы используется цикл **For Loop** (петля по заданию). Вызов цикла **For Loop** осуществляется через **Functions** ⇒ **Structures** ⇒ **For Loop**.
2. Рамку цикла нужно развернуть до необходимого размера и добавить регистр сдвига **Shift Register**, который осуществляет передачу результата вычислений из предыдущей операции в последующую. Для вызова регистра сдвига следует установить курсор на боковой границе рамки цикла и ПКМ вызвать команду **Add Shift Register**. Появившийся символ треугольника в рамке слева означает искомую переменную (u или i) на предыдущем шаге интегрирования, справа – на последующем шаге (рис. 5.2). Начальное условие (в нашем случае нулевое) задается цифровым управляющим элементом, подключаемым к входному (левому) контакту регистра сдвига.

Примечание. В среде LabVIEW номер операции обозначается буквой i в синей рамке внутри цикла **For Loop**. Для того, чтобы ток и номер операции не обозначались одной и той же буквой, рекомендуется вписать в эту рамку букву k инструментом «ввод текста» (курсором A).

3. Внутрь цикла при помощи формульного узла, вызываемого по пути **Functions** ⇒ **Structures** ⇒ **Formula Node**, записываются расчетные формулы (3) для расчета напряжения и (1) для расчета тока (после каждой формулы ставится точка с запятой). Поскольку строятся зависимости тока и напряжения от времени, сюда же следует ввести значение времени

$$t = k\Delta t \,. \qquad (5)$$

4. Для ввода исходных данных курсор ставится на рамку формульного узла и по пути **ПКМ** ⇒ **Add Input** на рамке появляются окошки, куда вписываются буквенные обозначения, имеющиеся в формулах. К этим окошкам присоединяются цифровые управляющие элементы с соответствующими числовыми данными. Выходные величины выводятся в окошки, вызываемые аналогично по пути **ПКМ** ⇒ **Add Output**. К этим окошкам подключаются индикаторы (виртуальные осциллографы).
5. Для наблюдения зависимости напряжения и тока от времени при переходном процессе нужно вызвать два виртуальных двухкоординатных осциллографа (на лицевой панели: **Controls** ⇒ **Graph** ⇒ **XY Graph**). Для того, чтобы на вход осциллографа можно было подать две величины – время и напряжение (или ток), применяется кластер (связка), вызываемый на панели блок-схем нажатием ПКМ на иконку осциллографа и далее **Cluster Tools** ⇒ **Bundle**. К верхнему входу кластера подводится сигнал t, к нижнему – $u(t)$ или $i(t)$, выход подается на осциллограф.
6. После задания значений ЭДС, сопротивления, емкости, шага расчета Δt и количества операций N программу можно запустить на исполнение и рассчитать переходные процессы в цепи при различных значениях шага расчета в диапазоне примерно от $0{,}1t$ до $3t$. При наблюдении переходных процессов на экране

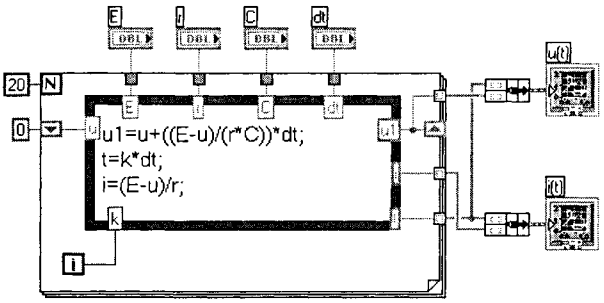


Рис. 5.3.

виртуального осциллографа нужно обратить внимание на картину процесса в области неустойчивости метода.

Блок-схема расчета по алгоритму явного метода Эйлера изображена на рис. 5.3.

Пример 5.2

Самостоятельно собрать блок-схему расчета переходного процесса в цепи неявным методом Эйлера. В этом случае напряжение определяется по уравнению (4), ток – по уравнению (1).

1. С этой целью можно скопировать блок-схему, собранную при выполнении п. 3 и заменить там уравнение (3) на уравнение (4). Для копирования схему нужно выделить с помощью инструмента «перемещение» («стрелка») (появляется пунктирная рамка) и при нажатой клавише **Ctrl** перенести на новое место.

Примечание. В старых версиях LabVIEW источники и приемники при такой операции не копируются (любой элемент может быть скопирован или удален только с той панели, с которой вызван). Поэтому остаются оборванные провода, которые можно удалить все сразу нажатием **Ctrl+B**.

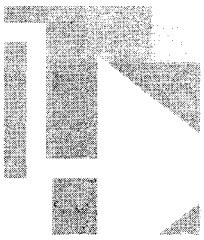
2. Запустить программу на исполнение. Рассчитать переходный процесс в цепи при различных значениях шага расчета в диапазоне примерно от 0,1τ до 3τ. Наблюдать на экране виртуального осциллографа переходный процесс в цепи. Обратить внимание на то, что картина процесса во всех случаях имеет одинаковый характер, область неустойчивости отсутствует.

Выводы

Слушатели вкратце ознакомились с численным интегрированием дифференциальных уравнений в среде LabVIEW. Была рассмотрена методика применения регистра сдвига при численных расчетах в цикле. Получен навык включения двухкоординатного виртуального осциллографа.

Лекция 6

Массивы



Вводятся понятия массивов данных и математических операций над массивами. Дается технология работы с массивами данных.

Массив – это набор элементов определенной размерности. Массивы объединяют элементы одного типа данных. Элементами массива называют группу составляющих его объектов. Размерность массива – это совокупность столбцов (длина) и строк (высота), а также глубина массива. Массив может быть одномерным (вектор), двумерным (матрица) или многомерным, и содержать до $2^{31}-1$ элементов в каждом направлении, насколько позволяет оперативная память.

Данные, составляющие массив, могут быть любого типа: численные, логические или строковые. Массив также может содержать элементы графического представления данных и кластеры. Массивы удобно использовать при работе с группами данных одного типа и при накоплении данных после повторяющихся вычислений.

Все элементы массива упорядочены. Каждому элементу присвоен индекс, причем нумерация элементов массива всегда начинается с 0. Таким образом, индексы массива находятся в диапазоне от 0 до (n-1), где n – число элементов в массиве.

Создание массива элементов управления и индикации

Для создания массива элементов управления или индикации данных необходимо выбрать шаблон массива из палитры **Controls** ⇒ **Array & Cluster** и поместить его на лицевую панель. Затем в шаблон массива поместить элемент управления или индикации данных (см. рис. 6.1). При этом терминал элемента на блок диаграмме приобретет цвет, соответствующий типу данных элементов массива.

Поместить элемент в шаблон массива следует до того, как он будет использоваться на блок-диаграмме. Если этого не сделать, то шаблон массива не будет инициализирован, и использовать массив будет нельзя.

Подобным образом можно создать массив-константу. Для этого необходимо выбрать шаблон **Functions** ⇒ **Array** ⇒ **Array constant** и поместить в него константу необходимого типа.

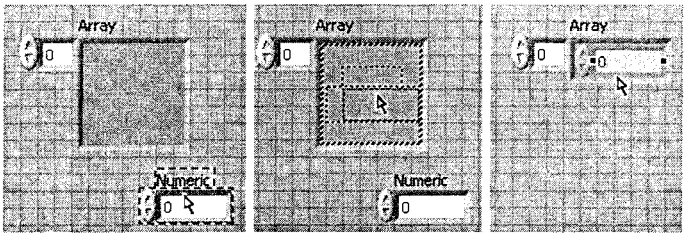


Рис. 6.1

На лицевой панели массив представляется двумя областями: зона индекса и зона видимости элементов. Сразу после создания массива виден только один элемент. Для того, что бы увидеть несколько элементов массива необходимо с помощью инструмента перемещение растянуть зону видимости элементов в горизонтальном или вертикальном направлении.

На рис. 6.2 показан массив, состоящий из элементов управления **Controls ⇒ Boolean ⇒ Push Button**. Свойства элементов, входящих в массив, можно редактировать непосредственно в зоне видимости элементов, как если бы элемент управления или индикации находился вне массива. Например, для элементов управления изменить размер (рис. 6.3).

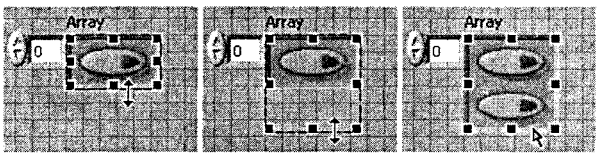


Рис. 6.2

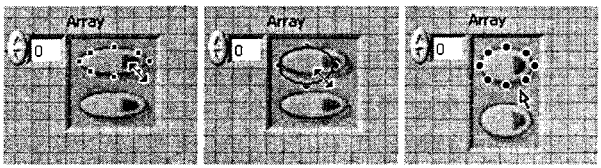


Рис. 6.3

Обратите внимание на то, что у всех элементов массива различаются только их значения, а все свойства: размер, цвет, точность, представление и т.д. могут быть только одинаковыми. Изменяя свойство у одного из элементов массива, вы изменяете свойства всех элементов.

В зоне индекса задается номер элемента массива, начиная с которого показываются элементы массива в зоне видимости элементов, т.е. индекс левого верхнего отображенного элемента. По умолчанию это значение 0. Это значит, что элементы массива показаны, начиная с нулевого элемента. Изменяя значение индекса можно наблюдать любой последовательный участок массива.

При желании можно удалить зону индекса. Для этого необходимо вызвать контекстное меню и выбрать пункты **Visible Items ⇒ Index Display**.

Двумерные массивы

Двумерный (2D) массив представляет собой прямоугольную таблицу (матрицу). Каждый элемент двухмерного массива характеризуется двумя индексами. Пример двухмерного массива размерностью 6 × 4 показан на рис. 6.4.

Для увеличения размерности массива необходимо щелкнуть правой кнопкой мыши по элементу индекса и выбрать из контекстного меню пункт **Add Dimension**.

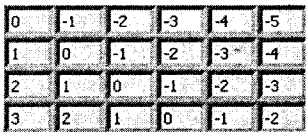


Рис. 6.4

Также можно использовать инструмент перемещение. Для этого надо просто изменить размер элемента индекса. Таким образом, можно увеличить размерность массива с одномерного до двумерного и выше, при этом зона видимости элементов становиться двумерной (рис. 6.5).

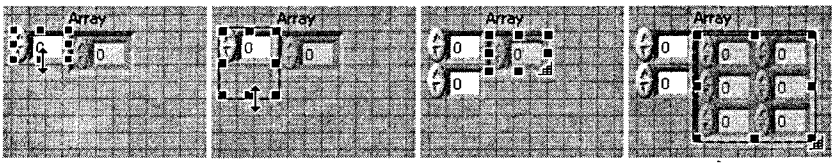


Рис. 6.5

Следует отметить, что для массивов размерностью от 3 и выше в зоне видимости элементов показывается двумерный срез массива. При этом числа в элементе индекса будут указывать индекс (координаты) левого верхнего отображаемого элемента.

На блок-диаграмме массив изображается утолщенным проводником, толщина которого зависит от размерности массива (рис 6.6).

– одномерный массив
– двумерный массив
– трехмерный массив.

Рис. 6.6

Цвет проводника соответствует типу элементов массива. При использовании массивов для согласования типов источника и приемника данных достаточно чтобы они имели одинаковую размерность, и элементы массивов были одного типа.

Математические функции (полиморфизм)

Для выполнения простейших математических операций над элементами массива можно использовать стандартные функции, расположенные в палитре **Functions** ⇒ **Numeric**. Все они являются полиморфными. Это означает, что на поля ввода этих функций могут поступать данные различных типов (скалярные величины, массивы). Например, можно использовать функцию **Add** для прибавления скалярной величины к массиву или сложения двух массивов. Если на одно поле ввода данных функции **Add** подать скалярную величину 2, а другое поле соединить с массивом, то функция прибавит 2 к каждому элементу массива.

Если на вход функции **Add** подать два массива одинаковой размерности, функция сложит каждый элемент первого массива с соответствующим элементом второго. Если с помощью функции **Add** сложить два массива разного размера, то функция сложит каждый элемент первого массива с соответствующим элементом второго и выдаст результат в виде массива с размером меньшего из двух исходных (рис. 6.7).

В LabVIEW, в отличие от большинства языков программирования, для того чтобы производить вычисления с элементами массивов, не потребуется использовать цикл. Большинство функций полиморфны и работают с массивами так же как со скалярными величинами. Например, для вычисления синуса от каждого элемента массива достаточно подать этот массив на вход соответствующей функции (рис. 6.8).

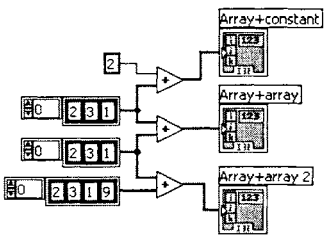
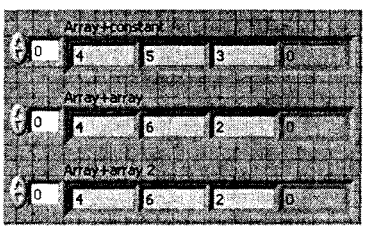


Рис. 6.7



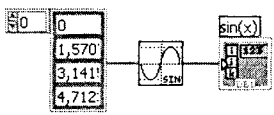


Рис. 6.8

Основные функции работы с массивами

Для работы с массивами предназначены следующие функции из палитры **Functions** ⇒ **Array**:

Таблица 6.1

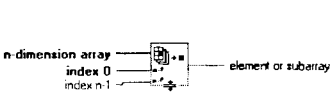
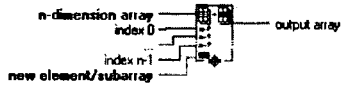
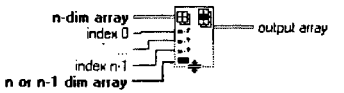
	***Array Size*** *– возвращает вектор размеров массива. Если массив n-мерный, на выходе функции Array Size будет вектор из n элементов. Так для одномерного массива из трех элементов функция* ***Array Size*** *выдаст значение 3, для двухмерного размером 5 × 10 результатом функции будет вектор из двух элементов 5 и 10.*
	***Index Array*** *– выдает элемент, соответствующий индексу, значение которого подается на поле ввода index. Функцию* ***Index Array*** *можно*

Таблица 6.1 (окончание)

использовать для выделения строки или столбца из двумерного массива и дальнейшего представления в виде подмассиво. Для этого надо подать двумерный массив на поле ввода данных функции. Функция **Index Array** должна иметь два поля **index**. Верхнее поле **index** указывает строку, а нижнее поле – столбец. Можно задействовать оба поля **index** для выбора отдельного элемента или только одна поле, для выбора строки или столбца.

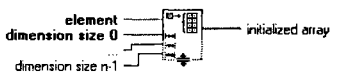


Replace Array Subset – заменяет часть массива, т.е. помещает значение или массив, поданный на терминал **new element/subarray** в исходный массив по координатам в полях **index**. Если не присоединять значений к терминалам **index** для какой-нибудь координаты, то будут заменены все элементы по этой координате. Выходной массив будет иметь одинаковую размерность и размер со входным.

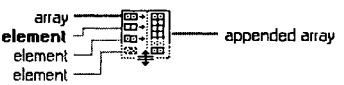


Insert Into Array – вставляет элемент или массив в исходный массив по координатам указанным в полях **index**. Если не присоединить проводники к терминалам **index**, то новые элементы добавятся в конец массива.

Delete From Array – удаляет элементы из массива начиная с номера элемента **index** и длинной **length**. На выходе результирующий массив и удаленная часть.

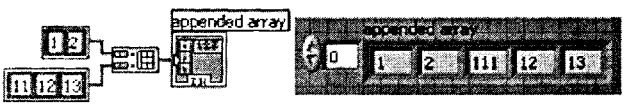


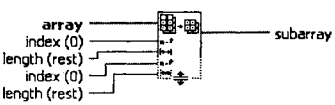
Initialize Array – создает массив заданной размерности, в котором каждый элемент инициализирован значением поля ввода данных **element**. Для увеличения размерности массива достаточно добавить поля ввода данных, растянув узел функции. Например, если для функции **Initialize Array** заданы следующие значения параметров: на поле element подается значение 4, а на поле **dimension size** значение 3, то на выходе получится одномерный массив, состоящий из трех элементов равных 4.



Build Array – объединяет несколько массивов или добавляет элемент в n-мерный массив. Изменение размера иконки функции увеличивает количество полей ввода данных, что позволяет увеличить количество добавляемых элементов. Например, можно получить из двух одномерных массивов двумерный:

Для объединения входных данных в массив той же размерности достаточно щелкнуть правой кнопкой мыши на функции и выбрать из контекстного меню пункт **Concatenate Inputs**:





Array Subset – выдает часть массива, начиная с индекса, поступившего на поле **index**, и длиной, указанной в поле **length**. Когда вы присоединяете массив к этой функции, узел меняет размер, автоматически создавая пару терминалов **index** и **length** для каждой координаты массива.

Автоматическое масштабирование функций работы с массивами

Функции **Index Array, Replace Array Subset, Insert Into Array, Delete From Array** и **Array Subset** при присоединении к ним массива автоматически изменяют количество входных терминалов, подстраиваясь под соответствующую размерность входного массива. Например, если к ним присоединить одномерный массив функция показывает один терминал для порядкового номера элемента, если двухмерный то два, один для номера строки второй для номера столбца и т.д. Так же можно получить на выходе не один элемент, а несколько (см. рис. 6.9). Для этого необходимо изменить размер иконки соответствующей функции, воспользовавшись инструментом перемещение.

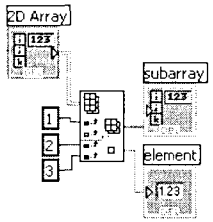


Рис. 6.9

Дополнительные функции работы с массивами

Таблица 6.2

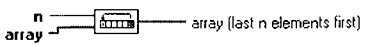
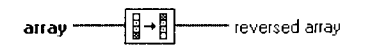
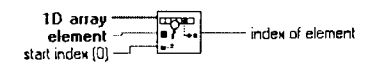
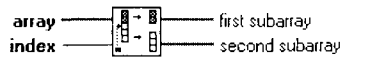
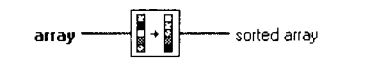
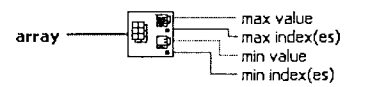
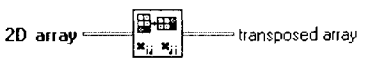
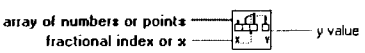
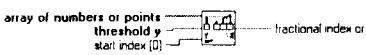
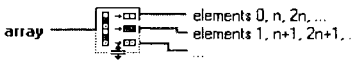
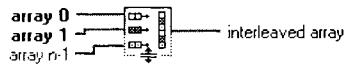
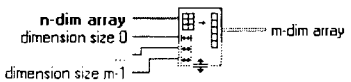
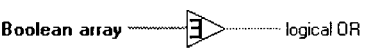
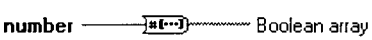
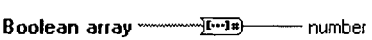
	***Rotate 1D Array*** *– при N>0 перемещает n последних элементов одномерного массиво в ночало, а остольные сдвигоет впрово на N, при N<0 происходит подобный сдвиг с переносом, но влево*
	***Reverse 1D Array*** *– перестановливает элементы одномерного массива в обратном порядке*
	***Search 1D Array*** *– поиск в массиве элементо со значением* ***element*** *ночиноя с* ***start index***
	***Split 1D Array*** *– розбивает одномерный массив на дво. первый моссив содержит элементы с индексами до* ***index-1*** *включительно, второй – остальные элементы*
	***Sort 1D Array*** *– сортирует элементы одномерного массива в порядке их возрастония. Если массив состоит из кластеров, функция сортирует массив по первым элементам кластеров. Если первые элементы кластера одинаковые, функция сровнивоет вторые и последующие элементы*
	***Array Max & Min*** *– находит максимальный и минимальный элементы в массиве, выдает их значение и индексы*
	***Transpose 2D Array*** *– транспонирует двумерный массив*
	***Interpolate 1D Array*** *– линейная интерполяция одномерного массива. На вход подается массив значений y и дробный индекс x, по котораму находится зночение y*
	***Threshold 1D Array*** *– функция обратная предыдущей, по задонным на входе массиву чисел* ***(array of numbers)*** *и порогу* ***(threshold)*** *вычисляет дробный индекс x*

Таблица 6.2 (окончание)

	***Decimate 1D Array*** *– разбивает исходный одномерный массив на n массивов, помещая в первый массив элементы с индексами 0, n, 2n... во второй массив 1, n+1, 2n+1... и т.д. где n – число выходов иконки функции*
	***Interleave 1D Arrays*** *– слияние массивов. Функция обратноя предыдущей*
	***Reshape Array*** *– изменяет размерность массива согласно количеству и значениям терминалов dimension size*

Функции для работы с массивами логических переменных

Таблица 6.3

	***And Array Elements*** *– Логическое «и» всех элементов массива*
	***Or Array Elements*** *– Логическое «или» всех элементов массива*
	***Number To Boolean Array,*** *и*
	***Boolean Array To Number*** *– две взаимообратные функции преобразующие целое число в двоичный код (в виде массива логических переменных) и обратно*

Выводы

Массив – удобный и гибкий тип данных позволяющий хранить, обрабатывать и отображать упорядоченный набор однотипных величин. Правильное понимание свойства полиморфизма функций может значительно упростить написание программ с использованием массивов.

Лекция 7

Цикл с фиксированным числом итераций (For)

Начинается изучение важных объектов блок-диаграммы – структур. Вводится цикл с фиксированным числом итераций (цикл ***For****), изучается поэлементная работа с массивами при использовании цикла* ***For****, организация доступа к предыдущим итерациям цикла.*

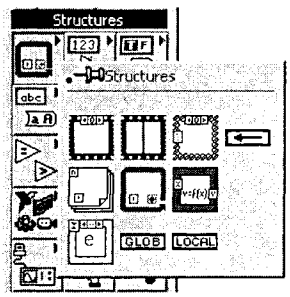


Рис. 7.1

Структуры

В этой лекции начинается описание важных объектов блок-диаграммы – структур. С помощью структур можно осуществить повторение отдельных частей программы, выполнение той или иной части программы в зависимости от какого-либо условия, выполнение программы в строго определенном порядке. Наверняка у вас уже имеется некоторый опыт программирования в текстовых средах разработки приложений. Так вот, некоторые структуры соответствуют циклу с фиксированным числом итераций (цикл **For**), циклу по условию (цикл **While**), оператору импликации (**if then else**). Вызвать любую структуру можно из палитры **Functions** ⇒ **Structures** (рис. 7.1).

Любая структура изображается в виде рамки, внутри которой содержится один или несколько участков программы. Каждый такой участок программы называется поддиаграммой. По краям структуры могут быть помещены входные и выходные терминалы. Вы можете наложить структуру на уже существующий участок программы или наоборот сначала поместить структуру, а за тем создавать элементы внутри нее.

Контекстное меню структуры вызывается при нажатии правой кнопки мыши на рамке структуры. Общими для всех структур пунктами контекстного меню являются:

Auto Grow – если флажок установлен, то при помещении объектов во внутрь структуры, она будет соответственно увеличивать размер.

Remove ... – удаление соответствующей структуры.

Replace with ... – изменить уже существующую структуру на структуру другого вида, подобную по функциональности.

Цикл For выполняет участок программы расположенный в поддиаграмме цикла определенное количество раз. Выберите его в палитре **Functions** (Функций). При этом изменится изображение курсора. Выделите область блок-диаграммы, в которой вы хотите разместить эту структуру. В процессе выделения не отпускайте кнопку мыши. Отмеченная область выделяется штриховым контуром. Выбрав область, отпустите кнопку мыши. Структура окажется на блок-диаграмме. Если в выделенной области находились другие объекты блок-диаграммы, они помещаются в тело цикла. Добавить новый объект внутрь структуры можно простым помещением его в область структуры. Кроме основной рамки цикла в нем присутствуют два терминала:

[N] – терминал общего числа итераций, определяет общее число итераций.

[i] – терминал счетчика итераций, содержит номер текущей итерации, начиная с 0.

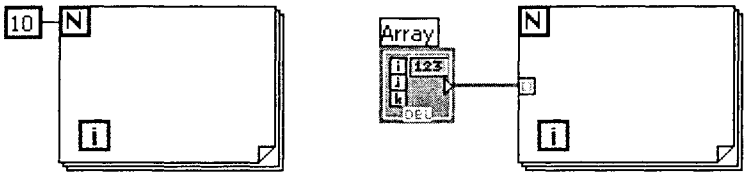
Данные могут поступать в цикл For (или выходить из него) через терминалы входных/выходных данных цикла. Терминалы входных/выходных данных цикла передают данные из структур и в структуры. Они представляют собой цветные прямоугольники и располагаются на границе области цикла. Прямоугольник принимает цвет типа данных, передаваемых по терминалу. Данные выходят из цикла по его завершении. Пока цикл не выполнил все положенные итерации выходные данные получить нельзя.

Число итераций цикла **For** должно быть известно до начала выполнения цикла. Имеется две возможности задать это число:

Непосредственно присоединить проводник к терминалу общего числа итераций [N]

Присоединить к одному из входных терминалов массив. В этом случае структура сама разберет массив на элементы.

Рис. 7.2



Цикл на рис. 7.2 выполнится ровно 10 раз, терминал счетчика итераций [i] будет принимать значения от 0 до 9. Цикл на рисунке справа выполниться столько раз, сколько элементов содержится в массиве, другими словами по итерации для каждого элемента массива. В случае если к терминалу [N] ничего не присоединено и нет разбираемых (**indexing**) массивов, LabVIEW выдаст сообщение об ошибке.

Автоматическая индексация

Цикл For может автоматически разбирать массив на элементы на входе и собирать из отдельных элементов массив на выходе. Это свойство называется автоиндексацией.

Пример 7.1. Автоиндексация

Соберите блок-диаграмму, показанную на рис. 7.3. Каждая последующая итерация добавляет в массив новый элемент. По завершении работы цикла, на выходе мы получим массив из N элементов. То, что на выходе действительно получится массив видно также и по толщине проводника данных. Снаружи структуры проводник данных стал толще.

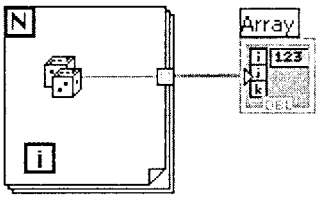


Рис. 7.3

Автоиндексация отключается щелчком правой кнопки мыши по терминалу входа/выхода из цикла и выбором пункта контекстного меню **Disable Indexing** (рис. 7.4). На выходе из цикла автоиндексацию следует отключать, в случае, когда нужно знать только последнее значение, а на входе в цикл, в случае, когда каждой итерации необходим доступ ко всему массиву, а не только к одному очередному его элементу.

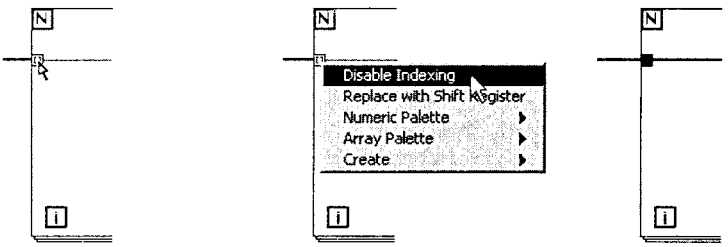


Рис. 7.4

Пример 7.2. Окружность

Построим изображение окружности на графике (рис. 7.5.).

В данном примере для перевода градусов в радианы используется узел **Convert Unit** (преобразовать размерность **Functions** ⇒ **Numeric** ⇒ **Conversion**), который значения терминала счетчика итераций [i], подсчитываемые в градусах, переводит в радианы. Это позволяет использовать функцию **Sine & Cosine**, аргумент которой должен быть представлен в радианах. Далее массивы объединяются и выводятся на двухкоординатный график осциллограмм. Попробуйте изменить программу, так что бы узлы **Convert Unit** и **Sine & Cosine** оказались вне цикла. Сравните полученный вариант с первоначальным.

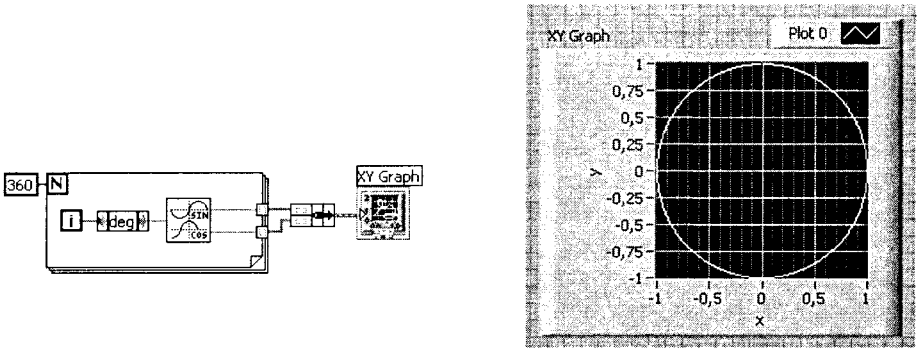


Рис. 7.5

Индексация нескольких массивов в одном цикле

В случае индексации нескольких массивов различной длины или при несовпадении числа элементов массива и числа поданного на N, число итераций цикла будет равно наименьшему значению. Иллюстрацией является рис. 7.6.

Здесь циклом индексируются 2 массива различной длины (2 и 3 элемента), а на терминал общего числа итераций подано число 4. Сколько раз выполниться цикл? Цикл выполнится 2 раза, по числу элементов наименьшего массива.

Для создания или разборки на элементы многомерного массива необходимо использовать несколько (по числу размерностей) вложенных циклов. Например, для двумерного массива потребуется два вложенных цикла, как это показано на рис. 7.7.

Внешний цикл разбирает массив на строки, внутренний цикл разбирает каждую строку на элементы. Из элемента вычитается единица, элементы собираются в строки, а внешний цикл собирает их в двумерный массив. Здесь следует отметить, что данный пример наглядно демонстрирует процесс индексации двумерных массивов. Использование вложенных циклов и автоиндексации для совершения алгебраических действий нецелесообразно. Как и большинство алгебраических функций, функция вычитания единицы полиморфна. Поэтому имеется возможность применить ее непосредственно ко всему массиву целиком без использования циклов. Сам факт, что вам известны принципы работы структуры For, не является достаточным для лишнего нагромождения блок-диаграммы. Старайтесь собрать блок-диаграмму как можно проще.

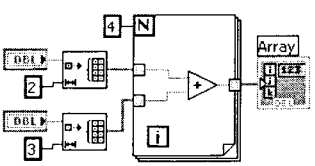


Рис. 7.6

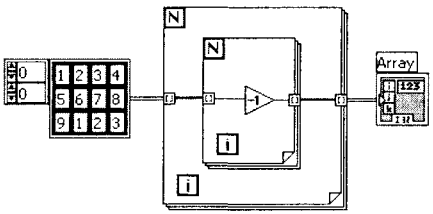


Рис. 7.7

Организация доступа к значениям предыдущих итераций цикла

При работе с циклами зачастую необходимо использовать значения, полученные в предыдущих итерациях. Так, в случае ВП, предназначенного для измерения температуры и отображения ее на графике, для расчета текущего среднего значения температуры необходимо использовать значения, полученные в предыдущих итерациях. Есть два пути доступа к этим данным: **Shift Register** (сдвиговый регистр) и **Feedback Node** (узел обратной связи).

Сдвиговый регистр (Shift Register)

Сдвиговые регистры используются при работе с циклами для передачи значений от текущей итерации цикла к следующей итерации. Сдвиговый регистр выглядит как пара терминалов, расположенных непосредственно друг против друга на противоположных вертикальных сторонах границы цикла. Правый терминал содержит стрелку «вверх» и сохраняет поступающие в него данные по завершению текущей итерации. LabVIEW передает данные с этого регистра в следующую итерацию цикла. Сдвиговый регистр создается щелчком правой кнопки мыши по левой или правой границе цикла и выбором из контекстного меню пункта **Add Shift Register**.

Сдвиговый регистр передает данные любого типа, он автоматически принимает тип первых поступивших на него данных. Данные, передаваемые на терминалы сдвигового регистра, должны быть одного типа.

Чтобы задать первоначальное значение сдвигового регистра, необходимо передать на его левый терминал любое значение извне цикла. Если исходное значение не задать, сдвиговый регистр использует значение, записанное в него во время последнего выполнения цикла. В случае, если цикл никогда не выполнялся (а это происходит когда на терминал общего числа итераций подается значение 0), сдвиговый регистр использует значение, используемое по умолчанию для данного типа данных.

Пример 7.3. Сдвиговый регистр

Использование сдвигового регистра показано на рис. 7.8. Пример, показанный на рисунке, вычисляет факториал числа. Запустите этот ВП в режиме отладки (**Highlight Execution**) и обратите внимание на данные, поступающие в и из сдвигового регистра. Сначала в левый терминал сдвигового регистра поступает единица. В первой итерации эта единица умножается также на единицу (напомним, что отсчет итераций начинается с нуля). Далее полученная единица поступает на правый терминал сдвигового регистра. В следующей итерации из левого терминала опять выходит единица, которая внутри цикла умножается на два. Два принимается правым терминалом сдвигового регистра. С левой стороны значение проводни-

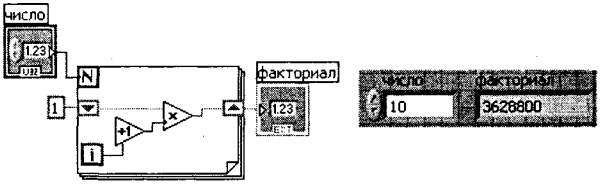


Рис. 7.8

ка теперь становится равным два, которое умножается уже на три. Таким образом, осуществляется подсчет факториала числа.

Если в текущей итерации используются данные не только предыдущей итерации, но и выполненных ранее итераций, следует использовать сдвиговый регистр с несколькими терминалами. Кроме того, что можно создать несколько сдвиговых регистров для передачи из одной итерации в другую значения нескольких переменных, в LabVIEW предусмотрена возможность создания стека сдвиговых регистров.

Стек сдвиговых регистров

Стек сдвиговых регистров представляет собой набор терминалов с левой стороны структуры, сохраняющих значения переменной в различных итерациях выполняемого цикла. Для создания стека сдвиговых регистров достаточно щелкнуть правой кнопкой мыши по левому терминалу и выбрать пункт контекстного меню **Add Element**.

При добавлении еще двух сдвиговых регистров к левому терминалу данные последних трех итераций переносятся на следующую итерацию. При этом значение последней итерации сохраняется в самом верхнем сдвиговом регистре. Второй терминал сохранят данные, переданные ему с предыдущей итерации, третий терминал хранит данные, полученные две итерации назад и т.д. В буквенных обозначениях это означает, что сохраняются данные i-1, i-2 и i-3 итераций.

Пример 7.4. Стек сдвиговых регистров

Пример иллюстрирует применение стека сдвиговых регистров (рис. 7.9.). В цикле в течение 100 итераций генерируется случайное число. С помощью стека сдвиговых регистров вычисляется среднее арифметическое четырех последних значений. Запустите ВП в режиме отладки и понаблюдайте за выполнением блок-диаграммы.

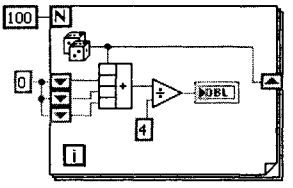


Рис. 7.9

Узел обратной связи

Узел обратной связи (**Feedback Node**) также как и сдвиговый регистр отвечает за перевод значения ка-

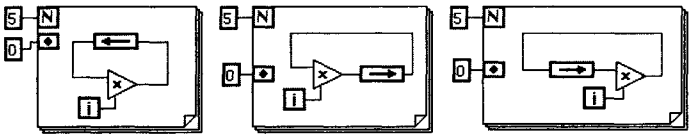


Рис. 7.10

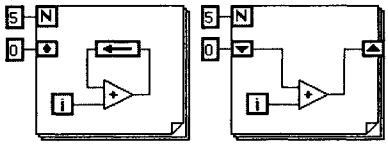


Рис. 7.11

кой-либо переменной из одной итерации в другую. Он появляется в цикле автоматически, когда вы соединяете поле вывода данных какого-либо узла (функции, подпрограммы) с его же полем ввода. Использование узлов обратной связи позволяет избегать большого количества проводников данных и соединений.

Вручную поместить узел обратной связи внутрь цикла можно, выбрав **Feedback Node** в палитре **Structures**. В зависимости от положения узла обратной связи относительно функции, которая его использует, стрелка на иконке может быть направлена либо вправо, либо влево. При этом направление передачи данных всегда определяется стрелкой. Таким образом, поля ввода и вывода также могут быть либо справа, либо слева. Наглядно это свойство узла обратной связи показано на рис. 7.10. Все три ВП выполняют одно и то же, различие состоит лишь в положении узла обратной связи.

Узел обратной связи и сдвиговый регистр полностью взаимозаменяемы, с помощью пункта контекстного меню **replace with...** один можно заменить другим. Следующий пример (рис. 7.11) демонстрирует соответствие узла обратной связи и сдвигового регистра. Оба цикла выполняют одно и то же действие суммируют числа от 0 до 4.

Выводы

Цикл **For** (с фиксированным числом итераций) применяется, в случае если требуется повторение определенного участка программы заранее известное число раз. Также если необходимо обработать данные в массиве или сформировать массив.

Лекция 8

Логические элементы управления и индикации

*Рассматривается логический тип данных, а также логические функции. Разбирается структура цикл с выходом по условию (цикл **While**).*

В LabVIEW существуют элементы управления и индикации, которые могут иметь только два состояния (включено, выключено). Такие элементы имеют логический тип (**Boolean**) и расположены в палитре **Controls** ⇒ **Boolean**. Например, это кнопки и «лампочки». Иначе говоря, данные логического типа могут иметь только два значения: «истина» (**True**) и «ложь» (**False**).

Механическое действие (Mechanical Action)

У логических элементов управления через контекстное меню можно установить механическое действие (**Mechanical Action**), в зависимости от которого элемент управления будет по разному реагировать на нажатие мыши и считывание ВП его значения.

- **Switch When Pressed** – Переключать в момент нажатия — Изменяет значение элемента управления каждый раз, когда пользователь нажимает на него, не зависимо от того, как часто ВП считывает его значение.
- **Switch When Released** – Переключать в момент отпускания — Изменяет значение элемента управления в момент отпускания кнопки мышки, не зависимо от того, как часто ВП считывает его значение. Можно рекомендовать использовать этот вариант вместо предыдущего, так как у пользователя, после того как он нажал клавишу мыши, остается шанс «передумать»: отвести мышь с кнопки и отпустить.
- **Switch Until Released** – Включено пока нажато — Изменяет значение элемента управления, пока пользователь нажимает кнопку мышки и удерживает, после отпускания клавиши мыши принимает исходное значение, наподобие дверного звонка, не зависимо от того, как часто ВП считывает его значение.

Latch When Pressed – Автовозврат после нажатия — Изменяет значение элемента управления, когда пользователь нажимает кнопку мыши и возвращается к первоначальному, как только ВП один раз считает значение.

Latch When Released – Автовозврат после отпускания — Изменяет значение элемента управления в момент отпускания кнопки мыши и возвращается к первоначальному, как только ВП один раз считает значение. Данное действие подобно диалоговым кнопкам **Windows**.

Latch Until Released – Автовозврат после удерживания — Изменяет значение элемента управления, когда пользователь нажимает и удерживает кнопку мыши и возвращается к первоначальному значению при опускании, но только после того как ВП минимум один раз считает значение.

Логические функции

Таблица 8.1

Вид	*Название*	*логика*		
		x	*y*	*результат*
x, y — x .and. y?	***And** – Логическое И*	*Истина*	*Истина*	*Истина*
		Истина	*Ложь*	*Ложь*
		Ложь	*Истина*	*Ложь*
		Ложь	*Ложь*	*Ложь*
x, y — x .or. y?	***Or** – Логическое ИЛИ*	*Истина*	*Истина*	*Истина*
		Истина	*Ложь*	*Истина*
		Ложь	*Истина*	*Истина*
		Ложь	*Ложь*	*Ложь*
x, y — x .xor. y?	***Exclusive Or** – Исключающее ИЛИ*	*Истина*	*Истина*	*Ложь*
		Истина	*Ложь*	*Истина*
		Ложь	*Истина*	*Истина*
		Ложь	*Ложь*	*Ложь*
x, y — x .implies. y?	***Implies** – Логическое ИЛИ между «y» и «не x».*	*Истина*	*Истина*	*Истина*
		Истина	*Ложь*	*Ложь*
		Ложь	*Истина*	*Истина*
		Ложь	*Ложь*	*Истина*
x — .not. x?	***Not** – Логическое НЕ*	*Истина*		*Ложь*
		Ложь		*Истина*

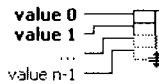
Для булевых функций «и», «или», «исключающее или» имеются соответствующие варианты с логическим отрицанием выхода (на выходном терминале появляется кружок):

x, y — .not. (x .and. y)?	***Not And***	*Истина*	*Истина*	*Ложь*
		Истина	*Ложь*	*Истина*
		Ложь	*Истина*	*Истина*
		Ложь	*Ложь*	*Истина*

Таблица 8.1 (окончание)

Вид	Название	логика		
		x	y	результат
x, y — .not. (x .or. y)?	Not Or	Истина	Истина	Ложь
		Истина	Ложь	Ложь
		Ложь	Истина	Ложь
		Ложь	Ложь	Истино
x, y — .not. (x .xor. y)?	Not Exclusive Or	Истина	Истина	Истина
		Истина	Ложь	Ложь
		Ложь	Истина	Ложь
		Ложь	Ложь	Истина

Boolean To (0,1) –Преобразует логическую истину в единицу, а ложь в ноль.



Compound Arithmetic – Операция над несколькими величинами. Используя инструмент **перемещение** можно изменить количество входных терминалов. Через пункт контекстного меню **Change mode** (изменить действие) можно выбрать сложение, умножение, логическое И, логическое ИЛИ, Исключающее ИЛИ

Функции сравнения (**Functions ⇒ Comparison**).

Таблица 8.2

Операции сравнения двух величин:		Операции сравнения с нулем:	
=	– равно?	=0	– ровно нулю?
≠	– не равно?	≠0	– не ровно нулю?
>	– больше?	>0	– больше нуля?
<	– меньше?	<0	– меньше нуля?
≥	– больше или равно?	≥0	– больше или ровна нулю?
≤	– меньше или равно?	≤0	– меньше или равно нулю?

In Range and Coerce – Определяет находится ли x между нижним и верхним пределом. В случае если x лежит вне данного отрезка на выход **coerced**(x) подается «отсеченное» значение по верхнему или нижнему пределу.

Select –в зависимости от значения на логическом входе выбирает одно из двух значений. Если на логическом входе будет значение «истина» (**True**), то выходе функция выдаст значение, поданное на вход t, если же на логическом входе «ложь» (**False**), то возвращается значение с поля f.

Цикл по условию (While)

Цикл **While** (по условию) работает до тех пор, пока логическое условие выхода из цикла не примет значение «истина». Во всем, что касается принципа работы цикла **While**, а также работы с объектами в цикле **While**, их размещения внутри цикла, использования сдвиговых регистров и узлов обратной связи цикл **While** аналогичен циклу **For**. Принципиальное различие этих циклов заключается в том, что цикл For выполняется некоторое число раз, задаваемое явно через терминал общего числа итераций или задаваемое неявно как число элементов индексируемого на входе цикла массива. Цикл же **While** выполняется неопределенное число раз, пока не будет выполнено заданное условие. В отличие от цикла **For** цикл **While** выполняется всегда. В случае если условие с самого начала выполнено, цикл выполняется 1 раз.

Элементы цикла **While**:

- терминал условия. Блок-диаграмма цикла **While** выполняется до тех пор, пока не выполнится условие выхода из цикла. По умолчанию, терминал условия выхода имеет вид, показанный слева. Это значит, что цикл будет выполняться до поступления на терминал условия выхода значения **True**. В этом случае терминал условия выхода называется терминалом **Stop If True** (остановить, если «истина»).

  Предусмотрена возможность изменения условия выхода и соответствующего ему изображения терминала условия выхода. Щелчком правой кнопки мыши по терминалу условия выхода или по границе цикла необходимо вызвать контекстное меню и выбрать пункт **Continue If True** (продолжить, если «истина»). Также можно воспользоваться инструментом *управление*, щелкнув им по терминалу условия. Напомним, что вызов палитры инструментов осуществляется через пункт меню **Window ⇒ Show Tools Palette**. Изображение терминала условия выхода поменяется на **Continue If True** (продолжить, если «истина»). В результате условием выхода из цикла становится поступающее на терминал условия значение **False**.

- терминал счетчика итераций. Содержит номер текущей итерации, начиная с 0.

Доступ к значениям предыдущих итераций цикла

Как уже было ранее упомянуто, сдвиговый регистр и узел обратной связи в цикле по условию (**While loop**) используются аналогично циклу с фиксированным числом итераций (**For loop**).

Автоиндексирование в цикле по условию

В основном автоиндексирование в цикле по условию (**While**) имеет тот же смысл что и в цикле с фиксированным числом итераций (**For**).

Надо заметить, что в цикле **While** (в отличие от цикла **for**) по умолчанию автоиндексирование для массивов выключено, т.е. массив не разбирается на элементы, а поступает в каждую итерации целиком.

Следует учесть, что цикл **While** может прекратить выполнение только при заранее определенном значении логической переменной, присоединенной к терминалу условия выхода из цикла, и не зависит от размера разбираемого (**Indexing**) массива. Т.е. выполнение цикла может закончиться раньше или позже, чем закончатся элементы массива. Очевидно, что в первом случае из массива будут извлечены не все элементы, а во втором, при попытке считывания после конца массива, в цикл будет поступать значение по умолчанию для данного типа (0 для чисел, пустой массив, пустая строка для массивов и строк соответственно и т.д.)

Пользуясь пунктом контекстного меню «**Replace**» можно изменить структуру Цикл по условию (**While loop**) на Цикл с фиксированным числом итераций (**For loop**).

Пример. 8.1. Цикл While

Самый просто пример использования цикла по условию уже собран в палитре **Express** ⇒ **Execution Control** ⇒ **While Loop**. Поместите это цикл на блок-диаграмму. Появится структура, показанная на рис. 8.1.

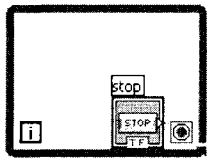


Рис. 8.1

Кроме собственно структуры **While** на диаграмме имеется терминал кнопки останова цикла. В данном случае этот цикл отвечает за выполнение блок-диаграммы, собранной внутри него. Остановить запущенную программу можно будет, нажав на клавишу **STOP** на лицевой панели ВП. Практически любую программу рекомендуется собирать внутри цикла **While**. Вам известно, что на инструментальной панели имеются кнопки непрерывного запуска программы **Run Continuously** и немедленной остановки выполнения программы **Abort Execution**. Данный пример иллюстрирует возможность программного непрерывного запуска программы и ее остановки. Однако есть, и в некоторых случаях существенная, разница между этими двумя способами выполнения программы. В случае использования цикла **While** и кнопки **STOP** блок-диаграмма внутри цикла всегда полностью завершит все положенные операции. В случае немедленной остановки выполнения программы через инструментальную панель выполнение программы обрывается немедленно. Это может повлечь за собой непредвиденную ошибку.

Задание 8.1. Решение нелинейного уравнения

Соберите программу по численному решению нелинейных уравнений методом бисекции (деления пополам). При решении нелинейного уравнения методом бисекции предполагается, что на некотором отрезке $[x_{min}, x_{max}]$ располагается только один корень уравнения $f(x) = 0$. В таком случае значения функции на границе отрезка будет иметь противоположные знаки. Если вычислить середину текущего отрезка и определить, какой знак принимает значение функции в этой точке, можно судить о том, в какой половине оказался корень уравнения. Таким образом, определяется новый отрезок. Процедура повторяется до тех пор, пока длина получаемого отрезка не станет меньше некоторого наперед заданного числа ε. В качестве корня уравнения принимается среднее значение последнего отрезка.

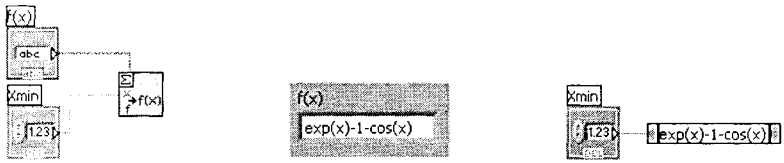


Рис. 8.2

Исходными данными является выражение $f(x)$, отрезок $[x_{min}, x_{max}]$ и число e. Выражение $f(x)$ задается строкой, а все остальные числа числовыми элементами управления. Поместите их на лицевую панель.

Чтобы получить значения функций $f(x_{min})$ и $f(x_{max})$ воспользуйтесь функцией **Functions ⇒ Analyze ⇒ Mathematics ⇒ Formula ⇒ Advanced Formula Parsing ⇒ Eval Single-Variable Scalar**. На вход этой функции подается строка с формулой и значение x. На выходе получаем значение функции в точке x.

Что делать, если у вас нет палитры **Advanced Formula Parsing**? Воспользуйтесь функцией **Numeric ⇒ Expression Node**. Однако в этом случае вам придется в узел выражения записать строку с формулой непосредственно на блок-диаграмме. Сравните использование функции **Single-Variable Scalar** и **Expression Node**. Они показаны на рис. 8.2.

С одной стороны решение, показанное справа, кажется более простым. Однако (как и в нашем случае) эту функцию придется использовать не один раз. Поэтому, если понадобится выражение $f(x)$ поменять, его придется изменить во всех узлах **Expression Node** на блок-диаграмме.

Поместите на блок-диаграмму цикл **While**. Т.к. границы отрезка от итерации к итерации будут изменяться, необходимо использовать сдвиговый регистр. Создайте в цикле два сдвиговых регистра для границ отрезка. Внутри цикла вычислите середину отрезка, а также значение функции в середине отрезка. Определите условие выхода из цикла: длина отрезка должна быть меньше наперед заданного числа ε. Попытайтесь реализовать это условие сами. Если все же возникают трудности, посмотрите полное решение, показанное в конце примера.

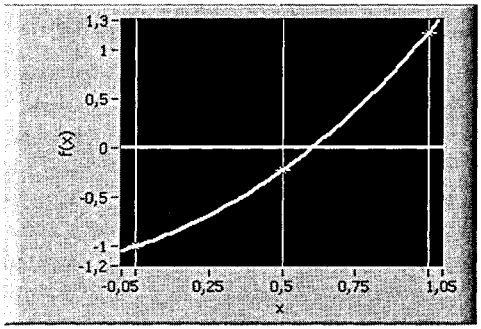


Рис. 8.3

Теперь разберемся в том, как реализовать процесс выбора текущего отрезка. Если значение функции в точке x_{min} и в середине отрезка одновременно больше или меньше нуля, то корень уравнения находится в другой половине отрезка. Если же знак функции в этих точках различен, то они и образуют новый отрезок. Это показано на рис. 8.3. На рисунке значение функции на правой границе отрезка [0,1] положительно, а в середине отрезка в точке 0,5 отрицательно. Это означает, что новым отрезком должна стать правая половина отрезка [0,5; 1].

Здесь следует отметить, что палитры функций для работы с логическими переменными находятся по пути **Functions ⇒ Boolean** и **Functions ⇒ Comparison**. В первой палитре представлены основные логические операции (такие как «и», «или» и т.д.), а во второй – операции сравнения (такие как «больше», «меньше» и т.д.).

Итак, перед циклом необходимо определить знак функции на одной из границ (в нашем примере это будет правая граница). Для этого воспользуйтесь операцией **Greater Then 0?** палитры **Functions ⇒ Comparison**. Получившуюся логическую переменную подайте на вход в цикл. В соответствии с нашим предположением о том, что на данном отрезке корень есть и при этом единственный, правая граница всегда будет либо положительной (как это показано на рис. 8.3), либо отрицательной. Далее уже внутри цикла знак функции необходимо определить для середины текущего отрезка. Здесь уже в зависимости от того, в левой или правой половине отрезка находится корень, знак функции будет меняться. Далее эти логические переменные необходимо сравнить. Если они обе имеют значение «истина» или «ложь», то следует выбирать правую половину отрезка. Если они одна из них «истина», а другая «ложь», то следует выбирать левую половину отрезка. Логическая операция, которая выдает значение «ложь» в первом случае, и значение «истина» во втором, называется **Exclusive Or** и находится в палитре **Functions ⇒ Boolean**.

В палитре **Functions ⇒ Comparison** находится еще одна необходимая нам функция – **Select**. Используйте две функции **Select** для определения левой и правой

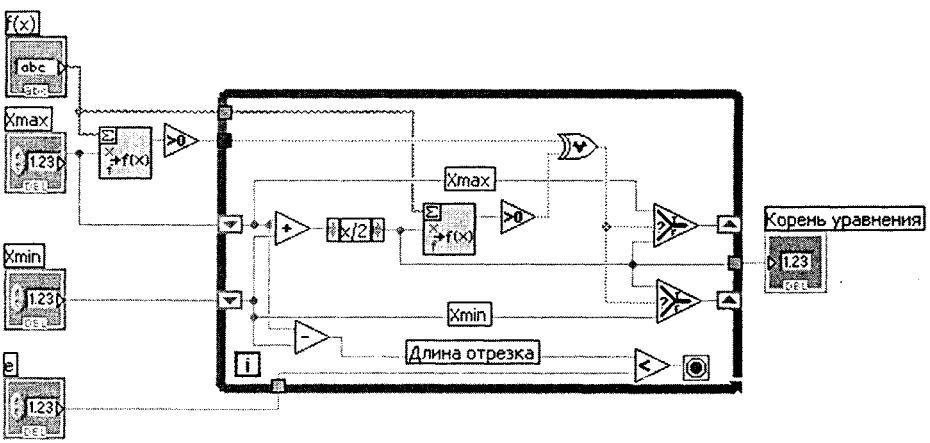


Рис. 8.4

границы нового отрезка. На вход *s* в обоих случаях следует подать значение полученной на выходе функции **Exclusive Or**. Как уже разбиралось, если на выходе **Exclusive Or** переменная принимает значение «истина», то правая граница сохраняет текущее положение. Значит, вход *t* соединяйте с соответствующим сдвиговым регистром правой границы. Вход *f* соединяйте с серединой отрезка. Во второй функции **Select** подключайте вход *t* к середине отрезка, а *f* – к левой границе. Выход функций **Select** подводите к сдвиговым регистрам.

Итак, блок-диаграмма готова. Осталось вывести результат вычислений на элемент индикации лицевой панели. Середину отрезка подводим к правой границе цикла. Образуется туннель, к которому и подключается элемент индикации. Окончательное решение этой задачи показано на рис. 8.4.

Перейдите на лицевую панель. Введите какую-либо функцию. Возьмите, например, классический пример $f(x) = exp(x)\text{-}1\text{-}cos(x)$. Задайте границы 0 и 1, задайте точность численного решения, например 0,0001. Запустите программу. Если блок-диаграмма построена правильно, то на элементе индикации будет число 0,601349. Понаблюдайте за ходом выполнения программы.

Еще раз обратите внимание на функциональное различие сдвиговых регистров и туннелей. Первые передают значение от итерации к итерации. Вторые предназначены для ввода данных в цикл, которые не меняются от итерации к итерации, и для вывода окончательных результатов работы цикла. При этом данные на входе и выходе считываются один раз: первые до начала работы цикла, а вторые после завершения работы.

Выводы

Цикл по условию (**While**) применяется в случае, когда количество итераций заранее не известно, и условие окончания цикла вырабатывается непосредственно внутри цикла. Использование для ВП внешнего цикла **While** с выходом по кнопке «стоп» является хорошим стилем программирования.

Лекция 9

Структура выбора (Case)

Структура последовательности (Sequence)

*Рассматриваются структуры вариант (**Case**) и последовательности (**Sequence**), позволяющие управлять порядком выполнения программы.*

Структура **Case** имеет две или более поддиаграммы вариантов. Только одна поддиаграмма варианта видима в данный момент времени и только одна поддиаграмма варианта работает при обращении к этой структуре. Входное значение терминала селектора структуры определяет, какая поддиаграмма будет выполняться в данный момент времени. В простейшем случае структура **Case** аналогична логическим операторам (**if...then...else**) в текстовых языках программирования.

Элементы структуры выбора:

селектор структуры **Case**, расположенный сверху графического изображения структуры, состоит из указателя значения варианта в центре и стрелок прокрутки по сторонам. Эти стрелки используются для просмотра возможных вариантов;

терминал селектора варианта. Значение, подаваемое на терминал селектора варианта, определяет, какая именно поддиаграмма структуры (вариант) будет выполняться. Допустимо использовать целочисленный, логический, строковый типы, а также тип перечисления в качестве значения, подаваемого на терминал варианта. Терминал варианта может располагаться в любом месте левой границы структуры **Case**. Если терминал варианта логического типа, то структура состоит из двух логических вариантов **True** и **False**. Если терминал варианта имеет один из следующих типов: целочисленный, строковый или перечисления, то количество вариантов может достигать 2^{31}-1 вариантов.

Для использования структуры **Case** необходимо отметить вариант по умолчанию. Вариант по умолчанию или поддиаграмма по умолчанию выполняется, если значение терминала варианта выходит за пределы диапазона или не существуют вариантов для возможных значений терминала варианта. Щелчок правой кнопки мыши на границе структуры **Case** позволяет добавлять, дублировать, перемещать и удалять варианты (поддиаграммы), а также отмечать вариант по умолчанию.

Структура **Case** допускает использование входных и выходных терминалов данных. Терминалы входных данных доступны во всех поддиаграммах, но их использование поддиаграммой структуры необязательно. Создание выходного терминала на одной поддиаграмме приводит к его появлению на других поддиаграммах в том же самом месте границы структуры. Если хотя бы в одной поддиаграмме выходной терминал не определен, то поле этого терминала окрашивается в белый цвет, что говорит об ошибке создания структуры. Необходимо определять значения выходных терминалов во всех вариантах (поддиаграммах). Кроме того, выходные терминалы должны иметь значения совместимых типов.

Задание 9.1. Ввод пароля

Рассмотрите пример использования структуры Case в качестве распределения доступа к каким-либо функциям вашей программы.

Поместите на лицевую панель строковый элемент управления **String** (**String & Path ⇒ String Control**). Измените его метку на следующую: «Введите пароль».

Поместите на блок-диаграмму структуру **Case** (**Structures ⇒ Case Structure**).

Соедините терминал элемента управления **String** с терминалом селектора варианта структуры **Case**. По умолчанию в структуре было два варианта: **«True»** и **«False»**, **Default**. Применительно к выбору одной их этих двух вариантов по тексту, введенному в поле элемента управления **String**, это означает, что блок-диаграмма первого варианта будет выполняться тогда, когда будет введено слово **True**, вторая блок-диаграмма будет выполняться в остальных случаях.

Вместо **«True»** введите предполагаемый пароль, например 123. На обе блок-диаграммы поместите стандартное диалоговое окно с одной кнопкой (**Time & Dialog ⇒ One Button Dialog**).

Создайте строковую константу для входной величины **message** обоих диалоговых окон (в контекстном меню указанного входа выберите **Create ⇒ Constant**). На блок-диаграмме «123» в поле строковой константы введите «Пароль правильный», на второй блок-диаграмме – «Пароль неправильный».

Лицевая панель и блок-диаграмма (оба варианта) представлены на рис. 9.1.

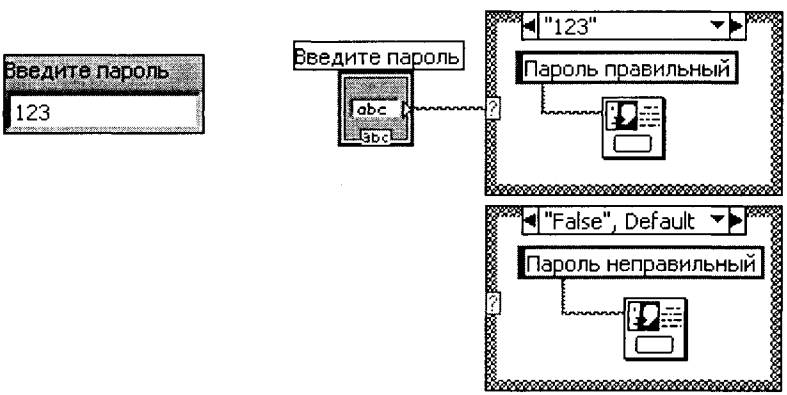


Рис. 9.1

Таким образом, при вводе в поле элемента управления **String** цифр 123 выпадает диалоговое окно с текстом «Пароль правильный». При вводе любых других символов выпадает диалоговое окно с текстом «Пароль неправильный». Дополнив блок-диаграммы обоих вариантов какими-либо выполняемыми функциями, вы ограничиваете доступ к одной из них паролем «123».

Задание 9.2. Калькулятор

Создайте ВП в виде калькулятора. Входные значения задавайте двумя элементами управления. В зависимости от выбора арифметической операции на выходе эти два числа будут складываться, вычитаться, умножаться или делиться. Создайте два числовых элемента управления a и b. Создайте нумерованный элемент управления **Controls ⇒ Ring&Enum ⇒ Enum**. В его свойствах (чтобы попасть на страницу свойств, в контекстном меню выберите пункт **Properties**) перейдите на вкладку редактирования пунктов Edit Items. И в таблице в колонку меток **Label** введите четыре строки: «a+b», «a-b», «a*b», «a/b». Каждой метке соответствует число от нуля до трех.

Перейдите на блок-диаграмму и поместите на нее структуру варианта **Case**. При подключении терминала элемента управления Enum к терминалу селектора варианта он поменяет свой цвет на синий (целочисленный тип), а в селекторе структуры заголовки вариантов **«True»** и **«False»** поменяются на «a+b» и «a-b». Выберите в контекстном меню структуры **Add Case For Every Value**. LabVIEW добавит вариант для каждого пункта элемента управления **Enum**. При этом название варианта, отображаемое в селекторе структуры, будет взято из пунктов элемента управления **Enum**. Для выполнения программы решающую роль имеют номер варианта: 0, 1, 2 или 3, на что и указывает синий цвет терминала элемента управления **Enum**, проводника данных и терминала селектора варианта. Однако для удобства как конечных пользователей ВП, так и программистов, варианты маркируются названиями.

Наберите в каждой поддиаграмме структуры соответствующую блок-диаграмму и выведите результат на элемент индикации. Здесь следует подчеркнуть, что все поддиаграммы структуры вариант находятся внутри одной структуры. Готовая блок-диаграмма представлена на рис. 9.2, а лицевая панель на рис. 9.3. Пусть то, что на рисунке блок-диаграммы показаны четыре структуры варианта рядом друг с другом, вас не вводит в заблуждение. На самом деле все четыре поддиаграммы находятся внутри первой структуры. Видна может быть всего лишь одна поддиаграмма, а остальные три как в колоде карт сложены позади первой. Переключение между ними осуществляется с помощью стрелок.

В заданиях использовались стандартные случаи использования структуры варианта, поэтому поддиаграммы создавались автоматически. Создавать свои варианты, можно, используя контекстное меню структуры. В контекстном меню доступны следующие пункты:

- **Add Case After** – добавляет вариант, следующий за текущим вариантом.
- **Add Case Before** – добавляет вариант до текущего варианта.

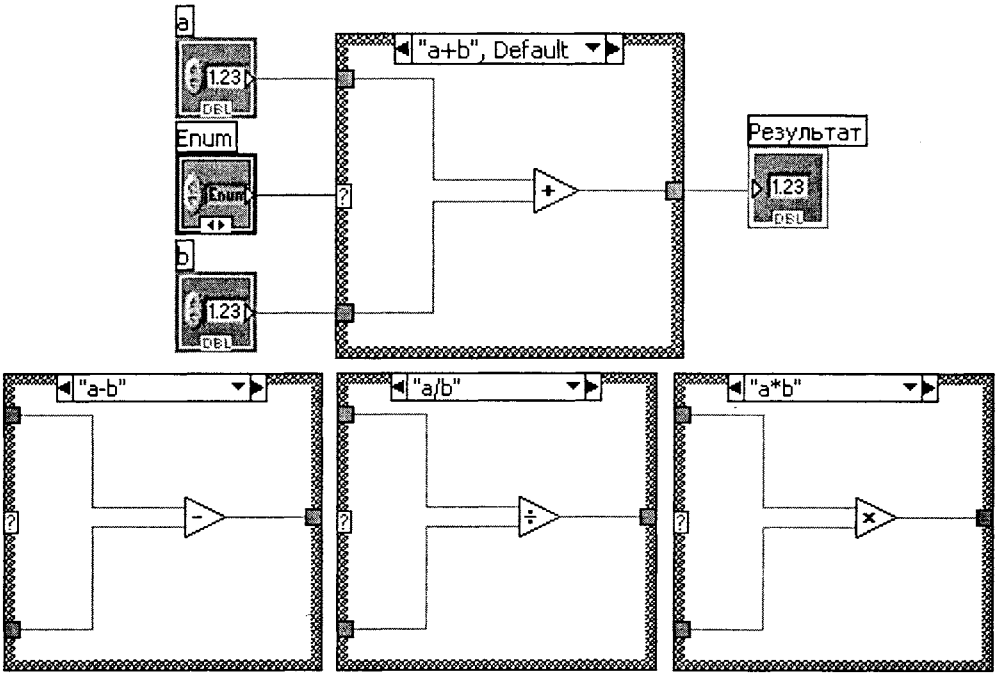


Рис. 9.2

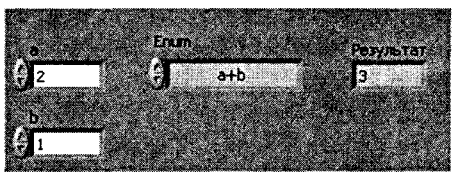


Рис. 9.3

- **Duplicate This Case** – дублирует текущий вариант. Этим пунктом целесообразно пользоваться, когда поддиаграмма еще одного варианта очень похожа за исключением некоторых деталей.
- **Delete This Case** – удаляет текущий вариант.
- **Remove Empty Cases** – удаляет пустые варианты.
- **Show Case** – показывает вариант. Этот пункт аналогичен стрелке рядом с названием варианта в селекторе вариантов.
- **Swap Diagram With Case** – меняет поддиаграмму текущего варианта на поддиаграмму другого варианта.
- **Rearrange Case** – вызывает диалоговое окно, в котором вы сможете поменять порядок расположения вариантов.
- **Remove Default** – убирает из структуры вариант по умолчанию, но сам вариант по умолчанию со своей поддиаграммой не удаляется. Если в структуре не назначен вариант по умолчанию, то этот пункт меню отсутствует.

Структура последовательности (Sequence)

Структура последовательности представляет собой одну или несколько поддиаграмм (кадров) которые исполняются подряд. Существуют два типа структур последовательности: структура открытой последовательности и многослойная структура.

Использование структур последовательности позволяет управлять порядком выполнения программы. В случае если вы хотите быть уверенным в том, что некоторая часть программы выполниться строго после другой части программы используйте структуру последовательности.

Структура открытой последовательности (Flat Sequence Structure)

Структура открытой последовательности выполняется кадр за кадром слева на право. Вы можете добавлять или удалять кадры из последовательности, используя контекстное меню. Когда вы добавляете или удаляете поддиаграммы, структура изменяет размер автоматически. Например, на рис. 9.4 изображена открытая структура последовательности, с помощью которой осуществляется подача двух звуковых сигналов с паузой между ними в 1 секунду.

Пользуясь пунктом контекстного меню **Replace** можно изменить тип структуры на многослойную.

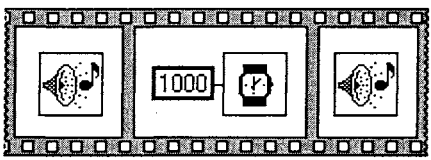


Рис. 9.4

Задание 9.3. Время выполнения программы

Создадайте ВП, который сможет подсчитать время выполнения какой-либо несложной операции, например вычисления синуса $\pi/6$. Для этого следует воспользоваться функцией получения текущего времени в миллисекундах **Functions** ⇒ **Time & Dialog** ⇒ **Tick Count (ms)** два раза: до вычисления синуса и после. Для того чтобы время с помощью указанной функции было получено строго в этой последовательности, и используйте структуру последовательности.

Время вычисления синуса на современных машинах настолько мало, что подсчет однократного выполнения этой операции вряд ли может быть выполнен. Вычисление синуса некоторого аргумента целесообразно выполнить много раз, к примеру, 10^8 раз. Подсчет времени таким способом уже может дать представление о времени выполнения какой-либо несложной операции.

Попытайтесь собрать блок-диаграмму такого ВП самостоятельно. Результат получите в секундах. Сравните блок-диаграмму вашего ВП с нашим (рис. 9.5). Запустите ВП. У нас вычисление синуса $\pi/6$ 10^8 раз выполняется за 1,936 секунды. Какие результаты показывает ваш ВП?

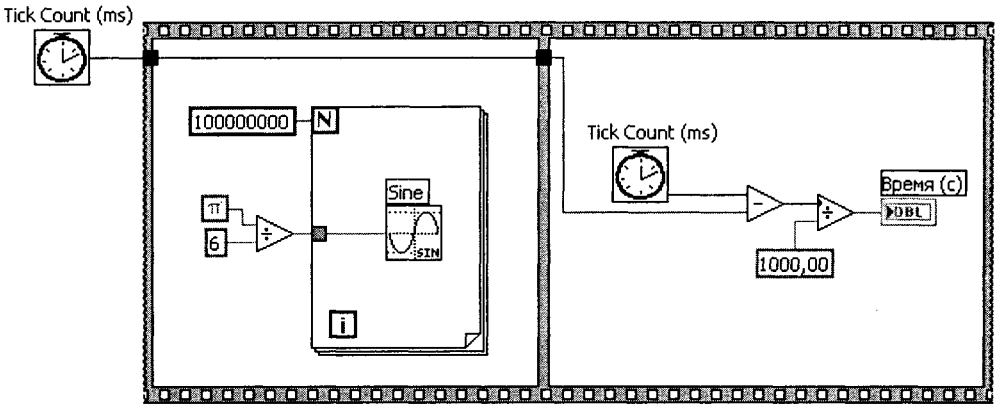


Рис. 9.5

Структура многослойной последовательности (Stacked Sequence Structure)

Структура многослойной последовательности содержит пронумерованные поддиаграммы (0, 1... и т.д.) которые выполняются по порядку. На блок диаграмме (в отличае от открытой последовательности) одновременно вы можете видеть только одну поддиаграмму. Переход от одной к другой поддиаграмме осуществляется с помощью селектора структуры последовательности. Если вы хотите сэкономить место на блок-диаграмме, используйте многослойную последовательность.

От одной поддиаграммы к другой данные передаются через терминалы локальных переменных. Для того, что бы создать терминал требуется на рамке структуры вызвать контекстное меню и выбрать пункт **Add Sequence Local**. После того как вы присоедините источник данных к терминалу локальных переменных, в нем появляется стрелка, направленная наружу, это значит, что терминал является приемником данных (рис. 9.6). Во всех последующих терминал будет являться источникам данных и стрелка в нем будет направлена внутрь. В кадрах предшествующих кадру источника данных терминал выглядит заштрихованным, и вы не можете его использовать.

Пользуясь пунктом контекстного меню **Replace**, можно сменить тип структуры с многослойной на последовательную или изменить структуру на структуру выбора (**Case**).

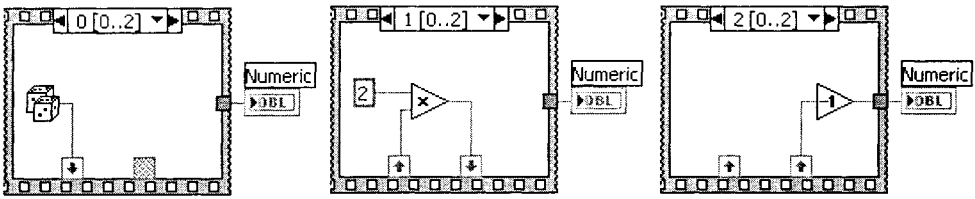


Рис. 9.6

Выводы

Структуры последовательности и выбора позволяют управлять способом выполнения программы. Структуру выбора используют в случае необходимости выполнять различный код программы при различных значениях переменных (логического, целочисленного, строкового или перечисляемого типов). Структуры последовательности позволяют выполнять участки программы в определенной последовательности. Использование открытой структуры последовательности более предпочтительно по сравнению с многослойной, т.к. в первом случае все части блок-диаграммы видны одновременно. Использование многослойной последовательности может быть оправданно, когда необходимо сэкономить место на блок-диаграмме.

Лекция 10

Структура обработки данных события (Event)

*Рассматривается структура события (**Event**), позволяющая эффективно обрабатывать действия пользователя.*

Структура события используется для синхронизации действий пользователя на лицевой панели с выполнением блок-диаграммы. Применение данной структуры позволяет выполнять определенную поддиаграмму каждый раз, когда пользователь совершает соответствующее действие (нажимает кнопку, изменяет значение элемента управления, перемещает мышь и т.д.). Без использования структуры события придется опрашивать состояние объектов лицевой панели в цикле, проверяя, не произошли ли какие-либо изменения. Опрашивание состояния объектов лицевой панели требует существенного количества процессорного времени и есть вероятность, что предполагаемые изменения будут пропущены, если они произошли слишком быстро. В этом качестве структура события позволяет избежать процесса опрашивания состояния лицевой панели для определения произведенных пользователем действий. Использование структуры событий сокращает требования программы к ресурсам процессора, упрощает код блок-диаграммы и гарантирует, что блок-диаграмма установит любые действия пользователя.

Элементы структуры события:

селектор структуры события – служит для перехода между поддиаграммами;


терминал времени ожидания – определяет время, которое структура события будет ожидать совершения события, прежде чем продолжиться выполнение остальной программы;

узел данных события – когда происходит событие, через этот узел передаются данные о событии в поддиаграмму. Набор данных зависит от типа события. Например, для события «нажатие на клавишу мыши» предаются координаты мыши, состояние кнопок мыши, для события «изменение значения» – старое значение и новое значение;

узел фильтра события – появляется при обработке таких событий, в названии которых присутствует вопросительный знак. Параметры,

доступные в узле фильтра событий можно изменить в процессе обработки события. Например, «закрытие приложения?» в зависимости от значения логической переменной, поданной на этот узел, событие будет обработано или проигнорировано;

узел динамического события – по умолчанию не показывается, для отображения следует выбрать из контекстного меню пункт **Show Dynamic Event Terminals**.

Структура события работает так же, как и структура варианта. В зависимости от того, какое событие происходит, структура выполняет тот или иной вариант блок-диаграммы. Для каждого варианта можно настроить одно или несколько событий, по которым он будет выполняться. Во время выполнения кода при запуске структуры варианта LabVIEW ждет указанного события и потом выполняет соответствующую поддиаграмму. После выполнения всех записанных в этой поддиаграмме действий выполнение структуры завершается.

При создании структуры события в ней уже есть вариант, который выполняется по истечению времени ожидания (**Timeout**). Время ожидания в миллисекундах подключается к терминалу времени ожидания. В случае если никаких данных на этот терминал не поступает или на него подано -1, поддиаграмма этого варианта не выполняется вообще. Чтобы добавить событие и перейти к диалоговому окну (рис. 10.1) редактора событий выберите в контекстном меню структуры **Add Event**

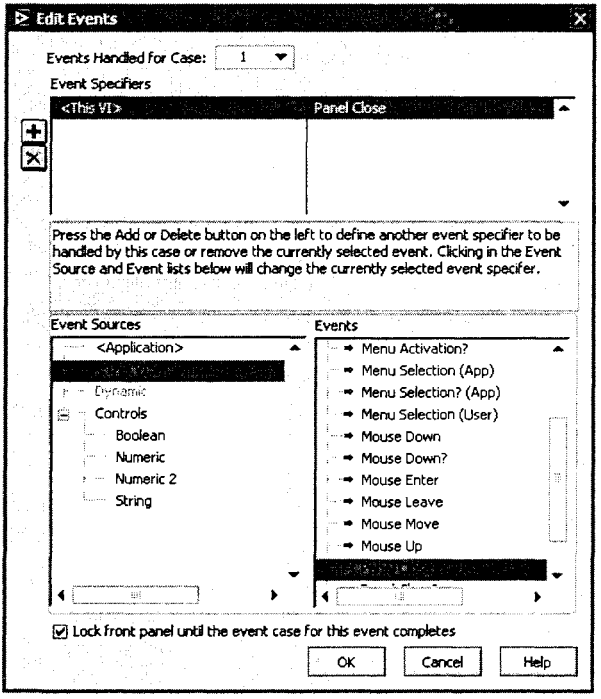


Рис. 10.1

Case. В поле **Event Specifiers** отображается описание текущего одного или нескольких событий для варианта, который выбран чуть выше в ниспадающем меню **Event Handled for Case**. Удалить или добавить событие для текущего варианта можно с помощью кнопок, расположенных левее этого поля.

В поле **Event Sources** представлены источники события. В них входят четыре основные категории: приложение (**Application**), текущий ВП (**This VI**), динамический источник (**Dynamic**), элементы управления (**Controls**). В поле **Events** производится выбор события. В табл. 10.1 представлены обрабатываемые события.

Таблица 10.1

Событие	*Описание*
	Приложение
Application Exit;	
Application Exit?	*Выход из LabVIEW.*
Timeout	*Превышение времени ожидания. Это событие при создании структуры создается по умолчанию.*
Текущий ВП	
Key Down;	
Key Down?	*Нажатие любых клавиш на клавиатуре в любом месте лицевой панели.*
Key Repeat;	
Key Repeat?	*Нажатие и удержание любой клавиши в любом месте лицевой панели.*
Key Up	*Отпускание клавиши на клавиатуре.*
Menu Activation?	*Вызов меню при помощи мыши либо горячих клавиш, например, Alt+F для открытия меню File. К этому событию относится также и вызов горячими клавишами какого-либо пункта меню, например, Ctrl+C при копировании текста. К событию не относится нажатие различных редактирующих клавиш, таких как «вставить» (Insert), «удалить» (Delete), «в начало» (Home), «в конец» (End), «на страницу вверх» (Page Up), «на страницу вниз» (Page Down), а также стрелок.*
Menu Selection (App);	
Menu Selection? (App)	*Выбор какого-либо пункта из ниспадающего меню приложения, например, Help ⇒ Show Context Help.*
Menu Selection (User)	*Выбор пункта пользовательского меню. В главе 16 показан пример использования этого события для создания пункта меню «О программе».*
Mouse Down;	
Mouse Down?	*Нажатие на клавиши мыши.*
Mouse Enter	*Перемещение курсора в пределы лицевой панели.*
Mouse Leave	*Перемещение курсора из пределов лицевой панели.*
Mouse Move	*Движение мышью.*
Mouse Up	*Отпускание клавиши мыши.*
Panel Close;	
Panel Close?	*Закрытие лицевой панели ВП.*
Panel Resize	*Изменение размеров лицевой панели, сворачивание или разворачивание окна, восстановление окна лицевой панели к исходным размерам.*

Таблица 10.1 (окончание)

Событие	Описание
	Приложение
Элементы управления и индикации	
Key Down;	
Key Down?;	
Key Repeat;	
Key Repeat?;	
Key Up;	
Mouse Down;	
Mouse Down?;	
Mouse Enter;	
Mouse Leave;	
Mouse Move;	
Mouse Up	*То же самое событие, что и одноименное событие в категории текущего ВП, но относящееся только к определенному элементу управления или индикации, а не ко всей лицевой панели.*
Value Change	*Изменение значения элемента управления или индикации.*
ListBox Events, MultiColumn ListBox, Tree	
Double Click	*Двойной щелчок по элементу списка.*
Tree	
Drag; Drag?	*Перетаскивание элемента дерева в новое место.*
Drop; Drop?	*Перетаскивание элемента дерева в навое места и удоление элемента в старом.*
Item Clase; Item Close?	*Сварачивание ветки в дереве.*
Item Open;	
Item Open?	*Разворачивание ветки в дереве.*

В таблице встречается двойное обозначение одного и того же события. Существует два вида событий пользовательского интерфейса – уведомление и фильтрация. Уведомляющие события показывают, что пользователь уже совершил некоторое действие, например, изменил значение элемента управления. Фильтрующие события информируют вас о действиях пользователя перед тем, как их обработает LabVIEW. Это позволяет вам каким-то образом отреагировать на действия пользователя, в том числе и отменить его действия. Для фильтрующих событий в обозначении используется знак вопроса. В обозначении уведомляющих событий знака вопроса нет.

После того как вы закончили редактирование событий, нажмите **OK**. Если вам понадобится еще раз обратиться к этому окну, в контекстном меню структуры событий выберите **Edit Events Handled by This Case**.

Когда вы перейдете на блок-диаграмму, обратите внимание, что в каждом варианте у левого края структуры появились узлы данных события, а в случае использования фильтрующих событий – у правого края структуры появились также и узлы фильтра событий. Как было выше упомянуто, узел данных событий передает возможные параметры события в поддиаграмму варианта. В табл. 10.2 представлены некоторые параметры узла данных события.

Таблица 10.2

Параметр	*Описание*	*Событие*
Source	*Целое число, которое характеризует источник события, например, действия пользователя или* ***ActiveX****.*	*Присутствует у всех событий.*
Type	*Целое число, которое характеризует тип события, такое как изменение значения, превышение времени ожидание или нажатие клавиши.*	*Присутствует у всех событий.*
Time	*Время внутренних часов в миллисекундах, когда событие произошло.*	*Присутствует у всех событий.*
VIRef	*Ссылка на ВП, в котором событие произошло*	*Присутствует у всех событий категории текущего ВП.*
CtlRef	*Ссылка на элемент управления или индикации, которому относится событие.*	*Присутствует у всех событий к категории элементов управления и индикации.*
FocusObj	*Ссылка на объект, который имеет клавиатурную факусировку. Если событие рассматривается для конкретного объекта лицевой панели, то ссылка относится к конкретной составляющей объекта (метке, шкале и т.д.).*	*Присутствует у всех событий, описывающих действия над клавишами клавиатуры.*
MenuRef	*Ссылка на пункт меню, из которого был выбран элемент.*	*Присутствует у всех событий, описывающих работу с меню.*
ItemTag	*Специальное имя выбранного пункта меню описывающих работу с меню.*	*Присутствует у событий,*
ItemPath	*Путь выбранного пункта меню*	*Присутствует у событий, описывающих работу с меню.*
Coords	*Координаты курсора в тот момент, когда событие произошло. Координаты отсчитываются от начала координат лицевой панели.*	*Присутствует у всех событий, описывающих действия над клавишами мыши.*
Button	*Целое число, которое характеризует, какая клавиша мыши была нажата. Для левой клавиши мыши оно равно 1, для правой – 2.*	*Присутствует у всех событий, описывающих действия над клавишами мыши.*
OldVal, NewVal	*Значение элемента управления или индикации до или после его изменения соответственно.*	*Присутствует у события изменения значения* ***Value Chonge Event****.*

В узле данных события можно убирать лишние терминалы. Для этого просто уменьшите размер узла данных события или в контекстном меню выберите пункт **Remove Element**. Если вам потребуется добавить параметр, увеличьте размер узла или в контекстном меню выберите пункт **Add Element**. Поменять параметр, например, с параметра **OldVal** на параметр **NewVal** можно просто, кликнув по терминалу или выбрав в контекстном меню пункт **Select Item**.

Узел фильтра событий может содержать изменяемые параметры. Практически все параметры узла фильтра событий повторяют параметры узла данных события. Однако есть и специальные параметры. К такому параметру относится параметр **Discard**. Его применение рассматривается в следующем примере.

Пример 10.1. Обработка события закрытия ВП

Используя структуру события можно перехватить событие, когда пользователь закрывает ВП (использует «крестик» в правом верхнем углу окна). Это может быть удобно, например, если Вы хотите совершить некоторые действия, прежде чем завершить работу приложения (сохранить настройки, закрыть файлы и т.д.).

Здесь структура обработки события содержит две поддиаграммы (рис. 10.2)

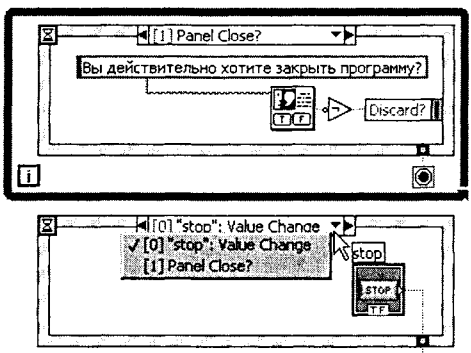


Рис. 10.2

[0] **«stop»:Value Change** – выполняется в случае изменения значения кнопки «stop», моделирует «нормальный» выход из программы

[1] **Panel Close?** – выполняется в случае попытки пользователя закрыть окно программы. Содержит узел **«Two Button Dialog»** (**Function ⇒ Time & Dialog ⇒ Two Button Dialog**),который выдает пользователю окно показанное на рис. 10.3.

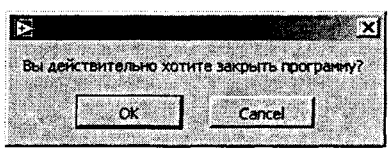


Рис. 10.3

Закрытие программы происходит только в случае, если пользователь выберет кнопку OK. В обратном случае на узел **«Discard?»** (сброс) поступает **«True»** и закрытия окна не происходит (событие сбрасывается).

Задание 10.2. Секундомер

Рассмотрим еще работу структуры события, создав полезное приложение с функциями обычного секундомера. Для этого проделайте следующие операции:

На блок-диаграмму поместите структуру событий, выбрав **Structures ⇒ Event Structures**. В контекстном меню структуры событий выберите **Edit Events Handled by This Case**. В поле источника события **Event Sources** выберите этот ВП **This VI**. В поле событие Event выберите **Mouse Down**. Таким образом, текущий вариант **Case** указанная структура будет выполнять, когда по лицевой панели щелкнут мышкой.

Снимите флажок с **Lock front panel until the event case for this event completes**, что позволит работать с лицевой панелью, пока указанный **Case** будет выполняться. Нажмите **OK**.

Используя палитру **Express ⇒ Execution Control**, поместите на блок-диаграмму цикл **While** с предусмотренной кнопкой останова программы. Поместите внутри цикла ВП затраченного времени, выбрав **Time & Dialog ⇒ Elapsed Time**. В появившемся диалоговом окне снимите флажок с **Automatically reset after time target**. Эта функция сбрасывает счетчик времени по достижению какого-либо времени. В данном случае эта функция не нужна.

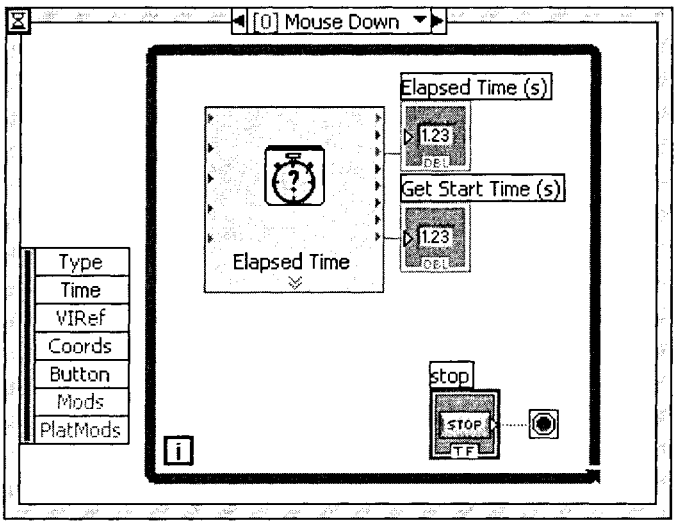


Рис. 10.4

Создайте для выходов **Elapsed Time (s)** и **Get start Time (s)** элементы индикации, выбрав в обоих случаях в контекстном меню **Create** ⇒ **Indicator**. Блок-диаграмма будет выглядеть так, как показано на рис. 10.4.

Перейдите на лицевую панель. В свойствах элементов индикации перейдите на вкладку **Format and Precision**. Для этого в контекстном меню элемента индикации **Elapsed Time (s)** выберите **Properties**. В поле выбора формата отображения числа выберите **Relative Time**. Для элемента индикации **Get start Time (s)** в качестве формата отображения следует выбрать **Absolute Time**. Это позволит наблюдать время в привычном для нас формате.

Запустите ВП. При щелчке мышкой по лицевой панели, **Elapsed Time (s)** отсчитывает время, **Get start Time (s)** отображает момент начала отсчета. Остановить секундомер можно, нажав на кнопку **Stop**.

В заключение отметим, что структуру события следует применять осторожно.

Во-первых, избегайте использовать структуру события вне цикла (не помещайте структуру рядом с циклом). Поскольку в этом случае структура обработает событие всего один раз. Если событие наступит во второй раз, оно не будет обработано, т.к. структура событий уже выполнила свои функции ранее. Во-вторых, избегайте использовать две структуры события в одном цикле. Настроить различные типы событий вы сможете и в одной структуре. В общем, следует тщательно продумывать какие события обрабатывать, какие приемы пользовательского интерфейса при этом применять (к примеру, диалоговые окна).

Выводы

Структура события позволяет обработать различного рода действий пользователя: нажатие клавиш, выбор меню, изменение параметров окна. Уведомляющие события используют для обработки уже произошедшего события. Фильтрующие события применяют для реагирования на еще не случившееся событие с возможностью отменить действия пользователя.

Лекция 11

Кластеры

Изучается составной тип данных кластер, его создание и редактирование, функции. Особое внимание уделяется кластерам ошибок, позволяющим отследить и обработать случаи незапланированного поведения программы.

Кластеры объединяют элементы разных типов данных. Кластеры играют ту же роль, что и структуры в текстовых языках программирования.

Объединение нескольких групп данных в кластер устраняет беспорядок на блок-диаграмме и уменьшает количество полей ввода/вывода данных, необходимых подпрограмме ВП. Максимально возможное количество терминалов ввода/вывода данных ВП равно 28. Если лицевая панель содержит более 28 элементов, которые необходимо использовать в ВП, можно некоторые из них объединить в кластер и связать кластер с полем ввода/вывода данных. Как и массив, кластер может быть элементом управления или индикации, однако при этом кластер не может содержать одновременно элементы управления и индикации.

В кластере, как и в массиве, все элементы упорядочены, но обратиться по индексу к ним нельзя, необходимо сначала разделить их. Для этого предназначена функция **Unbundle By Name**, которая обеспечивает доступ к определенным элементам кластера по их имени.

Создание кластеров из элементов управления и индикации

Для создания кластеров из элементов управления и индикации следует выбрать шаблон кластера на палитре **Controls** ⇒ **Array & Cluster** и поместить его на лицевую панель. После этого шаблон кластера следует заполнить элементами. Изменить размер кластера можно с помощью курсора.

На рис. 11.1 показан кластер, содержащий три элемента управления.

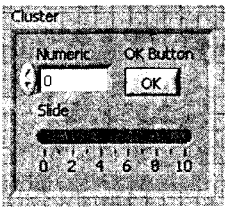


Рис. 11.1

Порядок элементов в кластере

Каждый элемент кластера имеет свой логический порядковый номер, не связанный с положением элемента в шаблоне. Первому помещенному в кластер элементу автоматически присваивается номер 0, второму элементу – 1 и так далее. При удалении элемента порядковые номера автоматически изменяются.

Порядок элементов в кластере определяет то, как элементы кластера будут распределены по терминалам функций **Bundle** (объединения) и **Unbundle** (разделения) на блок-диаграмме.

Посмотреть и изменить порядковый номер объекта, помещенного в кластер, можно, щелкнув правой кнопкой мыши по краю кластера и выбрав из контекстного меню пункт **Reorder Controls In Cluster**. Панель инструментов и кластер примут вид, показанный на рис. 11.2.

В белом поле (4) указан текущий порядковый номер элемента, в черном (5) – новый порядковый номер. Для установки порядкового номера элемента нужно в поле ввода текста **Click to set** to ввести число и нажать на элемент. Порядковый номер элемента изменится. При этом корректируются порядковые номера других элементов. Сохранить изменения можно, нажав кнопку **Confirm** на панели инструментов. Вернуть первоначальные установки можно, нажав кнопку **Cancel**.

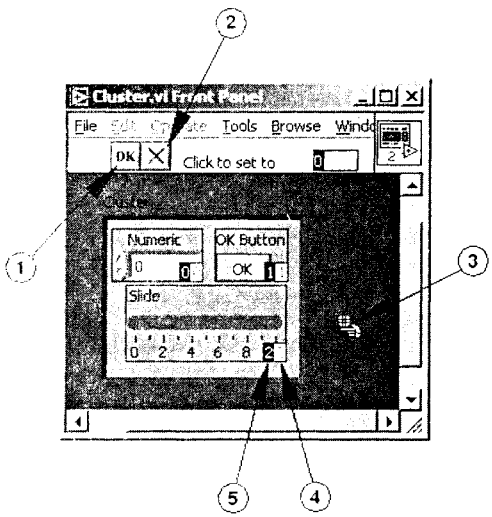


Рис. 11.2

1. *Кнопка подтверждения (**Confirm button**)*
2. *Кнопка отмены (**Cancel button**)*
3. *Курсор определения порядка (**Cluster order cursor**)*
4. *Текущий порядковый номер (**Current order**)*
5. *Новый порядковый номер (**New order**)*

Соответствующие элементы, определенные в кластерах одинаковыми порядковыми номерами, должны иметь совместимые типы данных. Например, в одном кластере элемент 0 является числовым элементом управления, а элемент 1 – строковым элементом управления. Во втором кластере элемент 0 – числовой элемент индикации и элемент 1 – строковый элемент индикации, тогда кластер элементов управления корректно соединится с кластером элементов индикации.

Если изменить порядковые номера элементов в одном из кластеров, проводник данных между кластерами будет разорван, так как типы данных элементов кластеров не будут соответствовать друг другу.

Создание кластера констант

На блок-диаграмме можно создать кластер констант, выбрав в палитре **Functions ⇒ Cluster** шаблон **Cluster Constant** и поместив в него числовую константу или другой объект данных, логический или строковый.

Если на лицевой панели кластер уже существует, то кластер констант на блок-диаграмме, содержащий те же элементы, можно создать, просто перетащив кластер с лицевой панели на блок-диаграмму или, щелкнув правой кнопкой мыши на кластере, выбрать из контекстного меню пункт **Create ⇒ Constant**.

Функции работы с кластерами

Для создания и управления кластерами используются функции, расположенные на палитре **Functions ⇒ Cluster**. Функции **Bundle** и **Bundle by Name** используются для сборки и управления кластерами. Функции **Unbundle** и **Unbundle by Name** используются для разборки кластеров.

Эти функции также можно вызвать, щелкнув правой кнопкой мыши по терминалу данных кластера и выбрав из контекстного меню подменю **Cluster Tools**. Функции **Bundle** и **Unbundle** автоматически содержат правильное количество полей ввода/вывода данных. Функции **Bundle by Name** и **Unbundle by Name** в полях ввода/вывода данных содержат имя первого элемента кластера.

Сборка кластеров

Для сборки отдельных элементов в кластер используется функция **Bundle**. Эта же функция используется для изменения данных в элементе уже существующего кластера. Инструмент ***перемещение*** используется для добавления полей ввода данных, для этого также можно щелкнуть правой кнопкой по полю ввода данных и выбрать из контекстного меню пункт **Add Input**. На рис. 11.3 собирается кластер из трех элементов: численного, строкового и булевого типа.

На поле ввода данных **cluster** (которое находиться наверху функции) можно подать уже имеющийся кластер, в этом случае узел **Bundle** выполняет функцию **замены** значений всех или некоторых элементов кластера. Количество полей ввода данных функции должно соответствовать количеству элементов во входящем кластере. Например, на рис. 11.4 показано изменение значений двух полей кластера.

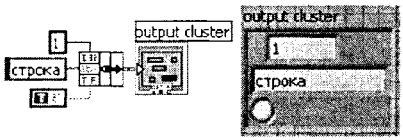


Рис. 11.3

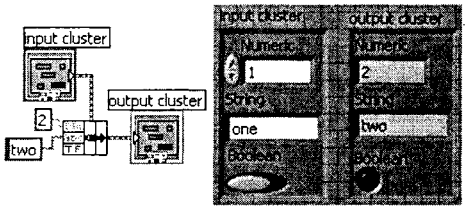


Рис. 11.4

Функция **Bundle by Name** работает так же как функция **Bundle**, но вместо обращения к элементу кластера по его порядковому номеру обращается к нему по его собственной метке (имени). При этом можно получить доступ только к элементам, имеющим собственную метку. Количество полей ввода данных не требует соответствия с количеством элементов в кластере. На рис. 11.5 показан пример использования функции **Bundle by Name** для полученного выше кластера.

Пример функционально повторяет пример рис. 11.4, т.е. изменяет значение двух элементов в кластере, но обращение к элементам происходит по их именам.

С помощью инструмента ***управление*** можно щелкнуть по полю ввода данных терминала и выбрать желаемый элемент из выпадающего меню. Можно также щелкнуть правой кнопкой мыши по полю ввода данных и выбрать элемент в разделе контекстного меню **Select Item**.

Использовать функцию **Bundle by Name** следует при работе со структурами данных, которые могут меняться в процессе работы. Чтобы добавить новый элемент в кластер или изменить порядковый номер элемента, нет необходимости вновь подключать функцию **Bundle by Name**, так как имя элемента все еще действительно.

Разделение кластера

Функция **Unbundle** используется для разбиения кластеров на отдельные элементы.

Функция **Unbundle by Name** используется для выделения из кластера элементов по определенному имени. Количество полей вывода данных не зависит от количества элементов в кластере.

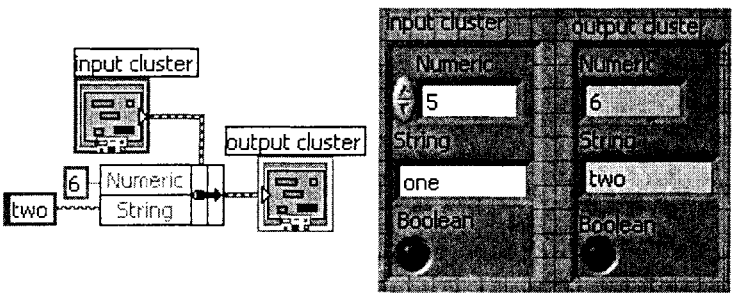


Рис. 11.5

С помощью инструмента ***управление*** можно щелкнуть по полю вывода данных и выбрать желаемый элемент из контекстного меню. Можно также щелкнуть правой кнопкой мыши по полю вывода данных и выбрать из контекстного меню пункт **Select Item**.

Так, функция **Unbundle** при использовании кластера, показанного на рис. 11.6, имеет три поля вывода данных, которые соотносятся с тремя элементами кластера. Необходимо знать порядок элементов в кластере для корректного сопоставления элементов имеющих одинаковый тип (как **Thermometer** и **Numeric** в данном примере). Если использовать функцию **Unbundle by Name**, то полей вывода данных может быть произвольное количество, и обращаться к отдельным элементам можно в произвольном порядке.

Пример 11.1. Масштабирование кластера (рис.11.7)

Каждый элемент в кластере имеет свой масштабный коэффициент. Предположим, что исходные данные, значения давления, скорости потока и температуры были получены с соответствующих датчиков напряжения. Затем ВП масштабирует эти значения и выдает фактические значения физической величины.

Рис. 11.6

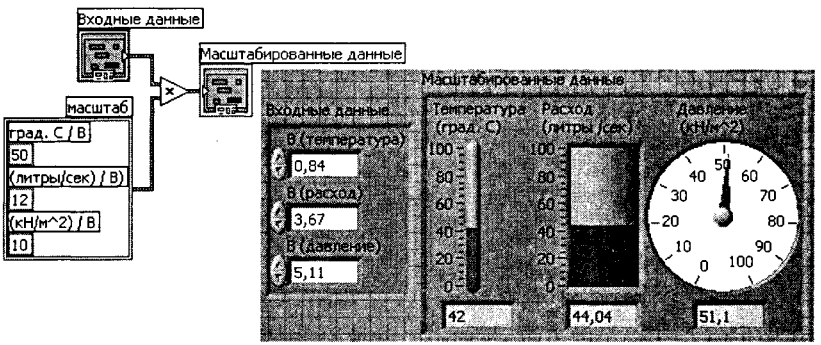
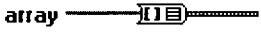


Рис. 11.7

Преобразование кластера в массив

В палитрах **Function ⇒ Array** и **Function ⇒ Cluster** имеются две функции позволяющие преобразовать массив в кластер и наоборот кластер в массив.



Array To Cluster – преобразует одномерный массив в кластер из элементов того же типа. Через контекстное меню этого узла пункт Cluster Size можно установить размер результирующего кластера

Cluster To Array – преобразует кластер из однотипных элементов в массив

Пример 11.2. Преобразования массива в кластер и наоборот

На рис. 11.8 Вы видите, как одномерный массив из трех элементов преобразуется сначала в кластер. Этот кластер выводится на лицевую панель, затем кластер преобразуется обратно в массив. Обратите внимание на определение размера кластера для функции **Array To Cluster**.

Кластеры ошибок

Даже в самой отлаженной программе встречаются ошибки, поэтому никогда нельзя предусмотреть все проблемы, которые могут возникнуть у пользователя. Без механизма проверки ошибок, о ВП можно сказать только то, что он не работает. Проверка ошибок позволяет узнать, в каком месте и почему произошел сбой.

При программировании любых операций ввода/вывода стоит подумать о возможном появлении ошибок. Почти все операции ввода/вывода возвращают информацию об ошибке. Чтобы правильно обрабатывать ошибки, в ВП нужно осо-

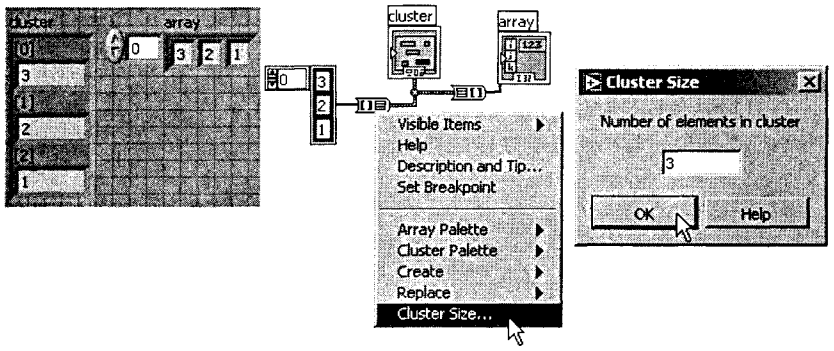


Рис. 11.8

бо тщательно выполнять проверку для таких операций ввода/вывода, как файловые и последовательные операции, операции работы с приборами, операции получения данных, а также процессы передачи информации.

Проверка на ошибки в ВП выявляет, в частности:

- Неправильную инициализацию связи с внешним устройством или записи в него некорректной информации.
- Ситуацию когда внешнее устройство не включено или не работает.
- Изменение путей к необходимым файлам (например по причине переустановки системного программного обеспечения).

Обработка ошибок

Ошибки при работе над любым проектом неизбежны. Поэтому при создании проекта важным этапом является отладка приложения, обработка ошибок. Обработка ошибок подразумевает сопоставление какого-либо действия возможным непланируемым событиям, например, вывода диалогового окна. В LabVIEW не реализована автоматическая обработка ошибок. Это сделано для того, чтобы можно было самостоятельно выбирать метод, которым обрабатываются ошибки. Например, если для ВП истекло время ожидания ввода/вывода, можно сделать так, чтобы не прекращалась работа всего приложения. Можно также заставить ВП повторить попытку через некоторое время. Процесс обработки ошибок в LabVIEW происходит на блок-диаграмме.

Существует два способа возврата ошибок в ВП и функциях: с помощью числа, обозначающего код ошибки и с помощью кластера ошибок. Как правило, функции используют число – код ошибки, а ВП принимают на вход и выдают на выходе информацию об ошибках в виде кластера.

Обработка ошибок в LabVIEW также построена на модели поточного программирования. Как и другие данные, информация об ошибках проходит через ВП. Для передачи информации об ошибках через ВП необходимо использовать входной и выходной кластеры ошибок, а также включить в конце ВП обработчик ошибок для определения того, были ли сбои в процессе работы ВП.

При выполнении ВП LabVIEW следит за появлением ошибок, и, как только где-нибудь происходит сбой, составляющие части ВП перестают выполняться и только передают ошибку дальше, на выход. Для обработки появляющихся в ВП ошибок в конце потока выполнения обычно используется показанный на рис. 11.9 простой обработчик ошибок **Simple Error Handler**. **Simple Error Handler** находится на палитре **Functions ⇒ Time and Dialog**.

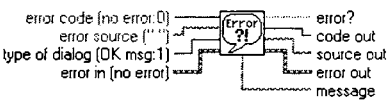


Рис. 11.9

Подсоедините кластер ошибок к полю входных данных «**Error In**» (по умолчанию ошибки нет). В случае возникновения ошибки данный ВП выводит диалоговое окно с информацией о возникшей ошибке.

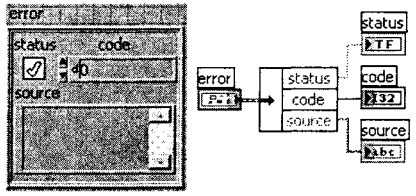


Рис. 11.10

Кластеры ошибок

На рис. 11.10 приведены компоненты кластеров ошибок, расположенных на палитре **Controls** ⇒ **Array & Cluster**.

- **status** является логической величиной, принимающей значение **True** в случае возникновения ошибки. Большинство ВП, функций и структур, которые принимают логические данные, используют этот параметр. При возникновении ошибки кластер ошибок передает функции значение **True**.
- **code** является целым 32-х битным числом со знаком, которое соответствует ошибке. В случае если **status** имеет значение **False**, а code отличен от нуля, то, скорее всего, это предупреждение, а не фатальная ошибка.
- **source** является строкой, которая определяет место возникновения ошибки.

Для создания входа и выхода ошибок в подпрограммах ВП используются кластеры ошибок из элементов управления и индикации.

Объяснение ошибки

При появлении ошибки можно щелкнуть правой кнопкой мыши внутри кластера и из контекстного меню (рис. 11.11) выбрать пункт **Explain Error**. Появится диалоговое окно **Explain Error**, содержащее информацию об ошибке. В контекстном меню также есть пункт **Explain Warning**, если в ВП нет ошибок, но есть предупреждения.

Диалоговое окно **Explain Error** также можно вызвать из меню **Help**.

Использование цикла пока (While) при обработке ошибок

Кластер ошибок может быть подсоединен к терминалу условия цикла **While** для остановки цикла. Когда кластер ошибок подсоединен к терминалу условия, на тер-

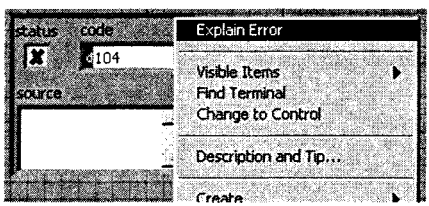


Рис. 11.11

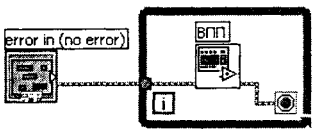


Рис. 11.12

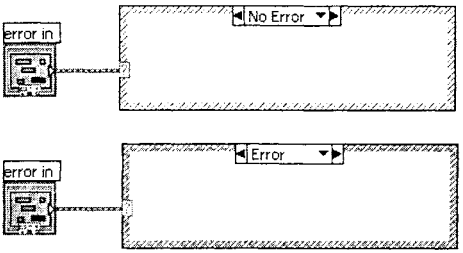


Рис. 11.13

минал подаются только значения параметра **status** – **True** или **False**. Например, цикл **While** показанный на рис. 11.12 выполняется до тех пор, пока ВПП не возвратит в выходном кластере наличие ошибки (поле **status** примет значение **true**).

Если к терминалу условия подсоединен кластер ошибок, пункты контекстного меню меняются с **Stop if True** и **Continue if True** на **Stop on Error** и **Continue while Error**.

Использование структуры варианта (Case) при обработке ошибок

Кластер ошибок может управлять «структурой варианта» (**Case**), в этом случае есть только два варианта структуры: нет ошибки (**No Error**) и ошибка (**Error**), для которых граница структуры имеет красный и зеленый цвет соответственно (см. рис. 11.13). Структура **Case** выполняет вариант, основываясь на информации о наличии ошибки.

В диалоге **New** (сразу после запуска LabVIEW) присутствует шаблон **«SubVI with Error Handling»** (ВПП с обработкой ошибок). Данный шаблон предназначен для создания ВПП который имеет входной и выходной терминал ошибки и две поддиаграммы структуры **Case**: на случай наличия и отсутствия ошибки на входе. Данный шаблон рекомендуется использовать для создания ВПП в которых требуется обработка ошибок.

Выводы

Тип данных кластер – наиболее универсальный, применяющийся практически в любом ВП. Используется при группировке различных по типу данных, тем самым, уменьшая количество проводников на блок-диаграмме. Использование кластеров ошибки помогает не только отслеживать появление ошибок в программе и гибко их обрабатывать, но и упорядочить выполнение стандартных последовательностей действий.

Лекция 12

Графическое представление данных

Рассматриваются широкие возможности LabVIEW по визуализации данных, включающие график диаграмм, график осциллограмм и двухкоординатный график.

В LabVIEW имеются разнообразные и достаточно гибкие средства для графического представления данных. Можно использовать различные графики, на которых можно отображать одну или несколько кривых, настроить цвет, тип представления, масштаб шкал и т.д.

График диаграмм

График диаграмм (**Waveform Chart**)– специальный элемент индикации в виде одного и более графиков. График диаграмм расположен на палитре **Controls** ⇒ **Graph**. На рис. 12.1 показан пример графика диаграмм с двумя графиками: экспериментальные данные и их среднее значение.

График диаграмм использует три различных режима отображения данных: **strip chart**, **scope chart** и **sweep chart** (см. рис. 12.2). Режим по умолчанию – **strip chart**.

Задание режима осуществляется щелчком правой клавишей мыши по диаграмме и выбором пункта **Advanced** ⇒ **Update Mode** из контекстного меню.

Режим **strip** chart представляет собой экран, прокручиваемый слева направо, подобно бумажной ленте. Режимы **scope chart** и **sweep chart** подобны экрану осциллографа и отличаются большей скоростью отображения данных по сравнению с **strip chart**. В режиме **scope chart** по достижении правой границы поле графика очищается, и заполнение диаграммы начинается с левой границы. Режим **sweep chart**, в отличие от режима **scope chart**, не очищает поле графика, а отделяет новые данные от старых вертикальной линией – маркером.

Соединение графиков

Для создания диаграмм достаточно соединить поле вывода скалярной величины с терминалом данных графика диаграмм. В примере на рис. 12.3 тип данных на терминале графика диаграмм, соответствует входному типу данных.

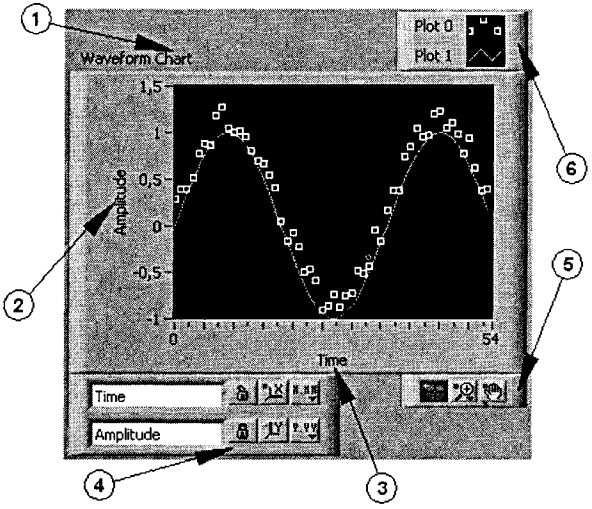


Рис. 12.1

1. *Название* ***(Label)***
2. *Шкала Y* ***(Y-scale)***
3. *Шкала X* ***(X-scale)***
4. *Панель управления шкалами* ***(Scale legend)***
5. *Палитра инструментов для работы с графиком* ***(Graph palette)***
6. *Панель управления графиком* ***(Plot legend)***

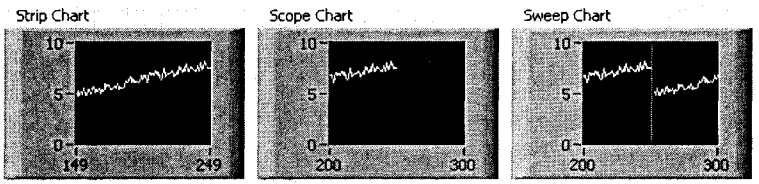


Рис. 12.2

График диаграмм может отображать несколько графиков. Для объединения отображаемых данных используется функция **Bundle**, расположенная в палитре **Functions ⇒ Cluster**. Например, блок-диаграмма, показанная на рис. 12.4, с помощью функции **Bundle** объединяет выходные данные трех подпрограмм ВП для последующего отображения трех кривых на графике диаграмм.

Терминал данных графика диаграмм имеет кластерный тип данных в соответствии с полем вывода функции **Bundle**. Для увеличения количества полей ввода данных функции **Bundle** необходимо с помощью инструмента перемещение изменить ее размер.

Имеется два типа отображения данных **Stack Plots** (кривые расположены друг под другом) и **Overlay Plots** (все кривые на одном графике), выбрать требуемый тип можно через контекстное меню.

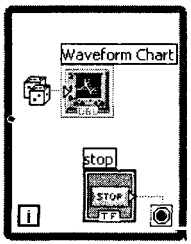


Рис. 12.3

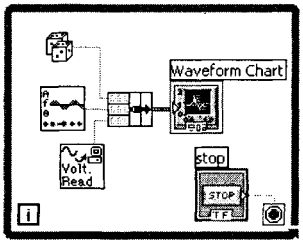


Рис. 12.4

График осциллограмм и двухкоординатный график осциллограмм

На графики в виде осциллограмм обычно подают массив данных. На рис. 12.5 показаны элементы графика.

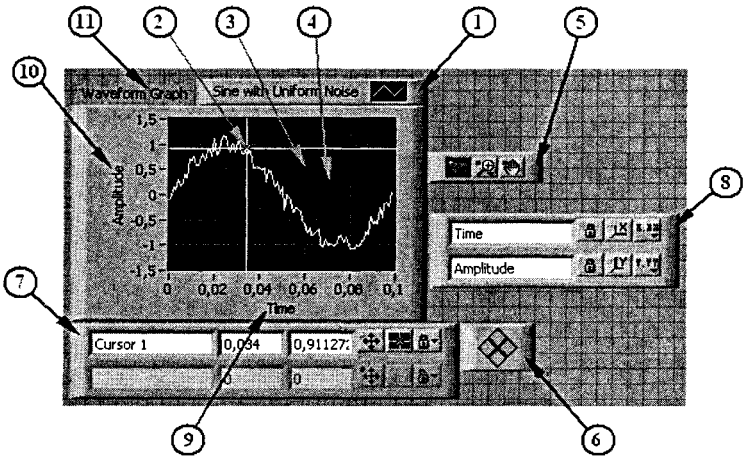


Рис. 12.5

1. Панель управления свойствами осциллограмм (**Plot legend**)
2. Курсор (**Cursor**)
3. Основная размерная сетка (**Grid mark**)
4. Дополнительная размерная сетка (**Mini-grid mark**)
5. Палитра элементов управления графиком (**Graph palette**)
6. Панель перемещения курсора (**Cursor mover**)
7. Панель управления свойствами курсора (**Cursor legend**)
8. Панель управления шкалой (**Scale legend**)
9. Шкала X (**X-scale**)
10. Шкала Y (**Y-scale**)
11. Собственная метка графика (**Label**)

График осциллограмм (**Waveform Graph**) и двухкоординатный график осциллограмм (**X-Y Graph**) также расположены на палитре **Controls** ⇒ **Graph**. График осциллограмм отображает только однозначные функции, такие как $y = f(x)$, с точками, равномерно распределенными по оси X. Двухкоординатный график осциллограмм отображает любой набор точек, будь то равномерно распределенная выборка или нет.

Для изображения множества осциллограмм необходимо изменить размер панели **Plot legend**. График множества осциллограмм используется с целью экономии пространства на лицевой панели и для сравнения осциллограмм данных между собой. График осциллограмм и двухкоординатный график осциллограмм автоматически поддерживают режим отображения множества осциллограмм.

Одиночный график осциллограмм

Одиночный график осциллограмм работает с одномерными массивами и представляет данные массива в виде точек на графике, с приращением по оси X равным 1 и началом в точке $x = 0$. Графики также отображают кластеры, с установленным начальным значением x, Δx и массивом данных по шкале y.

В примере показанном на рис. 12.6 использован **Expression Node** (Узел выражений) расположенный в **Function** ⇒ **Numeric**. Функция **mod** вычисляет остаток от деления двух целых чисел.

График множества осциллограмм

График множества осциллограмм работает с двумерными массивами данных, где каждая строка массива есть одиночная осциллограмма данных и представляет данные массива в виде точек на графике, с приращением по оси X равным **1** и началом в точке $x = 0$.

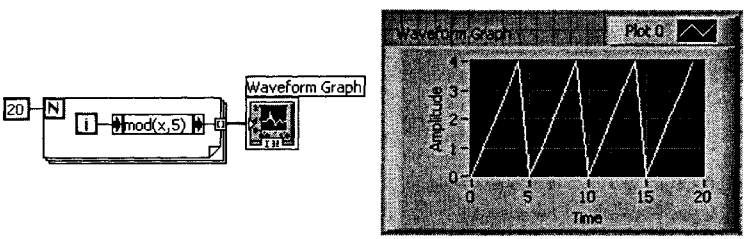


Рис. 12.6

Пример 12.1. График множества осциллограмм

В примере на рис. 12.7 использована функция **Quotient & Remainder** (расположенная в палитре **Functions** ⇒ **Numeric**) вычисляющая частное и остаток (аналогично **mod**) двух чисел.

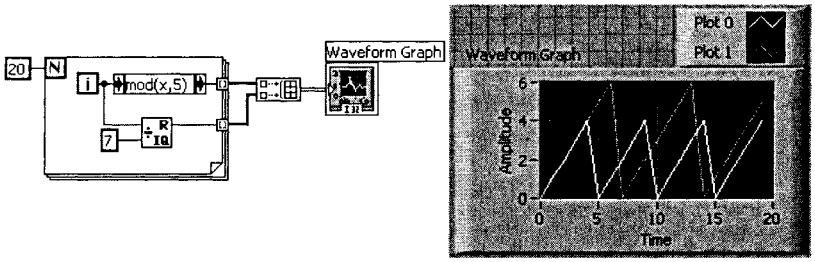


Рис. 12.7

Для того, что бы иметь возможность настроить вид обеих функций необходимо при помощи инструмента ***перемещение*** изменить размер панели управления свойствами осциллограмм.

Для представления каждого столбца двумерного массива данных в виде осциллограммы на графике необходимо соединить массив с терминалом графика, затем щелкнуть правой кнопкой мыши по полю графика и выбрать пункт контекстного меню **Transpose Array** (транспонирование массива).

Графики множества осциллограмм так же отображают кластеры, состоящие из начального значения x, Δx и двумерного массива данных по шкале y. График представляет данные по шкале y в виде точек с приращением Δx по оси x и началом в точке $x = 0$. Пример показан на рис. 12.8.

При помощи контекстного меню панели управления свойствами осциллограмм вы можете настроить вид точек на графике, например сделать их круглыми.

Графики множества осциллограмм отображают также и кластеры с установленным начальным значением x, Δx и массивом данных, содержащим кластеры. Каждый кластер содержит массив точек, отображающих данные по шкале *Y*. Для создания массива кластеров следует использовать функцию **Bundle**, которая объединяет массивы в кластеры. Далее, с помощью функции **Build Array** создается массив кластеров. Можно также использовать функцию **Build Cluster Array**, которая создает массив кластеров с определенными полями ввода данных.

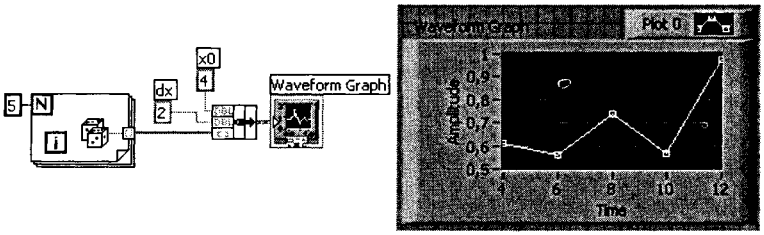


Рис. 12.8

Одиночные двухкоординатные графики осциллограмм

Одиночный двухкоординатный график осциллограмм работает с кластерами, содержащими массивы x и y. Двухкоординатный график осциллограмм также воспринимает массивы точек, где каждая точка является кластером, содержащим значения по шкалам x и y.

Обе показанные на рис.12.9 блок диаграммы при выполнении выводят одинаковые графики. Обратите внимание на различие типов данных: кластер из двух одномерных массивов и массив из кластеров, содержащих пару численных значений.

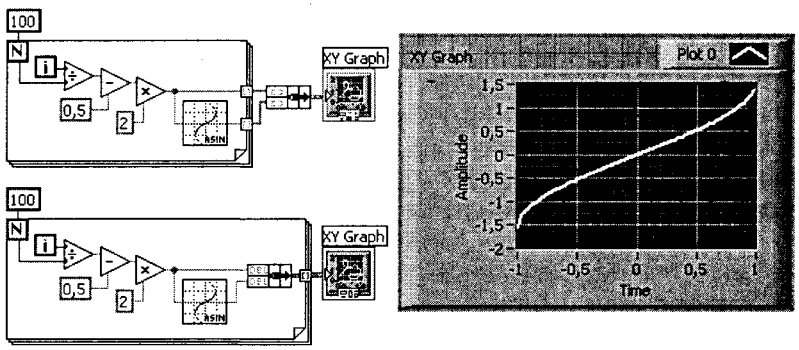


Рис. 12.9

Двухкоординатные графики множества осциллограмм

Двухкоординатные графики множества осциллограмм работают с массивами осциллограмм, в которых осциллограмма данных является кластером, содержащим массивы значений x и y. Двухкоординатные графики множества осциллограмм воспринимают также массивы множества осциллограмм, где каждая осциллограмма представляет собой массив точек. Каждая точка – это группа данных, содержащая значения по x и y.

Графики интенсивности

Графики и таблицы интенсивности (**Intensity graphs and charts**) удобны для представления двумерных данных. Например, для представления топографии местности, где амплитудой является высота над уровнем моря. Как и в случае с графиками диаграмм и осциллограмм, график интенсивности имеет постоянный размер дисплея, а дисплей таблицы интенсивности обладает возможностью прокрутки. Графики и таблицы интенсивности принимают на вход двумерный массив данных, где каждое число соответствует определенному цвету. Положение данного цвета на графике определяется индексами элемента в массиве. Графики и таблицы интенсивности имеют возможность использовать до 256 различных цветов.

Настройки графиков и таблиц интенсивности

Графики и таблицы интенсивности имеют много общих свойств с графиками диаграмм и осциллограмм, которые можно показать или спрятать, выбрав пункт контекстного меню **Visible Items**. Так как в графиках и таблицах интенсивности появляется третье измерение, то необходим дополнительный элемент – элемент управления цветовой шкалой, который определяет диапазон и способ цветового отображения данных. На рис. 12.10 составляющие части графика интенсивности.

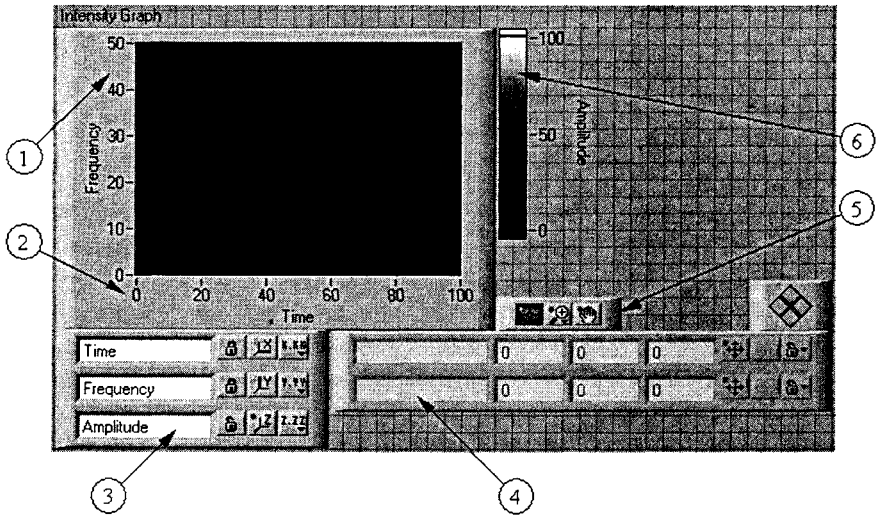


Рис. 12.10

1. *Шкала Y (**Y scale**)*
2. *Шкала X (**X scale**)*
3. *Панель управления шкалами (**Scale legend**)*
4. *Панель управления курсорами (**Scale legend**)*
5. *Палитра инструментов для работы с графиком (**Graph Palette**)*
6. *Шкала Z (цветовая шкала) (**Z scale (color ramp)**)*

Для того чтобы поменять цвет, ассоциированный с маркером, нужно выбрать пункт **Marker Color** в контекстном меню и выбрать цвет в окне выбора цвета. Контекстное меню вызывается инструментами ***управление*** или ***перемещение*** нажатием правой кнопки мыши по маркеру, расположенному около цветовой шкалы. Для добавления маркера к цветовой шкале необходимо нажать правой кнопкой мыши на цветовую палитру и выбрать пункт **Add Marker** из контекстного меню. Чтобы изменить значение какого-либо маркера на цветовой шкале нужно переместить маркер к требуемому значению инструментом ***управление*** или использовать инструмент ***ввод текста*** для ввода нового значения в текстовое поле маркера.

На рис. 12.11 изображен массив размера 4 × 4, представленный на графике интенсивности. На графике показан транспонированный массив.

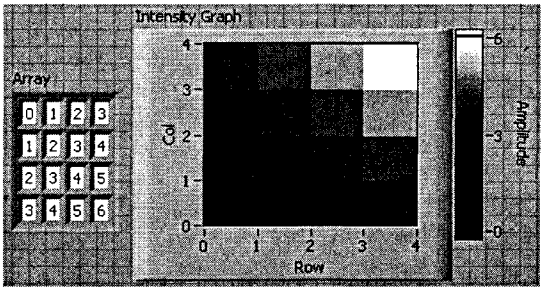


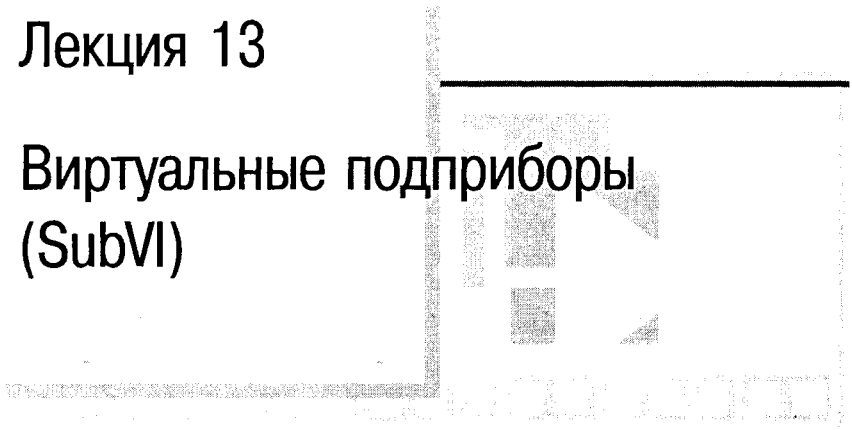
Рис. 12.11

Выводы

Наиболее универсальным из числа рассмотренных возможностей LabVIEW по визуализации данных является двухкоординатный график. Для последовательного отображения данных, как на бумажной ленте самописца (с возможностью просмотра предшествующих данных) необходимо использовать график диаграмм. Одномерные массивами и реальные сигналы удобно отображать на графике осциллограмм. Для отображения двухмерных данных в виде цветового поля целесообразно использовать график интенсивности.

Лекция 13

Виртуальные подприборы (SubVI)



Изучаются создание, редактирование и использование виртуальных подприборов. Описывается процесс редактирования иконки и работы с соединительной панелью.

После того как ВП сформирован, создана его иконка и настроена соединительная панель, виртуальный прибор можно использовать как подпрограмму в других ВП. Виртуальный прибор, используемый внутри другого виртуального прибора, называется Виртуальным Подприбором (ВПП). ВПП соответствует подпрограмме в текстовых языках программирования. Узел ВПП соответствует вызову подпрограммы. Узел – это графическое представление подпрограммы ВП, а не собственно исполняемый код ВПП, так же как вызов подпрограммы в текстовых языках программирования не есть сам исполняемый код подпрограммы. Использование подпрограмм ВП помогает быстро управлять изменениями и отладкой блок-диаграмм.

Создание и настройка ВПП

Следующий шаг после создания блок-диаграммы и формирования лицевой панели ВП – создание иконки ВП и настройка соединительной панели для использования виртуального прибора в качестве ВПП. Каждый виртуальный прибор в правом верхнем углу лицевой панели и в окне блок-диаграммы отображает иконку. Иконка – графическое представление прибора. Она может содержать текст, рисунок или и то и другое одновременно. Если ВП используется в качестве подпрограммы, то иконка идентифицирует его на блок-диаграмме другого ВП.

Установленная по умолчанию иконка ВП содержит номер, который указывает, сколько новых приборов открылись после запуска LabVIEW. Чтобы создать собственную иконку, отличную от заданной по умолчанию, нужно, щелкнув правой кнопкой мыши по иконке в правом верхнем углу лицевой панели или блок-диаграммы. Затем выбрать пункт **Edit Icon** (Редактирование иконки) из контекстного меню. **Icon Editor** (Редактор иконки) можно также вызвать двойным щелчком левой кнопки мыши в верхнем правом углу одной из панелей. Редактирование

иконки доступно также из пункта главного меню **File**, далее **VI Properties** (Свойства ВП), где в диалоговом окне **Category** (Категория) следует выбрать пункт **General** (Общие) и нажать кнопку **Edit Icon** (Редактирование иконки).

Редактирование иконки (Edit Icon)

Редактирование иконки выполняется в области, расположенной в центре окна **Icon Editor** (Редактора иконки), при помощи инструментов, расположенных слева от области редактирования (см. рис. 13.1). Вид иконки доступный на блок-диаграмме и в правом верхнем углу обеих панелей распложена справа от области редактирования, в соответствующем поле.

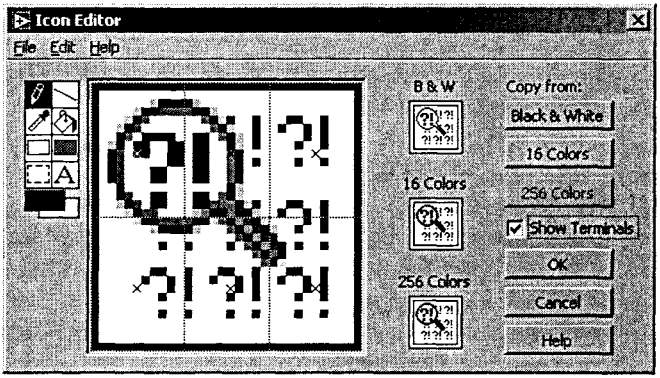


Рис. 13.1

Иконка может быть создана для черно-белого, 16-цветного или 256-цветного режима. Для печати, в случае отсутствия цветного принтера, LabVIEW использует черно-белую иконку. По умолчанию установлен 256-цветный режим.

Меню **Edit** (редактирование) используется для вырезания, копирования и вставки картинок из иконки или в нее. При выборе фрагмента иконки для вставки картинки, размер картинки изменяется для соответствия размеру выбранной области.

Для копирования цветной иконки в черно-белую (или наоборот) достаточно выбрать опцию **Copy from**, находящуюся в правой части диалогового окна **Icon Editor**. Нажать кнопку **OK** для окончательной замены.

В случае если сплошная граница вокруг иконки не нарисована, фон иконки будет прозрачным. При выборе иконки на блок-диаграмме маркеры выбора появляются вокруг каждого графического элемента иконки.

Набор инструментов для редактирования иконки расположен в левой части окна **Icon Editor** и выполняет следующие функции:

- инструмент ***карандаш*** позволяет рисовать или стирать по одной точке;


инструмент ***линия*** позволяет рисовать прямые линии. Для рисования вертикальных, горизонтальных и диагональных линий необходимо во время рисования нажать и удерживать клавишу **Shift**;


инструмент ***копирование цвета*** предназначен для копирования цвета символа в поле редактирования иконки;


инструмент ***заполнение цветом*** предназначен для заполнения ограниченной области заданным цветом переднего плана;


инструмент ***прямоугольник*** выводит в область редактирования прямоугольную границу заданным цветом переднего плана.
Двойной щелчок левой кнопкой мыши на прямоугольнике обводит иконку рамкой заданным цветом переднего плана.


Инструмент, ***заполненный цветом фона прямоугольник***, выводит в область редактирования прямоугольную границу заданным цветом переднего плана, заполненную цветом фона. Двойной щелчок левой кнопкой мыши на ***заполненном цветом фона прямоугольнике*** обводит иконку рамкой цвета символа и заполняет цветом фона;


инструмент ***выбор*** предназначен для выделения фрагмента иконки, что позволяет вырезать, копировать, перемещать или вносить другие изменения в выделенный фрагмент. Чтобы очистить область редактирования иконки достаточно дважды щелкнуть левой кнопкой мыши на инструменте ***выбор*** и нажать кнопку **Delete**:


инструмент ***ввод текста*** позволяет вводить текст в область редактирования иконки. Выбор шрифта производится двойным щелчком левой кнопкой мыши на инструменте ***ввод текста***;


инструмент ***передний план***, фон изображен в виде двух прямоугольников цветами фона и переднего плана (символа). При нажатии на каждый прямоугольник появляется палитра выбора цвета.

Опции в правой части **Icon Editor** предназначены для выполнения следующих задач:

- **Show Terminals** – показывает в области редактирования поля ввода/вывода данных.
- **OK** – сохраняет внесенные в иконку изменения
- **Cancel** – закрывает **Icon Editor** без сохранения изменений.

Настройка соединительной панели (Connector).

Для использования ВП в качестве подпрограммы ВП необходимо настроить соединительную панель (**Connector**).

Соединительная панель является совокупностью полей ввода/вывода данных, соответствующих элементам управления и индикации ВП. Соединительная панель определяет поля входных и выходных данных ВП. Таким образом, ВП можно использовать в качестве подпрограммы.

Каждому полю ввода или вывода данных назначается свой элемент лицевой панели. Для редактирования соединительной панели необходимо щелкнуть правой кнопкой мыши на иконке ВП и выбрать из контекстного меню пункт **Show**

Connector (Показать поля ввода/вывода данных). После этого вместо иконки появится соединительная панель, в которой каждый прямоугольник соответствует полю ввода или вывода данных. Количество полей ввода/вывода данных соответствует количеству элементов на лицевой панели. На рис.13.2 показана лицевая панель, содержащая два элемента управления и два элемента индикации. Таким образом, в соединительной панели LabVIEW показывает два поля ввода и два поля вывода данных.

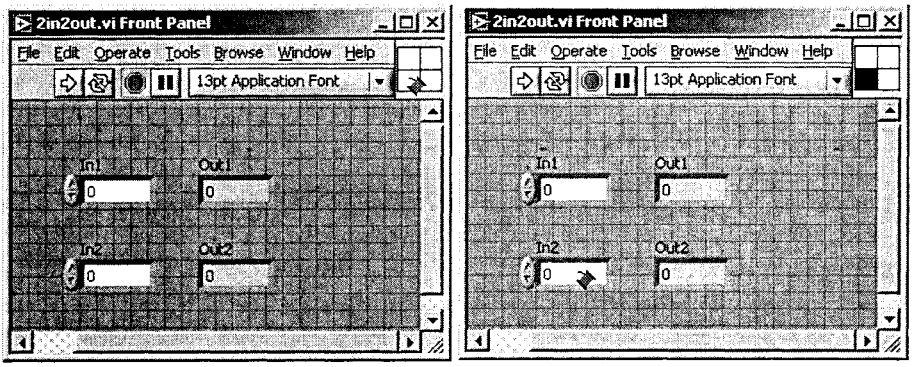


Рис. 13.2

Имеется возможность выбрать вид соединительной панели из шаблонов. Это осуществляется щелчком правой кнопки мыши на соединительной панели и выбором пункта **Patterns** (Шаблон) из контекстного меню. В шаблоне некоторые из полей ввода/вывода данных можно оставить без соединения и задействовать позднее при необходимости. Такая гибкость дает возможность вносить изменения с минимальным отражением на иерархии ВП. Причем не все элементы лицевой панели должны быть обязательно задействованы в соединительной панели.

Задействованные поля выделяются цветом, соответствующим типу данных элемента. Максимально возможное количество полей ввода/вывода данных ограничено 28, но следует избегать необходимости использования более 16 полей ввода/вывода данных. Наличие более 16 полей снижает удобство чтения программы.

Предусмотрена возможность изменять пространственное положение полей ввода-вывода соединительной панели с помощью соответствующего пункта контекстного меню: **Flip Horizontal** (отражение по горизонтали), **Flip Vertical** (по вертикали) или **Rotate 90 Degrees** (поворот на 90°).

Привязка полей ввода/вывода данных к элементам лицевой панели

После выбора шаблона соединительной панели необходимо каждому полю назначить свой элемент лицевой панели. Для упрощения использования ВПП следует

поля ввода данных размещать слева, а поля, связанные с элементами индикации, – справа на соединительной панели.

Чтобы назначить поля ввода или вывода данных, следует щелкнуть по выбранному полю левой кнопкой мыши, затем щелкнуть мышью на элементе, который необходимо связать с этим полем, после этого вывести курсор в свободное пространство лицевой панели и снова щелкнуть мышью. Задействованные поля примут цвет, определенный типом данных соответствующего элемента. Во время назначения полей ввода/вывода данных используется инструмент ***соединение***.

Можно также сначала щелкнуть левой кнопкой мыши по элементу, а потом по полю ввод/вывода данных.

Использование подпрограмм ВП

После создания ВП, оформления его иконки и настройки соединительной панели ВП может использоваться в качестве подпрограммы. Чтобы поместить подпрограмму ВП на блок-диаграмму, следует выбрать на палитре **Functions** (Функций) подраздел **Select a VI** (Выбор ВП), указать ВП и перенести его на блок-диаграмму.

Открытый ВП можно поместить на блок-диаграмму другого ВП, переместив на нее иконку этого ВП с помощью инструмента ***перемещение***.

Редактирование подпрограммы ВП

Вызов лицевой панели подпрограммы ВП из блок-диаграммы другого ВП производится двойным щелчком на нем инструментом ***управление*** или ***перемещение***. Это же можно сделать с помощью главного меню, выбрав в пункте **Browse** (Обзор) подпункт **This VI's SubVIs** (Подпрограммы этого ВП). Для вызова блок-диаграммы ВПП следует, удерживая клавишу **Ctrl**, дважды щелкнуть на нем левой кнопкой мыши.

Изменения, внесенные в подпрограмму ВП, доступны вызывающим его программам только после предварительного их сохранения.

Установка значимости полей ввода/вывода данных: обязательные, рекомендуемые и дополнительные (не обязательные)

При создании ВПП необходимо указать обязательные для соединения поля (также рекомендуемые и дополнительные) с целью предупреждения пользователя от ошибки.

Для указания значимости полей следует щелкнуть правой кнопкой мыши по соединительной панели, выбрать в контекстном меню пункт **This Connection Is** (Это поле...), установить метку на требуемую позицию: **Required** (Обязательное), **Recommended** (Рекомендуется) или **Optional** (Дополнительное).

Если поле ввода или вывода данных обязательно для соединения, то ВП не будет выполняться до тех пор, пока поле не будет правильно инициализировано. Если поле, рекомендованное для соединения, не задействовано, то ВП будет работать, но LabVIEW выдаст предупреждение в окне **Error List** (Список ошибок), при условии что в диалоговом окне **Error List** (Список ошибок) стоит метка в поле **Show Warnings** (Выдать предупреждение). LabVIEW не сообщает о незадействованных и не обязательных для соединения полях

По умолчанию LabVIEW устанавливает значимость созданного поля в позицию Recommended (Рекомендуется).

В окне контекстной справки **Context Help**, которое доступно из пункта главного меню **Help** ⇒ **Show Context Help**, обязательные для соединения поля обозначены жирным шрифтом, рекомендуемые – нормальным, а дополнительные (не обязательные) – светло-серым шрифтом при условии, что используется режим подробного просмотра **Detailed**. В **Simple** (Кратком) просмотре окна контекстной справки **Context Help** эта информация недоступна.

Создание ВПП из секции блок-диаграммы

Можно упростить блок-диаграмму ВП, создав из часто выполняемых операций подпрограмму ВП. Для этого с помощью инструмента ***перемещение*** необходимо выделить интересующую секцию блок-диаграммы и выбрать из пункта главного меню **Edit** (Редактирование) пункт **Create SubVI** (Создать ВПП). Выделенная секция сменится иконкой новой подпрограммы ВП. LabVIEW создаст элементы управления и индикации данных для нового ВПП и соединит поля ввода/вывода данных с существующими проводниками, как показано на рис. 13.3.

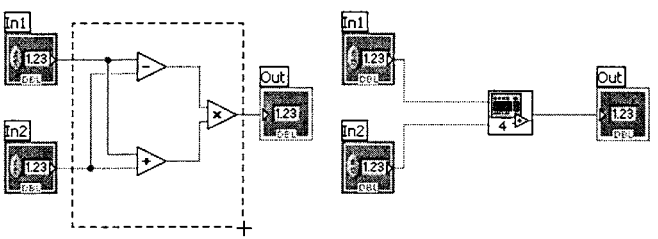


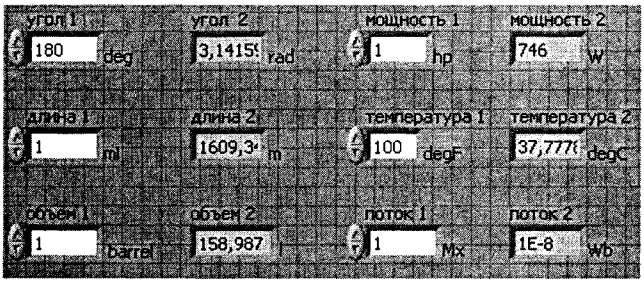
Рис. 13.3

Дважды щелкните правой кнопкой мыши по иконке подпрограммы ВП для редактирования соединительной панели и иконки и для сохранения ВПП.

Использование единиц измерения

Использование единиц измерения упрощает блок-диаграмму ВП, позволяет сделать работу с ВП более удобной. Пример на рис. 13.4 демонстрирует применение единиц измерения:

а)



б)

Рис. 13.4

Здесь для перевода величин из одних единиц измерения в другие используются метки размерности, т.е. мы не производим явных вычислений на блок-диаграмме.

Для того, чтобы элементам управления или индикации добавить единицы, надо сначала в контекстном меню выбрать команду **Visible Items ⇒ Unit label** (см. рис. 13.5) после этого станет видно поле размерности, в которое следует вписать название единиц измерения.

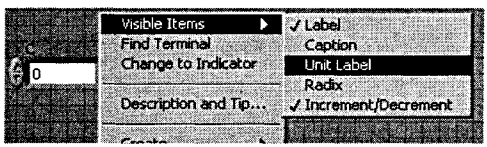
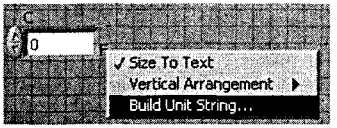


Рис. 13.5

Имеется так же возможность построения единиц измерения, для работы с ним нужно вызвать контекстное меню непосредственно поля размерности далее выбрать **Build Unit String**.

В поле **Units** диалога **Build Unit String** (см. рис. 13.6) представлен полный список физических единиц (если установлен флажок, то он группируется по соответствующим разделам). В левой части есть список кратных величин и их значений. Имеется возможность вводить степень для каждой из единиц.



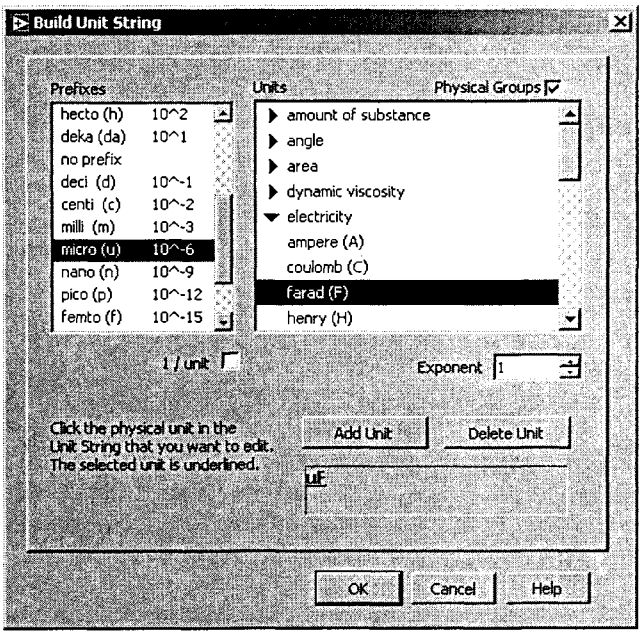


Рис. 13.6

Пример 13.1. Использование размерностей

Данный пример иллюстрирует расчет реактивного сопротивления последовательно соединенных емкостного и индуктивного элементов (рис 13.7).

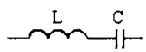


Рис. 13.7

Используется возможность LabVIEW ввода величин с единицами измерения. Исходными данными является частота [Гц], емкость [Ф] и индуктивность [Гн]. Угловая частота ω [$с^{-1}$] вычисляется как $\omega = 2\pi f$, индуктивное сопротивление [Ом] $X_L = \omega L$, емкостное [Ом] $X_C = 1/\omega C$, реактивное сопротивление последовательно соединенных индуктивности и емкости $X = X_L - X_C$. Блок диаграмма и лицевая панель представлены на рис. 13.8.

Несложно модифицировать программу, что бы можно было вводить величины в других единицах, например частоту в килогерцах, а емкость в микрофарадах. При этом не потребуется изменять блок-диаграмму, а достаточно изменить только единицы измерения. Например, как это показано на рис. 13.9:

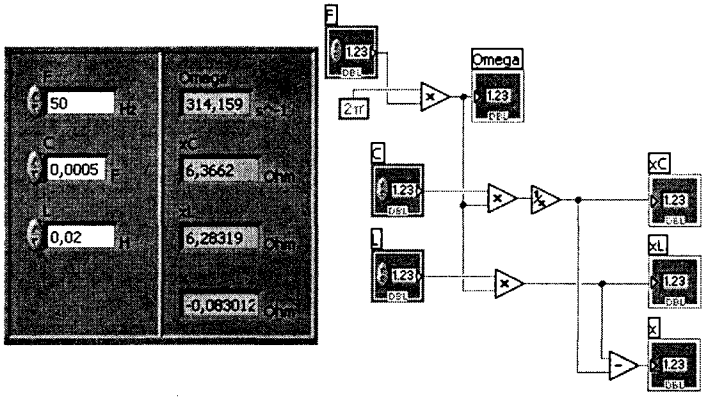


Рис. 13.8

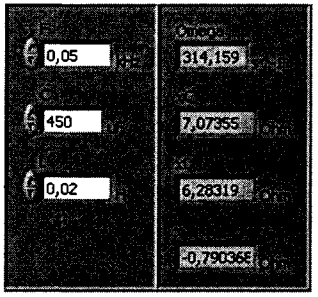


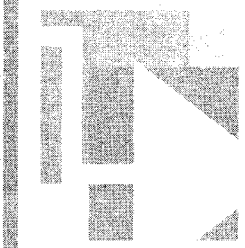
Рис. 13.9

Выводы

При разработке собственных программ особенно целесообразным оказывается использование ВПП. Это улучшает наглядность программ и облегчает их создание и модифицирование. Важной представляется возможность создания собственной иконки и задания важности терминала (обязательный для подсоединения проводника данных, рекомендуемый или необязательный). При работе с физическими величинами удобно использовать поля размерности.

Лекция 14

Строки



Описывается работа со строками. Разбираются различные приемы обработки текста. Завершается глава рассмотрением представления данных в табличной форме.

Строки представляют собой некоторый набор символов. Этот набор может быть привычным для чтения текстом. А может быть и специальным кодом. В первом случае строка на экране отображается в том же виде, в котором она записана. Во втором случае на экране отображается то, что было закодировано (например, знак табуляции или знак абзаца). Строки используются при решении следующих наиболее распространенных задач:

- Вывод на экран или принтер текстовых сообщений.
- Преобразование различных типов данных в строки и наоборот.
- Сохранение различных типов данных в файл. Обычно требуется сохранить числовые данные. Перед записью в файл числовые данные необходимо преобразовать в строки.
- Выбор входных данных функций, которые не могут быть представлены каким-либо типом данных и не имеют своего. Например IP адрес. В некоторых случаях путь к файлам и папкам также представляется строками.
- Использование сообщений в диалоговых окнах.

На лицевой панели строки появляются в виде таблиц, полей ввода текста и меток.

Создание строковых элементов управления и индикации

Для работы с текстом и метками используются строковые элементы управления и индикации, расположенные в палитре **Controls ⇒ String & Path**. Создание и редактирование текста в строке производится с помощью инструментов ***управление*** и ***ввод текста***. Для изменения размера строкового объекта на лицевой панели используется инструмент ***перемещение***. Для экономии места на лицевой панели можно использовать полосу прокрутки. Для этого необходимо щелкнуть правой

кнопкой мыши по строковому объекту и выбрать в контекстном меню пункт **Visible Items ⇒ Scrollbar**.

Тип отображения строкового объекта выбирается в его контекстном меню. Типы отображения строки и примеры заполнения поля ввода текста показаны в табл. 14.1.

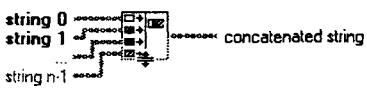
Таблица 14.1

Тип отображения	*Описание*	*Пример*
*Режим стандартного отображения (**Normal Display**)*	*Отображает стандартные ASCII коды, используя шрифт элемента управления. Управляющие коды для печати выводятся на экран в виде квадратов.*	Строка
*Режим отображения с обратным слэшем непечатаемых управляющих кодов ('\' **Codes Display**)*	*Выводит код для всех непечатаемых управляющих символов и русских букв.*	\D1\F2\F0\EE\EA\
*Режим скрытого отображения текста (**Password Display**)*	*Выводит * для всех кодов текстового пространства*	****
*Режим отображения 16-тиричных ASCII кодов (**Hex Display**)*	*Выводит значение 16-тиричные коды для каждого символа.*	D7F3 ECE8

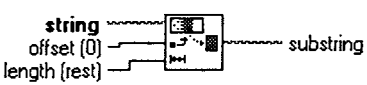
Функции работы со строками

Для редактирования строк и управления ими на блок-диаграмме следует пользоваться функциями обработки строк, расположенными в палитре **Functions ⇒ String**. Рассмотрим основные функции работы со строками.

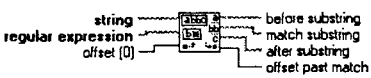
String Length – выдает количество символов в строке, включая пробелы. Пример показан на рис. 14.1.



Concatenate Strings – объединяет строки и одномерные массивы строк в отдельную строку. Для увеличения полей ввода данных функции следует изменить размер иконки. Пример использования функции представлен на рис. 14.2.



String Subset – выдает подстроку определенной длины **length**, начиная со значения **offset** (смещение). Смещение первого элемента в строке равно 0. Пример на рис. 14.3.



Match Pattern – ищет в строке повторяющуюся последовательность, поданную на вход **regular expression**, и, если находит соответствие, разбивает строку на три подстроки. Начало поиска определяется смещением **offset**. Если соответствие не найдено, поле вывода данных **match substring** является пустым, а значение поля вывода дан-

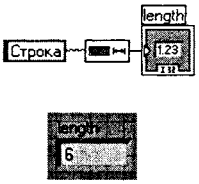


Рис. 14.1

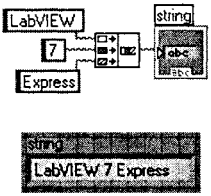


Рис. 14.2

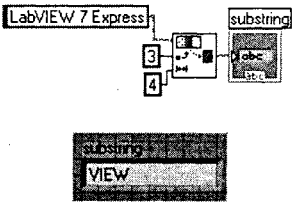


Рис. 14.3

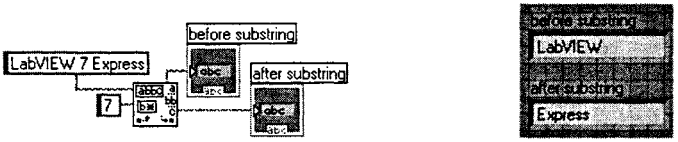


Рис. 14.4

ных **offset past** match (смещение повторяющейся последовательности в строке) равно -1. Пример на рис. 14.4.

Преобразование числа в строку. Функция Format Into String

Функция **Format Into String** преобразует данные различных типов в строку.

Например, если необходимо получить в виде одной строки слово «Напряжение» и численное значение следует собрать ВП, как это показано на рис. 14.5.

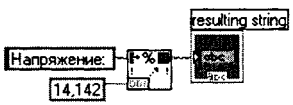


Рис. 14.5

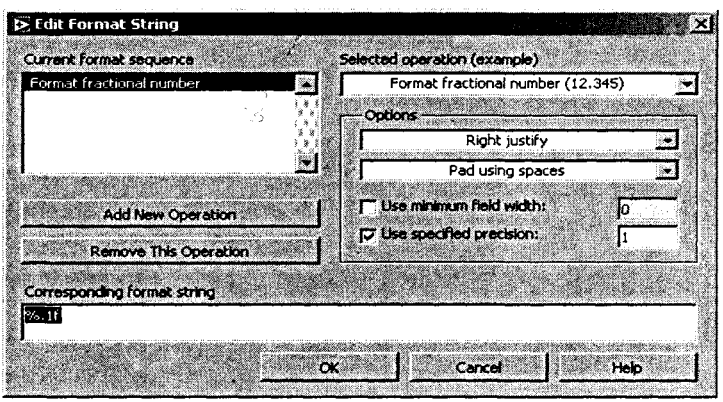


Рис. 14.6

Далее, для определения формата преобразуемого числа, в контекстном меню можно выбрать пункт **Edit Format String** (или просто произвести двойной клик мышкой на иконке функции). По умолчанию формат числа дробный.

Рис. 14.7

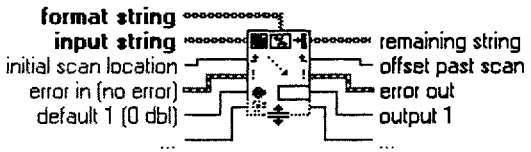
В появившемся диалоге (см. рис. 14.6) можно изменить формат отображения для каждого входного параметра: представление числа, выравнивание, точность. В данном примере выберем количество знаков после запятой (**use specified precision**). После выхода из диалога на блок диаграмме появиться строка-шаблон для вывода (см. рис. 14.7). В данном случае %.1f означает вывод числа в дробном виде с 1 знаком после запятой.

Для получения более подробной информации о синтаксисе форматов, следует обратиться к встроенной в LabVIEW справочной информации (**LabVIEW Help**).

Имеется возможность увеличить количество параметров, изменив размеры иконки.

Преобразование строк в числовые данные

Для преобразования строки в числовые данные следует использовать функцию **Scan From String**.



Функция **Scan From String** преобразует строку, содержащую допустимые числовые символы, такие как «0»-«9», «+», «-», «e», «E» и разделитель «.», в данные числового формата. Функция начинает просмотр строки, подаваемой на поле ввода данных **input string** с номера символа, задаваемого на поле **initial search location**. Основываясь на формате строки, функция может просматривать входящую строку и искать различные типы данных, таких как числовые или логические данные. Для увеличения количества полей вывода данных следует изменить размер иконки функции.

Например, при значениях на полях ввода данных «Напряжение: 14,1» функция выдает результат 14.1, как показано на рис. 14.8.

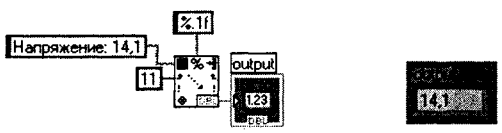


Рис. 14.8

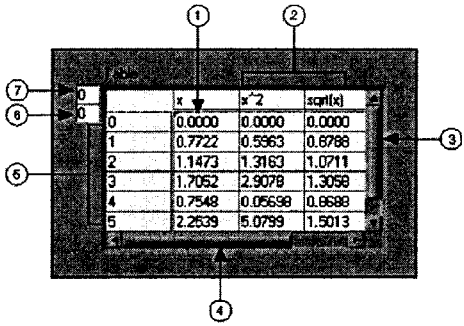


Рис. 14.9

1. *Ячейка таблицы*
2. *Заголовок строки*
3. *Заголовок столбца*
4. *Индекс по горизонтали*
5. *Вертикальная полоса прокрутки*
6. *Индекс по вертикали*
7. *Горизонтальная полоса прокрутки*

В формате строки **%** – указывает начало формата строки, ***f*** – указывает тип данных с плавающей запятой. Для создания и редактирования формата строки следует в контекстном меню выбрать пункт **Edit Scan String**.

Таблицы

Элемент управления Таблица, расположенный в палитре **Controls ⇒ List & Table** предназначен для создания таблиц на лицевой панели. Каждая ячейка находится в строке и столбце таблицы. Поэтому таблица представляет собой двумерный массив строк. На рис. 14.9 показана таблица и ее составные части.

Для инициализации значений ячеек таблицы используется инструмент ***управление*** или ***ввод текста***, с помощью которых достаточно ввести текст в выделенную ячейку.

Таблица представляет собой двумерный массив строк. Для ее использования в качестве элемента индикации, необходимо двумерный массив чисел преобразовать в двумерный массив строк.

Задание 14.1. Сортировка таблицы

Обработайте таблицу в ВП таким образом, чтобы имелась возможность сортировать различные столбцы таблицы по возрастанию. Саму сортировку осуществите с помощью функции **Sort 1D Array**, которая находится в палитре **Functions ⇒ Array**. Напомним, что эта функция сортирует элементы одномерного массива в порядке возрастания. Если массив состоит из кластеров, функция сортирует массив по первым элементам кластеров. В связи с этим таблицу следует преобразовать в массив таким образом, чтобы интересующий пользователя столбец стал первым. И потом

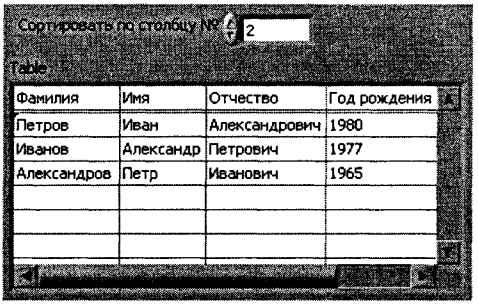


Рис. 14.10

сформировать из полученного массива кластер, который мы и будем сортировать. После осуществления сортировки кластер следует преобразовать массив и вернуть столбцы на свои первоначальные места.

Создайте саму таблицу и целочисленный элемент управления, который будет указывать столбец для сортировки. Заполните таблицу какими-либо данными. Можете позаимствовать данные на рис. 14.10.

По умолчанию заголовки столбцов и строк не видны. Чтобы заголовки столбцов стали видны в свойствах таблицы (диалоговое окно свойств вызывается из контекстного меню пунктом **Properties**) поставьте флажок напротив **Show Column Headers**. Кроме этого, чтобы появилась возможность редактировать заголовки, включите в контекстном меню таблицы опцию **Editable Headers**. Сейчас вам ничего не мешает сделать таблицу такой же, как это показано на рис. 14.10.

Как осуществить сдвиг столбца, по которому будет проводиться сортировка? Сдвиг столбца осуществите с помощью **Rotate 1D Array** из той же палитры **Functions ⇒ Array**. Одномерный массив, который подается на вход функции **Rotate 1D Array**, представляет собой строки нашей таблицы. Каждую строку получите с помощью цикла **For**. Если для каждой строки в отдельности поменять элементы местами, то в целом в таблице местами поменяются столбцы. Допустим, как и на рис. 14.10, необходимо сортировать таблицу по третьему столбцу (отсчет начинается с нуля). Иначе говоря, столбец с отчествами надо поместить на первое место. Функция **Rotate 1D Array** перемещает **n** элементов из начала в конец, если $n > 0$. И наоборот перемещает n элементов из конца в начало, если $n < 0$. Если на вход **n** подан 0, функция строку не изменяет. Столбцы таблицы надо сдвинуть на 2 столбца влево до сортировки, и на 2 столбца вправо после. Поэтому до использования функции сортировки необходимо подать -2 на вход n функции **Rotate 1D Array**, а после +2.

Преобразование массива в кластер и кластер в массив осуществляют функции **Array To Cluster** и **Cluster To Array** соответственно. При преобразовании кластера в массив никаких дополнительных операций проводить не требуется. А при преобразовании массива в кластер следует указать размер кластера. Предполагается, что структура таблицы известна заранее. Делается это через пункт **cluster size** контекстного меню функции. В нашем случае размер кластера равен 4.

Итак, вы разбили таблицу на отдельные строки. В каждой строке в цикле **For** поменяли местами столбцы. Сформировали из элементов строки кластер. На

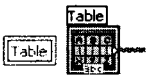


Рис. 14.11

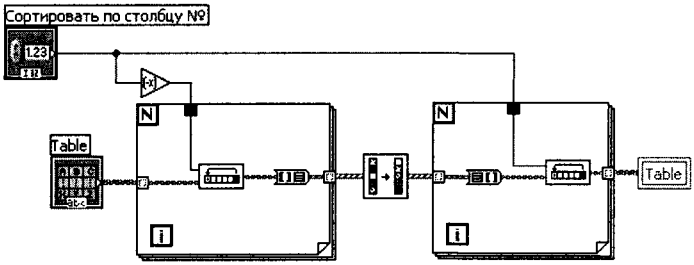


Рис. 14.12

выходе из цикла **For** собрали из кластеров массив. Отсортировали массив кластеров по первому элементу в кластере. Опять использовали цикл **For**, в котором из каждого элемента массива кластеров получили одномерный массив. Поменяли местами столбцы обратно. На выходе цикла **For** собрали отсортированные строки в таблицу. Остается вывести этот результат в ту же таблицу. Вариант с новой таблицей на лицевой панели в виде элемента индикации, безусловно, возможен. Однако такой способ отображения информации при работе с таблицами по вполне понятным причинам не является общепринятым. Необходимо, чтобы отсортированная таблица отображалась в исходной таблице. В таких случаях удобно использовать локальные переменные. Локальную переменную для любого элемента управления или индикации можно создать из контекстного меню элемента, выбрав пункт **Create ⇒ Local Variable**. Этот пункт контекстного меню доступен как на лицевой панели, так и на блок-диаграмме. После того, как вы создали локальную переменную, ее терминал появится рядом с терминалом элемента управления или индикации на блок-диаграмме (рис. 14.11).

Локальную переменную можно использовать в любом месте блок-диаграммы (если это не противоречит логике выполнения программы). Ее можно использовать как для чтения данных, так и для записи. В качестве последнего используйте локальную переменную, соединив ее с полученной отсортированной таблицей. ВП готов. Окончательная блок-диаграмма показана на рис. 14.12. Перейдите на лицевую панель и испытайте программу.

Подумайте, как сделать так, чтобы пользователь выбирал не номер столбца, а его название. Добавьте возможность сортировать столбцы не только по возрастанию, но и по убыванию. При этом управление должно осуществляться с лицевой панели ВП.

Выводы

Применение строк и функций работы с ними позволяет, во-первых, сделать интерфейс программы более дружественным, а, во-вторых, иметь возможность работать с текстовой информацией.

Описываются возможности LabVIEW по сохранению данных в файл, а также считыванию их из файла.

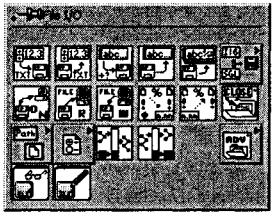
Лекция 15

Функции работы с файлами

Описывается возможность LabVIEW по сохранению данных в файл, а также считыванию их из файла.

Функции файлового ввода/вывода производят операции с файлами записи и считывания данных. Функции файлового ввода/вывода расположены в палитре **Functions ⇒ File I/O** и предназначены для:

- Открытия и закрытия файла.
- Считывания и записи из файла и записи данных в файл.
- Считывания и записи данных в виде таблицы символов.
- Перемещения и переименования файлов и каталогов.
- Изменения характеристик файла.
- Создания, изменения и считывания файлов конфигурации.



функции высокого уровня

функции низкого

Рис. 15.1

Палитра функций файлового ввода вывода, показанная на рис. 15.1, разделена на четыре части: функции высокого уровня **(high level File I/O)**, функции низкого уровня **(low level File I/O)**, подпалитра функций расширенных возможностей **(advanced File I/O)** и экспресс функции.

Основы файлового ввода/вывода

Стандартные операции ввода/вывода данных в/из файла состоят из следующей последовательности действий:

1. Создание или открытие файла. Указание месторасположения существующего файла или пути для создания нового файла с помощью диалогового окна LabVIEW. После открытия файла LabVIEW создает ссылку (**refnum**) на него.
2. Произведение операций считывания или записи данных.
3. Закрытие файла.
4. Обработка ошибок.

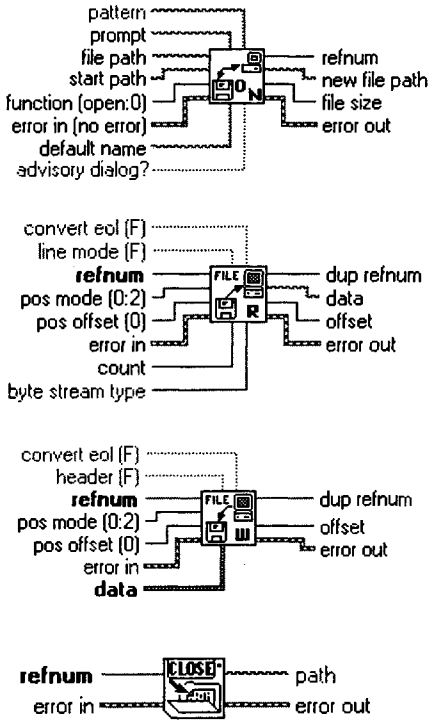
Функции файлового ввода/вывода низкого уровня

Функции файлового ввода/вывода низкого уровня расположены в средней строке палитры **Functions ⇒ File I/O**. Дополнительные функции работы с файлами (**Advanced File I/O**) расположены в палитре **Functions ⇒ File I/O ⇒ Advanced File Functions** и предназначены для управления отдельными операциями над файлами.

Функции файлового ввода/вывода низкого уровня используются для создания нового или обращения к ранее созданному файлу, записи и считывания данных и закрытия файла. Функции низкого уровня работы с файлами поддерживают все операции, необходимые при работе с файлами.

Для осуществления основных операций файлового ввода/вывода используются следующие ВП и функции:

Таблица 15.1



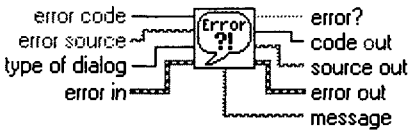
Open/Create/Replace File – открывает, перезаписывает существующий файл, или создает новый. Если **file path** (путь размещения файла) не указан, ВП выводит на экран диалоговое окно, в котором можно создать новый или выбрать уже существующий файл.

Read File – считывает данные из файла, определяемого по ссылке refnum, и выдает данные на поле вывода **data**, на поле **count** подается число считываемых данных. Считывание данных начинается с места, определяемого элементами **pos mode** и **pos offset**, и зависит от формата файла.

Write File – записывает данные в файл, определяемый по ссылке refnum. Запись начинается с места, определяемого полями ввода данных **pos mode** и **pos offset** для файла потока байтовых данных, и указателем конца файла для файла протоколированных данных.

Close File – закрывает указанный в ссылке **refnum** файл.

Файловые функции низкого уровня предают по цепочке информацию об ошибках. Для их обработки используются подпрограммы обработки ошибок, например:



Simple Error Handler VI (ВП Простой обработчик ошибок), расположенный в палитре **Functions ⇒ Time & Dialog**. Поля ввода **error in** и вывода error out информации об ошибках используются в каждом ВП для обмена информацией об ошибках между ВП.

Во время работы ВП проверяется наличие ошибок в каждом узле. Если ошибок нет, то ВП выполняется в обычном режиме. Если ошибка имеет место в одном ВП, то его выполнение прерывается, а информация об ошибке передается следующему ВП. Следующий ВП передает ошибку дальше. При этом сам ВП не выполняется. В конце выполнения всей цепочки ВП LabVIEW сообщает об ошибках.

Сохранение данных в новом или уже существующем файле

В файл, созданный (или открытый) с помощью функций файлового ввода/вывода, можно записать данные любого типа. При необходимости доступа к файлу со стороны других приложений или пользователей, следует записывать данные в виде строки ASCII символов

Доступ к файлу можно осуществить программным путем или с использованием диалогового окна. Для доступа к файлу с помощью диалогового окна на поле ввода **file path** подпрограммы ВП **Open/Create/Replace File VI** не следует подавать данные.

Пример 15.1. Запись строки в файл

На рис. 15.2 показано, как записать строку данных в файл при программном указании пути и имени файла. Если файл уже существует, то он перезаписывается, если нет – то создается новый файл.

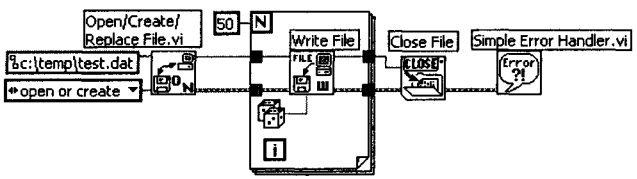


Рис. 15.2

Подпрограмма ВП **Open/Create/Replace File VI** открывает файл test1.dat, создает ссылку на файл и кластер ошибок.

Ссылка (**refnum**) является уникальным идентификатором для таких объектов как файл, прибор, сетевое соединение и т.п. При открытии файла, устройства или сетевого соединения LabVIEW создает ссылку на объект. Все операции с открытыми объектами выполняются с использованием ссылок.

Кластер ошибок и ссылка на файл последовательно передаются от узла к узлу. Поскольку узел не может выполниться, пока не определены все его входные поля данных, эти два параметра заставляют узлы работать в определенном порядке. Подпрограмма ВП **Open/Create/Replace File VI** передает ссылку на файл и кластер ошибок функции **Write File**, которая производит запись файла на диск. Функция **Close File** закрывает файл после получения кластера ошибок и ссылки на файл из функции **Write File**.

Подпрограмма ВП **Simple Error Handler VI** проверяет наличие ошибок и выводит информацию о них в диалоговом окне. Если в одном из узлов допущена ошибка, последующие узлы не выполняются, и кластер ошибок передается в подпрограмму ВП **Simple Error Handler VI**.

Форматирование строк таблицы символов

Для того чтобы записать данные в файл формата электронной таблицы, необходимо переформатировать строковые данные в строку таблицы, содержащую разделители, такие как символ табуляции. Во многих приложениях символ табуляции разделяет столбцы, а символ **end of line** (конец строки) разделяет строки. Для обеспечения совместимости между различными платформами следует использовать константу **end of line constant**, расположенную в палитре **Functions ⇒ String**. Константа осуществляет перевод строки.

Функция **Format Into File** предназначена для форматирования строк, путей к файлам, числовых и логических данных в текст, а также для записи текста в файл. Часто эта функция используется вместо двух операций – форматирования строки с помощью функции **Format Into String** или ВП **Build Text Express VI** и записи результата с помощью функций **Write Characters To File** или **Write File**.

Функция **Format Into File** предназначена для определения порядка, в котором данные записываются в тестовый файл. Однако ее нельзя применять для добавления данных в файл или перезаписи существующего файла. Для этих операций используется функция **Format Into String** совместно с функцией **Write File**. Путь к файлу или ссылку на него можно подать на поле **input file** или оставить это поле без соединения, чтобы указать имя файла в диалоговом окне.

Пример 15.2. Создание файла с таблицей

На рис 15.3 представлена блок-диаграмма, на которой подпрограмма ВП **Open/Create/Replace File VI** открывает файл. Цикл **For** выполняется пять раз. Функция **Format Into String** преобразует значения счетчика итераций и случайное число в строку. Также указываются символы **Tab constant** (табуляции) и **End of Line Constant** (конца строки) для создания двух столбцов и одной строки таблицы сим-

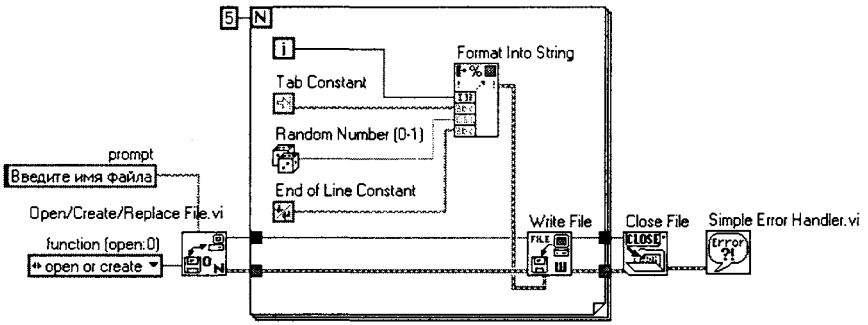


Рис. 15.3

волов. По окончании пяти итераций цикла файл закрывается и ВП проверяет наличие ошибок.

Этот ВП создает следующий текстовый файл, в котором стрелка (→) указывает символ табуляции, а символ ¶ указывает конец строки:

0→ 0,095978 ¶
1→ 0,322095 ¶
2→ 0,499802 ¶
3→ 0,819302 ¶
4→ 0,736656 ¶

Можно открыть данный текстовый файл в любом редакторе электронных таблиц для отображения на экране таблицы, показанной на рис. 15.4.

	A	B
1	0	0,095978
2	1	0,322095
3	2	0,499802
4	3	0,819302
5	4	0,736656

Рис. 15.4

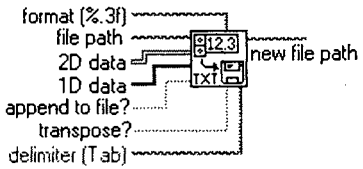
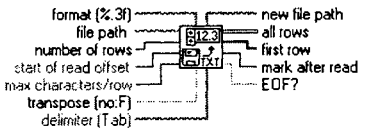
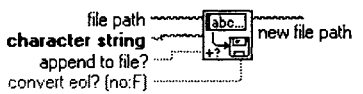
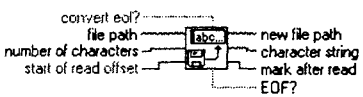
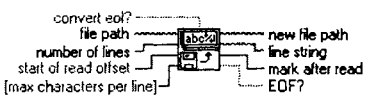
Функции файлового ввода/вывода высокого уровня

Функции файлового ввода/вывода высокого уровня расположены в верхней строке палитры **Functions ⇒ File I/O**. Они предназначены для выполнения основных операций по вводу/выводу данных.

Использование функций файлового ввода/вывода высокого уровня позволяет сократить время и усилия программистов при записи и считывании данных в/из файл(а). Функции файлового ввода/вывода высокого уровня выполняют запись и считывание данных и операции закрытия и открытия файла. При наличии ошибок файловые функции высокого уровня отображают диалоговое окно с описанием ошибок, в котором пользователю предлагается продолжить выполнение программы или остановить ее. Однако из-за того, что функции данного класса объединяют весь процесс работы с файлами в один ВП, переделать их под определенную задачу бывает трудно. Для специфических задач следует использовать функции файлового ввода/вывода низкого уровня.

Функции файлового ввода/вывода высокого уровня включают в себя:

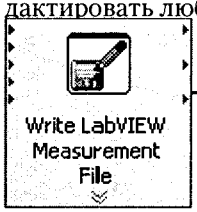
Таблица 15.2

	**Write to Spreadsheet File** – преобразует 2D или 1D массив числовых данных одинарной точности в текстовую строку и записывает строку в новый или добавляет в уже существующий файл. При этом можно также транспонировать данные. ВП открывает или создает файл перед записью и после всех операций закрывает его. Этот ВП используется для создания текстовых файлов, читаемых большинством текстовых редакторов и редакторов электронных таблиц.
	**Read From Spreadsheet File** – считывает определенное число строк от начального смещения **start of read offset** и преобразует данные в 2D массив числовых данных одинарной точности. ВП открывает файл перед чтением и после всех операций закрывает его. Этот ВП можно использовать для чтения таблицы символов, сохраненной в текстовом формате.
	**Write Characters to File** – записывает строку символов в новый файл или добавляет ее в уже существующий. ВП открывает или создает файл перед записью и после всех операций закрывает его.
	**Read Characters From File** – считывает количество символов **number of characters** от начального смещения **start of read offset**. ВП открывает файл перед чтением и после всех операций закрывает его.
	**Read Lines From File** – считывает определенное число строк из текстового или бинарного файла с положения **start of read offset**. ВП открывает файл перед чтением и закрывает его после.

Экспресс ВП

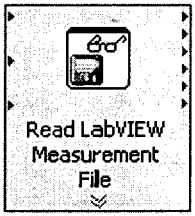
Для наибольшего удобства разработчика в LabVIEW имеются две экспресс функции для работы с файлами: **Write LabVIEW Measurement File** и **Read LabVIEW Measurement File**. Удобство экспресс ВП заключается в том, что они конфигурируются при помощи диалога и требуют минимального присоединения проводников. Файловые экспресс-функции работают со специальным типом файлов **LabVIEW Measurement File**, имеющим расширение .lvm. Данные в таких файлах

представлены в текстовом виде их при необходимости легко просмотреть и отредактировать любым текстовым редактором.



Write LabVIEW Measurement File–Запись в lvm-файл. Входные данные (скалярная величина, массив, осциллограмма, набор осциллограмм) подаются на терминал **Signals**, имеющий динамический тип (см. лекцию 16) В диалоге имеется возможность настроить:

- Сохранять поступающие (от разных запусков ВП) данные в один файл или сохранить серию файлов.
- Имя файла или маску по которой будут создаваться имена файлов.
- Описание файла.



Read LabVIEW Measurement File
Считывание из lvm-файла
На выход **Signals** поступают данные предварительно записанные функцией **Write LabVIEW Measurement File**
В диалоге можно настроить:

- Фиксированное имя файла
- Возможность запроса имени от пользователя.

Так же в диалоге можно произвести тестовое считывание и наглядно данные в виде таблицы

Необходимо заметить, что динамический тип кроме непосредственно данных может содержать и дополнительную информацию: имена осциллограмм, метки времени и т.д., для сохранения считывания которой файловые экспресс ВП наиболее удобны.

Выводы

В LabVIEW предусмотрен широкий спектр файловых функций, позволяющий разработчику выбрать именно те инструменты, которые ему необходимы. Имеются функции низкого уровня работающие с файлами как с последовательностью кластеров произвольного типа. Для работы с текстовыми файлами (например, для передачи табулированных данных в другие пакеты) имеется функции высокого

Лекция 16

Экспресс ВП, создание собственного меню

Рассматриваются дополнительные вопросы программирования в LabVIEW, не вошедшие в предыдущие темы. Представлена новейшая технология сборки блок-диаграммы при помощи легко конфигурируемых экспресс-ВП. Показывается процесс создания и настройки собственного меню.

В двух последующих главах описываются не связанные между собой дополнительные темы, которые не были представлены в других главах. В этой главе речь пойдет о появившейся только в 7-й версии LabVIEW новой разработке National Instruments Экспресс ВП, предназначенные для создания программного обеспечения с функциями измерения и последующего анализа данных. При этом все сложности при написании кода такого программного обеспечения сведены к минимуму. Пользователю надо только выбрать из стандартных заданий необходимое. В конце главы рассказывается, как добавить свои пункты меню в главное меню или в его разделы.

Экспресс ВП

Экспресс ВП представляют собой функции, которые в отличие от обычных функций предназначены для выполнения ряда общих задач. Параметры конкретной задачи настраиваются в отдельном диалоговом окне. В зависимости от выбранных задач у иконки Экспресс ВП на блок-диаграмме появляются необходимые входы и выходы. Иконка Экспресс ВП отличается от обычных функций синем полем, окружающем изображение иконки. Экспресс ВП расположены в палитре **Function** ⇒ **Express**. В табл. 16.1 описаны некоторые Экспресс ВП.

Таблица 16.1

Внешний вид	Название и назначение
	***Display Message to User*** – *Вывести сообщение пользователю.* *Выводит настраиваемое диалоговое окно с одной или двумя кнопками.*
	***Prompt User for Input*** – *Запросить пользовотельский ввод.* *Вызывает настроиваемое диалоговое окно, в котором пользовотель может ввести любое количество переменных численного, логического или строкового типа.*
	***Build Text*** – *Создать текст.* *Создает строку из любого количество входных переменных численного, логического или строкового типа. Существует возможность настроить вид отображения для каждой переменной.*
	***Simulate Signal*** – *Моделировать сигнал.* *Может генерировать сигнал: синусоидальный, прямоугольный или треугольный. Имеется возможность настроить параметры сигнала, добавить шум.*
	***Write LabVIEW Measurement File*** – *Записать данные в статистический файл (.lvm).* *Записывает данные любого типо в фойл. В основном предназначено для сохранения экспериментальных данных результатов измерения.*
	***Read LabVIEW Measurement File*** – *Считать данные из статистического файла (.lvm).* *Считывает данные заронее определенного типа из файло.*
	***Report*** – *Отчет.* *Создает отчет, в виде html-файло или распечотывает но отчет на принтере.*
	***Formula*** – *Формула.* *Предоставляет простой интерфейс для создония зависимостей (в виде формул) от нескольких переменных.*
	***DAQ Assistant*** – *Помощник сборо данных.* *Создает и запускает задания по сбору данных.*

Использование экспресс ВП значительно упрощает набор блок-диаграммы, поскольку каждый экспресс ВП можно настроить под конкретную задачу. При этом в блок-диаграмме экспресс ВП будет показан в виде одной иконки. Иначе говоря, в палитре функций **Express** уже создано множество стандартных подпрограмм, на которые вам не потребуется тратить свое время. При вызове экспресс ВП появляется диалоговое окно, в котором и производятся все необходимые настройки. Покажем на примере, как происходит работа с экспресс ВП.

Пример 16.1. Экспресс-ВП Build Text Express VI

Экспресс-ВП **Build Text**, расположенный в палитре **Functions ⇒ Express ⇒ Output** производит объединение входных строк. Если входные величины имеют не строковый тип данных, то они преобразуются в строку в соответствии с настройками этого экспресс-ВП.

При помещении Экспресс-ВП **Build Text** на блок-диаграмму появляется диалоговое окно настроек **Configure Build Text** (рис. 16.1). В этом примере значение напряжения подается на вход экспресс-ВП и преобразуется к формату данных с плавающей запятой с 2-мя числами после запятой. Затем это значение добавляется к концу строки «Напряжение равно».

При такой настройке экспресс-ВП выглядит так, как показано на рис. 16.2.

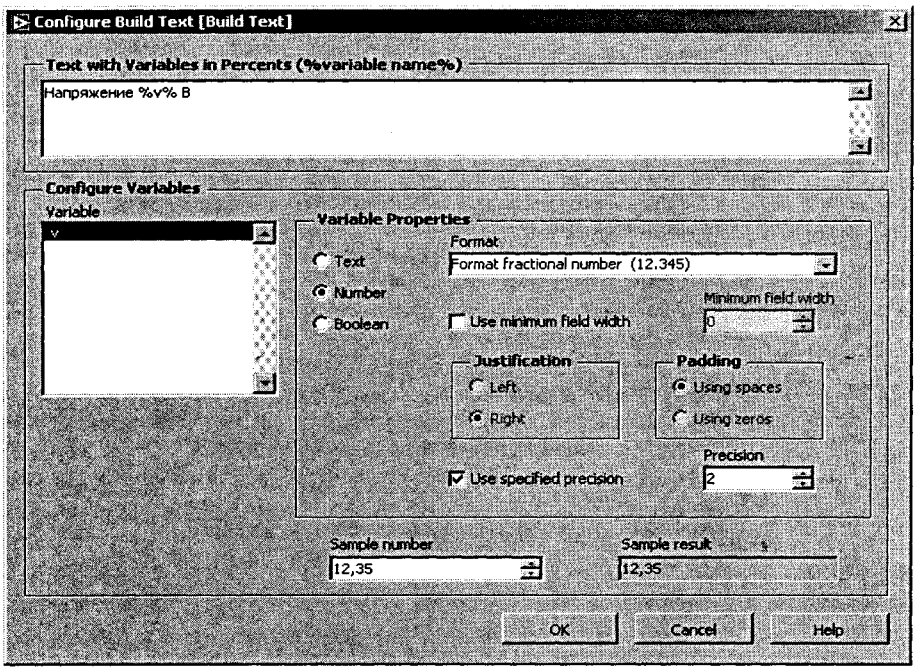


Рис. 16.1

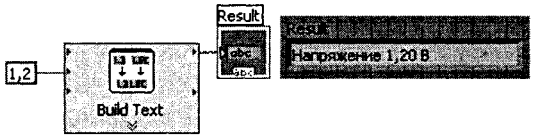


Рис. 16.2

Динамический тип данных (Dynamic Data Type)

Большинство экспресс ВП работают с динамическим типом данных, который может содержать различные данные, например, число, осциллограмму (о типе данных осциллограмма речь пойдет в главе 19) и т.п., а также свойства этих данных. Свойствами могут быть время получения данных, их название. Провода и терминалы динамического типа данных представляются на блок диаграмме темно-синим цветом. Вы можете соединить терминал этого типа с любым типом элемента индикации (включая числовой, булевого и т.п.). LabVIEW автоматически определит, что именно вывести на этот элемент индикации: график, число или логическую переменную. Что такое динамический тип данных можно увидеть из следующих примеров.

Если вы для сбора данных используете экспресс ВП **DAQ Assistant** (его применение подробно рассмотрено в главе 20) и выводите снятые данные на график, то на графике будет показан как сам график, его название (оно указывается отдельно), так и информация о времени, в которое были получены эти данные. При этом шкала X графика автоматически настраивается в соответствии с указанным временем. Если вы для разложения полученного сигнала в ряд Фурье используете экспресс ВП **Spectral Measurements** и выводите полученный результат на график, то указанный результат автоматически отображается в частотной области в соответствии с атрибутами, установленными внутри динамического типа данных.

В отличие от статических типов данных, с которыми работают обычные функции, в контекстном меню поле вывода экспресс ВП с динамическим типом данных можно выбрать соответствующих элемент индикации: **Create ⇒ Graph Indicator** для отображения данных на графике или **Create ⇒ Numeric Indicator** для отображения численного значения.

Обычные функции не принимают динамический тип данных. В случае необходимости динамический тип данных можно преобразовать в статический при помощи соответствующих функций:

 **Convert from Dynamic Data** – преобразование из динамического типа.

 **Converting to Dynamic Data** – преобразование в динамический тип.

Вызов этих функций осуществляется из палитры **Express ⇒ Signal Manipulation**. Эти функции также представляют собой экспресс ВП. Поэтому при их помещении на блок-диаграмму появляется диалоговое окно, в котором и производится настройка формата данных, который вы желаете получить на выходе или подать на вход этих функций.

Преобразование экспресс-ВП в подпрограмму ВП

Предусмотрена возможность создания подпрограммы ВП из настроенного под ваши нужды экспресс-ВП. Для этого достаточно щелкнуть правой кнопкой мыши по экспресс-ВП и выбрать пункт **Open Front Panel** (открыть лицевую панель) в контекстном меню.

1. Для создания подпрограммы ВП из сконфигурированного экспресс-ВП необходимо выполнить следующую последовательность действий:
2. Настроить экспресс-ВП;
3. Щелкнуть правой кнопкой мыши по экспресс-ВП и выбрать пункт **Open Front Panel** (открыть лицевую панель) в контекстном меню;
4. В появившемся диалоговом окне с предупреждением нажать на кнопку **Convert** (преобразовать), после этого появится лицевая панель ВП;
5. Отредактировать ВП;
6. Выбрать пункт **Operate ⇒ Make Current Values Default** или выделять мышью каждый элемент управления и выбирать пункт контекстного меню **Make Current Values Default** для сохранения значений каждого элемента управления;

7. Сохранить ВП, новая подпрограмма ВП заменит экспресс-ВП на блок-диаграмме.

После того, как вы проделаете указанные операции, обратно преобразовать ВПП в экспресс-ВП вы уже не сможете.

Создание собственного меню

Возможность создания своего меню в ВП проиллюстрируем примером. В нем добавим стандартный пункт **«About»** (в программах с русскоязычным интерфейсом этот пункт меню называется «О программе»). С помощью этого пункта во многих программах можно узнать об авторах, версии программы и т.д.

Общее управление пунктами меню производится в редакторе меню **Edit ⇒ Run-Time Menu**. Диалоговое окно редактора показано на рис. 16.3.

В окне слева представлен список пунктов меню, а справа свойства выбранного пункта.

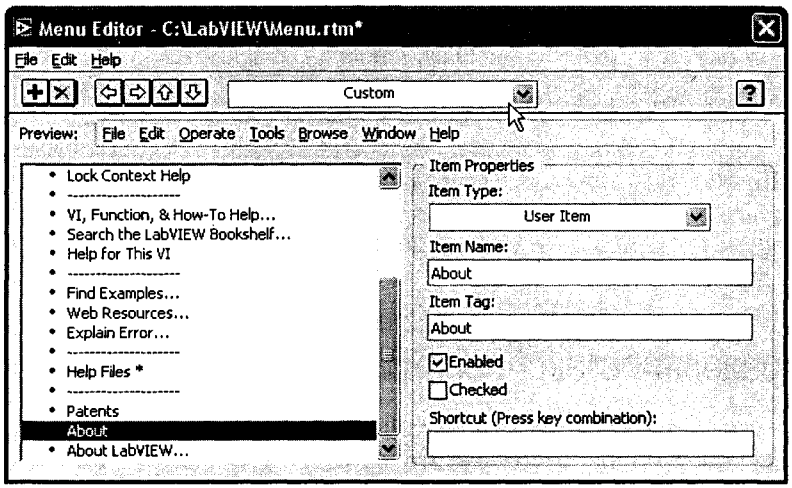


Рис. 16.3

Задание 16.1. Добавление пункта меню «About»

В ниспадающем меню сверху выберите то меню, с которым наш ВП будет работать. Всего три пункта: **Default**, **Minimal**, **Custom**. Первый относится к тому меню, которое установлено по умолчанию. Второй представляет собой минимальный набор общепринятых пунктов меню. Последний вы и будете использовать. Он позволяет создать собственное меню. При выборе этого пользовательского меню рядом находящиеся кнопки управления пунктами меню активизируются. Можно перемещать, добавлять и удалять отдельные пункты. Поскольку в примере предполагается только добавить один пункт. То в пользовательское меню следует добавить все пункты меню, устанавливаемого по умолчанию. Для этого в поле **Item**

Type выберите **Application Item ⇒ File ⇒ Entire Menu**. Этим вы добавили все ниспадающее меню **File**. Проделайте то же самое и для остальных элементов **Application Item**.

Теперь перейдите в меню **Help**. Добавляемый пункт обычно находится в разделе справки. Допустим, новый пункт будет располагаться между элементами **Patents** и **About LabVIEW**. Тогда установите курсор на элементе **Patents**. Нажмите кнопку добавления нового элемента в меню (она самая левая в ряду кнопок управления). Справа в поле **Item Name** введите имя элемента: **About**. В поле **Item Tag** также появится строка **About**. Эта строка представляет собой ссылку на этот элемент меню и ее следует запомнить. Ее надо будет использовать на блок-диаграмме ВП.

В редакторе меню осталось только сохранить пользовательское меню в отдельный файл. Теперь перейдите непосредственно к ВП. Вы можете создать новый ВП или использовать уже существующий проект. Если на блок-диаграмме нет цикла **While**, то создайте его. Предусмотрите кнопку останова выполнения программы.

Поместите на блок-диаграмму цикла **While** структуру события **Event**. В контекстном меню структуры добавьте **Add Event Case**. В поле **Event Sources** следует выбрать **This VI**. В поле **Events – Menu Selection (User)**. Таким образом, вы к имеющемуся варианту структуры событий добавляете еще один вариант, блок-диаграмма которого будет выполняться, когда будет выбран пункт пользовательского меню (в данном случае это добавленный нами пункт **About**). Подтвердите все изменения, нажав кнопку **OK**.

Появится узел данных события. Один из его терминалов – **ItemTag**. При выборе пункта About на выходе терминала будет строка, записанная в поле **Item Tag** соответствующего элемента меню в редакторе меню. Если вы ничего в этом поле не меняли, то в строке будет слово **About**. Воспользовавшись структурой варианта, можно для каждого добавленного элемента меню (в нашем случае один добавленный элемент, но их может быть несколько) сопоставить свою блок-диаграмму. Поместите на блок-диаграмму структуры события структуру варианта **Case**. Соедините терминал **ItemTag** с терминалом селектора варианта. Вместо True в заголовке варианта запишите **«About»**. А внутри этого варианта создайте строковую константу, например, «Эта программа создана мной!». Эту постоянную подключите к функции **Time & Dialog ⇒ One Button Dialog**. В целом все готово. Если при выполнении программы пользователь обратится к пункту меню **About**, то в структуре события будет выполняться блок-диаграмма этого события, т.е. обращения к пункту пользовательского меню. В случае, если этот пункт является пунктом **About**, то будет выполняться блок-диаграмма варианта «About». В случае, если это другой пункт (вы можете добавить не только этот пункт, но и другие пункты), то будут выполняться блок-диаграмма по умолчанию или другие созданные вами блок-диаграммы для других пунктов меню. Однако следует учитывать еще и тот момент, что в случае если пользователь не обращается к какому-либо пункту меню, то ВП должен выполняться в соответствии с основным замыслом разработчика. Для этого используйте вариант события **Timeout**, выполняемый по истечении времени ожидания. Он при использовании структуры события создается по умолчанию. На его блок-диаграмму ничего добавлять не надо. Поскольку в случае, если к меню обращений не было, ничего произойти не должно. Время ожидания устанавливает-

ся в миллисекундах и подключается к терминалу времени ожидания в левом верхнем углу структуры события. Допустим, время будет составлять 20 мс.

Окончательная блок-диаграмма показана на рис. 16.4 (скрытая пустая блок-диаграмма варианта **Timeout** не показана).

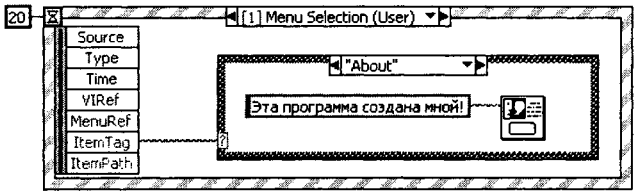


Рис. 16.4

Запустите ВП. Выберите в пункт меню **Help** ⇒ **About**. Должно появиться диалоговое окно с сообщением «Эта программа создана мной!». В заключение следует добавить, что в LabVIEW предусмотрена палитра функций для работы с элементами меню **Application Control** ⇒ **Menu**. Используйте функции этой панели для работы с элементами меню непосредственно на блок-диаграмме.

Выводы

Экспресс-ВП в отличие от функций позволяют решать широкий класс однотипных задач, ориентированных на сбор, обработку и визуализацию данных. Настройка различных параметров производится в отдельном диалоговом окне. Работа с экспресс-ВП значительно упрощает сборку блок-диаграммы. При помощи редактора меню имеется возможность создать собственное меню или модифицировать стандартное.

Лекция 17

Формирование отчетов, изменение внешнего вида объектов лицевой панели, менеджер библиотек

Продолжается рассмотрение дополнительных вопросов программирования в LabVIEW. Описываются создание отчетов, позволяющих наглядно представить данные, возможности изменения внешнего вида стандартных элементов управления и индикации. Рассматривается работа с библиотеками.

Формирование отчетов

Покажем на примере, как использовать функции генерации отчетов палитры **Report Generation**. Создадим ВП, который формирует отчет в HTML файл.

Задание 17.1. Формирование отчета

Перед формированием самого отчета необходимо выбрать для него основу. Предположим, что вы изучаете основные характеристики случайного процесса. Получите выборку из ста исходов при помощи генератора случайных чисел. Полученную выборку отобразите на графике и вычислите среднее значение и дисперсию. Здесь следует отметить, что LabVIEW предоставляет обширную библиотеку функций для анализа данных, которые доступны из палитры Analyze. Там же можно найти и множество математических функций, в том числе и статистических. В этой связи при решении этой задачи не имеет смысла собирать блок-диаграмму с использованием палитры **Numeric**. Вполне естественно, что в LabVIEW уже есть готовые функции для определения моментов различного порядка. Поэтому воспользуйтесь функциями палитры **Analyze** ⇒ **Mathematics** ⇒ **Probability and Statistics**. Решение этой задачи показано на рис. 17.1.

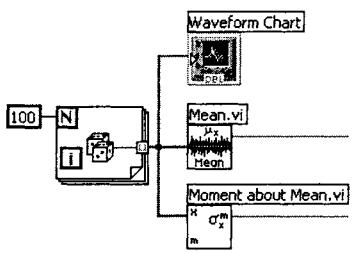


Рис. 17.1

Начать формирование отчета следует с ВП создания нового отчета **New Report**. Поместите его на блок-диаграмму. По умолчанию этот ВП создает новый стандартный отчет. Чтобы создать HTML отчет, надо на вход **report type** подать единицу или создать для него константу и поменять **Standard Report** на HTML.

Добавьте в отчет изображение графика выборочной функции. Добавление изображения какого-либо элемента управления осуществляет ВП **Report Generation ⇒ Append Control Image To Report**. Выход **report out** ВП **New Report** соедините с входом **report in** функции **Append Control Image To Report**. То же самое проделайте с выходом и входом ошибки. В дальнейшем выход report out и error out одного ВП соединяйте с соответствующим входом другого. Для ВП **Append Control Image To Report** необходимо указать, какой элемент управления следует поместить в отчет, а также описание изображения. Для того, чтобы сослаться на график, создайте для него ссылку. В его контекстном меню выберите **Create ⇒ Reference**. Эту ссылку и подайте на вход **Ctrl Reference**. К входу **description** подайте строку «Выборочная функция».

Чтобы сразу наблюдать результат, добавьте на блок-диаграмму ВП **Report Generation ⇒ HTML ⇒ Open HTML Report in Browser**. Запустите программу. В появившемся окне браузера вы сможете наблюдать график. В случае, если браузер не смог найти временной файл HTML отчета, возможно, что в пути к временной папке у вас имеются русские буквы. Поменять путь к временной папке можно в настройках LabVIEW в меню **Tools ⇒ Options**. Там на вкладке **Path** выберите Temporary Directory, снимите галочку с **Use Default** и назначьте другую директорию, например, C:\Temp. Если этот вариант со сменой папки вас по какой-либо причине не устраивает, воспользуйтесь функцией сохранения отчета в файл **Report Generation ⇒ Save Report To File**. Добавленные в следующих пунктах функции следует располагать до этой функции.

Добавьте в отчет комментарий к графику. Это можно сделать при помощи функции **Report Generation ⇒ Append Report Text**. Ко входу **text** подключите тот же самый текст «Выборочная функция». Чтобы текст начинался с новой строки ко входу **Append on new line?** подключите логическую константу **True**.

Добавьте в отчет горизонтальную черту, отделяющую график и его описание от остальной части отчета. Для этого следует воспользоваться функцией **Report Generation ⇒ HTML ⇒ Append Horizontal Line To Report**.

Далее поместим в отчет полученные характеристики нашей выборки случайного процесса. Результат представьте в виде таблицы:

Среднее значение	M(x)
Дисперсия	D(x)

Таблицу в отчет помещает функция **Append Table To Report**. Таблица должна представлять собой двумерный строковый массив. Попробуйте сформировать такой массив сами. Сравните ваше решение с нашим (рис. 17.2).

Кроме этого функция **Append Table To Report** позволяет добавить шапку таблицы. Шапка должна представлять собой одномерный строковый массив. Создайте такой массив с элементами «Величина» и «Значение» и подайте его на вход **column headers**. Чтобы у таблицы была видна сетка на вход **show grid lines** следует подать логическую переменную **True**.

Перед таблицей также как и в пункте 5 включите в отчет название таблицы «Характеристики».

После того, как все необходимые элементы в отчет добавлены, его следует сохранить в файл. Это можно сделать с помощью функции **Save Report To File**. Можно также просто открыть его в окне браузера.

Рис. 17.2

Итак, ВП полностью готов. Блок-диаграмма по формированию отчета показана на рис. 17.3. А сам полученный отчет представлен на рис. 17.4.

Изменение внешнего вида элементов управления и индикации

Окно редактирования внешнего вида элементов лицевой панели

Если вас по каким-либо причинам не устраивают стандартные элементы управления и индикации, вы можете изменить их внешний вид. LabVIEW позволяет

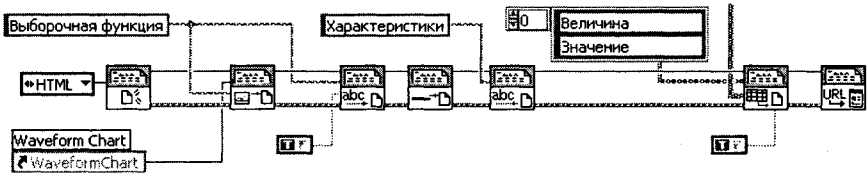
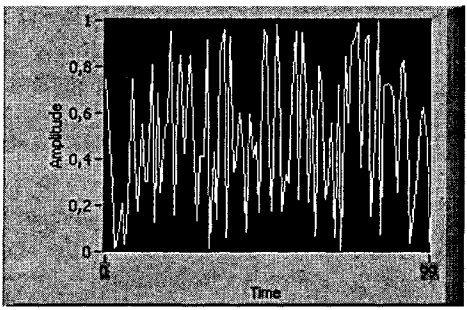


Рис. 17.3



Выборочная функция

Характеристики

Величина	Значение
Среднее значение	0,4974
Дисперсия	0,0916

Рис. 17.4

практически до неузнаваемости изменить внешний вид элементов на передней панели приложения при условии, что сохраняется их общая функциональность.

Пользователь может только изменить стандартный элемент управления и индикации. Поэтому начинать работу следует с выбора стандартного элемента лицевой панели, наиболее близкого по внешнему виду и функциональным возможностям к тому, что вы желаете получить.

Перейти в режим редактирования внешнего вида базового элемента управления и индикации можно, выбрав в его контекстном меню пункт **Advanced ⇒ Customize**... Откроется окно редактора (**Control Editor**) с логическим элементом индикации внутри. Есть и другой способ попасть в редактор элементов лицевой панели. Для этого нужно воспользоваться пунктом меню **File ⇒ New**... и в диалоге создания нового файла выбрать **Other Documents ⇒ Custom Control**. После чего нужно будет добавить на панель базовый элемент управления или индикации.

Панель редактора объектов лицевой панели очень похожа на лицевую панель ВП, за исключением того, что на ней можно разместить только один объект.

Редактор элементов управления и индикации имеет два режима работы. Первый из них носит название режим редактирования. В него вы попадаете автоматически в начале работы с редактором. В этом режиме можно менять размеры и пропорции элемента управления или индикации (как впрочем, и на лицевой панели ВП). Второй режим называется режимом настройки. В него можно попасть через меню **Operate ⇒ Change to Customize Mode** или нажав на кнопку  в панели инструментов.

Режим настройки

Рассмотрим режим настройки подробнее, так как именно он позволят радикально менять внешний вид объектов лицевой панели. При переходе в этот режим элемент управления или индикации разбивается на составные части и их можно редактировать отдельно.

Отдельные составные части элемента управления и индикации могут перекрывать друг друга. Чтобы выбрать элемент, находящийся на заднем плане, можно воспользоваться окном частей объекта **Control Parts**. Оно выводится на экран с помощью пункта меню **Window ⇒ Show Parts Window** (рис. 17.5). В центре окна выводится изображение составной части элемента управления или индикации, а под ним его название, местоположение и размер. Кнопки со стрелками используются для переключения между различными частями. В данном окне нельзя производить настройку частей. Оно предназначено только для выбора частей объекта.

С каждой частью элемента управления и индикации связно графическое изображение (а иногда и не одно), которое можно изменять по своему усмотрению. Все операции с изображениями в окне редактирования производятся через буфер обмена. Вы можете поместить изображение в буфер обмена из любого графического редактора (в буфер обмена любой объект помещается выделением и выбором пункта меню **Edit ⇒ Copy** или нажатием **ctrl+C**), а если картинка хранится в файле, воспользоваться пунктом меню **Edit ⇒ Import Picture from File**.

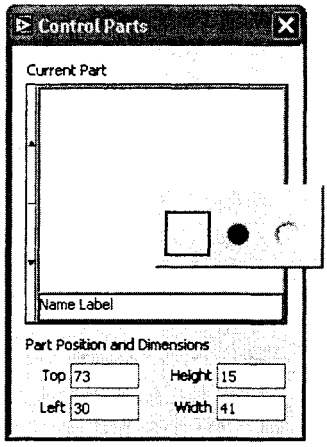


Рис. 17.5

Чтобы вместо стандартной поместить свою картинку, сначала нужно скопировать ее в буфер обмена. Затем выбрать встроенную картинку, которую вы хотите заменить. Сделать это можно, нажав правой кнопкой мыши на элемент управления или индикации и выбрав из контекстного меню пункт **Picture Item**. Перед вами появится список графических изображений, связанных с этой частью элемента управления или индикации. Как мы уже упоминали раньше, картинок может быть не одна, а две или даже четыре. Стрелки элемента управления **Pointer Slide** содержат две картинки. Одна предназначена для активного ползунка, другая – для остальных. Работа логического элемента управления имитируется с помощью четырех изображений. Первые две используются для состояния «ложь» и «истина», а третья и четвертая – для переходных состояний, когда рассматривается механическое действие при нажатии кнопок.

Замена встроенной картинки на рисунок из буфера производится с помощью пункта контекстного меню **Import Picture**. Другие пункты контекстного меню также предназначены для работы с изображениями. Вот их краткое описание:

- **Copy to Clipboard** – копирует отображаемую в данный момент картинку в буфер обмена. Этот пункт можно использовать, чтобы копировать изображение отдельных частей из одного элемента управления или индикации в другой.
- **Import Picture** – заменяет текущую картинку на ту, что находится в буфере обмена.
- **Import Picture at Same Size** – то же самое, что и в предыдущем пункте, но размер картинки из буфера автоматически погоняется под размер текущей картинки.
- **Revert** – все картинки заменяются на те, что были до того как вы воспользовались пунктом контекстного меню **Advanced ⇒ Customize** на лицевой панели ВП (если вы попали в окно редактирования другим способом, этот пункт меню будет неактивен).
- **Original Size** – восстанавливает изначальные размеры изображения. Используйте этот пункт, если вы растягивали или сжимали рисунок.

Режим редактирования

Режим редактирования дает возможность менять размеры и пропорции элемента управления и индикации. Обычно, то же самое можно сделать прямо на лицевой панели ВП, не прибегая к использованию окна редактирования. Исключением является тот случай, когда элемент управления или индикации необходимо сохранить, как точный эталон. Для этой цели используется строгое определение типа (**Strict Type Def.**), оно более подробно описано ниже.

Режим редактирования имеет одну особенность, которая проявляется при редактировании логических элементов управления и индикации. А именно он позволяет заменять стандартные изображения для истинного и ложного состояния. Пункт контекстного меню **Import Picture** содержит три подпункта для замены рисунков для ложного, истинного состояния (**False** и **True**) и для ободка вокруг объекта (**Decal**). Как обычно, для замены используются изображения из буфера обмена.

Как видно, этот способ немного проще того, что мы использовали при работе в режиме настройки, но в то же время и менее гибок, так как нельзя указать отдельный рисунок для переходных состояний.

Определение типа

Когда вы создаете новый элемент управления или индикации, вы можете сохранить его на диске в виде файла. Вставить его в любой ВП, даже на другом компьютере, можно через панель объектов лицевой панели: **Controls** ⇒ **All Controls** ⇒ **Select a Control**.

Разные экземпляры элемента управления или индикации, созданные на основе одного файла не зависят друг от друга. Изменения, сделанные в одном экземпляре, никак не сказываются на остальных. Это не всегда удобно. Например, вы наверняка захотите, чтобы в рамках одного проекта однотипные элементы управления и индикации выглядели одинаково. Добиться единообразия можно, сохранив элемент управления или индикации, как определение типа (**Type Def.**) или как строгое определение типа (**Strict Type Def.**). В этом случае все экземпляры элемента управления или индикации сохраняют связь с базовым файлом и изменения сделанные в одном экземпляре, немедленно отражаются и на других. Для изменения типа элемента используется выпадающий список «**Type Def. Status**»  в панели инструментов **Control Editor**.

Разница между определением типа и строгим определением типа заключается в степени, в которой каждый экземпляр элемента управления или индикации может отличаться от базового. Так элементы, сохраненные как строгое определение типа, не могут отличаться даже размерами. И как следствие, вы не сможете изменить размер элемента управления непосредственно на лицевой панели ВП. Для этого придется воспользоваться окном редактирования.

Диалоговое окно VI Library Manager

Диалоговое окно **VI Library Manager** необходимо для того, чтобы вы могли из объединенных общей функциональностью файлов создать один файл – библиотеку. Оно вызывается выбором пункта меню **Tools** ⇒ **VI Library Manager**. Окно показано на рис. 17.6.

В этом диалоговом окне вы можете создавать библиотеки ВП (файлы llb), работать с файлами в уже существующих библиотеках, копировать, переименовывать, удалять. Окно похоже на классический файловый менеджер с двумя полями, в которых показан список директорий и файлов. При этом файлы llb воспринимаются

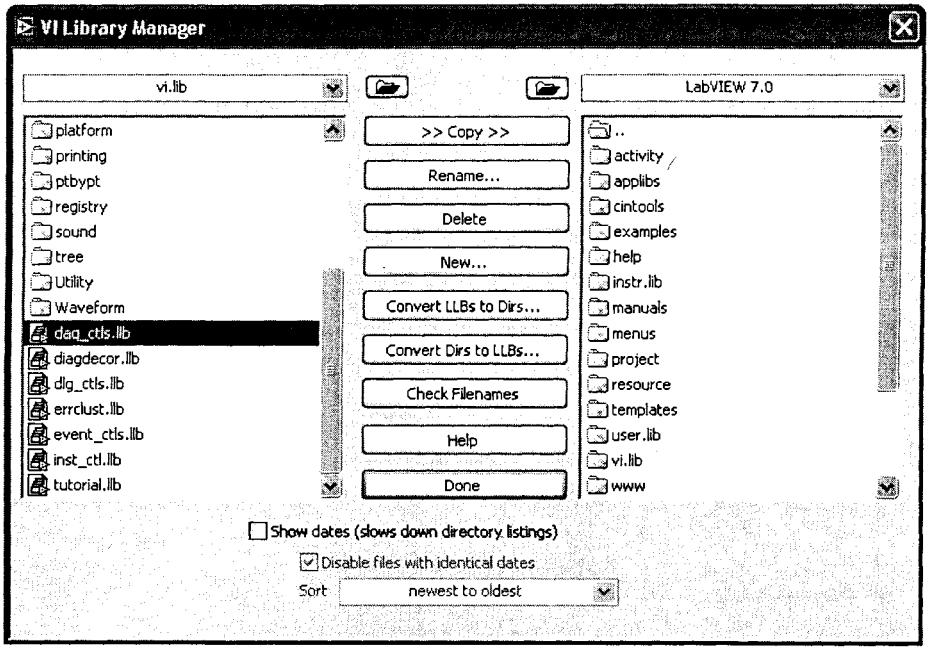


Рис. 17.6

как директории, в которые вы также можете заходить и работать с файлами в них. После того, как вы выбрали файл, активизируются кнопки, которые позволяют совершать следующие действия:

- **Copy** — копирует выбранный элемент в выбранную директорию второго поля;
- **Rename** — переименовывает выбранный файл или директорию;
- **Delete** — удаляет выбранный элемент из списка. У вас не получится удалить директорию, если она не является пустой;
- **New** — создает новую директорию или файл llb;
- **Convert LLBs to Dirs** — преобразует выделенную библиотеку llb в директорию. Если выбранным элементом является директория, то при нажатии на эту кнопку в директории находятся все llb файлы. После этого вы можете выбирать, как и какие llb файлы преобразовывать. Новая директория создается в том же месте, где расположен исходный llb файл;
- **Convert Dirs to LLBs** — преобразует выбранную директорию в файл llb;
- **Check Filenames** — производит в директории или библиотеке ВП поиск имен файлов, содержащих символы (:, \, /, ?, *, , , или). Проверяет, чтобы имена файлов содержали менее 31 символа. Опция **Check Filenames** проверяет файлы, в том числе и внутри библиотек llb. Файлы с именами, не содержащими указанных символов и содержащими менее 31 символа, могут быть

перенесены на другие платформы. Этот инструмент позволяет обнаружить потенциальные проблемы при перемещении файлов из библиотек ВП.;

- **Done** – выходит из диалогового окна **VI Library Manager**.

Выводы

Создание отчетов в LabVIEW позволяет использовать все основные объекты при представлении данных для последующей распечатки или сохранения в файл. Внешний вид стандартных элементов управления или индикации можно кардинальным образом изменять. Предусмотрена возможность собирать в одну библиотеку (один файл) набор ВП, объединенный общим назначением. Для реорганизации файлов в библиотеках существует инструмент **VI Library Manager**.

Лекция 18

Введение в сбор данных

Рассматривается общий подход работы с реальным сигналом, принципы считывания аналогового сигнала платой ввода-вывода. Описываются первые шаги по настройке программного обеспечения для сбора данных.

Среда LabVIEW включает в себя набор подпрограмм ВП, позволяющих конфигурировать, собирать и посылать данные на DAQ-устройства. Сама аббревиатура DAQ расшифровывается как Data Acquisition и переводится на русский язык как сбор данных. DAQ-устройства могут выполнять разнообразные функции: аналого-цифровое преобразование (**A/D**), цифро-аналоговое преобразование (**D/A**), цифровой ввод/вывод (**I/O**) и управление счетчиком/таймером. Каждое устройство имеет свой набор возможностей, у каждого устройства своя скорость обработки данных.

DAQ-устройства

Назначение DAQ-устройств

DAQ-устройство позволяет осуществлять сбор и генерирование данных. Его с одной стороны подключают к компьютеру, а с другой – к реальной электрической цепи. Получая сигнал от этой цепи в компьютер, QAQ-устройство обрабатывает его и передает компьютеру.

На базе персонального компьютера можно создать целую измерительную систему, в которую необходимо включить DAQ-устройство. При этом DAQ-устройство только преобразует входящий сигнал в дискретную форму, читаемую компьютером. Это означает, что одно и то же DAQ-устройство может производить множество различных измерений с помощью различных программ, которые считывают и обрабатывают данные. Хотя подобная гибкость позволяет вам использовать всего одно техническое устройство для множества видов измерений, вам потребуется значительное время для разработки соответствующих программ. Помочь с этой проблемой может LabVIEW, которая включает в себя множество функций для сбора данных и их последующего анализа.

Составление измерительных систем на базе компьютера и DAQ-устройства

Перед тем как DAQ-устройство сможет измерить физический сигнал, например температуру, преобразователь должен преобразовать измеряемую физическую величину в электрическую, обычно напряжение или ток. Возможно, вы считаете, что само по себе DAQ-устройство – это целая измерительная система, однако на самом деле оно представляет собой только одну составляющую такой системы. Вы не сможете подать сигнал сразу на DAQ-устройство. До этого исходный сигнал следует привести сигналу, который DAQ-устройство сможет считать и преобразовать в цифровой. В табл. 18.1 представлен список некоторых преобразователей.

Таблица 18.1

Явление	**Преобразователь**
Температура	*Термопара*
	Термометр сопротивления
	Термистор
	Датчики на интегральных схемах
Свет	*Вокуумный фотоэлемент*
	Фоторезистор
Звук	*Микрофон*
Сила и давление	*Тензометр*
	Пьезоэлектрический датчик
	Датчик напряжений
Местоположение (смещение)	*Потенциометр*
	Сейсмометр
	Линейный датчик на основе дифференциального тронсформатора
	Оптическое кодирующее устройство
	Поток жидкости Манометр
	Ультразвуковой расходометр
pH	*pH электроды*

После того как физическая величина преобразована в электрическую, ее уже можно измерять для получения необходимой вам информации: величины, скорости изменения, формы, частотного спектра и др. При этом в зависимости от поставленной задачи еще следует учесть технические средства по устранению помех и уже после этого подключать полученный сигнал к DAQ-устройству.

DAQ-системы могут быть подсоединены к компьютеру следующим образом:

- Съемное DAQ-устройство помещается в компьютер. Вы можете установить устройство в PCI-слот настольного компьютера или в PCMCIA слот мобильного компьютера с переносной измерительной системой DAQ.
- DAQ-устройство является внешним и подсоединяется к компьютеру через существующий порт (к примеру, последовательный порт или Ethernet порт).

Структурная схема измерительной системы обоих случаев показана на рис. 18.1.

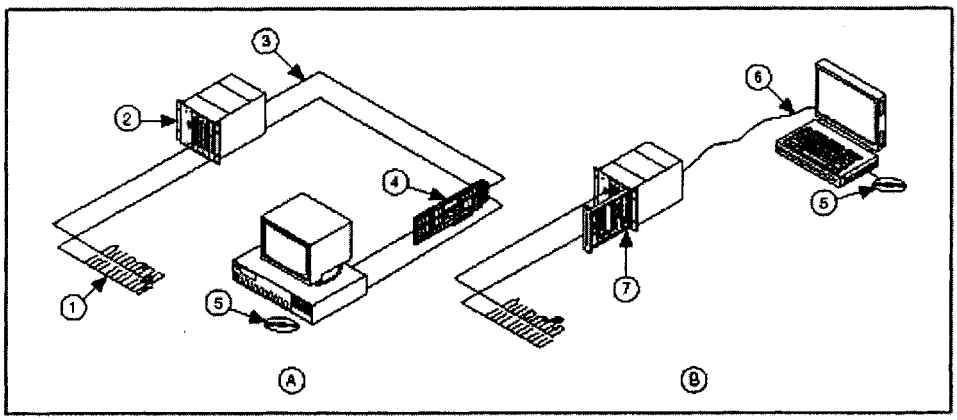


Рис. 18.1

1. *Датчики*
2. *Модуль согласования сигналов*
3. *Согласованные сигналы*
4. *Встроенное DAQ-устройство*
5. *Программное обеспечение*
6. *Связь с параллельным портом*
7. *Внешний DAQ-модуль*

Роль программного обеспечения

Через DAQ устройство компьютер получает исходные данные. Далее они обрабатываются в приложении, которое вы можете написать сами, используя возможности LabVIEW.

Обычно к DAQ-устройству прилагаются драйверы и программа. Драйверы представляют собой набор команд, которые устройство понимает. Программа посылает устройству различные команды, например, считать и вернуть показание напряжения. Программа к тому же анализирует собранные данные и отображает результаты анализа на экране.

Программное обеспечение измерительных устройств NI включает драйвер NI-DAQ, набор виртуальных приборов для настройки, сбора и отправки данных измерительному устройству.

Комплекс разработки приложений сбора данных состоит из среды программирования, **Measurement & Automation Explorer** и NI-DAQ. **Measurement & Automation Explorer** является высокоуровневым приложением, которое используется для тестирования и настройки DAQ-устройств. NI-DAQ состоит из следующих программных интерфейсов:

- традиционный NI-DAQ
- NI-DAQmx

Традиционный NI-DAQ – это усовершенствованная до версии NI-DAQ 6.9.x ранняя версия драйверов NI-DAQ.

NI-DAQmx – это последняя версия драйвера NI-DAQ с новыми ВП и функциями и инструментами для управления измерительными устройствами. Преимуществами NI-DAQmx перед предыдущими версиями NI-DAQ является **DAQ Assistant** – помощник по настройке каналов и измерительных задач устройства (о нем подробно будет рассказано в главе 20); увеличенная производительность, включающая более быстрый аналоговый ввод/вывод для создания DAQ приложений с использованием меньшего числа функций и ВП, чем в ранних версиях NI-DAQ.

Традиционный NI-DAQ и **NI-DAQmx** поддерживает различный набор устройств. Список поддерживаемых устройств опубликован по адресу ni.com/daq.

Настройка измерительных устройств

Перед тем, как начать разработку измерительных приложений, вам необходимо установить и настроить DAQ-устройства. Для установки DAQ-устройства выполните следующие действия:

1. Установите LabVIEW и драйвер. Инсталлятор LabVIEW устанавливает драйвер NI, если версия, приложенная с LabVIEW более новая, чем ранее установленная версия драйвера. Для уверенности в том, что текущая версия драйвера поддерживает устройство, установите драйвер с установочного диска, поставляемого с устройством.
2. Выключите компьютер.
3. Вставьте само устройство в соответствующий слот материнской платы. При этом следует ознакомиться с поставляемой с устройством документацией, чтобы понять, нет ли необходимости о дополнительных манипуляциях. Например, некоторые устройства имеют джамперы для выбора полярности аналогового входа и т.д. Не забудьте записать проделанные вами изменения.
4. Включите компьютер.
5. Настройте измерительное устройство, используя **Measurement & Automation Explorer**.

Measurement & Automation Explorer

Measurement & Automation Explorer (MAX) – это приложение, которое позволяет настроить программное и аппаратное обеспечение NI, запустить систему диагностики, добавить новые каналы и интерфейсы и наблюдать подсоединенные устройства и приборы. Оно устанавливается в процессе установки драйвера NI. При использовании традиционных NI-DAQ вам придется использовать MAX для настройки DAQ-устройств. Запустить MAX можно через иконку на рабочем столе или выбором пункта главного меню **Tools ⇒ Measurement & Automation Explorer** непосредственно в среде LabVIEW.

MAX необходимо запускать после установки DAQ-устройства на компьютер. Утилита конфигурации считывает информацию из реестра Windows, записанную Диспетчером устройств (**Device Manager**), и присваивает логическое имя для каждого DAQ-устройства. По логическому имени среда LabVIEW распознает DAQ-устройство. Начальное окно конфигурационной утилиты показано на рис. 18.2.

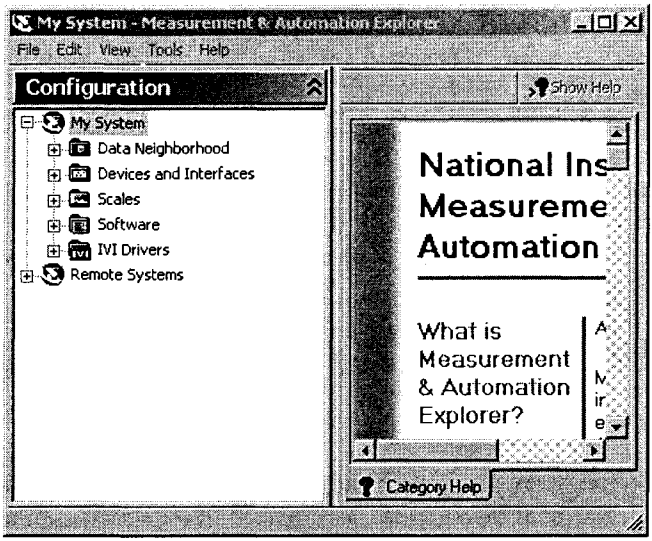


Рис. 18.2

Параметры устройства можно также установить с помощью утилит-конфигураций, входящих в комплект поставки устройств. **Measurement & Automation Explorer** позволяет сохранить логическое имя устройства и параметры конфигураций в реестр Windows.

Windows автоматически находит и настраивает DAQ-устройства, удовлетворяющие стандарту PnP, например, карту PCI-MIO-16-E.

В правой части окна расположена панель **Configuration** (рис. 18.3), в которой в виде дерева показаны составляющие системы, с которыми работает MAX:

- **My System** – локальный компьютер.
- **Data Neighborhood** – Виртуальные каналы. Дает возможность доступа к виртуальным каналам (именованным и заранее настроенным физическим). Вы можете создать и сконфигурировать виртуальный канал, вызвать тестовую панель, скопировать, удалить.
- **Devices and Interfaces** –Устройства и интерфейсы. Содержит список установленных и опознанных системой физических устройств: **DAQ, FieldPoint, GPIB, IMAQ, IVI, Motion, VISA, VXI** и т.п. Вы можете настроить существующие устройства, добавлять не найденные системой и удаленные устройства. Если устройство поддерживается и классическим драйвером и mx-драйвером, то оно (как показано на рис. 18.3) изображается в двух соответствующих категориях.
- **Scales** – Шкалы. Для более удобной работы с датчиками имеется возможность создать шкалу соответствия сигнала, получаемого с датчика, измеряемой датчиком величине. Шкала может быть линейной, задана полиномом или таблицей. Вы можете добавлять новые, удалять, просматривать и изменять свойства шкал.

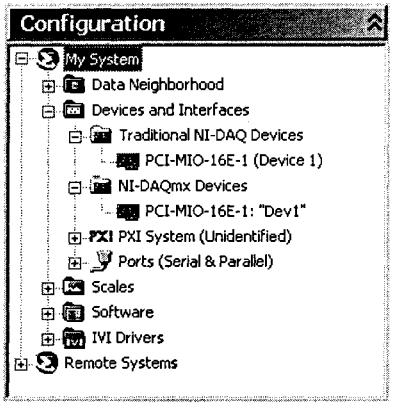


Рис. 18.3

- **Software** – программное обеспечение. Здесь представлен список программного обеспечения **National Instruments**, установленного на компьютере. Вы можете просматривать запускать и посредством интернет обновлять программное обеспечение.
- **Remote System** – удаленные системы, подключенные по сети другие компьютеры или устройства. Вы можете просматривать список удаленных систем, и конфигурировать некоторые их свойства.

Классические драйверы

Для устройств поддерживаемых классическими драйверами имеется возможность вызвать диалог **configure** (настройка), в котором в частности для аналогового ввода можно указать диапазон измеряемых величин и режим входа: дифференциальный, режим с общим проводом или режим с общим проводом, заземленным в конце (рис. 18.4).

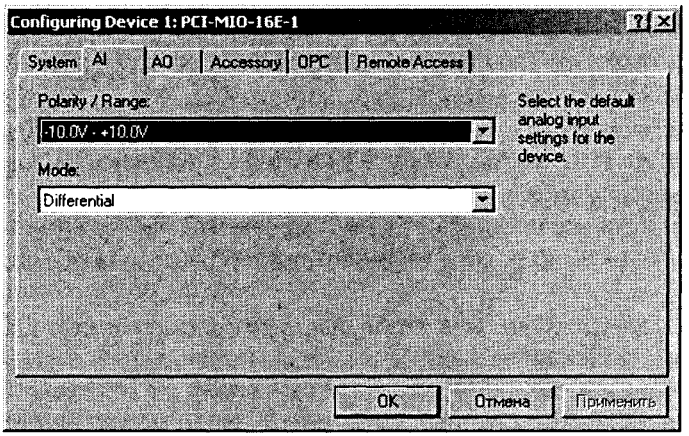


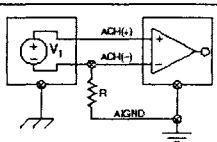
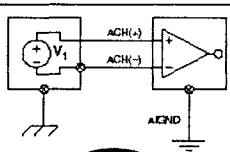
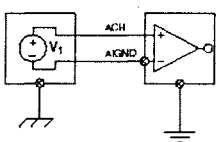
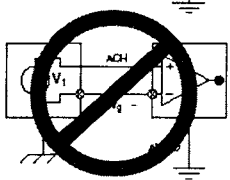
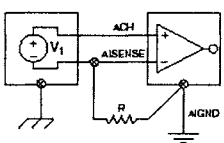
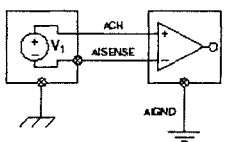
Рис. 18.4

В дифференциальном режиме входа (**Differential – DIFF**) выводы DAQ-устройства разбиваются на пары. Каждая пара выводов образует один канал. Ни один из вводов дифференциальной системы измерения не соотносится к какой-либо фиксированной базисной точке, например земле. Считывание сигнала производится между положительным и отрицательным выводами одного канала. Пара выводов при такой схеме называется **ACH(+)** и **ACH(-)**.

В двух других схемах измерений каналы образуются для каждого вывода в отдельности. Измерения проводятся для каждого канала относительно земли. В режиме с общим проводом, заземленном в конце (**Referenced Single- Ended Ground – RSE**) измерение осуществляется относительно точки **AIGND**, которая соединяется непосредственно с землей измерительной системы. В режиме с общим проводом, незаземленном в конце (**Non-referenced Single- Ended – NRSE**) измерение осуществляется относительно контрольной точки аналогового входа **AISENSE**, потенциал которой может быть отличным от потенциала земли измерительной системы **AIGND**.

В табл. 18.2 приводятся шесть возможных вариантов схем измерения.

Таблица 18.2

Вход Input	*Тип источника сигнала*	
	«Плавающий» источник сигнала Floating Signal Source	*Заземленный источник сигнала Grounded Signal Source*
Дифференциальный ***Differential (DIFF)***		
С общим проводом, заземленном в конце ***Single - Ended Ground Referenced (RSE)***		 *Не рекомендуется*
С общим проводом, не заземленном в конце ***Single – Ended Ground Nonreferenced (NRSE)***		

Выбор конфигурации входа зависит от конкретной задачи.

В приложении MAX имеется возможность вызвать тестовую панель, в которой можно протестировать все функции DAQ-устройства (рис. 18.5). Каждой задаче соответствует своя закладка: **Analog Input** – аналоговый ввод, **Analog Output** – аналоговый вывод, **Counter I/O** – управление счетчиком/таймером, **Digital I/O** – цифровой ввод-вывод.

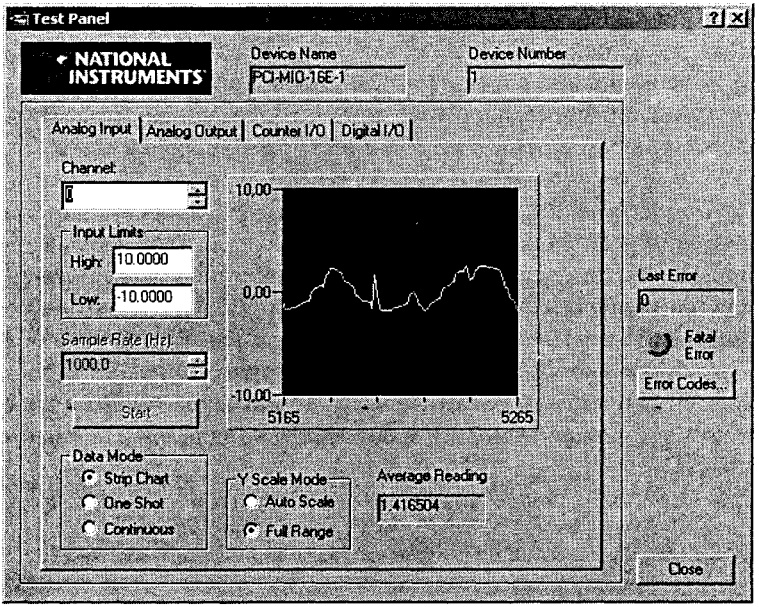


Рис. 18.5

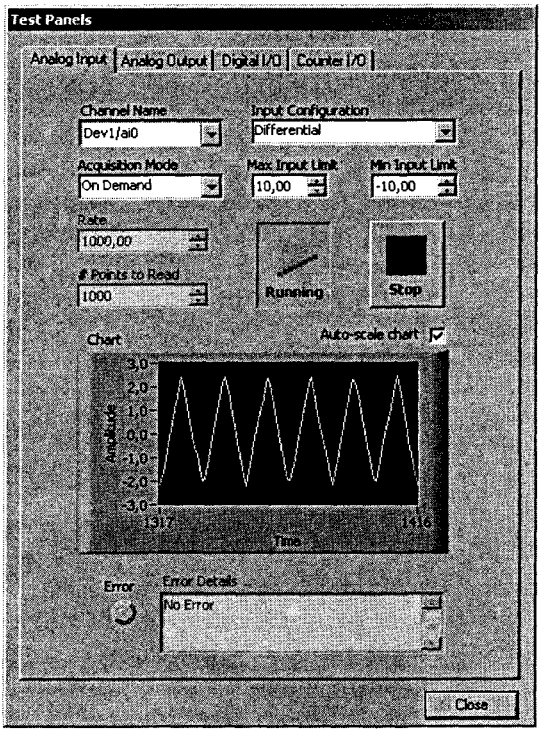


Рис. 18.6

DAQmx-драйверы

Для устройств поддерживаемых DAQmx-драйверами так же существует тестовая панель, функции которой аналогичны тестовой панели классических драйверов (см. рис. 18.6).

В MAX имеется возможность в сохранить в файл (**Export**) и восстановить из файла (**Import**) настройки. После того как система, устройства, шкалы и каналы настроены, вы можете сохранить настройки в файл, например для того, чтобы позже к ним можно было вернуться.

MAX является основным конфигурационным и тестовым приложением для DAQ-устройств.

Частота дискретизации (отсчетов)

В заключение первой лекции по сбору данных необходимо рассказать о самом принципе дискретизации сигнала. Дискретизация сигнала заключается в том, что при считывании аналоговых реальных сигналов DAQ-устройством каждой точке присваивается некоторый уровень, дискретное число. Если соединить точки сигнала и вывести на экран его изображение, то оно будет похоже на реальный сигнал (рис. 18.7).

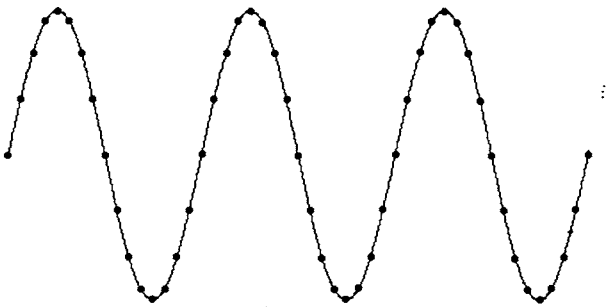


Рис. 18.7

Один из наиболее важнейших факторов при работе с аналоговым входом и выходом измерительной системы – это частота, с которой измерительное устройство считывает входящий сигнал или генерирует исходящий. Это частота называется частотой дискретизации или частотой отсчетов. Она определяет, с какой частотой аналого-цифровое или цифро-аналоговое преобразование производить. Большая частота сканирования позволяет считать больше точек в заданное время и может сформировать более точное представление об исходном сигнале в отличие от меньшей частоты дискретизации.

Подмена частот (при недостаточно высокой частоте дискретизации сигнала)

Небольшая частота дискретизации проявляется в наложении спектра, которое искажает представление об исходном сигнале. В результате может получиться так, что отсчитанный сигнал окажется сигналом с совсем другой частотой, чем исходный сигнал. Это хорошо показано на рис. 18.8. Избежать подмены частот можно увеличением частоты дискретизации сигнала.

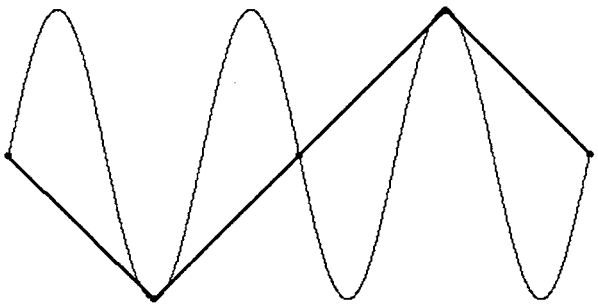


Рис. 18.8

В соответствии с теоремой Котельникова (теоремой отсчетов)[1] сигнал следует считывать с частотой дискретизации вдвое большей наибольшей частоты гармоники в сигнале. В таком случае исходный сигнал можно полностью восстановить.

Выводы

Компьютерные измерения необходимо начинать с построения измерительной системы, в которую входят датчики, модуль согласования, DAQ устройство, программное обеспечение. LabVIEW предоставляет широкие возможности для сбора данных. Она включает в себя драйверы для DAQ устройства производства National Instruments, функции аналогового ввода, аналогового вывода, цифрового ввода-вывода, синхронизации. После установки необходимого оборудования целесообразно запустить средство **Measurement & Automation Explorer**, которое позволяет настроить программное и аппаратное обеспечение, добавить виртуальные каналы, шкалы и т.д., проверить работоспособность оборудования. Кроме решения технических проблем при аналоговом сборе данных следует правильно выбирать частоту дискретизации.

[1] Теорема Котельникова (теорема отсчетов) формулируется следующим образом: непрерывный сигнал, спектр которого не содержит частот больших f_m, может быть однозначно представлен своими мгновенными значениями, разделенными одинаковыми интервалами времени, длина которых не должна превышать $f_m/2$.

Лекция 19

Сбор данных на базе традиционного NI-DAQ.

Рассматривается организация сбора данных при использовании классических средств LabVIEW.

Функций ввода/вывода сигнала оперируют типом данных **waveform** (осциллограмма) являющимся специальным типом для работы с оцифрованными данными. Рассмотрим данный тип и связанные с ним специальные функции.

Тип данных осциллограмма (waveform)

Waveform (сигнал) – тип данных, предназначенный для ввода, вывода, обработки и хранения реального сигнала. **Waveform** представляет из себя кластер, состоящий из четырех элементов:

t0 – дата и время начала ввода данных,
dx – интервал времени между выборками,
Y – массив данных,
attributes –свойства.

Для того, чтобы выделить элементы структуры waveform или собрать waveform из отдельных элементов, используется соответственно **Get Waveform Components** и **Build Waveform** (рис. 19.1) из меню **Functions ⇒ Waveform**.

Для генерации виртуального сигнала существует большой набор виртуальных инструментов (они расположены в **Functions ⇒ Waveform ⇒ Analog Waveform VIs and Functions ⇒ Waveform Generation**), в том числе:

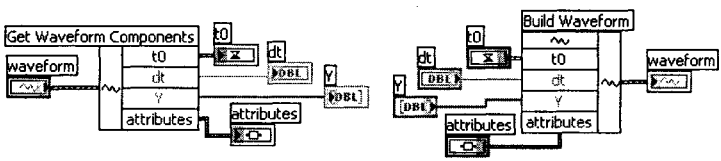
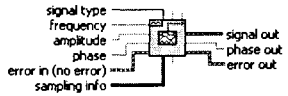
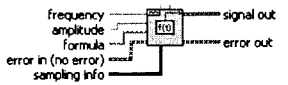


Рис. 19.1

Таблица 19.1

	***Basic Function Generator*** *– генерирует один (по выбору) из основных сиг-*
	***Formula Wavefarm*** *– генерирует сигнал по введенной формуле*
	***Sine Waveform*** *– генерирует синусоидальный сигнал.*

В качестве входных переменных для перечисленных виртуальных инструментов можно ввести частоту, амплитуду, начальную фазу, а также информацию о выборке **sampling info** – кластер из двух чисел **Fs** – частота дискретизации (выборок в секунду) и **#s** – количество точек. Кластер вызывается в палитре кластеров: **Functions ⇒ Cluster ⇒ Cluster Constant**, затем в него помещаются константы **Fs** и **#s** либо создается через контекстном меню соответствующего поля ввода **Create ⇒ Constant**. Пример ВП показан на рис. 19.2.

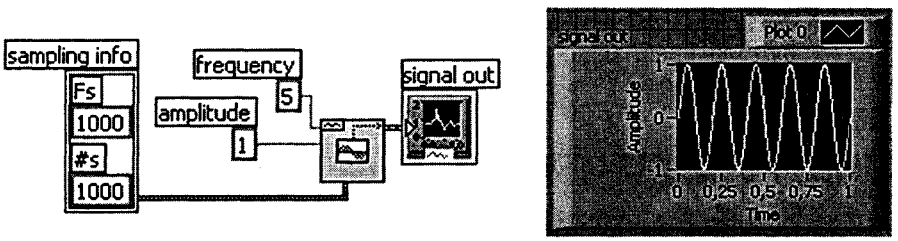


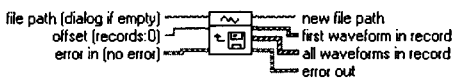
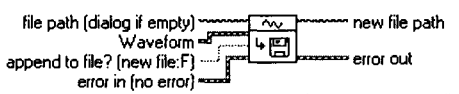
Рис. 19.2

Обратите внимание: в полученной структуре **waveform** количество точек (размер массива **Y**) равно **#s**, а **dx** равно **1/Fs**.

Отобразить сигнал waveform можно на виртуальном графике осциллограмм, в этом случае по оси абсцисс отображается время, а по оси ординат амплитуда.

В случае если возникает необходимость сохранить waveform в файл, следует воспользоваться виртуальными инструментами, расположенными в меню **Functions ⇒ Waveform ⇒ Waveform File I/O**:

Таблица 19.2

	***Read Waveform from File*** *– для считывания из файла*
	***Write Waveform to File*** *– для записи в файл*

Для записи и считывания необходимо задать имя файла; в случае если имя файла не будет задано, LabVIEW вызовет стандартный диалог «Открытие файла» из которого можно выбрать имя.

Аналоговый ввод реального сигнала

Для ввода сигнала в LabVIEW предусмотрены виртуальные инструменты из меню **Functions ⇒ NI Measurements ⇒ Data Acquisition ⇒ Analog Input** (рис 19.3).

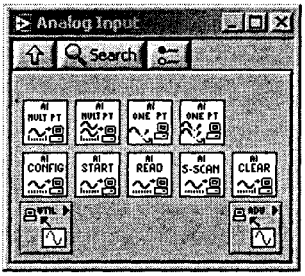
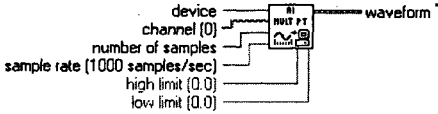
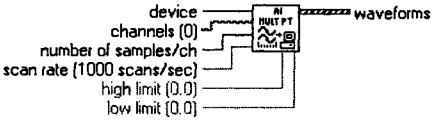
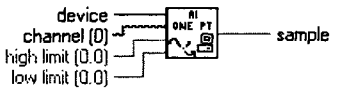
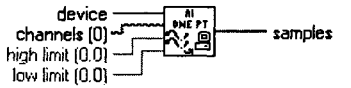


Рис. 19.3

Простые функции аналогового ввода

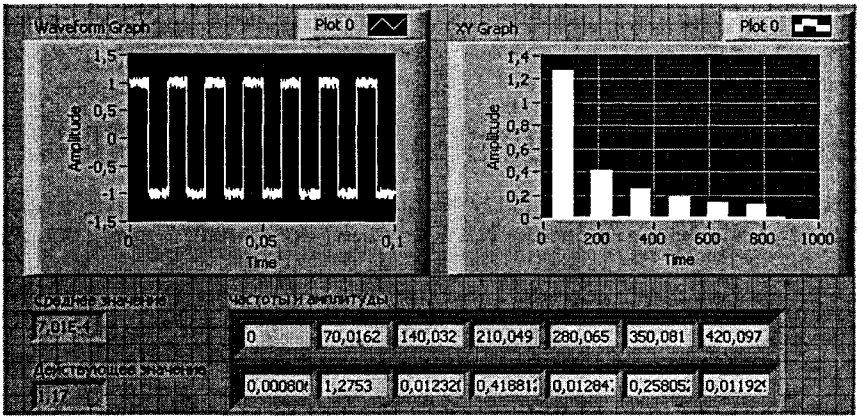
В табл. 19.3 перечислены функции аналогового ввода высокого уровня.

Таблица 19.3

	***AI Acquire Waveform*** *Получить осциллограмму – считывание серии выборок по одному каналу.*
	***AI Acquire Waveforms*** *Получить осциллограммы –считывание одновременных серии выборок по нескольким каналам*
	***AI Sample Channel*** *Выборка из канала – однократное считывание по одному каналу*
	***AI Sample Channels*** *Выборка из каналов – однократное считывание по нескольким каналам*

Для всех функций входными параметрами являются:

- **Device (устройство)** – номер устройство присвоенный плате (см. работу с MAX).



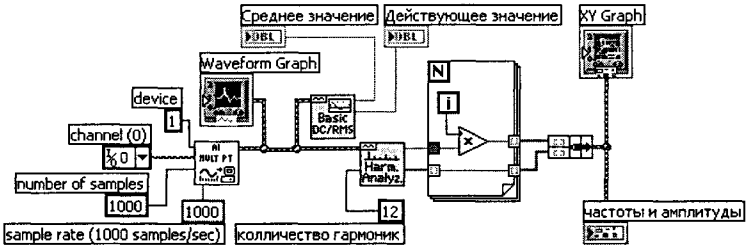


Рис. 19.4

- **Channel (канал)** – определяет физический канал на DAQ устройстве.
- **Number of samples** – количество выборок на канал.
- **Sample rate** – частота с которой производиться считывание.
- **High, Low limit** – верхнее и нижнее ограничение по уровню сигнала.

Пример 19.1. Простейший анализатор спектра

Рассмотрим ВП аналогового ввода и последующий анализ гармонического искажения сигнала (рис. 19.4)

В первую очередь функция **AI Acquire Waveform** преобразует реальный сигнал в массив выборок. На выходе waveform (массив выборок) который выводится на **Waveform Graph**, в результате как на осциллографе наблюдается входной сигнал. С помощью **Basic Averaged DC-RMS** находим постоянную составляющую **DC value** и действующее значение **RMS value** сигнала. Далее массив выборок обрабатывается функцией **Harmonic Distortion Analyzer**, на вход которой так же подаем количество гармоник (в нашем случае 12), на выходе получаем **detected fundamental frequency** – основную частоту и **components level** – массив амплитуд гармоник (начиная с постоянной составляющей). Далее, для того, что бы отобразить амплитудный спектр, создаем массив частот в цикле (начиная с нулевой для

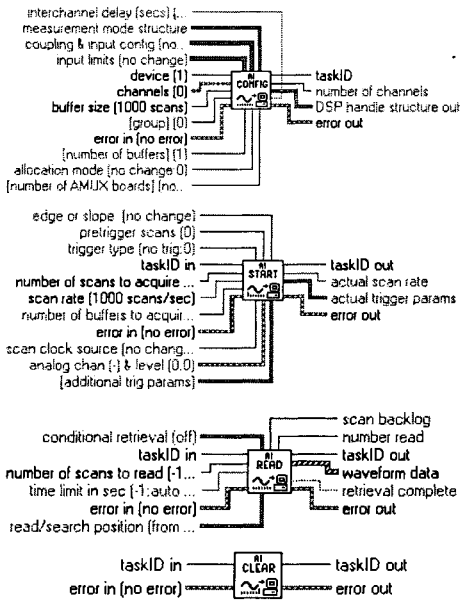
постоянной составляющей, далее для первой гармоники, для второй и т.д.). В результате получаем два массива амплитуд и частот гармоник, которые собираем в кластеры и отображаем на **XY Graph**. Для того, чтобы амплитуды гармоник увидеть в виде линейчатого спектра, требуется вызвать контекстное меню для легенды графика, и изменить свойства графика в подменю **Common Plots** на гистограмму.

Улучшенный аналоговый ввод

Непрерывный сбор данных с использованием буфера

В табл. 19.4 перечислены функции аналогового ввода низкого уровня.

Таблица 19.4



AI Config *– АЦП конфигурация*
Настраивает устройство непосредственно перед процедурой чтения данных. Необходимо задать номер устройства ***(device)****, опрашиваемые каналы* ***(input channels)****, и розмер буфера.*

AI Start *– АЦП старт*
Начинает считывание донных в буфер, можна задоть включение ***(triggering)*** *и частоту сконирования* ***(scan rate)****, 0 поданный на вход* ***number of scans to acquire*** *означает непрерывное считывание*

AI Read *– АЦП чтение*
Считывает данные из буфера. Необходимо задать количество считываемых выборок ***(number of scans to read)*** *Результатом является массив из осциллогромм, по одной но кождый канал*

AI Clear *– АЦП очистка*
Останавливает работу аналогового ввода.

Пример 19.2. Непрерывный аналоговый ввод с использованием буфера

Данный пример демонстрирует непрерывный ввод данных по одному каналу (рис. 19.5). Функция **AI Config** конфигурирует считывание для определенного устройства (номер которого подается на терминал **device**) и перечисленных в **input channels** каналов. Так же создается буфер размером **input buffer size**. ВП **AI Start** отвечает за запуск считывания, 0 поданный на терминал **number of scans to acquire** означает непрерывный ввод, так же необходимым параметром является **scan rate** (частота сканирования). Далее в цикле при помощи ВП **AI Read** происходит считывание данных и отображение их на графике осцилограмм. Считывание заканчивается в случае если пользователь нажмет кнопку **stop** или если возникнет ошибка в ВП **AI Read**. По окон-

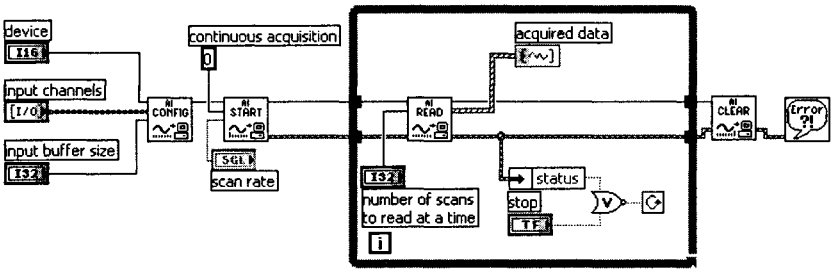


Рис. 19.5

чанию цикла ВП **AI Clear** завершает данное задание. В случае возникновения ошибок ВП **Simple Error Handler** выдаст окно с объяснением ошибки.

Если для конкретной задачи требуется синхронизация считывания (например, для того что бы сигнал не «прыгал» на графике, а «остановился» как на обыкновенном осциллографе) необходимо задать ВП **AI Start** источник запускающего сигнала (см. лекцию 20).

Выводы

Классические (**Traditional NI-DAQ**) функции позволяют решить любые задачи ввода аналоговых данных. При этом для простых задач существуют функции высокого уровня, в то время как для задач связанных с жесткой привязкой ко времени (применение запускающего сигнала, непрерывный сбор данных) необходимо разрабатывать собственную программу для конкретного случая.

Лекция 20

Запуск сбора данных. Использование DAQmx

Описывается организация сбора данных при использовании новейших расширенных возможностей DAQmx. Дается теоретическая справка о принципах включения сбора данных.

Включение (triggering)

Запуск процесса сбора или генерирования данных может определяться программным или аппаратным способом. С программным способом вы уже встречались в прошлой главе. В ней выполнение сбора данных осуществляется при запуске ВП. Аппаратный способ запуска процесса сбора или генерирования данных предполагает использование запускающего сигнала, сигнала, который и определяет момент начала запуска операции сбора или генерирования данных. Например, вам надо проверить отклик монтажной платы на импульсный сигнал. Вы можете использовать этот импульсный сигнал в качестве запускающего сигнала сбора данных. Если вы не используете подобный сигнал запуска, вам необходимо начать сбор данных до того, как вы приложили тестовый импульс. Таким образом, при аппаратном способе включения сбора или генерирования данных DAQ-устройство ожидает некоторое внешнее событие. Внешним событием может быть аналоговый или цифровой запускающий сигнал, характеристики аналогового или цифрового сигнала, такие как состояние, уровень, наклон. Рассмотрим возможные условия подробнее.

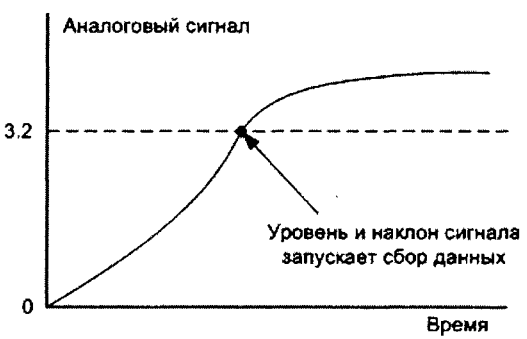


Рис. 20.1

Сбор данных начинается, когда аналоговый сигнал удовлетворяет определенным условиям, таким как достижение указанного уровня

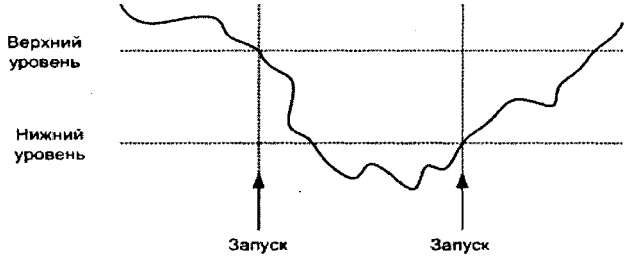


Рис. 20.2

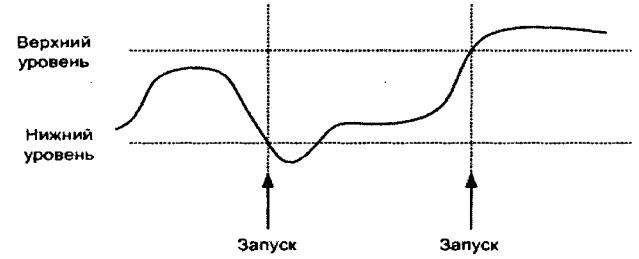


Рис. 20.3

сигнала или знака его производной. Когда измерительное устройство обнаруживает условия запуска, оно выполняет соответствующее действие, например, начинает измерения или отмечает, какая выборка отсчитывается, и т.д. На рис. 20.1 показано, что запускающий сигнал будет подан, когда измеряемый сигнал достигнет отметки 3,2. Это и инициализирует сбор данных.

Условие начала выполнения заданного действия измерительным устройством может выполняться тогда, когда аналоговый сигнал входит в интервал определенных значений напряжения или покидает его. Такие значения напряжения указываются верхним и нижним уровнями. На рис. 20.2 показаны моменты времени начала сбора данных, когда сигнал входит в указанный интервал напряжений.

На рис. 20.3 показаны моменты времени начала сбора данных, когда сигнал выходит из указанного интервала.

Запуск операций сбора или генерирования данных может начинаться по цифровому сигналу. Цифровой сигнал представляет собой сигнал, который имеет два дискретных уровня: высшая ступень (1) и низшая ступень (0). При смене уровней инициализируется сбор данных. На рис. 20.4 сбор или генерирование данных начинается после того, как цифровой сигнал переходит на низшую ступень.

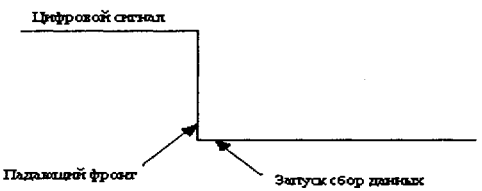


Рис. 20.4

Использование DAQmx

В палитре **DAQmx – Data Acquisition** содержатся все необходимые подпрограммы для осуществления операций аналогового и цифрового ввода-вывода и работы со счетчиками/таймерами (рис. 20.5). Многие задачи, не требующие расширенных возможностей синхронизации, могут быть выполнены с помощью экспресс-ВП **DAQmx Assistant**.

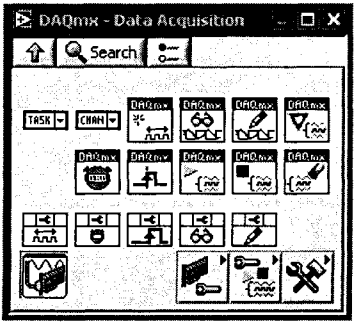


Рис. 20.5

Экспресс-ВП **DAQmx Assistant** позволяет легко и быстро решить стандартную задачу сбора данных. Такими задачами могут быть задачи аналогового измерения постоянного и переменного напряжения, температуры, сопротивления и т.д. Такими задачами может быть и аналоговый вывод, цифровой ввод/вывод и использование счетчиков. Выбор измерительной задачи достаточно широкий. Таким образом, задача измерения представляет собой просто формальное описание процесса измерений или генерации реального сигнала.

Задание 20.1. Измерение переменного напряжения с помощью экспресс-ВП DAQmx Assistant

Рассмотрите процесс измерения переменного напряжения. Добавьте на блок-диаграмму экспресс-ВП **DAQ Assistant**. При этом, как и в случае работы с любым другим экспресс-ВП, появится диалоговое окно, в котором осуществляется конфигурация задания. В процессе создания локального задания указывается необходимый тип измерения. В нашем случае нам надо выбрать аналоговый ввод **Analog Input**. На следующем шаге выберем напряжение **Voltage**. Далее надо выбрать канал, с которого будут считываться данные. Выберите физический канал DAQ устройства (например, **ai0**) и нажмите кнопку **Finish**.

Экспресс-ВП **DAQ Assistant** откроет новое диалоговое окно, при помощи которого можно установить все необходимые параметры задания. В верхней части этого диалогового окна (Рис. 20.6) настраиваются каналы и установки напряжения. В поле списка каналов Channel List можно добавлять и удалять каналы.

В нижней части этого окна настраиваются параметры выборки напряжения (рис. 20.7) и условия запуска задания (рис. 20.8).

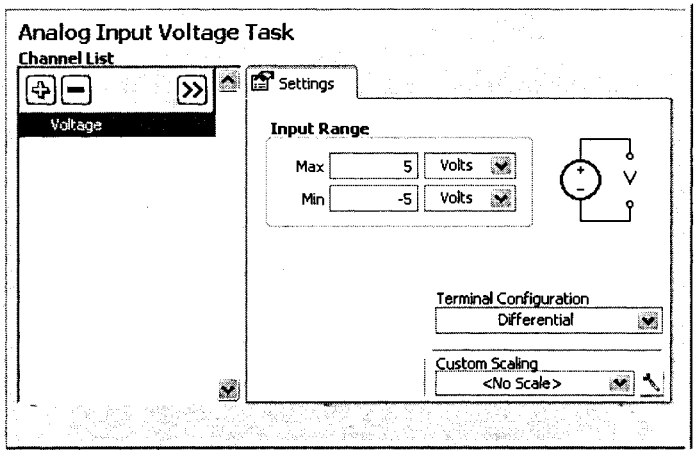


Рис. 20.6

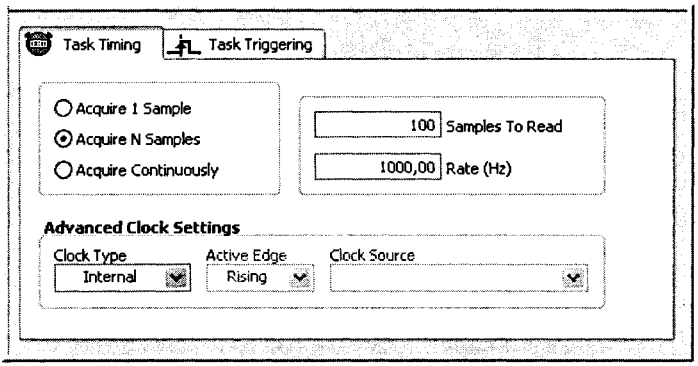


Рис. 20.7

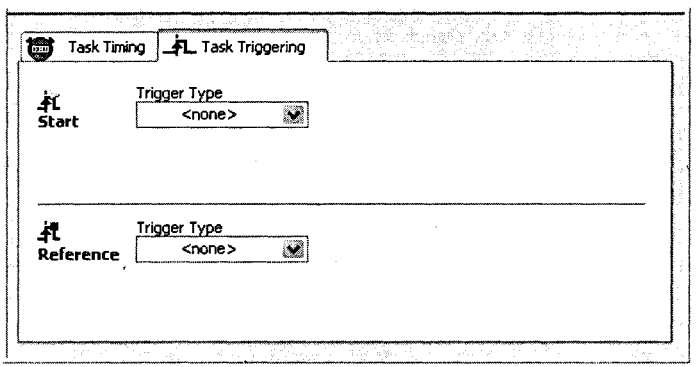


Рис. 20.8

В первой вкладке выберите производить **N** измерений **Acquire N Samples** или снимать измерения постоянно **Acquire Continuously**. Если есть необходимость, во второй вкладке выберите условия запуска по уровню (**Start**) и значению производной сигнала (**Reference**).

После завершения установки различных параметров вашего измерительного задания, в этом же окне можно посмотреть, что в результате получилось. Для этого нажмите кнопку **Test** в верхней панели. Появится еще одно диалоговое окно (рис. 20.9). Запустите измерительное задание, нажав на кнопку **Start**. На графике вы получите измеренный сигнал.

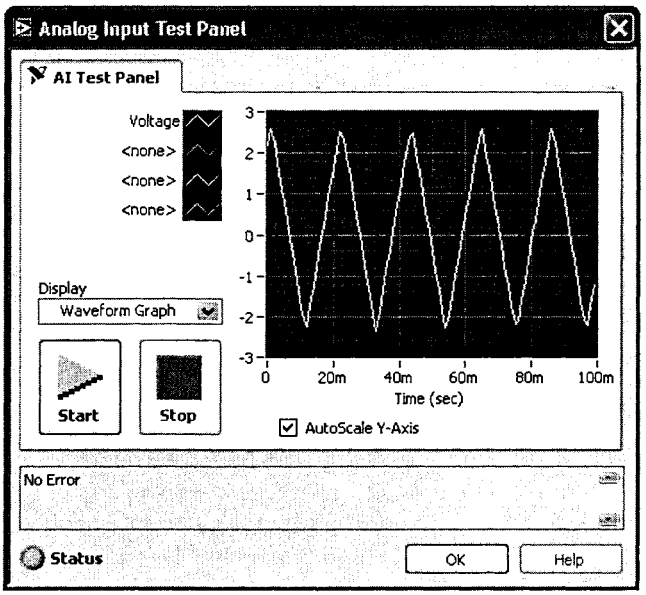


Рис. 20.9

Если в нижнем поле не указано никаких ошибок и логический элемент индикации **Status** горит зеленым цветом, а также, если вас устраивает график напряжения, завершите работу с **DAQ Assistant**.

После того, как вы несколько раз нажмете **OK** и вернетесь в окно блок-диаграммы, там появится иконка, с уже сформированными по вашему заданию входами и выходами. Добавьте на выходе с данными график, выбрав в контекстном меню **Create ⇒ Graph Indicator**. Ваш ВП по сбору и визуализации данных полностью готов. Блок-диаграмма такой программы показана на рис. 20.10.

В случае, если вашу задачу нельзя решить посредством экспресс ВП **DAQ Assistant**, собирать блок-диаграмму придется, использовав функции в палитре **DAQmx – Data Acquisition**. В табл. 20.1 представлено описание некоторых наиболее общих функций.

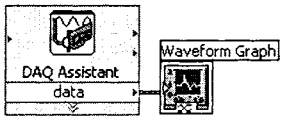


Рис. 20.10

Таблица 20.1

Иконка	Описание
	**DAQmx Create Virtual Channel** – Создает один или несколько виртуальных каналов и добавляет к ним задачу
	**DAQmx Read** – Считывает данные согласно заданию или по определенным каналам
	**DAQmx Write** – Записывает данные согласно заданию или по определенным каналам
	**DAQmx Timing** – Указывает число отсчетов. При необходимости создает буфер.
	**DAQmx Trigger** – Настраивает запуск задания
	**DAQmx Start Task** – Запускает задание
	**DAQmx Stop Task** – Останавливает задание
	**DAQmx Clear Task** – Очищает задание

В таблице используются термины **channel** (канал) и **task** (задание). Поясним, что они означают.

Понятие **NI-DAQmx** channels аналогично виртуальным каналам стандартного **NI-DAQ**. В стандартном **NI-DAQ** можно создавать виртуальные каналы, которые включают в себя совокупность настроек физического канала DAQ, типа измерений и информацию о нормировке значений. То есть понятие виртуальных каналов используется в качестве соответствия физических каналов проводимым измерениям. В стандартном **NI-DAQ** конфигурирование виртуальных каналов производилось в **MAX**. В **NI-DAQmx** настройка виртуальных каналов возможна как в **MAX**, так и в самом приложении

Задание представляет собой совокупность настроек одного или нескольких каналов, синхронизации, временных и других параметров. Задание описывает параметры операций сбора или генерирования данных, которые необходимо выполнить.

При помещении большинства функций палитры **DAQmx – Data Acquisition** на блок-диаграмму под иконкой функции появляется меню, в котором производится настройка функции (рис. 20.11). В частности для функции **DAQmx Read** в этом меню производится выбор, какой сигнал считывать и каким типом данных полученный сигнал представить. На рис. 20.12 показано, как производится выбор. В данном случае функция настраивается на дискретизацию некоторого количества точек аналогового сигнала по одному каналу. Результат будет представлен в виде осциллограммы.

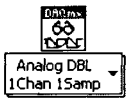


Рис. 20.11

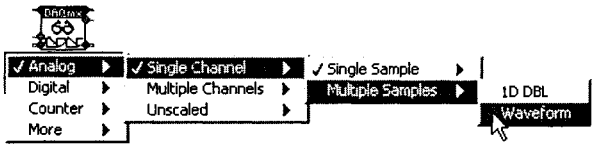


Рис. 20.12

Задание 20.2. Измерение переменного напряжения с помощью функций палитры DAQmx – Data Acquisition

Измерьте напряжение средствами **NI-DAQmx** (рис. 20.13). Получим выборку из 1000 точек так же, как это было сделано в примере 20.1.

Начать надо с создания новой задачи, за что отвечает функция **DAQmx Start task VI**. Потом на основе физического канала следует создать виртуальный канал. За это отвечает функция **DAQmx Create Virtual Channel VI**. Подайте на вход **task in** выход функции предыдущего пункта. Обязательным входом этой функции является вход **physical channels**, который определяет физический канал для сбора данных. Создав константу, выберите канал, с которого собираетесь снимать измерения. В нашем примере выбран канал **ACH0**, и он принадлежит DAQ устройству, определенному в **MAX** как **dev1**. Все остальные параметры функции оставьте по умолчанию. Под иконками функций **NI-DAQmx** располагается ниспадающее меню, в котором можно выбрать дополнительные параметры. Имеется возможность выбрать аналоговый ввод напряжения, тока, температуры, частоты и др., аналоговый вывод напряжения и тока, цифровой ввод и вывод, настроить синхронизацию. При этом в соответствии с выбранным действием корректируются и входы, выходы функции. В данном случае по умолчанию функция создания виртуального канала предназначена для снятия аналогового напряжения.

Далее с помощью функции **DAQmx Read VI** следует настроить считывание данных с указываемого виртуального канала. Поместите ее на блок-диаграмму и соедините с входами **tasks/channels** in и error соответствующие провода. В ниспадающем меню этой функции следует выбрать вариант для отсчитывания некоторого количества точек. По умолчанию эта функция считывает одно единственное значение. Чтобы выбрать дискретизацию 1000 значений напряжения, установите ее в положение **Analog ⇒ Single Channel ⇒ Multiply Samples ⇒ Waveform**. Появится дополнительный вход для числа точек в выборке **number of samples per channel**, на который и надо подать значение 1000.

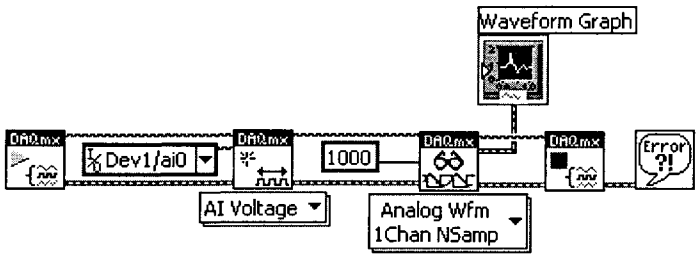


Рис. 20.13

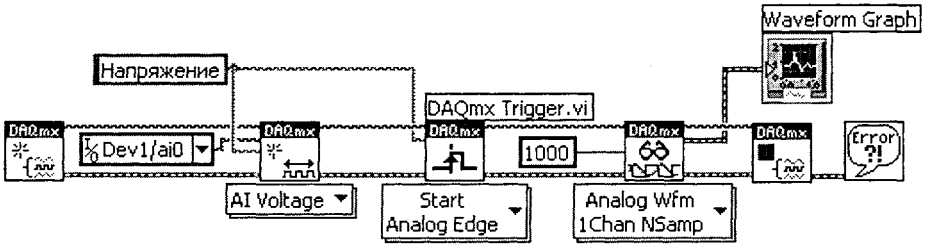


Рис. 20.14

Выведите результат на график диаграмм и завершите задачу с помощью функции **DAQmx Start task VI**. Запустите ВП. Если все сделано правильно и к соответствующему каналу DAQ-устройства подведено напряжение, на графике вы получите кривую мгновенного напряжения.

Задание 20.3. Измерение переменного напряжения с запуском по уровню и наклону сигнала

Осуществите сбор данных такие же, как и в прошлом примере, но с запуском сбора данных по уровню напряжения и его наклону.

Для этого в блок-диаграмму следует добавить функцию **DAQmx Triggering VI** (рис. 20.14). Ее необходимо поместить перед функцией считывания данных с виртуального канала и в ниспадающем меню выбрать начинать по аналоговому уровню в соответствии с условиями поставленной задачи **Start ⇒ Analog Edge**. Появятся дополнительные входы.

Входы **slope** и **level** можно оставить неподключенными, то есть использовать их значения по умолчанию. По умолчанию для включения сбора данных кривая напряжения должна возрастать и равняться нулю. Обязательным входом является вход **source**, на который надо подать имя виртуального канала. Создайте строковую постоянную и введите строку «Напряжение». Эту же постоянную надо подключить к входу **source** функции создания виртуального канала.

Выводы

Запуск сбора данных осуществляется аппаратным или программным способом. Программный способ запуска заключается в запуске приложения по сбору данных При необходимости учитывать различные условия запуска (аналоговый или цифровой запускающий сигнал, характеристики сигнала) используется аппаратный способ. Функции **NI-DAQmx** позволяют быстрее создать приложение по сбору данных. Экспресс-ВП **DAQmx Assistant** сводит все задачи по компьютерным измерениям к выбору основных параметров решаемой задачи. Для решения специфических измерительных задач применяются функции высокого уровня.

Лекция 21

Аналоговый вывод сигнала

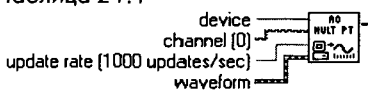
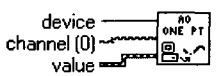
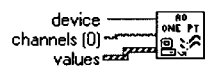
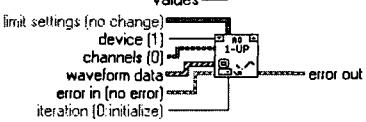
Описываются функции аналогового вывода сигналов. Рассматривается одновременный аналоговый ввод и вывод данных.

Подход к генерации сигнала в LabVIEW подобен подходу к чтению. Существуют функции высокого уровня, позволяющие быстро решить требуемую задачу. Существуют функции низкого уровня позволяющие обеспечить непрерывную генерацию сигнала и настройку всех необходимых параметров.

Для генерации сигнала в LabVIEW предусмотрены виртуальные приборы из меню **Functions ⇒ NI Measurements ⇒ Data Acquisition ⇒ Analog Output**:

В табл. 21.1 перечислены функции аналогового вывода высокого уровня.

Таблица 21.1

	***AO Generate Waveform*** *– Генерирует сигнал на один канал.*
	***AO Generate Waveforms*** *– Генерирует сигнал на несколько каналов.*
	***AO Update Channel*** *– Выводит заданную величину на один канал.*
	***AO Update Channels*** *– Выводит заданные величины на несколько каналов.*
	***AO Write One Update*** *– Выдоет единичное зночение напряжения для узозанных канолов устройства вывода.*

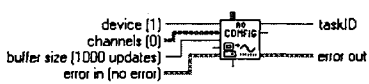
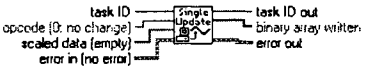
Для всех функций входными параметрами являются:

- **Device** (устройство) – номер устройство присвоенный плате (см. работу с MAX).

- **Channel** (канал) – определяет физический канал на DAQ-устройстве.
- **Update rate** – частота, с которой происходит запись данных.

В табл. 21.2 перечислены функции аналогового вывода высокого уровня.

Таблица 21.2

	***AO Config*** *– Организует задание (**task**), описывающее устройство, каналы и ограничения. Создает буфер заданного розмера для аналагового вывода.*
	***AO Single Update*** *– Осуществляет непрерывный вывод данных.*
	***AO Clear*** *– Останавливает и очищает задание аналогового вывода данных.*

Реальные нелинейные элементы в виртуальных схемах

Исследование и анализ процессов в нелинейных электрических цепях представляет собой весьма трудоемкую задачу как ввиду сложности и многообразия самих процессов, так и из-за трудности описания характеристик нелинейных элементов, поскольку эти характеристики зависят от множества факторов, учесть всю совокупность которых невозможно. Единственным способом получить надежные данные является эксперимент. Однако, здесь режимы работы зависят от параметров цепи, от частот, от начальных условий – поэтому экспериментальное исследование также оказывается чрезвычайно трудоемким.

Среда LabVIEW дает возможность преодолеть эти трудности тем, что позволяет исследовать реальные нелинейные элементы в полностью виртуальной среде: Целесообразность и перспективность такого подхода определяется тем, что расчетные характеристики нелинейных элементов, как правило, описываются весьма приближенно, многие их свойства, относящиеся главным образом к изменению режима или к динамическим особенностям процесса, не учитываются. Работу реального нелинейного элемента можно организовать таким образом, что источник реального сигнала создается при помощи системы LabVIEW, линейная часть схемы моделируется в среде LabVIEW, а наблюдение, измерение и обработка выходного сигнала осуществляются виртуальными приборами LabVIEW. Преимущества такого подхода очевидны: не представляет труда составить любую виртуальную схему и проанализировать на практике работу исследуемого устройства при реальных параметрах схемы. Следует, правда, отметить ограничение, – это напряжение до 5 В и малая мощность выходного сигнала, подаваемого на исследуемый элемент. Поэтому пока рассмотрена работа лишь таких виртуальных схем, в которых возможно проявление нелинейных свойств элементов при малых напряжениях и малых мощностях. Следует упомянуть и ограничения по частоте.

При включении реальных нелинейных элементов в виртуальную схему следует отметить тот факт, что в схеме LabVIEW элемент действует не как физический объект, а как его математическая модель, в нашем случае – вольтамперная характеристика. Поэтому последовательно с нелинейным элементом, на котором измеря-

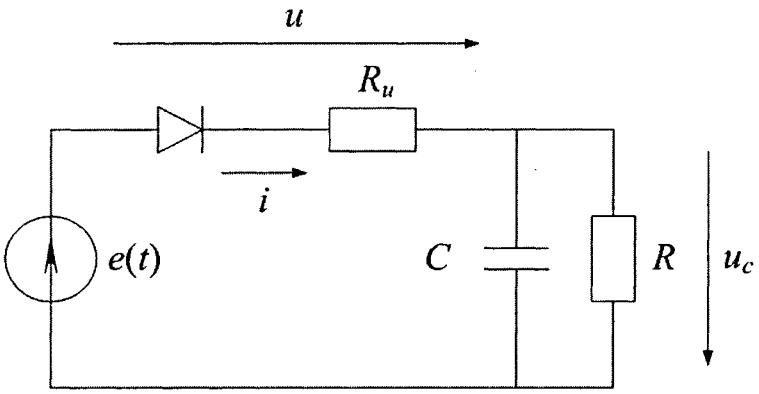


Рис. 21.1

ется напряжение, должен быть включен линейный резистор, позволяющий определить ток. Таким образом, входное устройство, соединяющее виртуальный прибор должно иметь как правило два входа (в самых простых схемах возможен один вход).

Задание 21.1. Исследование работы выпрямителя

Рассмотрим схему преобразователя переменного напряжения в постоянное – выпрямителя с фильтрацией выходного напряжения. Схема выпрямителя изображена на рис. 21.1. Питание цепи осуществляется от источника синусоидальной электродвижущей силы $e = E_m \sin\omega t$, нагрузкой является резистор R. В качестве фильтрующего элемента используется конденсатор С. Значения E_m, ω, R, C выбираются самостоятельно. Требуется построить зависимость выходного напряжения от времени.

В схеме будем использовать реальный диод и виртуальные источник синусоидальной электродвижущей силы, нагрузка и конденсатор фильтра. Поскольку в виртуальной схеме диод представлен своей вольтамперной характеристикой, необходимо последовательно с диодом включить измерительный резистор $R_и$ с малым сопротивлением, напряжение на котором пропорционально току диода.

Запишем основные уравнения. Напряжение нагрузки определяется дифференциальным уравнением:

$$C\frac{du_c(t)}{dt}+\frac{1}{R}u_c(t)=i(t) \qquad (1)$$

напряжение на диоде и измерительном резисторе

$$u(t)=e(t)-u_c(t) \qquad (2)$$

Численное интегрирование уравнения (1) проводится по явному методу Эйлера

$$u_c(t_{k+1})=u_c(t_k)+\frac{1}{C}(i(t_k)-\frac{1}{R}u_c(t_k))\Delta t \qquad (3)$$

где k – номер шага интегрирования.

Реальная часть схемы состоит из диода и измерительного резистора. Виртуальная часть схемы задается уравнениями (1-3), связывающими токи и напряжения на различных участках цепи. На каждом шаге интегрирования анализ уравнений (1-3) позволяет определить текущее напряжение ветви реальной части схемы, которое будем устанавливать с помощью функций аналогового вывода. А измеренное с помощью измерительного резистора значение тока будем использовать для расчета напряжения на конденсаторе и напряжения ветви с диодом и измерительным резистором в следующем шаге.

Итак, поместите на блок-диаграмму цикл **For**, задайте число итераций, например 500.

Добавьте формулы для расчета виртуальной схемы. Для этого удобнее всего воспользоваться структурой **Structures ⇒ Formula Node**. Поместите формульный узел на блок-диаграмму цикла. Добавьте вход, для чего в контекстном меню структуры выберите **Add Input**. Запишите имя переменной, например *uc*. Добавьте входы для переменных *Em, C, R, Ri, omega, t, dt, ui*. Все переменные следует записывать латиницей. Переменные *Em, C, R, Ri, omega* задаются пользователем. Создайте для каждой их них элемент управления. Учтите, что переменная *Ri* относится к реальному резистору, сопротивление которого должно быть известно. Для переменной dt создайте константу, равную, например, 0,1 мс. Текущее время *t* будет рассчитываться при умножении *dt* на номер итерации, поданной с терминала счетчика итераций. Переменная *uc* в (3) записана как $u_c(t_k)$. То есть это значение напряжения на емкости на прошлом шаге интегрирования. Поэтому значение для этой переменной следует подавать со сдвигового регистра. Подключите ко всем входам соответствующие провода. Переменная ui соответствует напряжению, измеренному на измерительном резисторе. О ее подключении речь пойдет ниже. В сам формульный узел запишите следующие уравнения:

```
uc_new = uc + (ui/Ri - (uc/R)) * dt/C;
e = Em * sin(5 * t);
udr = e - uc;
```

Точка с запятой используется для корректного окончания выражения в формульном узле. Ознакомиться с синтаксисом, используемым в формульном узле, можно в системе справки LabVIEW.

С правой стороны формульного узла добавьте три выхода. Выход создается аналогично входу, только в этом случае следует выбирать **Add Output**. Назовите переменные на выходе так, как они записаны внутри формульного узла: *uc_new, e, udr*. Первая переменная *uc_new* относится к новому вычисленному значению напряжения на емкости, которое в (3) записана как $u_c(t_{k+1})$. Выход uc_new соедините со сдвиговым регистром, чтобы передать новое значение в следующую итерацию цикла. Вторая переменная e соответствует напряжению источника синусоидальной электродвижущей силы в данный момент времени. Третья переменная *udr* определяет напряжение на реальном участке цепи (диоде и измерительном резисторе), которое устанавливается с помощью функций аналогового вывода. Все три выхода выведите на *XY* график, где для значений по оси абсцисс используйте текущее время *t*.

На блок-диаграмму добавьте функции **NI Measurements ⇒ Data Acquisition ⇒ Analog Input ⇒ AI Sample Channel.vi** и **NI Measurements ⇒ Data Acquisition ⇒ Analog Output ⇒ AO Update Channel.vi**. Первая функция однократно отсчитывает аналоговое напряжение с выбранного канала. Вторая однократно устанавливает напряжение выбранного канала. В соответствии с заданием первую функцию используйте для измерения напряжения на измерительном резисторе, а вторую для установления напряжения на последовательном соединении диода и резистора. По умолчанию выход **Sample** функции **AI Sample Channel.vi** обладает типом **Waveform** (сигнал). Измените его в контекстном меню поля вывода **Select Type ⇒ Scaled Value**. После этого выход **Sample** функции **AI Sample Channel.vi** соедините со входом **ui** формульного узла. Обеим функциям необходимо указать канал (аналогового ввода и аналогового вывода), к которому вы подключили реальную электрическую цепь. Вход **device** у этих функций можно оставить неподключенными, если используемое DAQ-устройство имеет номер 1, которое используется по умолчанию.

Таким образом, мы имеем, что в каждой итерации цикла с канала аналогового ввода считывается напряжение (ток) измерительного резистора, рассчитывается напряжение на виртуальном конденсаторе. Через рассчитанное напряжение на конденсаторе и напряжение на источнике определяется напряжение на реальных диоде и измерительном резисторе. Оно и устанавливается с помощью функций аналогового вывода.

Заключительным шагом по сборке блок-диаграммы установите на канале аналогового вывода напряжение равным нулю до и после работы цикла. До работы цикла необходимо установить нулевое напряжение для того, чтобы при последующей операции считывания напряжение на измерительном резисторе (а эта операция в нашем случае выполняется первой) было равным нулю. При этом следует учесть, что операция аналогового вывода должна выполняться строго до начала работы цикла. Поэтому на блок-диаграмме используйте структуру последовательности. Для корректного завершения работы цикла также необходимо установить на канале аналогового вывода напряжение равным нулю.

Сверьте составленную блок-диаграмму с блок-диаграммой, показанной на рис. 21.2. Запустите ВП. Установите значения E_m, C, R, ω равными 8 В, 2 10^{-6} Ф, 5000 Ом, 314 1/с соответственно. Кривые напряжений источника, конденсатора и последовательного соединения диода и резистора должны выглядеть так, как показано на рис. 21.3.

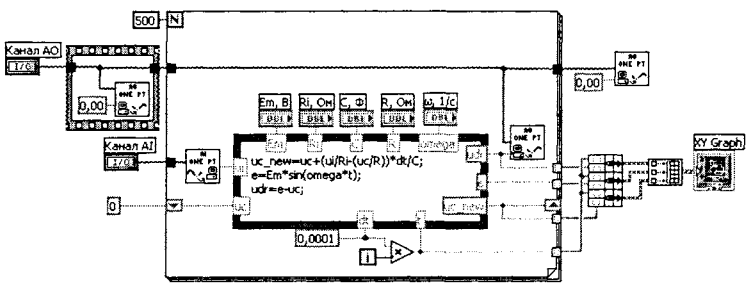


Рис. 21.2

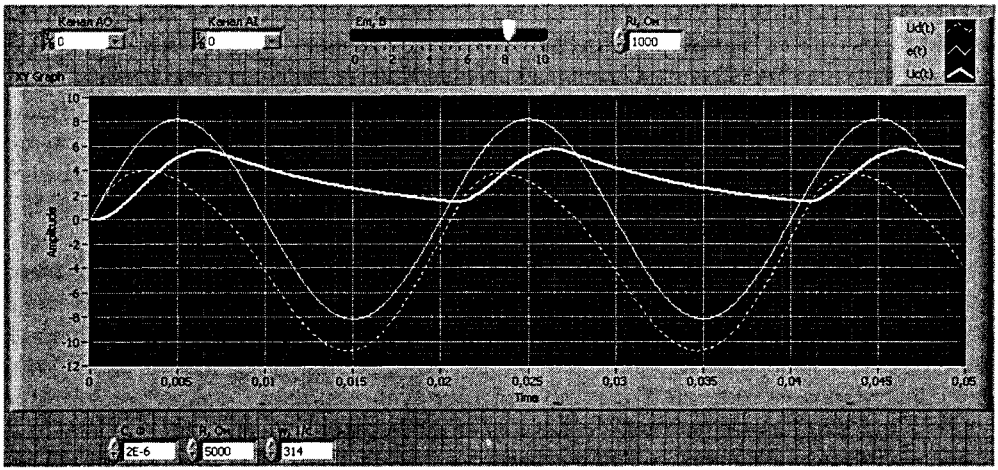


Рис. 21.3

Пример 21.2. Исследование работы выпрямителя в реальном времени

Рассмотрим, как программу задания 21.1 можно усовершенствовать (рис. 21.4).

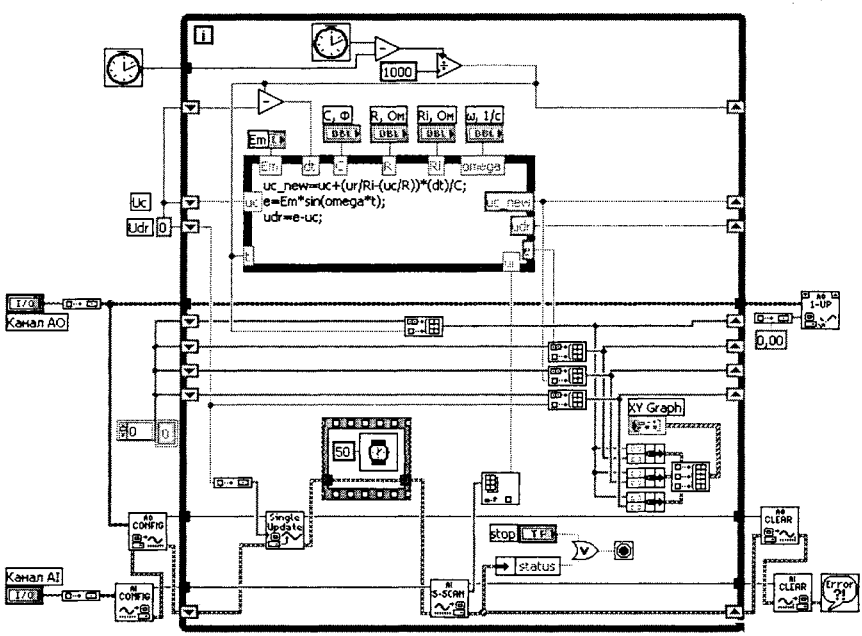


Рис. 21.4

Поскольку динамическая ВАХ реального нелинейного элемента может отличаться от статической (то есть зависит от частоты питающего напряжения), в примере добавлен расчет времени t и dt. Функция **Tick Count (ms)** выдает время в миллисекундах. Это накладывает некоторые ограничения по расчету времени в LabVIEW 7.0. В частности в данном примере именно этим обусловлено введение задержки времени 10 мс. В результате, чтобы рассматривать достаточное число точек на период, необходимо уменьшить частоту источника синусоидальной электродвижущей силы. Это ограничение при исследовании характера изменения мгновенных значений токов и напряжений в схемах с нелинейными элементами может быть некритичным.

Еще одним нововведением является использование ряда функций аналогового ввода и вывода низкого уровня. В любой функции высокого уровня, которая в принципе является ВП, уже содержится ряд функций низкого уровня. ВП высокого уровня имеют открытую структуру, поэтому можно открыть и изучить блок-диаграмму. В ВП **AI Sample Channel.vi** и **AO Update Channel.vi** кроме функций непосредственно считывания и записи имеются функции настройки задания **AI Config.vi** и **AO Config.vi**, которые описывают измерительную задачу. Их нет никакой необходимости вызывать в каждой итерации цикла. На рис. 21.4. видно, что функции **AI Config.vi** и **AO Config.vi**, а также функции очистки задания по завершению программы **AI Clear.vi** и **AO Clear.vi** вынесены за пределы цикла. Внутрь цикла помещены функции работы непосредственно с DAQ-устройством для однократного ввода и вывода **AI Single Scan.vi** и **AO Single Update.vi**. Для наглядности на рис. 21.4 иконки функций аналогового ввода и аналогового вывода расположены параллельно, образуя две цепочки.

Для жесткого задания последовательности выполнения функций аналогового ввода и вывода использована потоковая структура выполнения ВП. В частности, входы и выходы кластеров ошибок соединены в нужной последовательности выполнения функций. Как только возникает ошибки, все последующие ВП не выполняются (см. лекцию 11). Кроме этого теперь вместо цикла **For** используется цикл **While**, одно из условий работы которого это отсутствие каких-либо ошибок при работе функций аналогового ввода и вывода. Завершение работы к тому же происходит при нажатии на кнопку **Stop**.

Изучение процессов, проистекающих в схеме рис. 21.1., в реальном времени дополняется возможностью наблюдать построение графиков мгновенных напряжений в процессе работы цикла. Это обеспечивается тем, что на каждой итерации в массивы интересующих нас величин ($u_c(t)$, $e(t)$, $u_{dr}(t)$ и t), которые передается через сдвиговые регистры, добавляются новые значения напряжений и времени.

Рядом со сдвиговым регистром, передающим текущее значение $u_c(t)$, помещен еще один для $u_{dr}(t)$. Он нужен для обеспечения правильного порядка: вначале вывода напряжения на реальный участок и по истечению 10 мс измерения напряжения измерительного резистора. Рассчитанное значение напряжения на диоде и измерительном резисторе выводится на канал аналогового вывода не в текущей итерации, а в следующей.

При запуске ВП, блок-диаграмма которого изображена на рис. 21.4, рекомендуется взять следующие значения E_m, C, R, ω: 8 В, 5 10^{-4} Ф, 5000 Ом, 1 1/с соответственно.

Выводы

Получение достаточно простыми средствами экспериментальных характеристик нелинейных цепей, с возможностью вариации параметров в широком диапазоне и с параллельной математической обработкой результатов эксперимента, полезно как в научных исследованиях, так и в учебном процессе. Для решения задач управления реальными объектами необходимо освоить непрерывный и одновременный ввод и вывод данных.

Лекция 22

NI ELVIS

Рассматривается новый универсальный комплекс приборов NI ELVIS для создания образовательных лабораторий в вузах. Описывается его комплектация, основные возможности.

NI ELVIS (National Instruments Educational Laboratory Virtual Instrumentation Suite) представляет полный комплект технических устройств и программного обеспечения для проведения лабораторных работ практически любого типа. Комплектация NI ELVIS представлена на рис. 22.1.

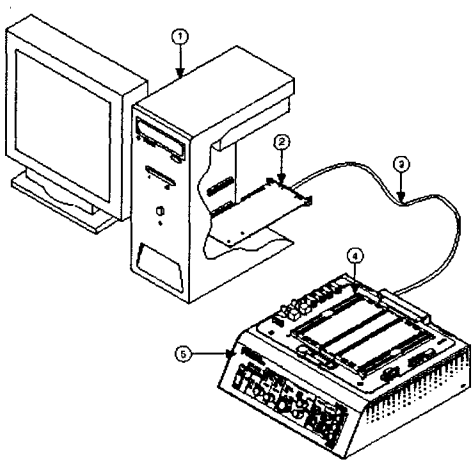


Рис. 22.1

1. *Компьютер, на котором запущен LabVIEW.*
2. *DAQ устройство.*
3. *Кабель 68-Pin Series.*
4. *Монтажная панель NI ELVIS.*
5. *Настольная станция NI ELVIS.*

В NI ELVIS входит многофункциональное DAQ-устройство, настольная станция, монтажная плата, а также набор виртуальных приборов, разработанных в среде LabVIEW и выполняющих функции различных измерительных приборов и устройств, используемых в обычных лабораториях вузов. Рассмотрим каждую составляющую подробнее.

DAQ-устройство

NI ELVIS спроектирован для работы с DAQ-устройствами National Instruments. DAQ-устройства National Instruments высокопроизводительны, многофункциональны. Они представляют собой аналоговые, цифровые и синхронизирующие устройства ввода-вывода, подключаемые к PCI-слоту компьютера. Для использования NI ELVIS установленное в компьютере DAQ-устройство должно поддерживать следующую минимальную конфигурацию:

- 16 каналов ввода аналоговых сигналов;
- 2 канала вывода аналоговых сигналов;
- 8 цифровых линий ввода-вывода;
- Два счетчика/таймера.

Настольная станция NI ELVIS

Настольная станция NI ELVIS обеспечивает соединение DAQ устройства с монтажной панелью. В нее встроены источники постоянного напряжения ±15 В и +5 В, регулируемые источники питания и генератор функций (синусоидальной, прямоугольной, треугольной). Кроме всего прочего настольная станция оснащена системой защиты DAQ-устройства от возможных повреждений в результате лабораторных ошибок. Работа с настольной станцией осуществляется через панель управления, которая показана на рис. 22.2.

Остановимся подробнее на некоторых объектах панели управления настольной станции. NI ELVIS взаимодействует с компьютером через восемь цифровых линий ввода-вывода DAQ устройства. Переключатель **Communications** управляет разводкой цифрового ввода-вывода к NI ELVIS. Во время обычной работы переключатель находится в положении **Normal**. В этом случае сигналы с цифрового ввода-вывода направляются на аппаратное обеспечение NI ELVIS, позволяя управлять им через программное обеспечение. Когда переключатель **Communications** установлен в положение **Bypass**, электрическая цепь NI ELVIS шунтируется. Что позволяет работать непосредственно с DAQ-устройством. В этом режиме программное обеспечение не работает. И при попытке воспользоваться виртуальными приборами NI ELVIS, на экран выводится сообщение о том, что переключатель находится в режиме транзитной передачи. Некоторые элементы управления виртуальных приборов к тому же затемнены и неактивны.

Переключатель **Manual**, имеющийся среди элементов управления регулируемых источников питания и генератора функций, устанавливает управление в ручной

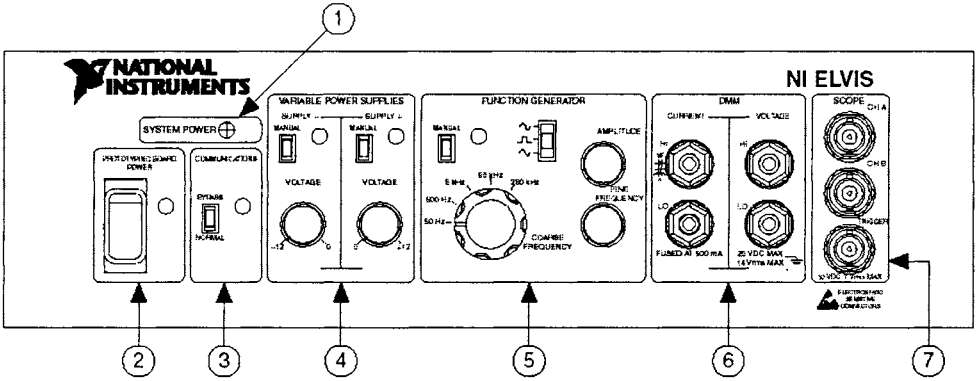


Рис. 22.2

1. *Светодиод системного питания;*
2. *Выключатель монтажной панели;*
3. *Переключатель режима работы настольной станции;*
4. *Элементы управления регулируемых источников питания;*
5. *Элементы управления генератора функций;*
6. *Разъемы цифрового мультиметра;*
7. *Разъемы осциллографа.*

или программный режим. В ручном режиме работы управление источниками и генератором осуществляется через панель управления настольной станции. В программном режиме управление осуществляется через соответствующие виртуальные приборы (речь о них пойдет в следующей лекции). При работе настольной станции в режиме транзитной передачи генератор функций и регулируемые источники питания доступны только в ручном режиме управления.

Элементы управления генератора функций кроме ручки регулирования амплитуды (максимальное значение амплитуды сигнала может составлять 2,5В) и выбора формы сигнала содержат ручку установки уровня требуемой частоты (**COARSE FREQUENCY**) и ручку плавной регулировки частоты (**FINE FREQUENCY**).

Разъемы цифрового мультиметра **CURRENT HI** и **LO**, **VOLTAGE HI** и **LO** типа «банан» и BNC разъемы осциллографа **CH A**, **CH B** и **TRIGGER** соответствуют выводам с теми же названиями на монтажной панели. Поэтому при использовании разъемов цифрового мультиметра и осциллографа будьте осторожны. Их нельзя подключать одновременно к различным сигналам. Это может повлечь за собой повреждение элементов схемы на монтажной панели.

Защитная панель NI ELVIS предназначена для защиты от коротких замыканий и превышения напряжения. Она сконструирована таким образом, чтобы ее можно было легко вынимать и заменять необходимые предохранители на новые. Все предохранители отвечают стандартным требованием. В связи с чем нет необходимости отправлять защитную панель для ремонта в сервисный центр National Instruments.

Монтажная панель NI ELVIS

Монтажная панель (рис. 22.3) подключается к настольной станции через стандартный PCI-слот. Поэтому вы можете спроектировать свою панель и работать с ней на настольной станции.

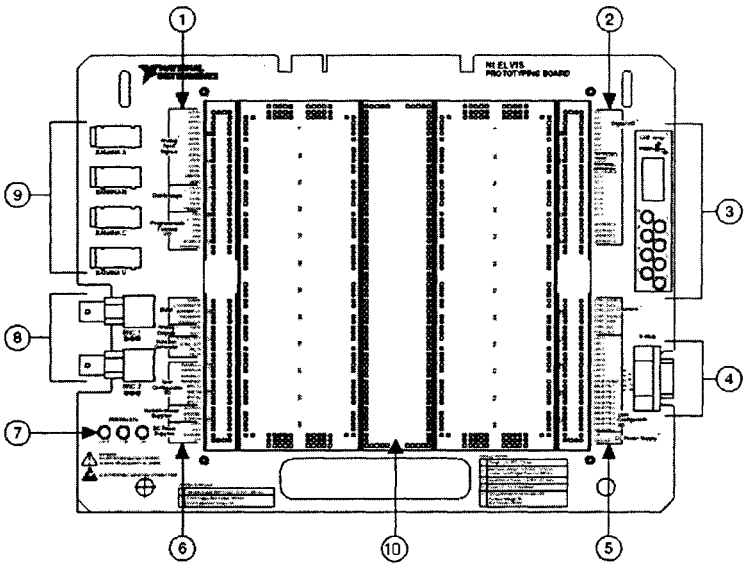


Рис. 22.3

1. *Аналоговый ввод, вводы осциллографа, а также ввод-вывод программируемых функций;*
2. *Цифровой ввод-вывод;*
3. *Набор световых индикаторов;*
4. *Разъем D-SUB;*
5. *Счетчик/таймер, пользовательский ввод-вывод, вывод источника постоянного напряжения;*
6. *Вводы цифрового мультиметра, аналоговый вывод, генератор функций, пользовательский ввод-вывод, выводы регулируемых источников питания, выводы источников постоянного напряжения;*
7. *Светодиоды питания;*
8. *BNC разъемы;*
9. *Разъемы типа «банан»;*
10. *Макетная панель.*

Предполагается, что выполняющий лабораторную работу на макетной панели NI ELVIS собирает аналоговую или цифровую электрическую цепь. Собранную цепь он может подключать к аналоговым или цифровым каналам ввода-вывода

DAQ-устройства, к встроенным в настольную станцию источникам постоянного напряжения, регулируемым источникам питания и к генератору функций. Кроме этого к собранной на макетной панели электрической цепи через разъемы типа «банан», BNC разъемы и разъем D-Sub можно подключать и внешние сигналы, измерительные приборы, источники энергии и т.п. Следует отметить, что сама аналоговая или цифровая электрическая цепь может быть собрана и не на макетной панели. Это в ряде случаев может оказаться исключительно полезным. Для ее подключения к монтажной панели NI ELVIS можно воспользоваться уже упомянутыми разъемами.

Сама макетная панель представляет собой набор выводов, объединенных в горизонтальные полосы по четыре и пять выводов и вертикальные полосы по 25 выводов. Выводы каждой такой полосы соединены между собой, что позволяет набирать схемы, образуя из них узлы. Горизонтальные полосы по четыре вывода в каждой слева и справа макетной панели уже подключены к различным каналам DAQ-устройства, устройствам и разъемам монтажной панели. Рядом нанесено обозначение к каждому такому выводу. Остальные выводы вы можете использовать по своему усмотрению. В табл. 22.1 представлено описание к обозначениям и выводам монтажной панели NI ELVIS. Выводы сгруппированы по своему функциональному назначению.

Таблица 22.1

Обозначение	Тип	Описание
ACH 0..5+	*Общий аналоговый ввод*	***Analog Input Channel 0..5 (+)*** *– Положительный дифференциальный ввод для каналов с 0 по 5 аналогового ввода*
ACH 0..5-	*Общий аналоговый ввод*	***Analog Input Channel 0..5 (-)*** *– Отрицательный дифференциальный ввод для каналов с 0 по 5 аналогового ввода*
AISENSE	*Общий аналоговый ввод*	***Analog Input Sense*** *– Нулевая точка для аналоговых каналов в режиме* ***NRSE.***
AIGND	*Общий аналоговый ввод*	***Analog Input Ground*** *– Земля для DAQ устройства. Это точка никак не связана с землей NI ELVIS.*
CH A..B+	*Осциллограф*	***Oscilloscope Channels A, B (+)*** *– Положительный вывод каналов осциллографа*
CH A..B-	*Осциллограф*	***Oscilloscope Channels A, B (-)*** *– отрицательный ввод каналов осциллографа*
TRIGGER	*Осциллограф*	***Oscilloscope Trigger*** *– Ввод пускового сигнала для осциллографа относительно* ***AIGND***
PFI 1..2	*Программируемая функция*	***Progrommable Function Input (PFI) 1..2, 5..7*** *–*
PFI 5..7	*ввода-вывода*	*программируемая функция ввода-вывода DAQ-устройства.*
SCANCLK	*Программируемая функция ввода-вывода*	***Scan Clock*** *– ввод подсоединен к штекеру* ***SCANCLK*** *DAQ-устройство.*
RESERVED	*Программируемая функция ввода-вывода*	*Ввод подсоединен к штекеру* ***EXTSTROBE**** *DAQ-устройства.*

Таблица 22.1 (продолжение)

Обозначение	Тип	Описание
3-WIRE	*Цифровой мультиметр*	***Three Wire** – Источник напряжения для трехпроводных транзисторных измерений*
CURRENT HI	*Цифровой мультиметр*	***Positive Current** – положительный ввод для всех измерений (кроме измерения напряжения) цифрового мультиметро*
CURRENT LO	*Цифровой мультиметр*	***Negative Current** – отрицательный ввод для всех измерений (кроме измерения напряжения) цифрового мультиметра*
VOLTAGE HI	*Цифровой мультиметр*	***Positive Voltage** – положительный ввод для вольтметро цифрового мультиметро*
VOLTAGE LO	*Цифровой мультиметр*	***Negative Voltage** – отрицательный ввод для вольтметро цифрового мультиметра*
DAC 0..1	*Общий аналоговый вывод*	***Analog Channel Output 0..1** – выводы для ЦАП DAQ-устройство*
FUNC_OUT	*Генератор функций*	***Function Output** – вывод генератоpo функций*
SYNC_OUT	*Генератор функций*	***Synchronization Output** – вывод сигнола тронзисторно-тронзисторной логики той же чостоты, что и сигнал вывода **FUNC_OUT***
AM_IN	*Генератор функций*	***Amplitude Modulation Input** – ввод для амплитудного модулятоpo генератоpo функций.*
FM_IN	*Генератор функций*	***Frequency Modulation Input** – ввад для частотного модулятоpo генератара функций.*
BANANA A..D	*Пользовательская настройка вводо-выводо*	***Banana Jacks A..D** – Выводы подключены к розъемам типа «банан» на монтажной панели*
BNC 1..2+	*Пользовательская настройка ввода-вывода*	***BNC Connectors 1..2 (+)** – Выводы подключены к BNC-разъемам на монтажной панели*
BNC 1..2-	*Пользовательская настройка ввода-вывода*	***BNC Connectors 1..2 (-)** – Выводы подключены к BNC-разъемам на монтажной панели*
SUPPLY+	*Регулируемые источники питания*	***Positive** – Вывод от 0 до 12 В регулируемого источника питания.*
SUPPLY-	*Регулируемые источники питания*	***Negative** – Вывод от -12 до 0 В регулируемого источника питания.*
GROUND	*Регулируемые источники питания, источники питания постоянного тока*	***Ground** – Заземление монтажной панели. Обе точки заземления соединены вместе.*
+15 V	*Источники питания постоянного тока*	***+15 V Source** – Вывод фиксированных +15 В источника питания относительно **NI ELVIS GROUND***
-15 V	*Источники питания постоянного тока*	***-15 V Source** – Вывод фиксированных -15 В источника питания относительно **NI ELVIS GROUND***
+5 V	*Источники питания постоянного тока*	***+5 V Source** – Вывод фиксировонных +5 В источника питания относительно **NI ELVIS GROUND***
DO 0..7	*Цифровой ввод-вывод*	***Digital Output Lines 0..7** – Выводы шины записи.*
WR ENABLE	*Цифровой ввод-вывод*	***Write Enable** – Вывод, через который сигнализируется об осуществлении записи на шину записи.*

Таблица 22.1 (окончание)

Обозначение	Тип	Описание
LATCH	*Цифровой ввод-вывод*	***Latch** – Вывод, но который подоется импульс, после того кок донные но шине зописи готовы.*
GLB RESET	*Цифровой ввод-вывод*	***Global Reset** – Вывод, через который сигнолизируется об общем сбросе цифровых конолов.*
RD ENABLE	*Цифровой ввод-вывод*	***Read Enable** – Вывод, через который сигнолизируется об осуществления считывония с шины считывония.*
DI 0..7	*Цифровой ввод-вывод*	***Digital Input Lines 0..7** – Выводы шины считывония.*
ADDRES 0..3	*Цифровой ввод-вывод*	***Address Lines 0..3** – Выводы адресной шины.*
CTR_SOURCE	*Счетчики*	***Counter 0 Source** – Вывод подключен к штекеру **GPCT0_SOURCE** DAQ-устройство.*
CTR0_GATE	*Счетчики*	***Counter 0 Gate** – Вывод подключен к штекеру **GPCT0_GATE** DAQ-устройства.*
CTR0_OUT	*Счетчики*	***Counter 0 Output** – Вывод подключен к штекеру **GPCT0_OUT** DAQ-устройства.*
CTR1_GATE	*Счетчики*	***Counter 1 Gate** – Вывод подключен к штекеру **GPCT1_GATE** DAQ-устройства.*
CTR1_OUT	*Счетчики*	***Counter 1 Output** – Вывод подключен к штекеру **GPCT1_OUT** DAQ-устройства.*
FREQ_OUT	*Счетчики*	***Frequency Output** – Вывод подключен к штекеру **FREQ_OUT** DAQ-устройства.*
LED 0..7	*Пользовательская ностройка ввода-вывода*	***LED 0..7** – Вывод световых индикаторов.*
DSUB SHIELD	*Пользовательская настройка ввода-вывода*	***D-Sub Shield** – Подключение к **D-Sub Shield**.*
DSUB PIN 1..9	*Пользовательскоя настройка ввода-вывода*	***D-Sub 1..9** – Подключение к штекером **D-Sub**.*

Монтажная панель NI ELVIS имеет 6 дифференциальных каналов ACH0..5. Эти каналы подключены непосредственно к вводам DAQ-устройства. В табл. 22.2 показано их подключение.

Таблица 22.2

NI ELVIS	DAQ-устройство	NI ELVIS	DAQ-устройство
ACH0+	*ACH0*	*ACH3-*	*ACH11*
ACH0-	*ACH8*	*ACH4+*	*ACH4*
ACH1+	*ACH1*	*ACH4-*	*ACH12*
ACH1-	*ACH9*	*ACH5+*	*ACH5*
ACH2+	*ACH2*	*ACH5-*	*ACH13*
ACH2-	*ACH10*	*AISENSE*	*AISENSE*
ACH3+	*ACH3*	*AIGND*	*AIGND*

Следует учитывать, что некоторые каналы аналогового ввода используются несколькими приборами. Поэтому их нельзя включать одновременно. К каналам **ACH<0..2>** виртуальные приборы не подключены, с ними можно работать без помех. Канал **ACH5** используется цифровым мультиметром при анализе входного сопротивления (например, измерение емкости, тест диода и т.д.). При работе с осциллографом от каналов **ACH3** и **ACH4** отключите какие-либо соединения, их использует виртуальный прибор осциллограф. В любом случае при одновременном использовании одного и того же канала на экран выводится сообщение о том, что продолжение работы какого-либо виртуального прибора невозможно.

Для подключения цифрового мультиметра на монтажной плате предусмотрены вывода тока и напряжения, также имеется и дополнительный терминал для трехпроводных транзисторных измерений (табл. 22.1). Дифференциальные входы вольтметра обозначаются **VOLTAGE HI** и **VOLTAGE LO**. Все остальные возможности цифрового мультиметра доступны через вводы **CURRENT HI** и **CURRENT LO**. Вывод **3-WIRE** одновременно с **CURRENT HI** и **CURRENT LO** используется для трехпроводных измерений.

Для подключения осциллографа на монтажной плате предусмотрены выводы **CH<A..B>+**, **CH<A..B>-** и **TRIGGER** (табл. 22.1). Во время работы виртуального прибора осциллографа они подключены к **ACH 3** и **ACH 4** соответственно.

NI ELVIS обеспечивает доступ к двум ЦАП DAQ-устройства. К ним относят каналы **DAC0** и **DAC1**. Эти каналы используются NI ELVIS для формирования произвольных сигналов. Кроме генератора произвольных сигналов NI ELVIS (о котором речь пойдет в следующей главе) к указанным каналам обращаются и другие виртуальные приборы, например, цифровой мультиметр и генератор функций. В случае одновременного использования каналов выводится сообщение о том, что возможен конфликт ресурсов.

Доступ к генератору сигналов на монтажной панели обеспечивается не только выводом **FUNC_OUT**, но и некоторыми дополнительными терминалами. **SYNC_OUT** выводит сигнал синхронизации той же частоты, что и выходной сигнал. Выводы **AM_IN** и **FM_IN** относятся к амплитудной и частотной модуляции соответственно. Программы для амплитудной модуляции управляются через канал **DAC0**, а для частотной – **DAC1**.

Регулируемые источники питания обеспечивают настраиваемые выходные напряжения от 0 до +12 В на выводе **SUPPLY+** и от -12 до 0 В на выводе **SUPPLY-**.

Схема цифрового ввода-вывода NI ELVIS состоит из 8-битной шины считывания и записи, четырех управляющих вводов и 4-битной адресной шины. Их назначение представлено в табл. 22.1.

Монтажная панель обеспечивает доступ к счетчику/таймеру DAQ-устройства. Имеется возможность управлять ими и через программное обеспечение. Назначение выводов представлено в табл. 22.1.

На монтажной панели есть некоторые разъемы и выводы, предназначенные для решения каких-либо дополнительных задач. Имеются четыре разъема типа «банан», два BNC разъема, а также разъем D-Sub. Каждый штекер разъема подключен к полосе выводов на макетной панели.

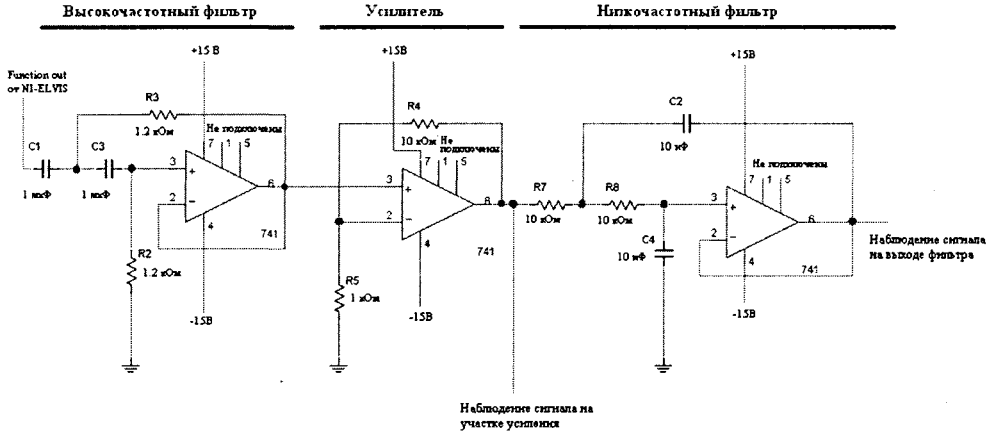


Рис. 22.4

Восемь световых индикаторов предназначены для цифрового вывода. Анод каждого светодиода соединен к выводу на макетной плате через резистор 220 Ом. Катод каждого светодиода заземлен.

Задание 22.1. Полосовой фильтр

В качестве примера рассмотрим лабораторную работу, целью которой является исследование характеристик полосового фильтра*, выполненного на операционных усилителях. Принципиальная электрическая схема полосового фильтра представлена на рис. 22.4.

Полосовой фильтр состоит из трех частей: высокочастотного фильтра, усилителя, низкочастотного фильтра. Граничная частота высокочастотного фильтра:

$$f_o = \frac{1}{2\pi\sqrt{R_2R_3C_1C_3}} = \frac{1}{2\pi\sqrt{1.2\cdot10^3\cdot1.2\cdot10^3\cdot1\cdot10^{-6}\cdot1\cdot10^{-6}}} \approx 132.6\text{Гц}.$$

Коэффициент усиления каскада усиления:

$$U_{out} = \left(1 + \frac{R_4}{R_5}\right) U_{in} = \left(1 + \frac{10\cdot10^3}{1\cdot10^3}\right) U_{in} \approx 11U_{in}.$$

Граничная частота низкочастотного фильтра:

$$f_o = \frac{1}{2\pi\sqrt{R_7R_8C_2C_4}} = \frac{1}{2\pi\sqrt{10\cdot10^3\cdot10\cdot10^3\cdot0.01\cdot10^{-6}\cdot0.01\cdot10^{-6}}} \approx 1591.5\text{Гц}.$$

В эксперименте используется три операционных усилителя LM741, два конденсатора с емкостью 1 мкФ, два конденсатора с емкостью 10 нФ, один резистор с сопротив-

* Полосовым фильтром называется устройство, подавляющее сигналы с частотами, лежащими вне его полосы пропускания.

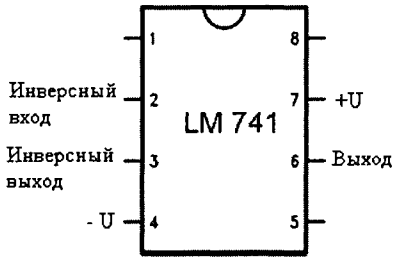


Рис. 22.5

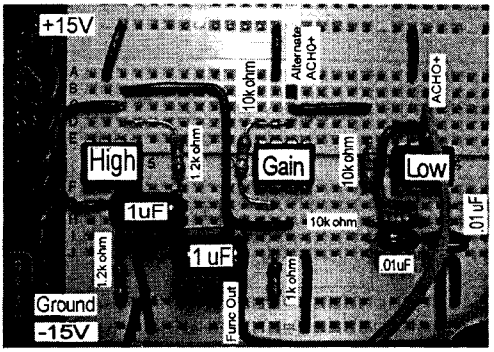


Рис. 22.6

лением 1 кОм, два резистора с сопротивлением 1.2 кОм и 3 резистора с сопротивлением 10 кОм. Схема выводов операционного усилителя показана на рис. 22.5.

В опыте предполагается на вход полосового фильтра подавать напряжение различной формы. Для чего используется генератор функций. Наблюдение сигнала можно проводить на двух участках: после стадии усиления и на выходе фильтра. С помощью виртуальных приборов NI ELVIS можно исследовать отклик и сравнивать его с теоретическим. Для любой лабораторной работы, в том числе и этой, имеется возможность создать собственный виртуальный прибор. Схема подключений элементов на макетной панели NI ELVIS показана на рис. 22.6.

Вертикальные полосы выводов подключены к выводам -15 В, земли и +15 В. Между ними и собрана вся схема. На рис. 22.6 не показано соединение вывода **ACH1+** с выводом генератора функций **FUNC_OUT**, а также вывода **ACH1-** с землей. Альтернативный узел наблюдения, который находится на участке между стадией усиления и низкочастотным фильтром, на рисунке заштрихован. Предполагается что провод, подключенный к выводу **ACH0+**, можно перемещать с выхода полосового фильтра в указанный альтернативный узел.

Выводы

Таким образом, NI ELVIS представляет собой целую измерительную систему, которая содержит оборудование NI ELVIS, DAQ-устройство и программное обеспечение LabVIEW, которое управляет оборудованием. Посредством NI ELVIS можно производить различные измерения, используя как драйвер NI-DAQ, так и драйвер NI ELVIS. При подключении к NI ELVIS можно пользоваться тремя стандартными измерительными функциями DAQ-устройств: аналоговым вводом и выводом, синхронизацией. Четвертая функция DAQ-устройства – цифровой ввод-вывод – невозможно использовать одновременно с измерительной системой NI ELVIS, поскольку через цифровой ввод-вывод осуществляется управление схемами настольной станции, а также регулируемыми источниками питания, генератором функций и цифровым мультиметром.

Лекция 23

Программное обеспечение NI ELVIS

С помощью программного обеспечения NI ELVIS управление настольной станцией и монтажной платой можно производить через компьютер. Виртуальные приборы имеют простой интерфейс и позволяют проводить измерения различных электрических величин, анализ собранных данных и выводить информацию на экран или в файл.

Программное обеспечение **NI ELVIS** включает следующие виртуальные приборы:

- **Digital Multimeter (DMM)** – цифровой мультиметр;
- **Oscilloscope (Scope)** – осциллограф;
- **Function Generator (FGEN)** – генератор функций;
- **Variable Power Supplies** – регулируемые источники питания;
- **Bode Analyzer** – частотно-фазовый анализатор;
- **Dynamic Signal Analyzer (DSA)** – анализатор динамических сигналов;
- **Arbitrary Waveform Generator** (ARB) – генератор сигналов произвольной формы;
- **Digital Bus Reader** – программа считывания цифровых сигналов;
- **Digital Bus Writer** – программа записи цифровых сигналов;
- **Impedance Analyzer** – анализатор входного сопротивления;
- **Two-Wire Current Voltage Analyzer** – двухпроводный вольтамперный анализатор;
- **Three-Wire Current Voltage Analyzer** – трехпроводный вольтамперный анализатор.

Модуль запуска виртуальных приборов – Instrument Launcher

Доступ ко всем виртуальным приборам **NI ELVIS** реализован через отдельную программу запуска, которая открывается по пути **Пуск** ⇒ **Программы** ⇒ **National Instruments** ⇒ **NI ELVIS 1.0** ⇒ **NI ELVIS**. Для загрузки какого-либо прибора,

необходимо нажать на соответствующую кнопку. Если кнопки запуска виртуальных приборов недоступны и затемнены, это может означать, что есть проблемы с DAQ-устройством либо настольной станцией. Возможно, что DAQ-устройство не сконфигурировано или что настольная станция отключена от питания. Программа выдаст сообщение о том, в чем может заключаться ошибка.

В этой главе дается краткое описание каждого из виртуальных приборов **NI ELVIS**. Порядок рассмотрения приборов соответствует их положению в списке виртуальных приборов программы запуска. Перед тем как перейти к описанию отдельных виртуальных приборов, следует остановиться на общих принципах работы с ними. Основными рабочими кнопками приборов являются кнопки запуска выполнения виртуального прибора и кнопки сохранения данных измерений и/или анализа в файл. Изображения кнопок и их описание представлены в табл. 23.1

Таблица 23.1

Изображение	*Название*	*Описание*
	***Run (Single)***	*Кнопка запуска виртуальнаго прибора. В случае, если рядом написано* ***Single****, программа выполняется один раз. В случае если рядом написано* ***Run****, программа будет выполняться постоянно.*
	***Run***	*Кнопка пастаянного запуска виртуального прибора в нажатом состоянии.*
	***Log***	*Кнопка сохранения результатав в файл.*

Цифровой мультиметр – Digital Multimeter (DMM)

Цифровой мультиметр поддерживает все основные функции мультиметра. Этот прибор может производить измерения следующих величин:

- Постоянное и переменное напряжение. Для измерения напряжения используются выходы **VOLTAGE HI** и **VOLTAGE LO**. Измеряемое постоянное напряжение должно быть в пределе ±20 В. Действующее значение измеряемого переменного напряжения не должно превышать 14 В.
- Постоянный и переменный ток. Для измерения тока используются выходы **CURRENT HI** и **CURRENT LO**. Ток не должен превышать 250 мА.
- Активное сопротивление. Для измерения активного сопротивления, а также для всех упоминаемых далее величин используются выходы **CURRENT HI** и **CURRENT LO**. Значение сопротивления должно находиться в пределах от 5 Ом до 3 МОм.
- Емкость. Пределы измерения составляют от 50пФ до 500мкФ.
- Индуктивность. Пределы измерения составляют от 100 мкГн до 100мГн.
- Тест работоспособности диода.
- Прозвонка.

Сопротивление, индуктивность и емкость не могут измеряться, когда генератор функций работает в ручном режиме.

Осциллограф – Oscilloscope (Scope)

Осциллограф обладает функциональностью стандартного осциллографа, который можно найти в лаборатории любого технического вуза. Осциллограф **NI ELVIS** имеет два канала. Имеется возможность масштабировать и регулировать ручки, изменяя временную развертку. Вы также можете выбрать источник пускового сигнала и режимные настройки. В зависимости от того, какое используется DAQ-устройство, можно выбрать между цифровым и аналоговым запуском развертки осциллографа. Кроме всего прочего воспользоваться осциллографом можно, подсоединяя его выводы как на макетной панели, так и через разъемы на лицевой панели настольной станции. Лицевая панель виртуального осциллографа показана на рис. 23.1.

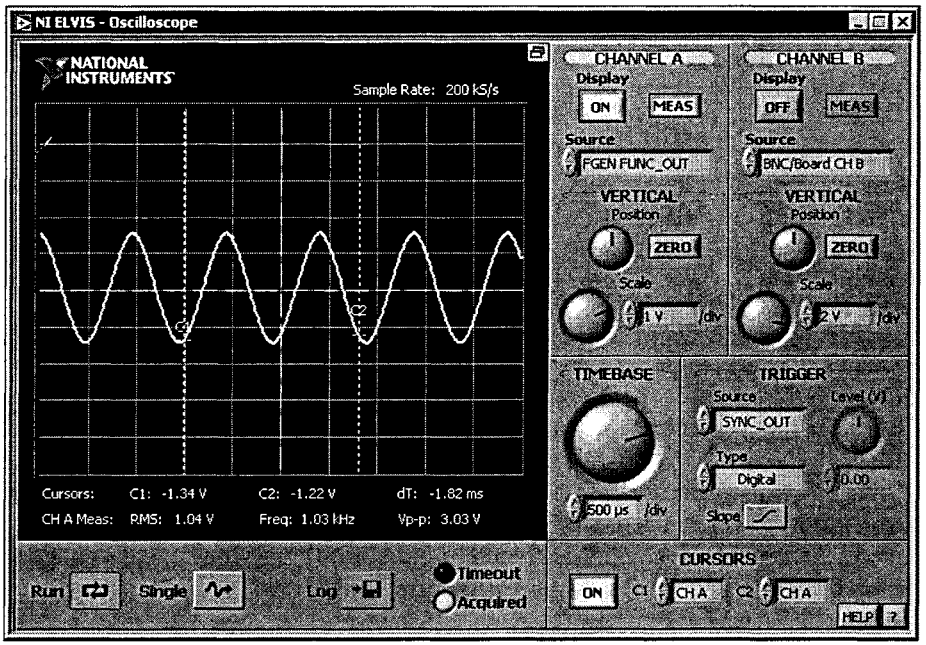


Рис. 23.1

Генератор функций – Function Generator (FGEN)

Генератор функций встроен в настольную станцию. На его выходе можно получить сигнал синусоидальной, прямоугольной и треугольной формы. Формирование сигнала, выдаваемого генератором функций как через элементы управления настольной станции (при ручном режиме управления), так и через элементы управления виртуального прибора (при программном режиме управления), осуществляется через настройку следующих элементов:

- форма сигнала (синусоидальная, треугольная, прямоугольная);
- амплитуда (8-битная установка амплитуды в пределах ±2,5 В);
- частота (основная разрешающая способность – быстрая 8-битная установка частоты в пределах от 5 Гц до 250 кГц и высокая разрешающая способность – медленная 12-битная установка частоты);
- постоянная составляющая (8-битная установка постоянной составляющей в пределах ±5 В);
- амплитудная и частотная модуляция (до 10 В).

Регулируемые источники питания – Variable Power Supplies

Регулируемые источники питания позволяют на выходе получить от -12 до 0 и от 0 до +12 В постоянного напряжения. Управление источниками осуществляется как в ручном, так и в программном режиме.

Для того, чтобы использовать регулируемые источники, подключите выводы регулируемых источников питания **SUPPLY+** и **SUPPLY–** к схеме. Установите необходимое напряжение через элементы управления лицевой панели настольной станции или виртуального прибора.

Частотно-фазовый анализатор – Bode Analyzer

Частотно-фазовый анализатор предназначен для построения фазо-частотной и амплитудно-частотной характеристик (АЧХ и ФЧХ) пассивных и активных линейных электрических цепей. Лицевая панель виртуального прибора показана на рис. 23.2.

В целях улучшения картины АЧХ и ФЧХ надо соответствующим образом подбирать амплитуду сигнала. На пассивные цепи следует воздействовать сигналом с высокой амплитудой, а активные цепи (с усилением) следует изучать сигналом с небольшой амплитудой, что позволит избежать насыщения выходного напряжения.

Задание 23.1. Снятие АЧХ и ФЧХ

Снимите АЧХ и ФЧХ электрической цепи. Для того, чтобы снять характеристики следует выполнить следующие действия:

1. На макетной панели собрать схему, которую вы собираетесь изучать.
2. К схеме подключить источник напряжения. К выводу **FUNC_OUT** один узел схемы, а к выводу **GROUND** – другой.
3. Соединить вывод **FUNC_OUT** с **ACH1+**, а вывод **GROUND** к **ACH1-**.
4. Соединить вывод **ACH0+** с изучаемым узлом, а **ACH0-** с выводом **GROUND**.
5. Запустить виртуальный прибор **Bode Analyzer**.

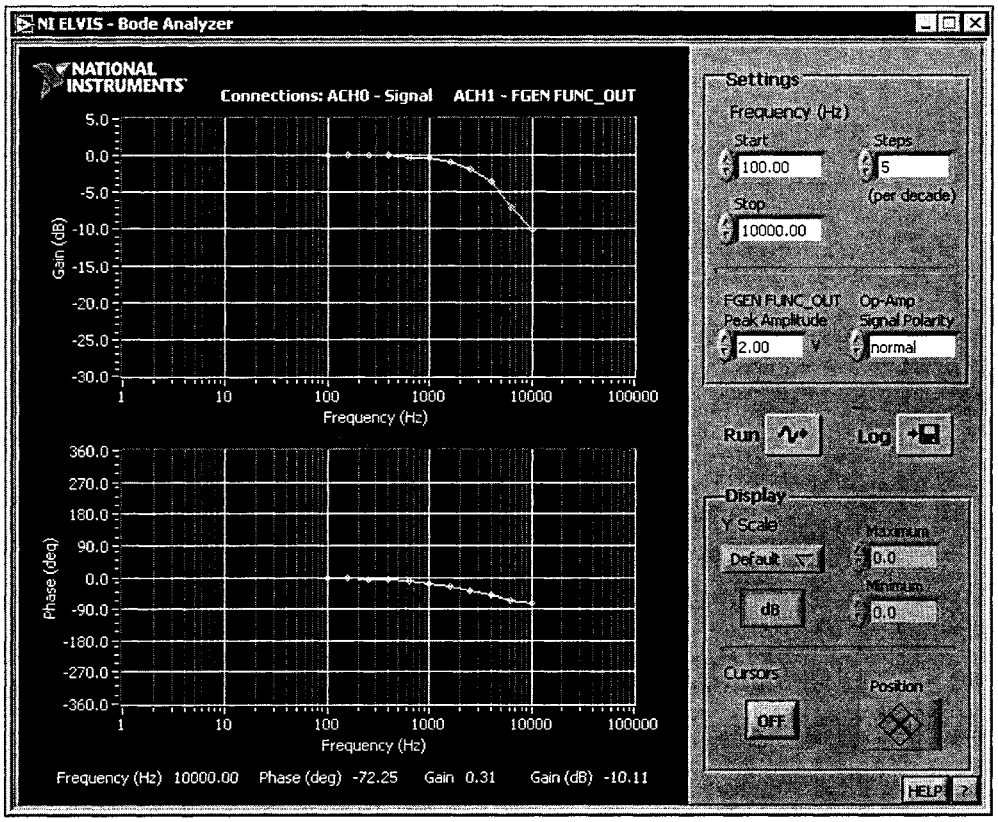


Рис. 23.2

6. Установить начальную и конечную частоту сбора данных в элементах управления **Start** и **Stop**.
7. Запустить анализ.

Анализатор динамических сигналов – Dynamic Signal Analyzer

Анализатор динамических сигналов (его лицевая панель показана на рис. 23.3) вычисляет и отображает усредненную среднеквадратическую мощность спектра одного из каналов. С его помощью можно также определять пиковую частотную составляющую, оценить рабочую частоту и мощность. Диапазон частот, для которых производятся измерения и вычисления зависит от возможностей DAQ-устройства. В зависимости от возможностей DAQ-устройства виртуальный прибор поддерживает цифровое и аналоговое включение. Для включения по цифровому сигналу запускающий сигнал следует подключать к выводу **TRIGGER**. При

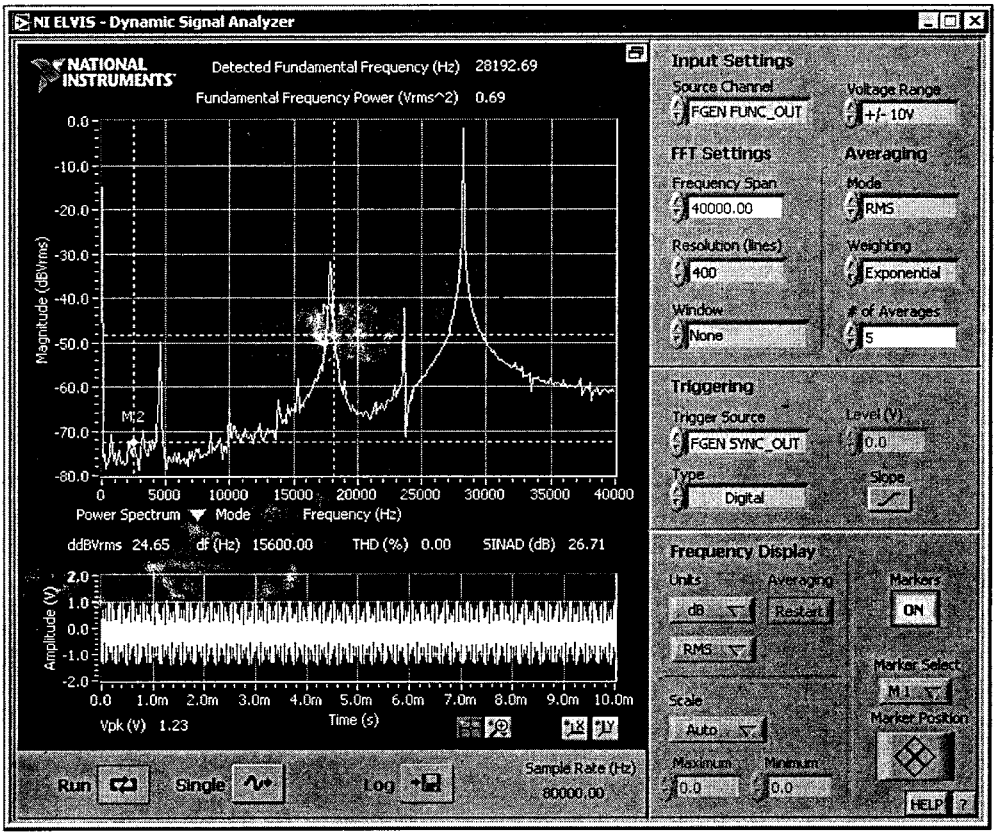


Рис. 23.3

включении по аналоговому сигналу можно установить уровень (**Level**) и наклон (**Slope**) запускающего сигнала.

Задание 23.2. Анализ динамических сигналов

Для того, чтобы получить функцию спектральной плотности и ее характеристики, выполните следующие действия:

1. Подключите измеряемый сигнал к BNC разъему на лицевой панели настольной станции или соответствующим выводам макетной панели. Выбор выводов, с которого будет считываться сигнал, осуществляется выбором канала источника **Source Channel**.
2. Запустите виртуальный прибор **Dynamic Signal Analyzer**. После запуска виртуальный прибор выполняется автоматически.
3. При необходимости в поле **Triggering** настройте данные запускающего сигнала.
4. При необходимости настройте дополнительные возможности анализатора.

Генератор сигналов произвольной формы – Arbitrary Waveform Generation

С помощью генератора сигналов произвольной формы можно образовать сигнал определяемой пользователем формы. Лицевая панель генератора сигналов произвольной формы показана на рис. 23.4. Вы можете создавать множество различных сигналов, используя программу Waveform Editor, которая включена в программное обеспечение **NI ELVIS**. Запустить редактор сигналов можно по пути **Пуск** ⇒ **Программы** ⇒ **National Instruments** ⇒ **NI ELVIS 1.0** ⇒ **Waveform Editor**. Далее, чтобы генератор сигналов произвольной формы обработал и подал на выход созданный вами сигнал, необходимо загрузить файлы с расширением ***.wdt**.

Поскольку обычное DAQ-устройство имеет два канала аналогового вывода, одновременно на выход можно подавать два сформированных сигнала.

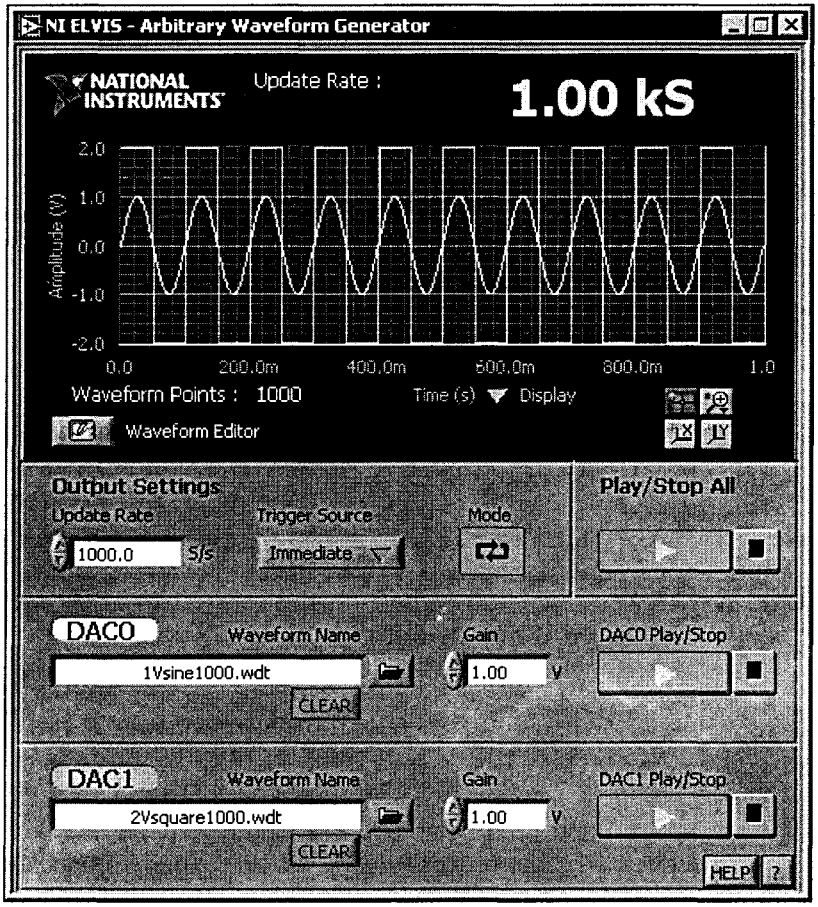


Рис. 23.4

Задание 23.3. Генерация сигнала произвольной формы

Для того, чтобы подать созданный вами сигнал на реальную электрическую цепь, выполните следующие действия:

1. Подключите выводы **DAC0** и/или **DAC1** к схеме.
2. Запустите виртуальный прибор **Arbitrary Waveform Generation**.
3. Загрузите файл (или файлы) с формой сигнала.
4. Установите частоту обновления (**Update Rate**). Аналоговая частота сигнала определяется частотой обновления и числом выборок за период.
5. Запустите процесс формирования сигнала (**Run**).

Цифровое считывающее и записывающее устройство – Digital Reader и Digital Writer

Цифровое считывающее устройство считывает цифровые данные с шины считывания. Подключение устройства осуществляется через выводы цифрового ввода **DI**.

Цифровое записывающее устройство передает шине записи пользовательские цифровые шаблоны. Вы можете сами составить шаблон или использовать готовые шаблоны. Подключение устройства осуществляется через выводы цифрового вывода **DO**.

Задание 23.4. Цифровой ввод-вывод

Подключите выводы DO к выводам световых индикаторов. Используя один из предложенных шаблонов, понаблюдайте за индикаторами.

Анализатор входного сопротивления – Impedance Analyzer

Анализатор входного сопротивления, как следует из названия, определяет входное сопротивление, активную и реактивную составляющие пассивного двухполюсного элемента. Его лицевая панель показана на рис. 23.5. Подключение к анализатору входного сопротивления осуществляется через выводы **CURRENT HI** и **CURRENT LO**.

Двухпроводный вольтамперный анализатор – Two-Wire Current-Voltage Analyzer

С помощью двухпроводного вольтамперного анализатора можно построить вольтамперную или внешнюю характеристику двухполюсника. Его лицевая панель показана на рис. 23.6. Измерения можно производить во всех четырех квадрантах в пределах ±10 В для напряжения и ±10 мА для тока. Виртуальный прибор предлага-

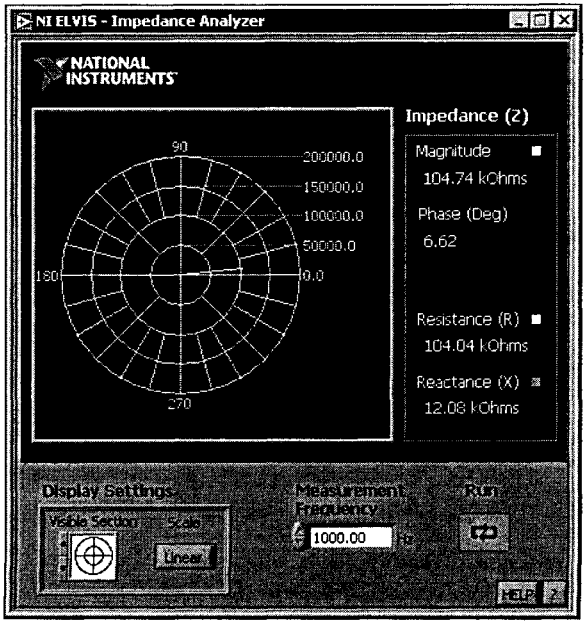


Рис. 23.5

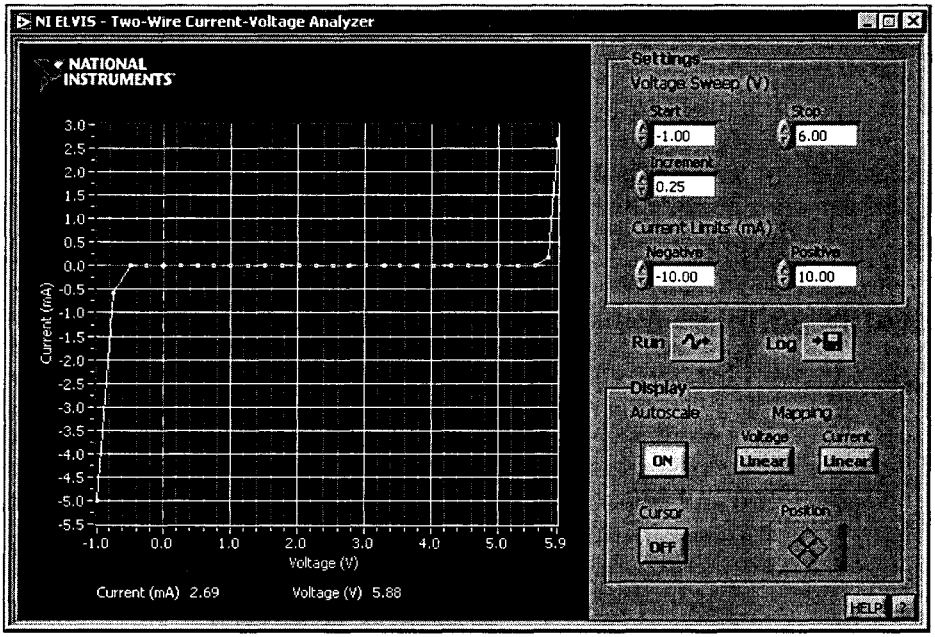


Рис. 23.6

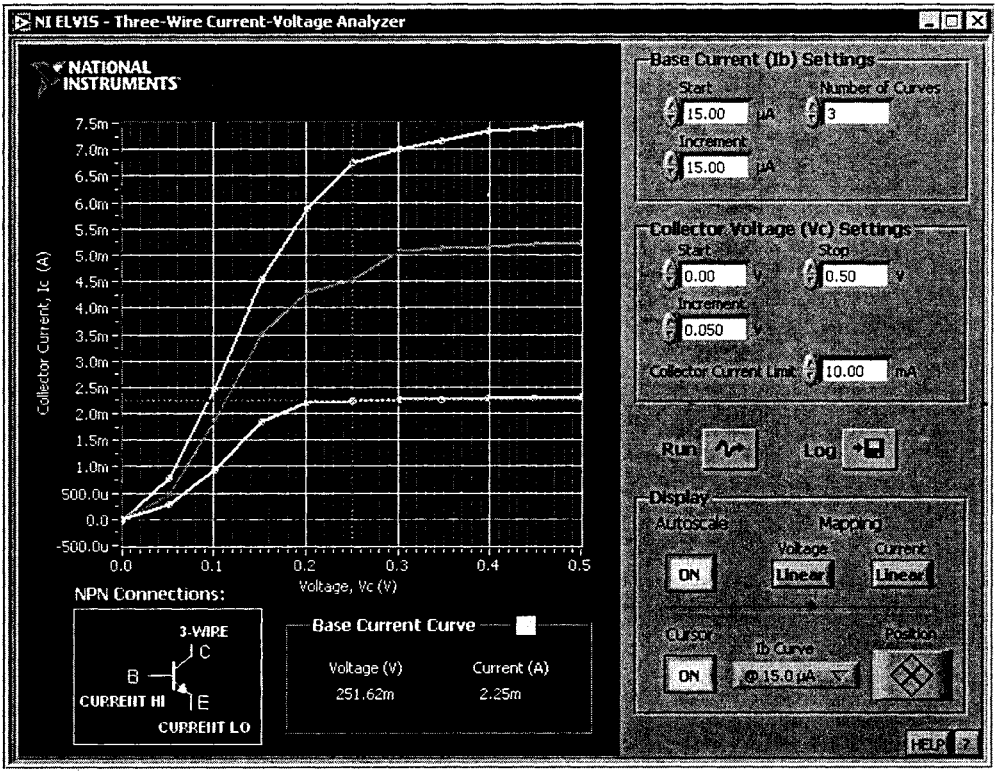


Рис. 23.7

ет достаточно гибкие настройки для пределов изменения тока и напряжения, а также сохранения данных в файл. Подключение двухполюсника осуществляется к выводам CURRENT HI и CURRENT LO.

Трехпроводный вольтамперный анализатор – Three-Wire Current-Voltage Analyzer

Трехпроводный вольтамперный анализатор предназначен для снятия внешних характеристик npn биполярных плоскостных транзисторов. Лицевая панель виртуального прибора показана на рис. 23.7. Виртуальный прибор может строить внешние характеристики в пределах от 0 до 10 В для напряжения коллектора и от 0 до 10мА для тока коллектора. Подключение транзистора осуществляется следующим образом: коллектор подключается к выводу 3-WIRE, база к – CURRENT HI, эмиттер к – CURRENT LO.

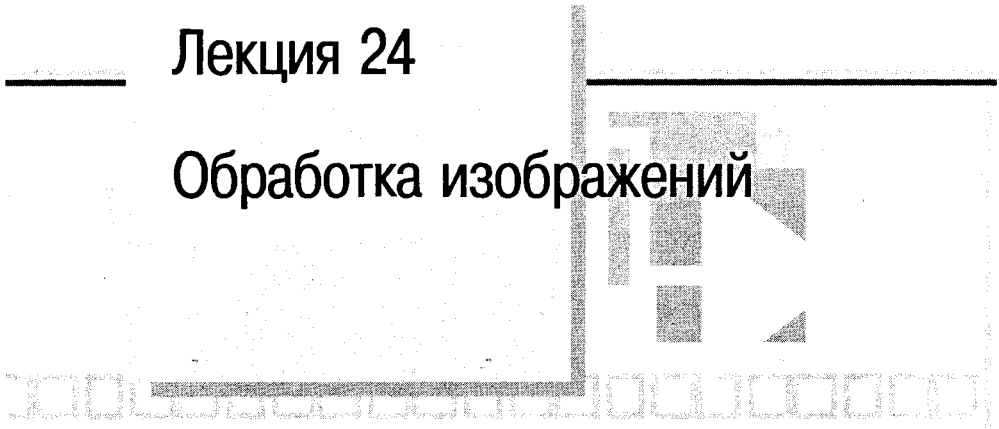
Программный код всех виртуальных приборов имеет открытую архитектуру. Его можно модифицировать в зависимости от предъявляемых к виртуальным приборам требований. Для всех приборов прилагаются драйверы под LabVIEW. На основе драйверов **NI ELVIS** (они находится в палитре функций **Instrument I/O** ⇒ **Instrument Drivers** ⇒ **NI ELVIS**) вы можете создавать свои виртуальные приборы.

Выводы

К собственно настольной станции **NI ELVIS** и DAQ-устройству прилагается программное обеспечение, представляющее собой набор виртуальных приборов, осуществляющих стандартные измерения. Цифровой мультиметр, осциллограф, частотно-фазовый анализатор, анализатор входного сопротивления, двухпроводный и трехпроводный вольтамперный анализатор позволяют измерять напряжение, ток, различные характеристики двухполюсников, строить амплитудно-частотные, фазо-частотные, а также вольтамперные характеристики. Виртуальные приборы генератор функций и регулируемые источники питания предназначены для управления источниками сигналов настольной станции программным образом. Для работы с цифровым вводом-выводом используются программы записи и считывания цифровых сигналов. Кроме перечисленных виртуальных приборов при выполнении различных лабораторных работ может понадобиться генератор сигналов произвольной формы.

Лекция 24

Обработка изображений



Рисунки и иллюстрации, несомненно, являются наиболее выразительным средством представления информации. Они повышают наглядность представляемых данных и облегчают работу пользователя. С помощью LabVIEW можно создавать практически любые графические изображения для представления результатов расчетов, измерений и т.п.

Представление графики в LabVIEW

Графические данные в LabVIEW могут храниться в трех формах: двумерном массиве (**pixmap**), одномерном массиве (кластер **image data**) и векторном рисунке (**picture**).

При использовании одномерного или двумерного массива в него записывается информация о цвете каждой точки изображения. Поэтому такой формат хранения изображения называется точечным (или растровым).

В случае использования двумерного массива каждой точке изображения с координатами (x,y) соответствует элемент массива, находящийся в строке под номером y и в столбце под номером x. Точка с координатами (0,0) располагается в левом верхнем углу изображения. При таком способе хранения структура массива данных полностью совпадает со структурой изображения. Поэтому можно легко получить информацию о цвете отдельной точки – для этого нужно извлечь из массива соответствующий элемент с помощью функции индексирования массива (**Index Array**).

Однако применение кластера **image data** для хранения растрового изображения является более предпочтительным. Основной элемент кластера – одномерный байтовый массив **image**, в котором собственно и размещается само изображение. На кодирование одной точки может расходоваться различное количество элементов массива **image**. Способ кодирования зависит от значения другого элемента кластера – **image depth**, который определяет глубину цвета или, проще говоря, количество цветов которые могут быть одновременно использованы в изображении. Если

используется 24-битная глубина цвета (16 млн. цветов), каждая точка кодируется тремя элементами **image** – по одному на красную, зеленую и синюю составляющие цвета. Для изображений с глубиной цвета 8 или 4 бит в **image** указывается номер цвета из палитры для каждой точки. Палитра при этом записывается в массиве **colors**, который тоже входит в кластер **image data**. Для монохромных изображений (глубина цвета 1) один элемент **image** кодирует сразу восемь точек – по одному биту на точку.

Кластер **image data** содержит еще один интересный элемент – байтовый массив **mask** (маска). На каждую точку изображения приходится один бит из этого массива. Если соответствующий точке бит из **mask** равен нулю, эта точка становится как бы прозрачной (маскируется). Для работы с масками предусмотрена специальная функция создания маски **Create Mask**. Мы рассмотрим ее чуть позже.

В противоположность растровому существует векторный формат изображения. При его использовании записывается не информация о цвете каждой точки, а команды рисования (например, **провести линию, нарисовать окружность**), с помощью которых это изображение получено. Для векторных изображений в LabVIEW предусмотрен специальный тип данных картинки – **picture**. С типом **picture** работают все функции LabVIEW для рисования графических примитивов (линий, окружностей, прямоугольников и т.д.). Графический элемент индикации также принимает данные типа **picture**, что, однако, вовсе не означает невозможности вывода на него растрового изображения – нужно только предварительно нарисовать его на объекте типа **picture**, с помощью функций **Draw Flattened Pixmap/Draw Unflattened Pixmap**.

Холст, кисти и краски

Последовательность создания рисунка в LabVIEW такова: вы берете пустую картинку (**All Functions ⇒ Graphics & Sound ⇒ Picture Functions ⇒ Empty Picture**) – как бы чистый холст и рисуете на нем что-нибудь, например точку. Каждая функция рисования принимает на один из входов исходный рисунок и выдает на выходе рисунок с добавленное фигурой. Поэтому, чтобы нарисовать еще одну точку на том же рисунке, нужно соединить выход одной функции рисования со входом другой.

Кстати говоря, для рисования точек в LabVIEW используется функция **Draw Point**. С принципом ее работы мы сейчас познакомимся поближе, тем более, что такие же, в общем, принципы положены в устройство и других функций рисования.

Итак, как вы уже знаете, одним из входов любой функции рисования, и **Draw Picture** не исключение, является вход типа **picture**. На него подается исходный рисунок.

На другой вход, который называется **point** (*x,y*), подаются координаты будущей точки – два целых числа, упакованные в кластер.

Параметр **color** указывает, каким цветом рисовать. Он имеет тип 32-разрядного целого, но если цвет рисунка известен заранее, при составлении блок-диаграммы, его лучше задать с помощью цветовой константы. Для этого на входе **color** нужно

выбрать из контекстного меню **Create ⇒ Constant**. На блок-диаграмме появится квадратик, цвет которого можно менять с помощью кисточки из панели инструментов (**Tools Palette**).

Наконец параметр **pen** позволяет управлять тем, как будут рисоваться нужные нам геометрические фигуры. Первый элемент кластера pen задает толщину пера (**Width**), а второй – тип линии (**Style**). Изменяя толщину пера можно рисовать жирные точки и линии. Тип линии при рисовании точек не используется, зато используется при рисовании других геометрических фигур. Возможные значения этого параметра приведены в табл. 24.1. Помните, однако, что если толщина пера задана больше 1, то все линии будут рисоваться сплошными независимо от выбранного типа линий.

Таблица 24.1

Значение	*Название*	*Тип линии*
0	*Solid*	*Сплошная линия*
1	*Dash*	*Штриховая линия*
2	*Dot*	*Пунктирная линия*
3	*Dash Dot*	*Штрих-пунктирная линия*
4	*Dash Dot Dot*	*Линия, чередующая штрих и два пунктира*

Зная как рисовать точки, пойдем дальше и научимся рисовать линии. Для этого в LabVIEW существует функция **Draw Line**. Ее входы **picture**, **color** и **pen** имеют то же назначение, что и одноименные параметры функции **Draw Point**, поэтому на них останавливаться не будем.

Новым для нас является параметр **end point**. Он представляет собой кластер из двух целых чисел и указывает точку конца линии. Но ведь у линии помимо конца есть еще и начало, откуда же LabVIEW узнает, откуда начинать рисование? Дело в том, что в объекте типа **picture** помимо собственно рисунка хранятся еще и так называемые координаты курсора, их то и использует функция **Draw Line** в качестве координат начала линии. Координаты курсора неявно устанавливаются функциями рисования по завершении своей работы. Так функция рисования точки устанавливает курсор в точке рисования, а функция рисования линии – в точке конца линии. Есть и отдельная функция управления положением курсора – **Move Pen**, она устанавливает курсор в точку с координатами, указанными в кластере **new position**.

С понятием курсора связан еще один параметр, который есть у функций **Draw Line** и **Move Pen**. Это логический параметр **absolute value?**. Если его значение истинно (по умолчанию так и есть), то все координаты, переданные функции, отсчитываются от левого верхнего угла рисунка, в противном случае отсчет ведется от текущего положения курсора.

Задание 24.1. Создание рисунка

Покажем, как на рисунок нанести синусоиду. На рис. 24.1 показана блок-диаграмма простой программы, использующей функцию рисования линии для построения графика функции $y = sin(x)$.

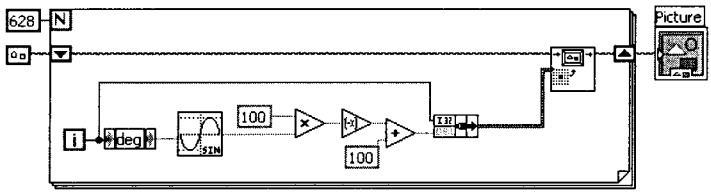


Рис. 24.1

Блок-диаграмма состоит из цикла **For**, с помощью которого мы будем менять значение переменной x. Для перевода градусов в радианы используется узел **Convert Unit**. Полученное число используется как аргумент функции вычисления синуса **All Functions ⇒ Numeric ⇒ Trigonometric ⇒ Sine**. Чтобы правильно отобразить полученное значение на рисунке нужно провести ряд преобразований. Во-первых, помножим его на сто для того, что бы увеличить масштаб. Во-вторых, вычтем из результата опять сто и помножим на −1. Последняя операция выполняет преобразование системы координат. Дело в том, что нам нужно, чтобы начало координат находилось по середине рисунка и ось y была бы направлена вверх, в то время как при рисовании считается, что начало координат находится в левом верхнем углу, а вертикальная ось направлена вниз.

Наконец, полученные значения координат точки упаковываются в кластер. Таким образом, на каждой итерации цикла будут вычисляться координаты новой точки графика.

Теперь займемся выводом его на экран. Для начала добавьте к циклу **shift register**, нажав на рамке цикла и выбрав соответствующий пункт из контекстного меню. В этом регистре будет запоминаться наш рисунок. В качестве начального значения к нему нужно присоединить пустую картинку (**All Functions ⇒ Graphics & Sound ⇒ Picture Functions ⇒ Empty Picture**). Конечное значение регистра нужно поместить в графический элемент индикации, на котором и будет отображаться рисунок.

И, наконец, в цикл нужно вставить функцию рисования линий **Draw Line**. Ее вход **picture** и выход **new picture** нужно подключить к регистру соответственно на левой и правой рамке цикла. Ко входу **end point** подключите кластер с координатами очередной точки. Так как **Draw Line** использует текущие координаты курсора для определения точки начала линии и в то же время устанавливает курсор в точку конца линии по окончании рисования, график будет состоять из отрезков последовательно соединенных прямых.

В LabVIEW предусмотрено еще много функций для рисования геометрических фигур. Их перечень приведен в табл. 24.2.

Таблица 24.2.

Изображение	*Название*	*Функция*
	*Draw Point*	*Нарисовать точку*

Таблица 24.2 (окончание)

Изображение	Название	Функция
	Move Pen	Переместить перо
	Draw Line	Нарисовать линию
	Draw Multiple Lines	Нарисовать ломанную линию
	Draw Rect	Нарисовать прямоугольник
	Draw Grayed Out Rect	Нарисовать полупрозрачный прямоугольник
	Draw Rounded Rect	Нарисовать прямоугольник с закругленными углами
	Draw Oval	Нарисовать эллипс
	Draw Arc	Нарисовать дугу (сектор)
	Draw Text at Point	Нарисовать текст в указанной точке
	Get Text Rect	Вычислить прямоугольник, охватывающий текст
	Draw Text in Rect	Нарисовать текст в прямоугольнике
	Draw Circle by Radius	Нарисовать окружность заданного радиуса
	Draw Unflattened Pixmap	Нарисовать картинку по двумерному массиву
	Draw Flattened Pixmap	Нарисовать картинку по одномерному массиву

Подписи к рисункам

Помимо нанесения на рисунок геометрических фигур, его можно украсить и текстовыми надписями. Такая возможность пригодится для вывода поясняющих подписей к графикам, чертежам и т.д.

Для вывода текста в LabVIEW имеется три функции. Прежде, чем перейти к их описанию, разберемся с тем, как LabVIEW печатает текстовую информацию. Вообще говоря, термин «печать» плохо подходит для описания этого процесса. Как и в любой другой системе с графическим интерфейсом пользователя, в LabVIEW текст «рисуется» на экране. Начертание каждой литеры хранится не в виде матрицы точек, а в виде описания геометрических кривых, из которых она состоит. Главным преимуществом такого способа, является легкость масштабирования и других трансформаций начертания (например, при использовании курсива), которые можно осуществить без ухудшения качества получающегося в итоге изображения. Однако при этом возникают и свои трудности. Так для того, чтобы поместить одну строку под другой, нужно знать, сколько точек займет выводимая строка в высоту, и самостоятельно учесть эту величину вместе с межстрочным интервалом при выборе точки вывода второй строки.

Итак, приступим к описанию текстовых функций. Начнем с общих для всех параметров. Во-первых, это параметр текстового типа **text**, в котором собственно и задается текст будущей надписи.

Два других параметра задают параметры шрифта. Параметр перечислимого типа **desired font** позволяет выбрать один из трех системных шрифтов. Если его значение равно нулю, то нужный шрифт формируется по описанию, передаваемому в кластере **[user specified font]**. Назначение элементов этого кластера приведено в табл. 24.3. Если системе не удастся создать шрифт, в точности подходящий под описание, она предложит наиболее подходящий по ее мнению. Особое внимание следует уделить элементу **Font Name**, в котором задается название шрифта. Здесь нельзя ошибиться даже в одной букве, иначе все другие параметры будут проигнорированы, и будет использоваться стандартный шрифт.

Таблица 24.3.

Элемент	*Описание*
Font Name	*Название шрифта*
Size	*Кегль шрифта*
Strikeout	*Перечеркнутый шрифт*
Italic	*Курсив*
Underline	*Шрифт с подчеркиванием*
Outline	*Контурный шрифт*
Shadow	*Оттененный шрифт*
Bold	*Полужирный шрифт*

Ориентацию текста можно менять с помощью параметра **text orientation** (таблица 24.4). По умолчанию текст печатается горизонтально слева направо.

Таблица 24.4.

Значение	*Название*	*Ориентация текста*
0	*None*	*Обычная ориентоция*
1	*Stacked*	*Вертикально в столбик*
2	*Clockwise*	*Повернуто на 90° по часовой стрелке*
3	*Counterclockwise*	*Повернуто на 90° против часовой стрелки*

Теперь поговорим о конкретных функциях вывода текста. Одна из них, **Get Text Rect**, вычисляет минимальный прямоугольник, в который помещается заданная строка текста. Результат представляет собой координаты левого верхнего и правого нижнего угла прямоугольника, упакованные в кластер. Используйте эту функцию, чтобы правильно расположить на одном рисунке несколько текстовых надписей, не допуская их наложения друг на друга.

Вторая функция под названием **Draw Text at Point**, занимается собственно рисованием текста. В качестве одного из параметров выступает так называемая базовая точка, координаты которой находятся в кластере **origin**. Она определяет коор-

динаты вывода текста совместно с параметром **alignment**. Последний является кластером из двух переменных перечислимого типа (**horizontal** и **vertical**): первая позволяет определять положение текста относительно базовой точки по горизонтали, а вторая – по вертикали. По горизонтали базовая точка может обозначать левый край текста, его середину или правый край (значение переменной **left**, **center** и **right** соответственно), а по вертикали верхний край, середину строки или нижний край (значение переменной **top**, **center** и **bottom** соответственно).

Наконец третья функция, **Draw Text in Rect**, рисует текст в указанном прямоугольнике, который задается в кластере **rect**. При этом все, что выходит за рамки прямоугольника, автоматически обрезается. При использовании этой функции гарантируется, что надпись не выйдет за отведенные под нее рамки и не испортит остальной рисунок.

Операции с графическими данными

Поддержка операций с графическими данными в последних версиях LabVIEW значительно улучшилась. Теперь можно легко реализовать полный цикл обработки, включающий загрузку графики из файла, модификацию и запись рисунка обратно в файл.

Пример 24.1. Титры

На рисунке 24.2 приведена блок-диаграмма ВП, осуществляющая как раз такой цикл. Воспользуемся возможностями ВП обработки графических данных для того, чтобы нанести на картинку титры.

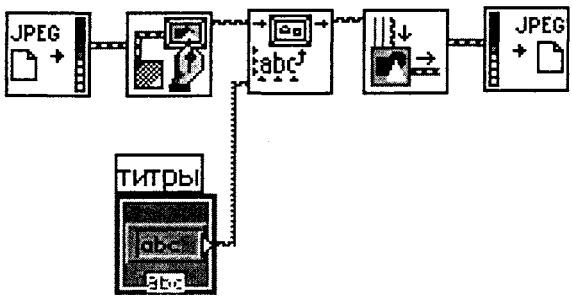


Рис. 24.2

Сначала происходит считывание графических данных из файла. Предназначенные для этого функции находятся на вкладке **All Functions ⇒Graphics & Sound ⇒ Graphics Formats**. Поддерживаются файлы в формате JPEG, BMP и PNG. На этой же вкладке находятся ВП для преобразования графических данных в форму двумерного массива и обратно.

Затем происходит отображение полученной из файла картинки на объект **picture** с помощью функции **Draw Flattened Pixmap**. Существует аналогичная по действию

функция **Draw Unflattened Pixmap**, которая принимает данные в виде двумерного массива. Если вы решите ей воспользоваться, обратите внимание на то, что она является полиморфной, то есть автоматически определяет глубину цвета передаваемого ей изображения. Однако, если вы используете 4- или 8-битное изображение, это нужно указать явно, выбрав в пункте контекстного меню **Select Type** нужную модификацию этой функции.

Возможности вывода картинки на объект **picture** мы рассмотрели ранее. В примере 24.1 используются функции рисования текста, с помощью которых на картинку помещаются титры. Затем используется функция **Picture to Pixmap** для перевода картинки обратно в форму одномерного массива. И, наконец, исправленный рисунок сохраняется в новый файл.

Упомянем еще две функции, предназначенные для работы с растровой графикой (они также находятся на вкладке **All Functions ⇒ Graphics & Sound ⇒ Picture Functions**). Первая из них – это **Get Image Subset**, которая предназначена для вырезания из картинки прямоугольной области. Координаты прямоугольника передаются в кластере **subset rect**. Эта функция работает с графикой в форме одномерного массива.

Вторая функция используется для наложения маски. Как мы уже говорили при рассмотрении видов представления графики в LabVIEW, кластер **image data** содержит специальный битовый массив, позволяющий сделать некоторые точки изображения прозрачными. Функция **Create Mask** записывает в этот массив такие значения, чтобы сделать прозрачными точки определенного цвета, указанного в параметре **Mask Color**. Дополнительный логический параметр **Combine Masks?** указывает, нужно ли скомбинировать новую маску со старой. Например, если вы хотите скрыть на картинке элементы желтого и зеленого цветов, нужно использовать последовательно две функции **Create Mask**, причем у второй параметр **Combine Masks?** должен иметь значение «истина». Последний параметр этой функции, **1-Bit Mask Value** предназначен для работы с монохромными (однобитными) изображениями. Он определяет выводить на экран точки со значением единица или, наоборот, со значением ноль.

Создание собственных элементов индикации

Возможности видоизменения стандартных индикаторов в LabVIEW довольно ограничены. Стандартные элементы индикации уже запрограммированы определенным образом, и изменить логику их поведения невозможно (ведь у элемента управления нет блок-диаграммы, только лицевая панель). Используя инструменты для работы с графикой, можно создавать индикаторы с принципиально новыми функциями.

Пример 24.2. Элемент индикации в виде рисунка

В качестве примера создадим числовой элемент индикации, который будет показывать число в виде кругового сектора.

Для этого сначала поместим на лицевую панель целочисленный элемент управления (**Numeric Controls ⇒ Vertical Pointer Slide**) и графический индикатор (**All Controls ⇒ Graph ⇒ Controls ⇒ Picture**). Затем соберем блок-диаграмму показанную на рис. 24.3.

Изучим принцип ее действия.

Число от элемента управления поступает на вход **arc size** функции **Draw Arc** и тем самым задает угловой размер сектора. Так как самый большой из возможных секторов имеет размер равный 360°, это же значение является максимальным числом, которое сможет отобразить наш индикатор. Чтобы снять это ограничение, нужно просто умножить входной параметр на число (360 / max), где max – требуемое предельное значение, отображаемое индикатором.

Ко входу **start angle** подключен ноль – сектор будет рисоваться начиная с верхней точки. Вход **Rect** определяет прямоугольную область, в которую будет вписан рисунок сектора. Ко входу **color** подключена цветовая константа зеленого цвета, а на входе **Fill** установлено логическое значение истина, чтобы получить закрашенный сектор.

Второй узел на блок-диаграмме **Draw Text in Rect** используется для вывода на элемент индикации числового значения. Для простоты текст выводится в тот же прямоугольник, что и сектор окружности. Ко входу **text** должен подключаться сигнал текстового типа. Чтобы перевести число в текстовую форму используется функция **All Functions ⇒ String ⇒ String/Number Conversions ⇒ Number to Decimal String**.

Наконец, полученное изображение выводится на графический элемент индикации (рис. 24.4).

Если вы запустите программу в режиме непрерывного исполнения и попробуете подвигать рычажок **Vertical Pointer Slide**, показания индикатора тоже начнут меняться. Можно заметить, что при смене показания наш индикатор начинает мер-

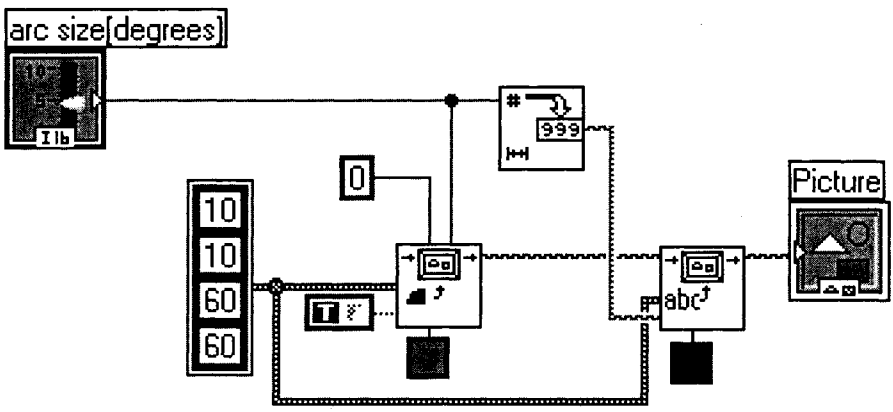


Рис. 24.3

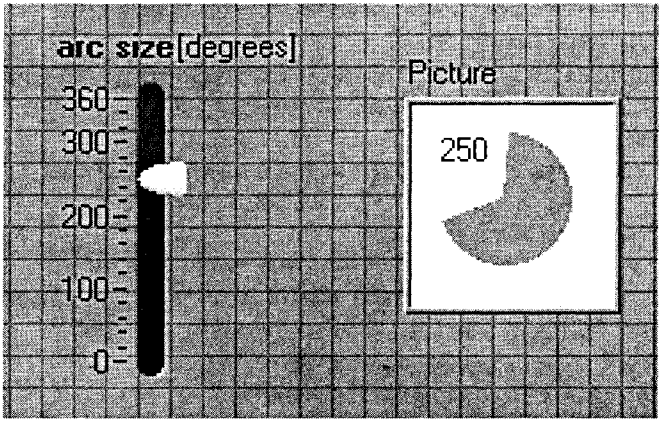


Рис. 24.4

цать. Это происходит из-за того, что перед тем как нарисовать новое изображение, LabVIEW полностью стирает старое. Чтобы избавиться от мерцания, нужно заставить LabVIEW сначала нарисовать новое изображение элемента индикации в памяти и только потом перенести на **Picture Control**. Этого можно добиться, выбрав из контекстного меню графического элемента индикации (на лицевой панели) пункт **Advanced ⇒ Smooth Updates**.

Как мы убедились, преобразовать информацию от некоторого датчика довольно просто. Наибольший интерес создание собственного элемента индикации представляет, когда есть необходимость на одном рисунке-индикаторе отразить показание сразу нескольких датчиков. Например, можно создать индикатор для диспетчера лифта, который будет одновременно показывать положение лифта в шахте и состояние дверей в нем.

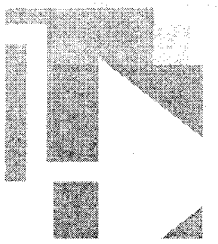
Созданный виртуальный прибор можно сохранить в файл и использовать в других приложениях.

Выводы

Использование рисунков и функций работы с ними позволяет, во-первых, сделать интерфейс ВП более дружественным, а, во-вторых, работать с графическими данными. С помощью функций для работы с графическими изображениями имеется возможность создать уникальный элемент индикации.

Лекция 25

Работа в сети



Изучаются вопросы создания и работы с удаленной лицевой панелью, ее публикацией в Интернете при помощи web-браузера и просмотра страницы в браузере.

В последующих двух главах вы познакомитесь с некоторыми инструментами LabVIEW, которые дают возможность публиковать виртуальные приборы в интернете, обмениваться данными через сеть, взаимодействовать с другими программами и т.д.

LabVIEW обладает рядом встроенных возможностей для организации связи через интернет, в том числе:

- Функции **TCP/IP** и **UDP**;
- Web-сервер;
- Сервер виртуальных приборов;
- Протокол **DataSocket** для обмена данными через LAN и интернет;
- Java-приложения;
- Элементы управления **ActiveX**;
- **E-mail**, **ftp** и **telnet**.

Web-сервер

Не следует путать 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 браузеру просматривать, наблюдать или управлять ВП. Сервер виртуальных приборов позволяет локальным и удаленным ВП вызывать функции и свойства вашего ВП.

Настройка web-сервера производится на странице **Web Server: Configuration** в **Tools ⇒ Options**. По умолчанию web-сервер отключен. Поэтому для начала отметьте флажком **Enable Web Server**. При желании измените корневой каталог **Root Directory**, в котором будут размещаться все необходимые файлы. По умолчанию web-сервер работает на стандартном 80 порту. Чтобы в системе не возникло конфликтов, убедитесь, что нет другого сервера, работающем на этом же порту. При не-

обходимости измените порт, предварительно сняв флажок с **Use Default**. Если есть необходимость, измените остальные настройки. Перезапустите LabVIEW. Теперь web-сервер LabVIEW запущен.

Вы можете соединиться с сервером из браузера по интернет адресу вашего компьютера. Если сервер и браузер находятся на одном компьютере, можно использовать адрес **localhost** или 127.0.0.1. Введите в адресную строку http://localhost. Загрузятся страницы, на которых подробно описано, как использовать web-сервер LabVIEW для опубликования ваших ВП и документов.

Все публикуемые web-сервером LabVIEW документы должны располагаться в корневой директории. Если вы не изменили ее, то корневой директорией является **LabVIEW 7.0/www**.

Остановимся подробнее на синтаксисе пути к файлам и URL. Когда вы сохраняете документ в корневой директории web-сервера, вы можете обратиться к нему, используя соответствующую ссылку (URL). Ссылки к документам любого web-сервера имеют следующий формат:

http://host.domain.com/path/to/document.htm

или

http://host.domain.com:8000/path/to/document.htm.

Пояснения для записи ссылок представлены в табл. 25.1

Таблица 25.1

http://	*Показывает браузеру, что проводится соединение по HTTP протоколу.*
host.domain.com	*Показывает браузеру адрес web-сервера (вашего компьютера).*
:8000	*Определяет порт, испальзуемый сервером. Если сервер использует порт 80, то это число опускается (как в первом примере).*
/path/to/document.htm	*Показывает путь к документу. Например, если корневой директорией является директория по умолчанию C:\LabVIEW 7.0\www, строка /path/to/document.htm относится к файлу C:\LabVIEW7.0\www\path\to\document.htm.*

Если путь к файлу содержит какие-либо специальные символы, такие как пробелы, специальные знаки пунктуации или символы ударения, следует зашифровать их в соответствии с условными обозначениями в ссылках. Каждый специальный символ должен быть заменен кодом %xx, где xx значение символа в шестнадцатеричной системе исчисления. Например, название файла «Experiment Page.htm» зашифровывается как «Experiment%20Page.htm», потому что 20 это значение пробела в шестнадцатеричной системе исчисления.

При помощи web-сервера LabVIEW кроме общепринятых форматов можно размещать документы трех видов: изображение лицевой панели вашего ВП в текущий момент времени, анимированное изображение лицевой панели ВП (только для пользователей Netscape) и размещение самого ВП с возможностью как наблюдения за происходящим, так и управления ВП. При любом способе размещения документа следует учесть, что публикуемый ВП должен быть загружен в память.

Чтобы поместить изображение лицевой панели ВП, web-сервер LabVIEW использует ссылку **.snap?**. Строка **.snap?** должна следовать перед названием ВП. Например, если у вас загружен ВП с именем **Chart.vi**, наберите в адресной строке

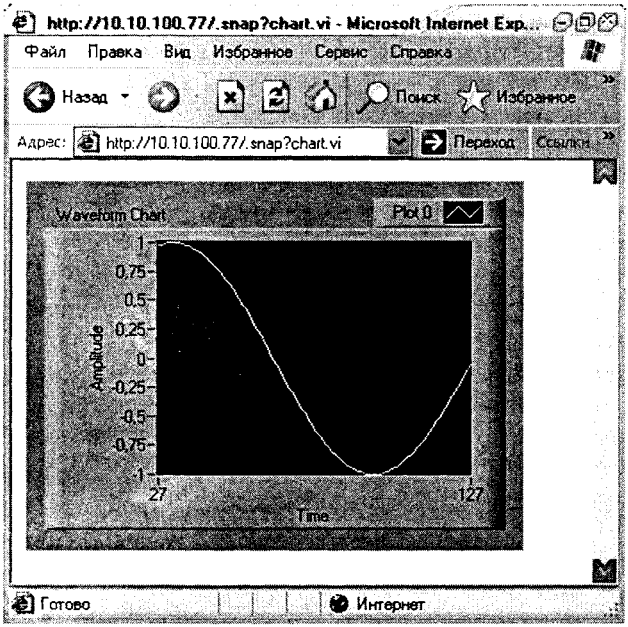


Рис. 25.1

браузера http://localhost/.snap?Chart.vi (в случае, если web-сервер запущен на другом компьютере, замените localhost на имя либо интернет адрес удаленного компьютера). Вы должны получить изображение лицевой панели ВП так, как показано на рисунке 25.1.

Вполне естественным выглядит желание поместить этот рисунок не в отдельном документе, а на какой-либо размещаемой вами странице (например, с текстом, поясняющим изображение). В этом случае вставьте в документ HTML соответствующую ссылку на изображение. Большинство редакторов HTML позволяют вставлять изображения и затем определять URL источника изображения. Напечатайте соответствующий URL в поле источника. Если вы редактируете HTML вручную, используйте один из следующих тэгов:

IMG SRC = «/.snap?Chart.vi»

IMG SRC = «http://host.domain.com/.snap?Chart.vi»

В первом примере используется относительная ссылка. Ее применяют, когда документ HTML и изображение ВП публикует один и тот же сервер. Во втором примере используется полная ссылка. Ее применяют, когда изображение ВП загружается с другого сервера.

Чтобы поместить анимированное изображение ВП используйте ссылку **.monitor?**. С строкой **.monitor?** работают так же, как и со строкой **.snap?**. Анимированное изображение ВП можно наблюдать только в браузере Netscape. В других браузерах пользователь загрузит изображение ВП в текущий момент времени.

Инструмент Web Publishing

Для упрощения процесса размещения информации о ваших ВП в сети интернет в LabVIEW предусмотрен инструмент **Web Publishing**. Запуск мастера производится в меню инструментов **Tools ⇒ Web Publishing Tool**. Окно инструмента **Web Publishing** показано на рис. 25.2.

Этот инструмент использует шаблон, который позволяет построить несложный документ HTML. С помощью шаблона (как это показано на рисунке) можно разместить на странице ВП с заголовком и пояснениями к нему.

В поле **Document Title** введите заголовок вашего документа (этот же текст отображается в строке заголовка окна браузера). В поле **Header** введите текст, размещаемый перед ВП. В поле **Footer** введите текст, размещаемый после ВП. Сам размещаемый ВП выберите в ниспадающем меню **VI Name**. В нем можно выбрать как один из открытых ВП, так и найти ВП на диске. Следующим шагом следует выбрать способ размещения вашего ВП в **Viewing Options**. Возможны три способа:

- **Embedded** – Внедряет удаленную панель ВП. Это позволяет управлять и наблюдать ВП удаленно в браузере.
- **Snapshot** – Отображает статическое изображение лицевой панели в браузере.
- **Monitor** – Отображает анимированное изображение лицевой панели в браузере.

Последние два способа размещения ВП были разобраны ранее.

Дополнительно окошко метки **Border** определяет, нужна ли черная граница вокруг изображения ВП. Окошко **Request Control** определяет, выполнять ли запрос об управлении внедренного ВП сразу после загрузки страницы.

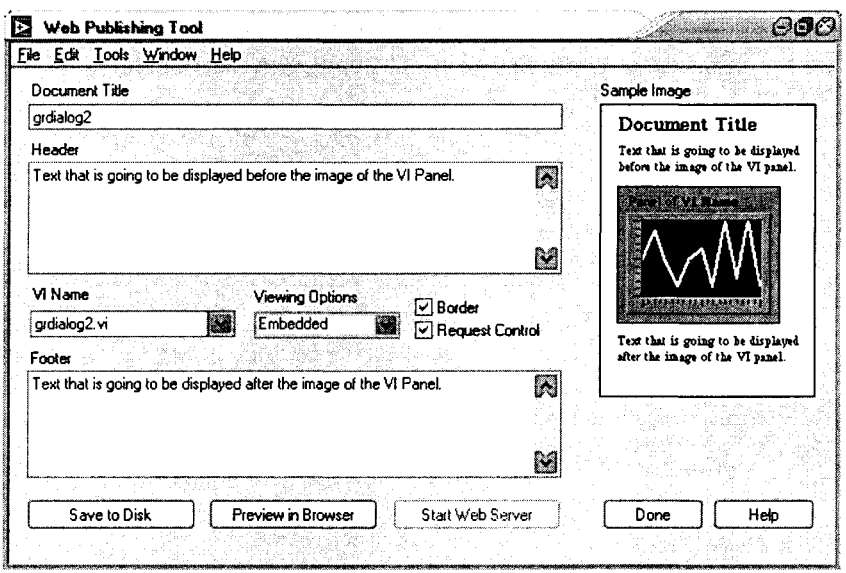


Рис. 25.2

После того, как вы записали все комментарии, выполнили все необходимые операции, воспользуйтесь предварительным просмотром страницы **Preview in Browser**. Если кнопка **Preview in Browser** неактивна, это означает, что web-сервер LabVIEW отключен. Его можно запустить кнопкой **Start Web Server**. Если кнопка **Preview in Browser** снова неактивна, то web-сервер не может быть запущен, возможно, из-за неправильной настройки TCP/IP вашего компьютера или из-за другого приложения, уже использующего указываемый в настройках порт.

Как только вы закончили создание вашей страницы, для ее сохранения нажмите **Save to Disk**. Сохранить документ следует внутри корневой директории сервера. После сохранения страницы мастер отобразит URL, который пользователи могут использовать для доступа к этому документу с удаленного компьютера. Нажав на кнопку **Connect**, вы запустите вашу страничку. Нажав на кнопку **OK**, вы вернетесь в основное диалоговое окно. Чтобы завершить работу с инструментом **Web Publishing**, нажмите **Done**.

В случае, если описанный шаблон вас не удовлетворяет, воспользуйтесь дополнительными редакторами. Наблюдение и управление ВП пользователями обеспечивается тэгом **object** и **embed** с соответствующей ссылкой на ВП. Вы также можете отредактировать страницу HTML, созданную инструментом **Web Publishing**.

Работа с удаленной лицевой панелью ВП возможна только в случае, если рассматриваемый ВП загружен в память компьютера, на котором запущен web-сервер.

Рассмотрим работу пользователя с внедренным в страницу ВП подробнее. Здесь следует отметить, что у пользователя должен быть установлен **LabVIEW Run-Time Engine** соответствующей версии, который позволит запускать приложения LabVIEW без установки непосредственно LabVIEW. Пользователи запрашивают управление ВП выбором **Request Control of VI** внизу лицевой панели, отображаемой браузером, или в контекстном меню лицевой панели. Если в данный момент нет другого пользователя, управляющего ВП, появляется сообщение о предоставлении управления (рис. 25.3).

В случае, если управление уже предоставлено другому пользователю, сервер поставит запрос в очередь. Управлять ВП пользователь сможет тогда, когда сервер обработает его запрос по завершении работы другим пользователем (пользователями) или истечении времени ограничения работы с ВП. Чтобы завершить работу с удаленной лицевой панелью и закрыть соединение, следует просто закрыть окно браузера. В случае, если пользователь желает только снять с себя право управления ВП с возможностью продолжать наблюдение, ему надо выбрать **Release Control of VI** в меню выбора внизу лицевой панели либо в контекстном меню лицевой панели. Кроме описанных функций этого меню пользователю доступны функции, которые позволяют просмотреть последнее сообщение от сервера **Show Last Message**, просмотреть оставшееся время управления **Show Control Time Remaining**.

Следует отметить, что, если вы желаете иметь возможность сохранять получаемые данные на удаленном компьютере, вам следует использовать протокол **DataSocket** или **TCP**. Об этом речь пойдет в следующей главе.

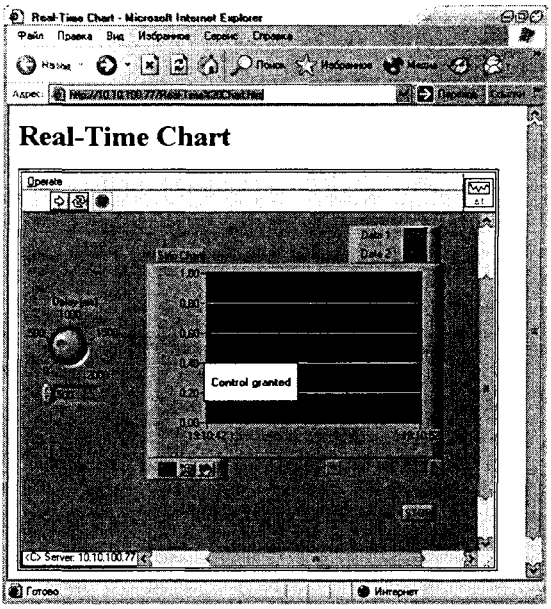


Рис. 25.3

При работе с удаленными лицевыми панелями браузер не позволяет свободно использовать пользовательский интерфейс вашего приложения. Поэтому имейте в виду, что некоторые приложения (в частности при использовании диалоговых окон и окон ВПП) не смогут выполнять заданные функции через браузер.

Кроме этого, некоторые ВП не смогут работать удаленно так же, как они работают локально. Внедренные на лицевую панель элементы управления **ActiveX** не отобразятся на удаленной лицевой панели, поскольку они прорисовываются и функционируют независимо от LabVIEW. В ВП может быть предусмотрено обращение к стандартному диалоговому окну поиска файла. Поскольку удаленный пользователь не может просматривать файловую систему, при попытке использовать этот диалог система выдаст сообщение об ошибке. В связи с этим кнопка **Browse** элемента управления **Path** в удаленных панелях неактивна. Более подробно об ограничениях, накладываемых на использование возможностей LabVIEW при размещении ВП на web-сервере, можно прочитать в сопроводительной документации LabVIEW.

Управление соединениями удаленного использования лицевой панели осуществляется через панель **Remote Panel Connection Manager**. Окно этой панели показано на рис. 25.4.

Вызвать это диалоговое окно можно в меню **Tools**. В нем показана таблица подсоединенных пользователей и график скорости подключения. В главном меню этого окна специфичными является пункт **Graph**. В пункте меню **Graph** можно спрятать график, выбрав **Hide Graph** (соответственно восстановить его можно, выбрав **Show Graph**). К тому же можно настроить параметры сбора и отображения

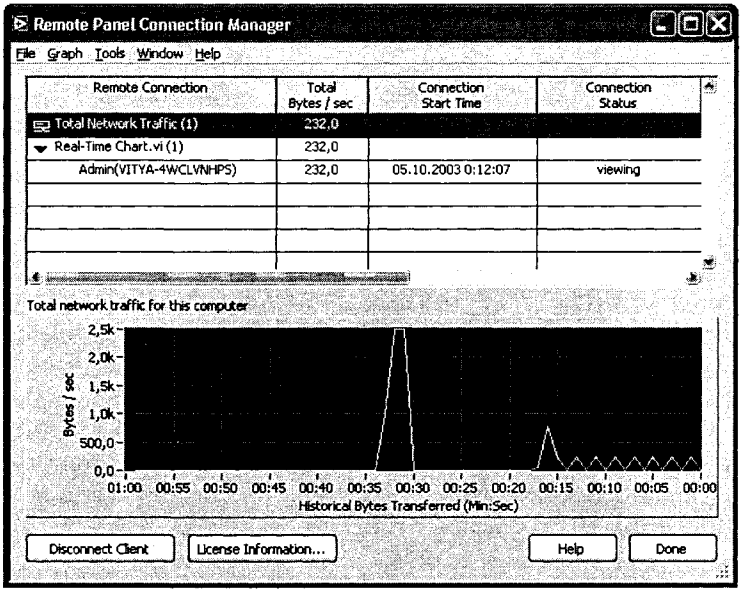


Рис. 25.4

информации о трафике. **Log History** устанавливает время, которое LabVIEW хранит данные клиента. По умолчанию стоит 5 минут, что означает, что есть информация о переданных данных за последние пять минут. **Display History** устанавливает время, информация в течение которого отображается на графике. Например, на рисунке это время установлено одной минуте. Эту величину можно установить либо равной времени **Log History**, либо меньшей.

При работе с таблицей подключенных пользователей можно просматривать график скорости подключения как для конкретного пользователя, так и для конкретного ВП. Имеется возможность отключать пользователей. Для этого надо выбрать пользователя и нажать на кнопку **Disconnect Client**.

Передавать или отнимать права управления ВП можно через лицевую панель ВП. Интерфейс передачи прав управления между пользователем и сервером похож на интерфейс получения прав в браузере. В контекстном меню лицевой панели или в меню, которое находится в левом нижнем углу лицевой панели, доступны следующие операции: **Lock Control (Unlock Control)** – блокирует (разблокирует) возможность передачи управления пользователям, **Switch Controller** – передает управление web-серверу, **Show Last Massage** – показывает последнее сообщение web-сервера.

В заключение о средствах наблюдения и управления лицевыми панелями через браузер отметим, что у обладателей **LabVIEW Full Development System** и **Application Builder** есть лицензия **remote panel license**, которая позволяет только одному клиенту работать с удаленной лицевой панелью. У обладателей **LabVIEW Professional Development System** и **Application Builder** есть лицензия для пяти

клиентов. Если первые не могут увеличить максимальное число пользователей одновременного подключения к web-серверу, то вторые для этой цели могут воспользоваться средством **NI License Manager**. Необходимость увеличить максимальное количество одновременных подключений можно оценить, нажав на кнопку **License Information** в диалоговом окне **Remote Panel Connection Manager**. В появившемся диалоговом окне (рис. 25.5) отображается список IP- адресов, в подключении которым было отказано из-за превышения максимального числа пользователей.

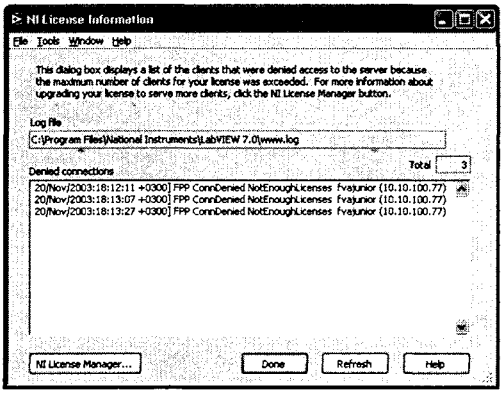


Рис. 25.5

Что касается пользователя, то для него выводится сообщение о том, что число пользователей превышает максимальное: **Remote panel connection exceeds maximum number of licenses**. Вы можете составить свое сообщение (в том числе и на русском языке). Для этого создайте в папке **labview/www** файл **LicenseErrorMessage.txt**, в котором и поместите свое сообщение.

Доступ к Web-серверу

По умолчанию к вашему web-серверу имеют доступ все компьютеры. Чтобы настроить, какие именно компьютеры могут иметь доступ, перейдите на страницу **Web Server: Browser Access** в диалоговом окне **Options** (рис. 25.6).

В поле с правой стороны вводится адрес компьютера. При нажатии на **Allow Viewing and Controlling** (разрешает наблюдение и управление лицевой панелью ВП), **Allow Viewing** (разрешает наблюдение лицевой панели) или **Deny Access** (запрещает доступ) доступ с этого компьютера будет разрешен или запрещен. Чтобы добавить указанный адрес в список доступа, нажмите **Add**. Две зеленые галочки слева от адреса появляются, когда вы разрешили наблюдение и управление лицевой панелью ВП, одна зеленая галочка появляется, когда вы разрешили только наблюдение, а красный крестик появляется, когда вы запретили доступ. Если у записи нет никаких значков, она неправильно построена. Отредактируйте или удалите ее. Для удаления записи выберите ее и нажмите **Remove**.

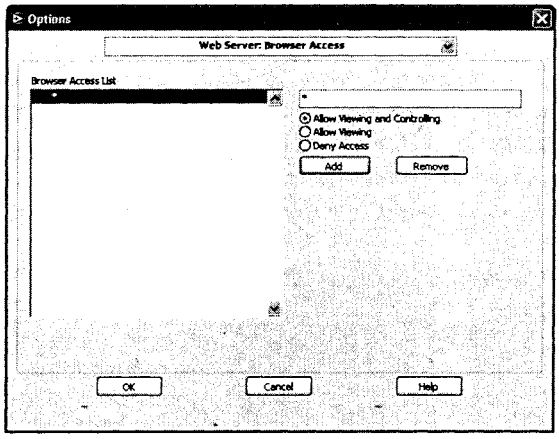


Рис. 25.6

Вводить можно IP-адрес, например 130.164.140.12, или имя домена, например **www.ni.com**. Чтобы определить группу адресов, используйте специальный символ *. Например, используя запись ***.domain.com**, вы устанавливаете правило для всего домена domain.com. Записью 130.164.* вы можете установить правило для подсети. Использовать символ * вы можете только в начале имени домена или в конце IP-адреса. Разрешение для записи, расположенной ниже в списке, преобладает над предыдущими разрешениями. Для изменения порядка записей в списке можно курсором переносить записи. Например, если вы запрещаете доступ ко всем адресам, оканчивающимся на .test.site.com, но после этой записи добавляете запись для предоставления доступа адресам a.test.site.com и b.test.site.com, доступ для них будет разрешен.

Примеры в табл. 25.2 показывают, как правильно использовать символ *.

Таблица 25.2

	Список	Статус
✓	*	*Розрешоет доступ всем одресам.*
✓	*.site.com	*Разрешает доступ всем адресам с окончанием.site.com.*
✗	public.site.com	*Запрещоет доступ этому адресу, даже если предыдущая запись разрешила доступ.*
✓	a.test.site.cam	*Разрешает доступ этому адресу, даже если предыдущая запись его запретила.*
✗	*.test.site.cam	*Зопрещает доступ адресам с окончанием .test.site.com.*
✓	130.164.123.123	*Разрешает доступ этому адресу, даже если прошлая запись его запретила.*
✗	130.164.123.*	*Запрещает доступ всем IP адресам, начинающимся на 130.164.123.*

По умолчанию доступ предоставляется ко всем ВП. Настройка, к каким ВП можно подключиться, производится на странице **Web Server: Visible VIs** в диалоговом окне **Options** (рис. 25.7).

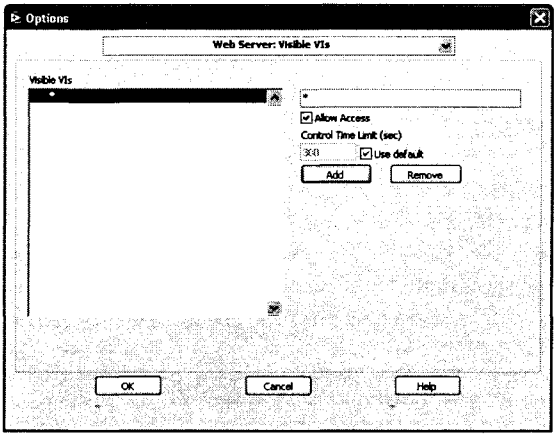


Рис. 25.7

Ввод списка ВП осуществляется аналогично вводу списка доступа компьютеров, рассмотренному ранее. Только в данном случае вводится имя ВП или путь к директории. Вы можете использовать специальные символы, представленные в табл. 25.3.

Таблица 25.3

Символ	Действие
?	*Равняется одному любому символу, исключая разделитель в написании пути.*
*	*Равняется любому набору символов, исключая разделитель в написании пути.*
**	*Равняется любому набору символов, включая разделитель в написании пути.*

Примеры в табл. 26.4 показывают, как правильно использовать символы.

Таблица 25.4

Список		Статус
✓	*	*Разрешает доступ ко всем ВП.*
✓	c:\labview\server*	*Разрешает доступ ко всем ВП из директории c:\labview\server.*
✓	c:\labview\test**	*Разрешает доступ ко всем ВП из директории c:\labview\test и ее поддиректориях.*
✗	c:\labview\test\private.vi	*Запрещает доступ к этому ВП, даже если предыдущие записи его разрешают.*
✓	srvr_*.vi	*Разрешает доступ к любому ВП, имя которого начинается со строки srvr_ и заканчивается строкой .vi.*
✗	Open?.vi	*Запрещает доступ ко всем ВП, имя которых Open?, где ? представляет собой любой символ кроме разделителя при написании пути.*

На этой же странице можно установить ограничение по времени работы с удаленным клиентом при наличии ожидающих управления пользователей.

Удаленная панель

В случае, если у пользователей в их распоряжении имеется LabVIEW, можно избежать опубликования ваших проектов на страницах в сети интернет. Аналогичную работу с внедренным в страницу ВП можно осуществить через диалоговое окно подключения к удаленной панели непосредственно через LabVIEW. Указанное окно вызывается через пункт меню **Operate ⇒ Connect to Remote Panel** (рис. 25.8).

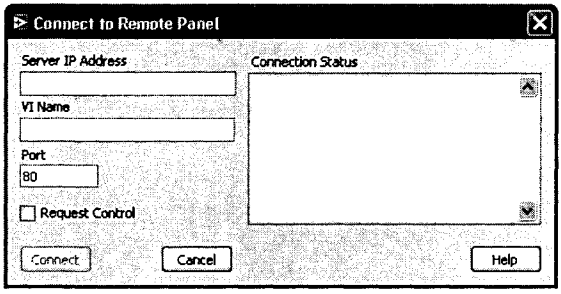


Рис. 25.8

В поле **Server IP Address** вводится IP-адрес удаленного компьютера, в поле **VI Name** – имя ВП. При нажатии на кнопку **Connect** удаленная панель осуществит подключение к ВП. Взаимодействие пользователя с удаленной панелью происходит абсолютно так же, как и с удаленной панелью, загруженной в браузере. В случае возникновения ошибок при соединении, необходимая информация появляется в поле **Connection Status**.

Выводы

Встроенный web-сервер позволяет размещать страницы в сети интернет, не используя какое-либо дополнительное программное обеспечение. Инструмент **Web Publishing** значительно упрощает процесс создания страниц с изображением лицевой панели или внедренным ВП. Имеется возможность удаленной работы с ВП.

Лекция 26

Технология DataSocket

*Описывается передача данных посредством технологии **DataSocket** в сетях интернет и интранет.*

Возможность размещения в сети интернет страниц с изображением своего ВП или даже с внедрением ВП в страницу можно рассматривать как значительное удобство при представлении вашей работы. Цели использования web-сервера могут быть различными от передачи просто изображения ВП клиентам до использования удаленной панели для выполнения лабораторной работы при дистанционном образовании. Пользователи могут наблюдать за тем, что происходит на лицевой панели вашего ВП, а могут и управлять вашим ВП. Однако более важным для пользователей в их работе может оказаться возможность обмениваться не картинками, а исходными, измеренными и обработанными данными. National Instruments предлагает технологию **DataSocket**, которая упрощает передачу данных между компьютерами и приложениями, совершенствует средства автоматизации физических измерений. **DataSocket** – это новая технология, основанная на промышленном стандарте TCP/IP. При этом использование этой технологии не требует от специалистов знаний и опыта работы с низкоуровневым программированием **TCP**. Все что требуется от программиста, это открыть соединение **DataSocket** и записать необходимые данные.

Для размещения и считывания данных при использовании технологии **DataSocket**, можно использовать следующие протоколы:

- **DataSocket Transport Protocol (dstp)** – собственный протокол National Instruments для передачи данных. Для использования этого протокола следует запустить сервер **DataSocket**.
- **OLE for Process Protocol (opc)** – протокол, разработанный специально для размещения данных, поступающих в реальном масштабе времени. Для использования этого протокола следует запустить сервер OPC.
- **logoc** – внутренняя технология National Instruments для передачи данных между сетью и локальным компьютером.

Кроме этих протоколов поддерживаются также **ftp** и **file**.

Поскольку протоколы **dstp**, **opc** и **logos** могут обновлять удаленные и локальные элементы управления и индикации, используйте их для размещения и считывания изменяющихся данных. Для считывания данных из файлов используйте протоколы **ftp** и **file**. Они не позволят обновлять объекты лицевой панели. Примеры записи ссылок для указанных протоколов представлены в табл. 26.1

Таблица 26.1

Протокол	*Пример*
dstp	*dstp://servername.com/item*
	где dstp – наименование протокола, servernome.com – адрес сервера, item – уникальная метка данных.
opc	*opc:\national instruments.opctest\item*
logos	*logos://computer_name/process/data_item_name*
ftp	*ftp://ftp.ni.com/datasocket/ping.wav*
file	*file:c:\mydata\ping.wav*
	file:\\machine\mydata\ping.wav

Технология **DataSocket** позволяет размещать и считывать данные следующих типов:

- текст, воспроизводимый строковым элементом индикации;
- табличный текст, рассматриваемый как массив;
- звук (файлы с расширением .wav);
- переменные типа вариант.

DataSocket состоит из двух компонентов: **DataSocket API** и **DataSocket** сервера. **DataSocket API** представляет собой программный интерфейс приложения, который взаимодействует с различными типами данных различных языков программирования. **DataSocket API** автоматически преобразует данные измерений в пересылаемый по сети поток байтов. Считывающее приложение **DataSocket** автоматически преобразует поток байтов обратно в исходную форму. Подобное автоматическое преобразование устраняет сложность работы с сетью, которое предполагает написание значительного объема кода при использовании библиотек **TCP/IP**.

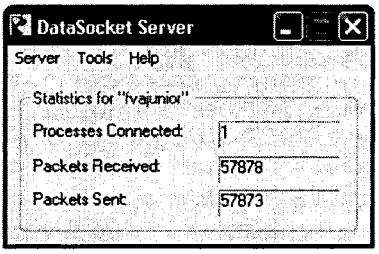


Рис. 26.1

DataSocket сервер обеспечивает интернет соединение. Запуск **DataSocket** сервера осуществляется через меню **Пуск** ⇒ **Программы** ⇒ **National Instruments** ⇒ **DataSocket** ⇒ **DataSocket Server** (рис. 26.1). Кроме LabVIEW технология **DataSocket** поддерживается другими средами программирования, такими как C++, Visual Basic и Java. Сервер **DataSocket** является отдельной программой, которая управляет подключением клиентов. Размещение данных предполагает использование трех частей: издатель, **DataSocket Server** и подписчик.

Использование DataSocket на лицевой панели

Технология **DataSocket** обеспечивает доступ к нескольким объектам ввода и вывода данных через лицевую панель. Технология достаточно проста. Если вы желаете разделить элемент индикации с другими компьютерами в сети, введите его URL. Пользователи на других компьютерах поместят этот элемент индикации на лицевую панель, введут URL, и тогда смогут получить данные этого элемента индикации. Таким образом, вы размещаете данные объекта лицевой панели, а пользователи обращаются к этому объекту через свою лицевую панель и считывают информацию этого объекта.

Соединение по протоколу **DataSocket** принципиально отличается от подключения к ВП посредством web-сервера тем, что здесь осуществляется взаимный обмен данными. Каждому элементу управления или индикации лицевой панели можно указать адрес для размещения либо считывания его данных через подключение **DataSocket**. При этом графическое изображение элементов, как это происходит при работе с удаленным ВП, не передается.

Соединение **DataSocket** используют для размещения данных элемента управления. При этом удаленные пользователи настраивают свои элементы управления или индикации на считывание информации с этого элемента управления. Например, вы поместили на лицевую панель элемент управления **Knob**. Пользователь на другом компьютере может подключиться к этому элементу управления, чтобы соединить его к своим подпрограммам и функциям или просто наблюдать его изменение на элементе индикации. Это показано на рис. 26.2. Слева показан элемент управления, данные которого опубликовали с помощью протокола **DataSocket**. Справа показан элемент управления и элемент индикации, которые настроили считывать данные первого элемента управления. На рисунке видно, что независимо от того, каким именно выбран элемент управления, являющийся источником данных, вид подключенных к нему элементов выбирает удаленный пользователь. Однако эти объекты должны поддерживать одинаковый тип данных. Зеленые прямоугольники у рассматриваемых объектов показывают, что соединение активно (рис. 26.3). В случае ошибки подключения этот прямоугольник окрашен красным цветом. При наведении на прямоугольник во всплывающей подсказке можно прочесть, в чем состоит ошибка. Когда ВП не запущен, прямоугольник окрашен в серый цвет.

а) б)

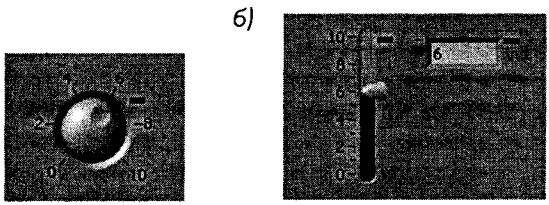


Рис. 26.2

Рис. 26.3

То же относится и к размещению данных элемента индикации. Удаленные пользователи также могут использовать элемент управления или индикации для применения получаемой информации в своих ВП или наблюдения ее изменения.

Таким образом, вы можете разместить данные элемента управления и индикации или подключиться к ним с целью наблюдения или использования данных в своем ВП. Кроме того, вы можете одновременно отправлять и считывать данные элемента управления. Т.е. удаленные пользователи в своих ВП устанавливают свои значения элемента управления, и эта информация передается вашему ВП. Когда вы запускаете ВП, элемент управления вашей лицевой панели восстанавливает текущее значение элемента управления другого ВП или приложения, которое было размещено через соединение **DataSocket**. Когда пользователь изменяет значение элемента управления, через соединение **DataSocket** новое значение передается вашему элементу управления. Если после этого вы установите другое значение, оно будет передано другим пользователям.

Настройка описываемой функции осуществляется в диалоговом окне **DataSocket Connection** (рис. 26.4). Для вызова диалогового окна **DataSocket Connection** в контекстном меню какого-либо объекта лицевой панели выберите **Data Operation ⇒ DataSocket Connection.**

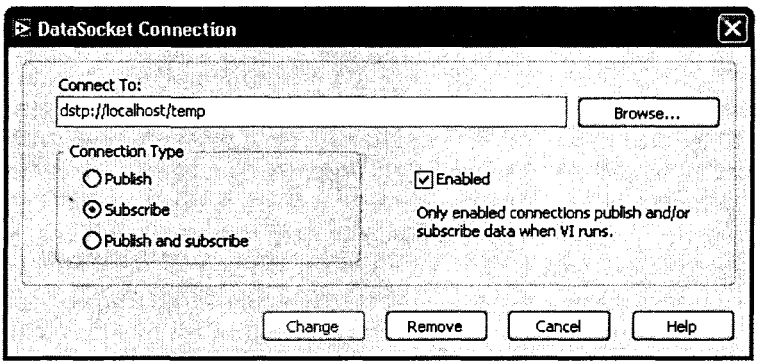


Рис. 26.4

В строке **Connect To** записывают адрес (URL), по которому вы публикуете или считываете данные. Синтаксис URL очень похож на общепринятый, используемый в браузерах.

Далее следует выбрать, что именно вы желаете сделать: разделять информацию с другими пользователями, считывать ее или одновременно разделять и считывать. В зависимости от этого выберите **Publish**, **Subscribe** или **Publish and Subscribe**.

Флажок **Enabled** показывает, разрешено ли подключение **DataSocket** для этого элемента. Если вы желаете закрыть соединение, и при этом сохранить адрес ссылки, уберите этот флажок.

Чтобы завершить операцию, нажмите **Attach**. Если же вы желаете удалить данные соединения, то нажмите **Remove**.

Использование DataSocket на блок-диаграмме

Функции DataSocket

Использование функций **DataSocket (Communication ⇒ DataSocket)** на блок-диаграмме позволяет считывать и записывать данные программно. Палитра функций **DataSocket** показана на рис. 26.5.

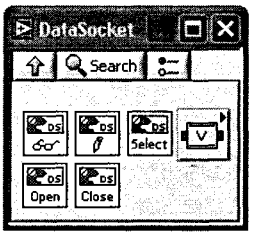


Рис. 26.5

Функции палитры **DataSocket** представлены в табл. 26.2

Таблица 26.2

Иконка	*Назначение*
	***DataSocket Read*** *– Считывает значение данных, указываемых в* ***connection in****. Вход* ***connection in*** *может быть строковой ссылкой* ***DataSocket*** *(используется по умолчанию) или ссылка* ***refnum*** *соединения* ***DataSocket****.*
	***DataSocket Write*** *– Записывает значения донных, указывоемых в* ***connection in****.*
	***DataSocket Select URL*** *– Вызывает диологовое окно для выбара источника или приемника данных и возвращает ссылку на указанные данные. Использовать этот ВП рекомендуется только в том случае, когда URL данных неизвестен и его требуется найти.*
	***DataSocket Open*** *– Открывает соединение* ***DataSocket*** *с указываемой ссылкой URL в указанном режиме mode. Режим может быть одним из пяти:* ***Read*** *– режим считывания,* ***Write*** *– режим записи,* ***ReadWrite*** *– режим записи и считывания,* ***BufferedRead*** *– буферированный режим считывания,* ***BufferedReadWrite*** *– буферированный режим считывания и записи.*
	***DataSocket Close*** *– Закрывает указываемое соединение* ***DataSocket****.*

Пример 26.1. Использование функции DataSocket Write

Рассмотрим блок-диаграмму, показанную на рис. 26.6. Функция записи позволяет передать данные через соединение **DataSocket**. В данном случае используется числовое значение. В общем случае функции **DataSocket** являются полиморфными и работают с большинством типов данных. Использование строкового параметра dstp://localhost/temp означает то, что эти данные впоследствии могут быть считаны по метке temp с локальной машины.

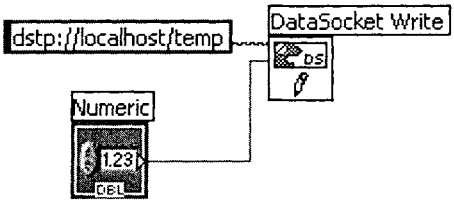


Рис. 26.6

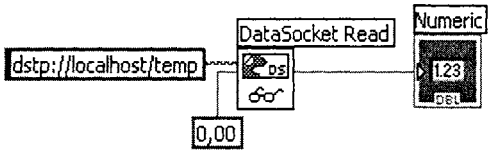


Рис. 26.7

Пример 26.2. Использование DataSocket Read

Рассмотрим блок-диаграмму, показанную на рис. 26.7. Функция чтения позволяет считывать данные через соединение **DataSocket**. На вход **connection in** подается ссылка, по которой будут считываться данные. С помощью входа **type** указывается предполагаемый тип данных. В случае, если последний не определен, то функция чтения данных возвращает данные в виде варианта, с которым можно работать через палитру функций **Variant**, к которой можно перейти через палитру функций **DataSocket**. В частности, чтобы преобразовать вариант в какой-либо тип данных, можно воспользоваться функцией **Variant To Data**.

Буферирование данных

При использовании протокола **DataSocket** по умолчанию подписчики считывают самые последние данные. Когда значения записываются на сервер быстрее, чем клиент их считывает, более новые значения записываются вместо старых, необработанных значений. Такая потеря необработанных данных может наблюдаться как на сервере, так и в клиенте. Это не является проблемой, если вам необходимо получать самые последние данные. Однако некоторые задачи предполагают использование каждого значения, размещенного на сервере. Вам нужно буферизовать данные со стороны клиента, а также настроить буферизацию со стороны сервера при помощи **DataSocket Server Manager** (вызывается из того же меню программ, что и сервер **Programs ⇒ National Instruments ⇒ DataSocket ⇒ DataSocket Server Manager**). Кроме протокола **DataSocket** буферизация применяется и в других протоколах, таких как **opc**, **logos**, **file**. Рассмотрим пример, позволяющий буферировать данные. Описание указанного примера вы можете также найти в системе **LabVIEW Help**.

Задание 26.1. Буферирование данных

Составьте блок-диаграмму, показанную на рис. 26.8.

Для этого выполните следующие действия:

Откройте соединение **DataSocket**. Поместите на блок-диаграмму функцию **DataSocket Open**. Ко входу URL функции **DataSocket Open** подключите ссылку на данные, названные **bufdata**. Эти данные должны быть опубликованы на сервере

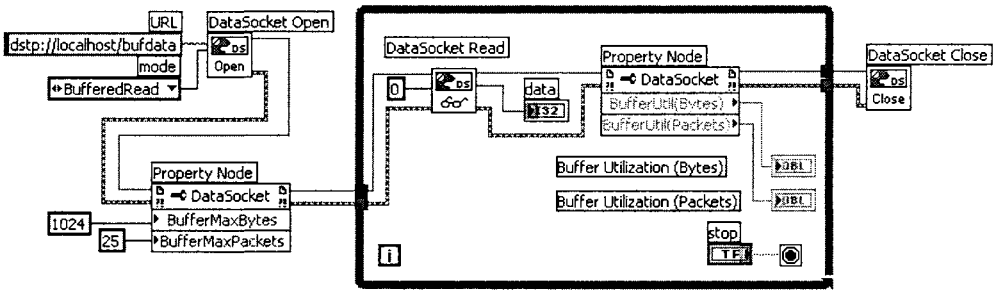


Рис. 26.8

локального компьютера (о чем говорит имя сервера **localhost**). Чтобы рассмотреть результаты этого примера, следует запустить на локальной машине сервер **DataSocket** и опубликовать данные с меткой **bufdata**. Буферизация данных будет осуществляться только в том случае, если на вход **mode** функции **DataSocket Open** подано **BufferedRead** или **BufferedReadWrite**. Еще три возможных варианта **Read**, **Write** и **ReadWrite** соответствуют соответственно командам **Subscribe**, **Publish** и **Subscribe and Publish**, выбираемым на лицевой панели. Следует отметить, что промежуточное хранение данных невозможно при работе с соединением **DataSocket** через лицевую панель. Итак, создайте константу для входа **mode** и выберите **BufferedRead**.

Установите размер буфера для промежуточного хранения данных клиента **DataSocket**. Поместите на блок-диаграмму узел **Property Node** (**Application Control** ⇒ **Property Node**). В контекстном меню установите, что осуществляется настройка параметров **DataSocket** (**Select Class** ⇒ **DataSocket** ⇒ **DataSocket**). Выберите **Buffer Maximum Bytes**, добавьте еще строку, которая автоматически установится на **Buffer Maximum Packets**. В контекстном меню установите, что все указанные параметры следует считывать (**Change All To Write**). Создайте константы для каждого из параметров. Первый установите 1024, а второй – 25. Это означает, что максимальный размер буфера **DataSocket** устанавливается равным 1024 байт и максимальное число пакетов, временно сохраняемое на стороне клиента, равно 25.

Настройте считывание данных. Поместите на блок-диаграмму структуру **While Loop** (для прекращения работы цикла предусмотрите кнопку завершения работы). На блок-диаграмму структуры поместите функцию **DataSocketRead**. Укажите тип данных, считываемых с сервера. Для этого соедините константу соответствующего типа со входом **type**. Создайте элемент индикации для считываемых данных.

Обеспечьте наблюдение за используемым размером буфера. Для этого на блок-диаграмму структуры поместите узел **Property Node**. Выберите управление параметрами **DataSocket** так, как описано в пункте 2. Выберите параметры **Buffer Utilization Bytes** и **Buffer Utilization Packets**. Создайте для них элементы индикации.

Закройте соединение **DataSocket**, поместив на блок-диаграмму функцию **DataSocket Close**. Начиная с функции **DataSocket Open**, последовательно соедините все

выходы и входы функций ссылкой на подключение **DataSocket**. То же самое сделайте с выходами и входами ошибок.

Таким образом, можно выполнять считывание данных без потери данных в результате перезаписи старых значений более новыми. Однако следует отметить, что буферизация **DataSocket** не гарантирует доставки данных в случае, если буфер сервера или клиента переполнен. Тогда происходит перезапись старых значений новыми. Рассмотренный пример позволяет наблюдать используемые возможности буфера, которые можно использовать для вывода сообщений о переполнении буфера. Чтобы обнаружить потерянные данные в потоке данных, на стороне сервера задайте публикуемым данным уникальный идентификатор. В результате проверка потерянных данных может осуществляться по пропущенным идентификаторам данных на стороне подписчика. О том, какие инструменты для решения этой задачи можно использовать, речь пойдет ниже.

Тип данных вариант

В некоторых случаях ВП или другое приложение, которое считывает данные, не могут преобразовать их обратно в исходный тип данных. Ранее упоминалось, что протокол **DataSocket** работает с данными типа вариант. Это обусловлено тем, что протокол может использоваться различным приложениями, не имеющими отношения непосредственно к LabVIEW. В таких случаях необходимо использовать функции для работы с типом данных вариант.

Что касается упомянутой выше задачи проверки потери данных, то ее решение сводится к тому, что на стороне сервера исходные данные преобразуются в вариант, к которому при помощи функции **Set Variant Attribute** добавляется признак – идентификационный номер с некоторой меткой, допустим с меткой **id**. На стороне сервера по метке **id** идентификационный номер можно выделить с помощью функции **Get Variant Attribute**. Покажем на примере, каким образом могут использоваться функции палитры **Variant** во время передачи данных посредством протокола **DataSocket**.

Задание 26.2. Добавление к измеренным данным отметки времени

Рассмотрим процесс добавления отметки времени к вашим данным. Составим блок-диаграмму, изображенную на рис. 26.9.

Для этого выполним следующие действия:

Поместим на блок-диаграмму структуру **While Loop**. Добавим кнопку завершения работы структуры. Добавим на блок-диаграмму структуры функцию ожидания **Wait (ms)**. Установим время ожидания равным 1 секунде.

Преобразуем необходимые данные в вариант. Соединим терминал элемента управления **Voltage** со входом функции **To Variant**.

Добавим вашим данным (в нашем примере эти данные названы **Voltage**) информацию о том, в какое время они получены. Поместим на блок-диаграмму структуры

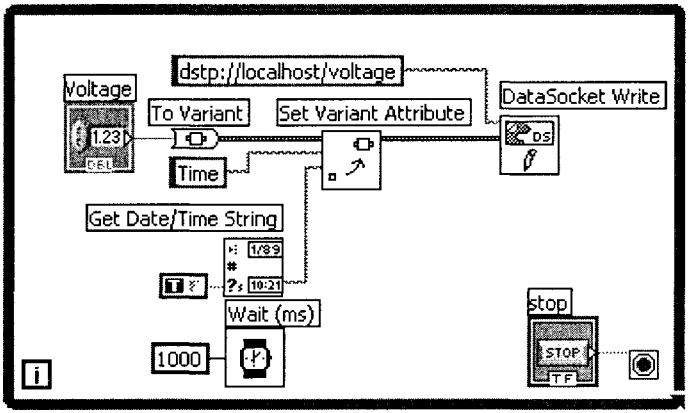


Рис. 26.9

функцию **Set Variant Attribute**. Ко входу **Variant** подключим провод с вашими данными, преобразованными в вариант. На входе **name** укажем метку добавляемого признака, например **Time**. Ко входу **value** нужно подключить само значение добавляемого признака. В нашем случае это текущее время. Текущее время в виде строки получим с помощью функции **Get Date/Time String**. Поскольку для того, чтобы на выходе этой функции было именно текущее время, следует значение входа **want seconds? (F)** установить в положение **True**. Иначе эта функция выдаст текущую дату. И наконец, выход **time string** функции **Get Date/Time String** соединим со входом **value** функции **Set Variant Attribute**.

Разместим полученные данные на сервере. Поместим на блок-диаграмму структуры **While Loop** функцию **DataSocket Write**. Укажем обязательные входные данные: к **connection in** подключите строку с URL, со входом **data** соедините выход **Variant** out функции **Set Variant Attribute**. В нашем примере ссылку запишем так: dstp://localhost/voltage. Строку localhost следует заменить именем компьютера или интернет адресом, если сервер запущен на другой машине.

Таким образом, на сервере размещаются данные об измеренном напряжении. Для того, чтобы на сервере можно было бы получить кроме самого значения напряжения также и время измерения, данные о напряжении преобразуются в вариант. К рассматриваемым данным добавляется отметка о текущем времени. В результате на сервере размещаются как сами данные, так и время, в которое они были получены.

Задание 26.3. Получение измеренных данных и отметок времени

Рассмотрим процесс преобразования данных к исходному типу со стороны клиента. Составим блок-диаграмму, изображенную на рис. 26.10.

Для этого выполним следующие действия:

Поместим на блок-диаграмму структуру **While Loop**. Добавьте кнопку завершения работы структуры.

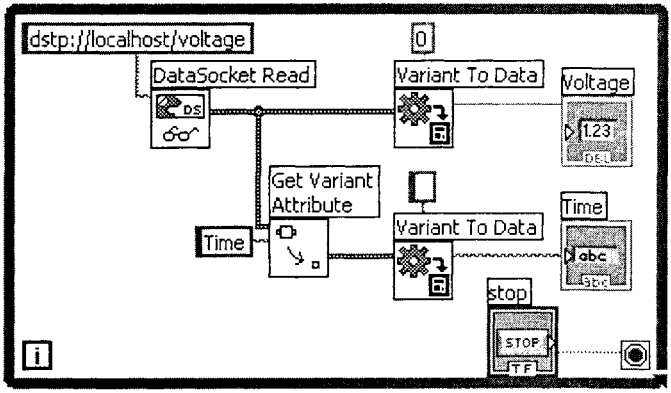


Рис. 26.10

Настроим чтение данных с сервера. Поместите функцию **DataSocket Read** на блок-диаграмму структуры. Укажите URL для обязательного входа **connection in**. В нашем примере ссылку запишем так: dstp://localhost/voltage. Строку **localhost** следует заменить именем компьютера или интернет адресом, если сервер запущен на другой машине.

Выделим из варианта исходные данные. С помощью функции **Variant To Data** выделите изначально рассматриваемые данные с меткой **Voltage**. Тип извлекаемых данных задается на вход **type**. Можно соединить его с любой константой. Перед тем как использовать функцию **Variant To Data** для получения времени, его следует считать с помощью функции **Get Variant Attribute**. Для того, чтобы считать указанное свойство (в данном случае времени), необходимо его название соединить со входом **name** функции **Get Variant Attribute**. Аналогично с получением значения напряжения на вход **type** следует задать строковый тип данных.

Таким образом, клиент считывает с сервера данные об измеренном напряжении и одновременно получает информацию о том, в какой момент времени это измерение было сделано.

Выводы

Протокол **DataSocket** позволяет обмениваться данными с компьютерами посредством локальной сети или интернета, а также с другими приложениями. Непосредственно предавать данные элементов управления и индикации следует через лицевую панель. Для решения специальных задач в LabVIEW предусмотрена палитра функций **DataSocket**. Для того, чтобы свести потери данных к минимуму целесообразно применять буферирование данных.

Лекция 27

Разработка больших проектов

Описывается возможность управления проектами, состоящими из большого количества виртуальных подприборов, при помощи инструментов просмотра иерархии и сравнения.

Под большим проектом можно понимать любой ВП, в котором вы используете достаточное количество структур с несколькими вложенными блок-диаграммами, в котором имеется несколько уровней вызова ВПП. В котором, в конце концов, у вас возникают сложности, когда вы пытаетесь вспомнить, как работает ВПП, сделанными вами ранее.

Предполагается, что при изучении последующих глав вы уже пытались или даже создали ВП, который можно считать сложным проектом. Вы уже встретились с некоторыми трудностями при работе с файлами и ВПП. Материал последних глав посвящен описанию инструментов, облегчающих разработку большого проекта, позволяющих быстрее ориентироваться среди множества объектов блок-диаграммы. Кроме этого будет показано, каким образом можно упростить сам код вашего ВП, на что обратить внимание, чтобы не нагромождать блок-диаграмму излишними объектами. Далее представлены возможности LabVIEW для создания разного рода документации к вашему продукту. И, наконец, описывается, как из готового проекта создать независимо от LabVIEW выполняемую программу.

Иерархия виртуальных приборов

При создании сложных проектов с множеством подпрограмм удобно изображать структуру проекта, иерархию используемых компонентов. Окно иерархии LabVIEW позволяет рассмотреть весь проект, графически представленный диаграммой вызываемых подпрограмм, определений типов и глобальных переменных. Окно иерархии ВП вызывается из пункта меню **Browse** ⇒ **Show VI Hierarchy**. Пример иерархии представлен на рис. 27.1.

В окне иерархии представлены иконки всех узлов, к которым обращаются загруженные в память ВП. При наведении курсора на какой-либо объект LabVIEW по-

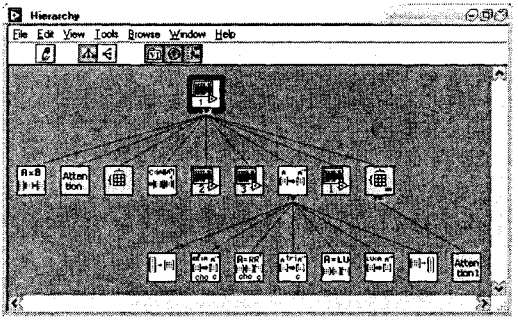


Рис. 27.1

казывает его имя. С иконками ВП в окне иерархии можно работать также как и на блок-диаграмме, в частности: переносить ВП из окна иерархии на блок-диаграмму в качестве ВПП, выбирать и копировать один или несколько узлов в буфер, затем вставляя их в какие-либо блок-диаграммы. Двойное нажатие клавиши мыши по иконке ВП открывает его лицевую панель.

Наблюдение за иерархией рассматриваемого ВП обеспечивается стрелками сворачивания или разворачивания иконок ВПП текущего узла. Если узел имеет ВПП, то под его иконкой имеется стрелка (рис. 27.2). Черная стрелка (на рисунке направленная вниз) означает, что на более низшем уровне показаны все ВПП выделенного красной рамкой узла. При нажатии на эту стрелку, все ВПП сворачиваются, а стрелка меняет направление и становится красной.

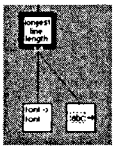


Рис. 27.2

Скрывая и раскрывая различные части проекта, имеется возможность исследовать и наглядно представить его структуру. Вывести всю иерархию ВП можно, выбрав из контекстного меню **Show All VIs**. Список всех ВП, загруженных в память, также можно получить программными средствами с помощью свойства ВП **All VIs in Memory**.

Панель инструментов окна **Hierarchy** содержит кнопки, представленные в табл. 27.1

Таблица 27.1

	*Во время работы с иерархией ВП, перемещая, скрывая или показывая узлы, может нарушиться построение иерархии. При разворачивании и сворачивании иерархий ВПП могут появиться пересечения связывающих линий. При перемещении узлы могут наезжать один на другой. Кнопка* ***Redo Layout*** *позволяет восстановить первоначальную упорядоченность диаграммы.*
	*Кнопка* ***Vertical Layout*** *устанавливает вертикальное направление разворачивания иерархии. Рассматриваемый ВП помещается наверх. Вызываемые им ВПП располагаются на уровень ниже. Еще ниже располагаются подпрограммы ВПП и т.д.*
	*Кнопка* ***Horizontal Layout*** *разворачивает иерархию слева направо.*
	*Кнопка* ***Include VI Lib*** *позволяет исключить ВП, предоставляемые LabVIEW в библиотеке виртуальных приборов, которые размещаются в директории* ***labview\vi.lib****. По умолчанию ВП из библиотеки LabVIEW включаются в иерархию ВП.*

Таблица 27.1 (окончание)

	*Кнопка* ***Include Globals*** *устанавливает показывать или не показывать в иерархии ВП глобальные переменные.*
	*Кнопка* ***Include Type Definitions*** *устанавливает отображать или не отображать в иерархии ВП определения типов.*

При работе с иерархией ВП следует обладать информацией о специальных знаках, которые могут сопровождать иконки узлов. В случае, если к ВПП подведены не все возможные линии от вызывающих их ВП, то сбоку появляется синяя стрелка. На рис. 27.3 показан пример возникновения синих стрелок. На левом рисунке нижние иконки сопровождаются стрелками, а на правом – нет, поскольку показаны все связи.

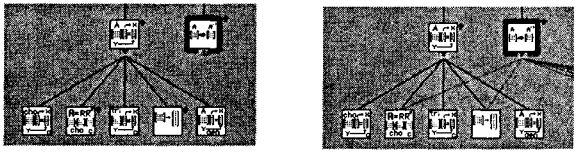


Рис. 27.3

В случае, если в настройках ВПП указано, что при его вызове работа всего проекта приостанавливается (включение данной опции производится через пункт **VI Properties** контекстного меню иконки и выбора на вкладке **Execution** опции **Suspend When Called**), то на иконке появляется зеленый восклицательный знак. Если во время выполнения ВП действие программы уже приостановлено, то на иконке появляется красный восклицательный знак.

Искать необходимый объект в окне иерархии можно самостоятельно, просматривая иконки и всплывающие надписи к ним. Имеет смысл также проводить поиск узла по имени.

Запустите поиск, печатая имя узла, который вы желаете найти. Когда окно иерархии активно, по любому печатаемому тексту осуществляется поиск. В то время как вы печатаете текст, в левом верхнем углу появляется строка поиска, отображающая печатаемый текст (как показано на рис. 27.4). LabVIEW подсвечивает узел с именем, которое совпадает с текстом в строке поиска.

Если узлов с именем, начинающимся на текст в строке поиска, больше одного, нажмите клавишу **Enter**, чтобы перейти к следующему узлу, удовлетворяющему условиям поиска. Нажмите клавиши **Shift-Enter**, чтобы перейти к предыдущему узлу, удовлетворяющему условиям поиска.

В заключение опишем еще одну возможность упростить работу с иерархией проекта. Кроме вызова диалогового окна иерархии в пункте **Browse** главного меню также можно выбрать пункты, которые формируют список различных категорий ВП. **This VI's Callers** отображает список ВП, которые вызывают текущий ВП в качестве подприбора. **This VI's SubVIs** показывает список подприборов ВП (рис. 27.5). Этот список не включает подприборы второго и более уровня в иерар-

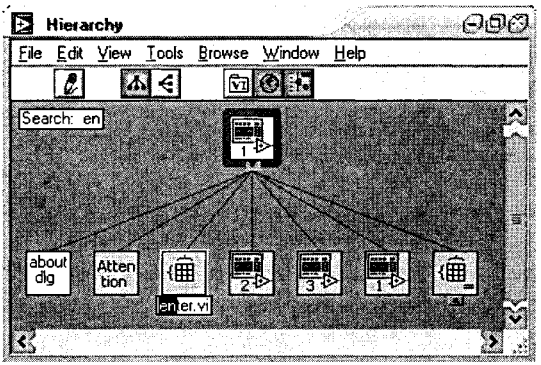


Рис. 27.4

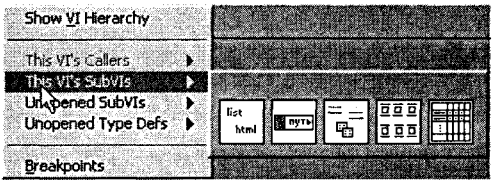


Рис. 27.5

хии. **Unopened SubVIs** отображает список всех неоткрытых подприборов текущего ВП, в том числе и подприборы второго и более уровня в иерархии. **Unopened Type Defs** показывает список всего неоткрытых определителей типа текущего ВП.

Инструмент сравнения проектов

При работе с большим проектом иногда полезно сохранять все файлы проекта в другой каталог. Это позволит зафиксировать проект на каком-то промежуточном этапе, например, после отладки какого-либо функционального блока. Возможны ситуации, когда работа над проектом проводится на разных компьютерах. При этом достаточно легко запутаться, в каких файлах внесены изменения. Причем добавляться изменения могут как к более старым файлам, так и к новым. Разобраться с одинаковыми файлами различных версий поможет инструмент **Compare**. Инструмент **Compare** позволяет сравнивать как отдельные ВП, так и их иерархии.

Сравнение двух виртуальных приборов

Для сравнения двух ВП вызывается диалоговое окно **Compare VIs**, выбрав **Tools ⇒ Compare ⇒ Compare VIs**. Оно показано на рис. 27.6.

Следует отметить, что перед запуском процесса сравнения сравниваемые файлы необходимо загрузить в память компьютера. А это означает, что у них должны быть разные имена. Если в LabVIEW открыты два ВП, то они автоматически вписываются в поля файлов сравнения. Иначе их следует выбрать, нажав на кнопку **Select**.

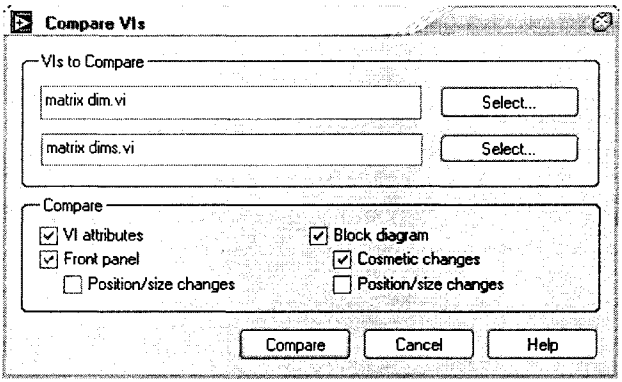


Рис. 27.6

Рассмотрим свойства сравнения Compare, по которым LabVIEW осуществит сравнение ВП. **VI attributes** позволит сравнить свойства двух ВП, установленных в диалоговом окне **VI Properties** пункта главного меню **File**. **Front panel** сравнивает объекты лицевой панели. При этом можно включить или отключить сравнения местоположения и размеров объектов лицевой панели выбором **Position/size changes**. **Block diagram** позволит найти различия в блок-диаграммах. При этом можно включить или исключить из рассмотрения косметические изменения **Cosmetic changes**, то есть те изменения, которые не влияют на процесс выполнения ВП (например, различие в цвете, видимые поддиаграммы в структурах **Case** и **Stacked Sequence** и т.д.). Отдельно выбирается, сравнивать ли ВП по местоположению и размеру объектов блок-диаграммы. Чтобы произвести анализ, нажмите **Compare**.

Результаты сравнения появляются в диалоговом окне **Differences** (см. рис. 27.7). Кроме этого диалогового окна сверху экрана появляются дополнительные окна лицевых панелей и/или блок-диаграмм (в зависимости от настроек сравнения). Окно **Differences** состоит из двух полей: поля **Differences**, в котором дан список различий двух ВП, и поля **Details**, которое раскрывает подробности этих различий. Нажмите на **Show Difference**, LabVIEW выделит объекты лицевой панели или

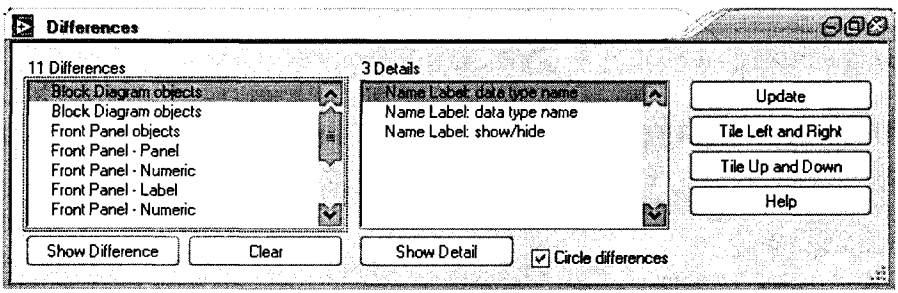


Рис. 27.7

блок-диаграммы, между которыми имеется различие. Этой кнопке диалогового окна соответствует двойное нажатие левой клавишей мыши по выбранному элементу из списка. Нажмите на **Show Detail** (этой кнопке также соответствует двойное нажатие клавиши мыши), LabVIEW покажет конкретную составляющую этого различия. Просмотренные вами различия и их подробности помечаются галочкой слева от списка. Если установить флажок **Circle differences**, то ко всему прочему, выделяемые объекты будут обведены красным кружком. Пример того, как осуществляется сравнение, показан на рис. 27.8.

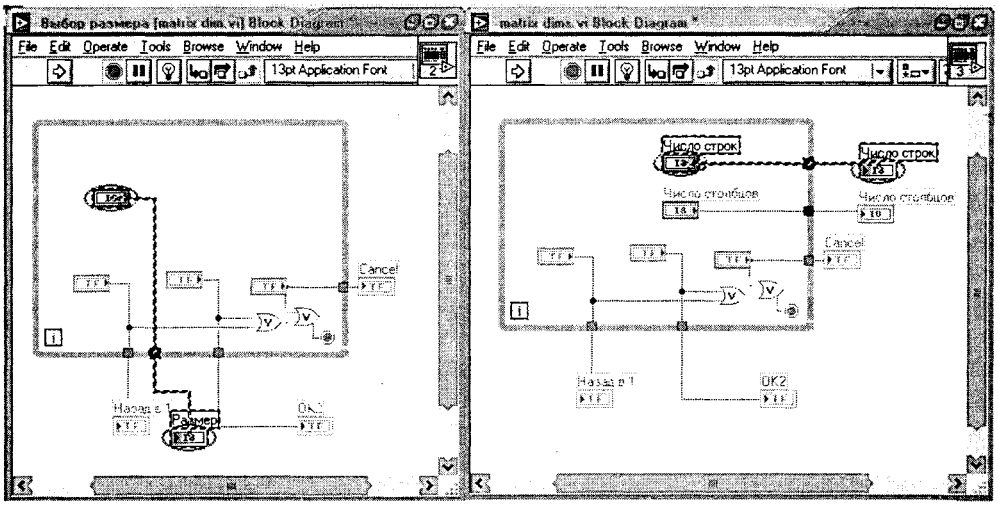


Рис. 27.8

На рисунке окна блок-диаграмм расположены справа и слева (исходный или первый ВП показывается слева). Это соответствует режиму отображения **Tile Left and Right**. Чтобы расположить их сверху и снизу нажмите **Tile Up and Down** (исходный или первый ВП показывается сверху).

Просмотрите список различий, внесите необходимые изменения. Нажмите **Update**. Это позволит обновить список и продолжить работать с оставшимися различиями.

После завершения работы закройте окно **Differences**. При этом LabVIEW закроет и дополнительные окна лицевых панелей и блок-диаграмм.

Сравнение двух иерархий

Для сравнения двух иерархий вызовите диалоговое окно **Compare VI Hierarchies**, выбрав **Tools ⇒Compare ⇒ Compare VI Hierarchies**. Это диалоговое окно показано на рис. 27.9.

В отличие от сравнения двух ВП для осуществления процесса сравнения двух иерархий ВП не требуется загружать файлы в память, переименовывать их. Достаточно просто запустить LabVIEW.

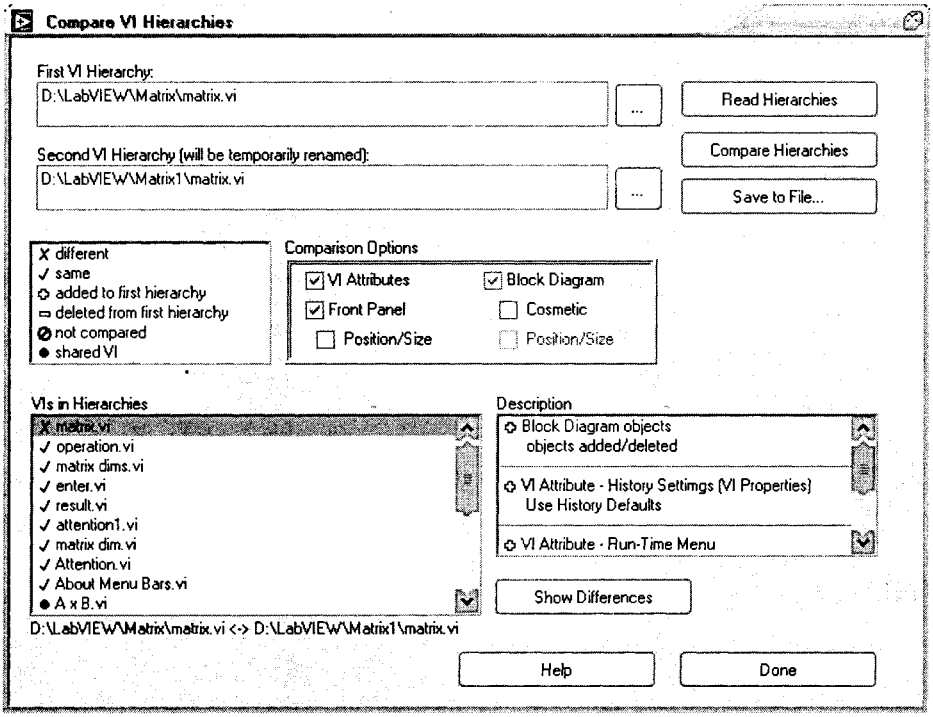


Рис. 27.9

Выберите первый и второй главные файлы иерархии ВП. Пути к файлам появятся в полях **First VI Hierarchy** и **Second VI Hierarchy**. Поскольку LabVIEW не может загрузить одновременно два ВП с одним и тем же именем, LabVIEW в случае необходимости временно переименовывает второй файл, копируя ВП во временный каталог и добавляя к имени приставку **cmp**.

Установите свойства сравнения **Comparison Options**. Назначение каждой из опций рассмотрено ранее.

Предварительное сравнение можно выполнить, нажав на **Read Hierarchies**. LabVIEW считает и проанализирует обе иерархии. Выведет список ВПП. Покажет новые и удаленные файлы. Полный список файлов представляется в поле **VIs in Hierarchies**. Подробное сравнение всех ВП в иерархии осуществляется при нажатии на **Compare Hierarchies**. Если до этого иерархии не были считаны и проанализированы на наличие различий в структуре иерархии, LabVIEW сначала сделает это. Сам процесс сравнения в зависимости от количества файлов и их размера может занять некоторое время. На правой стороне диалогового окна показано, сколько ВП уже обработано, а сколько осталось. По окончании обработки в поле **Description** появится список различий для конкретного ВП.

Слева находится поле **Legend**, в котором поясняются символы, используемые в полях **VIs in Hierarchies и Description**. Их описание приведено в таблице.

Таблица 27.2

	Описание	Назначение
✗	*different*	*Показывает, что ВП различаются.*
✓	*same*	*Показывает, что ВП не различаются.*
✪	*added to first hierarchy*	*Показывает, что ВП существует в первой иерархии, но его нет во второй.*
▭	*deleted from first hierarchy*	*Показывает, что ВП существует во второй иерархии, но его нет в первой.*
⊘	*not compared*	*Показывает, что ВП еще не были сравнены.*
●	*shared VI*	*Показывает, что обе иерархии совместно используют ВП из библиотеки vi.lib.*

Имеется возможность сохранить проведенный анализ в текстовый файл. Для этого нажмите на кнопку **Save to File**.

Кнопка **Show Differences** вызывает окно **Differences** выбранного ВП. Это позволяет более подробно остановиться на изменениях конкретного ВП.

Выводы

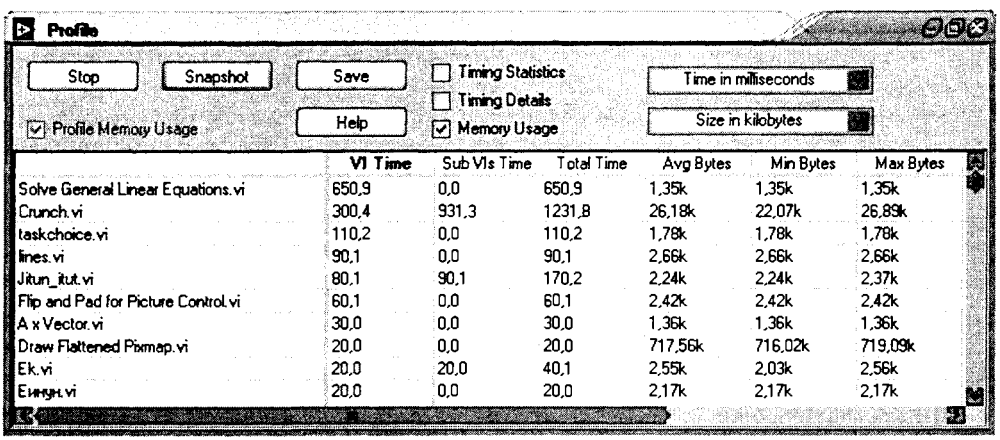
Окно иерархии составляет схематичное изображение создаваемого проекта со всеми функциями и ВПП, глобальными переменными, уникальными элементами управления и индикации, их связями. Инструмент сравнения двух ВП позволяет сравнивать ВП по различным характеристиками (отличия на лицевой панели и/или блок-диаграмме). Инструмент сравнения иерархий предоставляет подробную информацию об отличиях в файлах в двух иерархий.

Лекция 28

Производительность и управление памятью. Контроль за исходным кодом

Рассматриваются инструменты, позволяющие оценить скорость работы и требуемую память для отдельных составляющих программы, общий размер проекта. Даются советы по увеличению производительности.

VI Performance Profiler представляет собой мощное средство для определения того, на что ваше приложение тратит время и как оно использует память. В окне **Profile** показываются данные времени и объема используемой памяти для каждого ВП вашего проекта. Вызов диалогового окна **Profile** осуществляется выбором пункта меню **Tools** ⇒ **Advanced** ⇒ **Profile VIs**. Пример окна показан на рис. 28.1.



	VI Time	Sub VIs Time	Total Time	Avg Bytes	Min Bytes	Max Bytes
Solve General Linear Equations.vi	650,9	0,0	650,9	1,35k	1,35k	1,35k
Crunch.vi	300,4	931,3	1231,8	26,18k	22,07k	26,89k
task.choice.vi	110,2	0,0	110,2	1,78k	1,78k	1,78k
lines.vi	90,1	0,0	90,1	2,66k	2,66k	2,66k
Jitun_itut.vi	80,1	90,1	170,2	2,24k	2,24k	2,37k
Flip and Pad for Picture Control.vi	60,1	0,0	60,1	2,42k	2,42k	2,42k
A x Vector.vi	30,0	0,0	30,0	1,36k	1,36k	1,36k
Draw Flattened Pixmap.vi	20,0	0,0	20,0	717,56k	716,02k	719,09k
Ek.vi	20,0	20,0	40,1	2,55k	2,03k	2,56k
Ежун.vi	20,0	0,0	20,0	2,17k	2,17k	2,17k

Рис. 28.1

Перед запуском подсчета временных затрат и работы проекта следует выбрать, какие именно данные вас интересуют. По умолчанию собирается основная информация о проекте, которая всегда отображается в таблице. В нее входят следующие составляющие:

- **VI Time** – общее время выполнения ВП, в том числе и время, которое потребовалось пользователю при работе с лицевой панелью прибора;
- **SubVIs Time** – общее время выполнения подпрограмм ВП;
- **Total Time** – суммарное время выполнения ВП и его ВПП.

Более полную информацию можно получить, выбрав **Timing Statistics**. В нее входят:

- **#Runs** – число полных выполнений ВП;
- **Average** – среднее время выполнения ВП;
- **Shortest** – минимальное время выполнения ВП;
- **Longest** – максимальное время выполнения ВП.

Опция **Timing Details** позволит проанализировать время выполнения конкретных задач. В нее входят:

- **Diagram** – время выполнения кода блок-диаграммы;
- **Display** – время обновления данных элементов управления и индикации;
- **Draw** – время прорисовки лицевой панели.

Время прорисовки включает следующие компоненты:

- время, которое просто требуется для прорисовки лицевой панели, когда открывается окно с лицевой панелью;
- время, которое требуется для прорисовки объектов, которые перекрывают друг на друга, фон которых сделан прозрачным. Эти объекты заново прорисовываются, когда их значения меняются, в отличие от других объектов, которые прорисовываются один раз.
- **Tracking** – время, затрачиваемое на передвижение курсора при работе пользователя с лицевой панелью. Это время может быть большим для некоторых операций, например при изменении изображения графиков, выборе пунктов всплывающих меню, выборе значений элементов управления.
- **Locals** – время, затрачиваемое на запись или считывание данных локальных переменных на блок-диаграмме. Опыт показывает, что это время может быть значительным, особенно когда локальные переменные обрабатывают большие по объему и сложные по структуре данные.

Для того, чтобы собрать информацию об используемой памяти, пометьте флажком опцию **Profile Memory Usage**. После того, как вы это сделаете, появится возможность выбрать опцию **Memory Usage**. Двойной выбор обусловлен тем, что в случае если нет флажка в опции **Profile Memory Usage**, то при сборке информации используемая память не определяется. Второй же флажок просто позволяет добавить в таблицу собранную информацию или убрать ее из таблицы. Эта секция показывает два набора данных – данные, относящиеся к числу используемых байт, и данные, относящиеся к числу используемых блоков. Блок – это сегмент памяти, в котором хранится информация о данных. Например, массив целочисленных переменных использует для хранения некоторое количество байт или один блок. Используются независимые блоки памяти для массивов, строк, путей к файлам и графическим изображениям. Большое число блоков в памяти нагружает ваше приложение и может стать причиной снижения производительности в целом.

Категории, на которые делится вся информация об используемой памяти:

- **Average Bytes** – среднее число байт, используемых для хранения данных при выполнении программы;
- **Min Bytes** – минимальное число байт, используемых для хранения данных при выполнении программы;
- **Max Bytes** – максимальное число байт, используемых для хранения данных при выполнении программы;
- **Average Blocks** – среднее число блоков, используемых для хранения данных при выполнении программы;
- **Min Blocks** – минимальное число блоков, используемых для хранения данных при выполнении программы;
- **Max Blocks** – максимальное число байт, используемых для хранения данных при выполнении программы.

Процесс получения указанной информации осуществляется двумя кнопками: **Start/Stop и Snapshot**. Первая запускает или останавливает сбор информации, вторая позволяет наблюдать текущее состояние таблицы.

Некоторые советы по увеличению производительности

Хотя компилятор формирует код, который выполняется в основном очень быстро, иногда требуется получить наилучшую производительность ваших ВП. Обратите внимание на следующие возможные причины:

- ввод/вывод (файлы, сбор данных, работа с сетью);
- отображение информации на экране (большие, перекрывающиеся объекты, большое количество графиков, неправильный выбор типа данных);
- неправильное использование объектов блок-диаграммы (неэффективное использование массивов и строк, неэффективное использование структур).

Другие факторы также могут влиять на производительность, однако обычно их влияние несущественно.

Обычно ввод/вывод является причиной непроизводительных затрат времени. Часто оно занимает больше времени, чем вычислительные операции. Например, простая операция считывания с последовательного порта занимает несколько миллисекунд. Причина такой величины кроется в том, что осуществляется передача информации через несколько слоев операционной системы. Поэтому производительность улучшается, если вы организуете свое приложение таким образом, что в нем при каждом вызове выполняется передача большого объема данных, вместо выполнения большого числа вызовов ввода/вывода, используя небольшой объем данных.

Например, вы создаете ВП сбора данных. В вашем распоряжении есть функция передачи данных одной точки **AI Sample Channel VI** и функция передачи данных нескольких точек **AI Acquire Waveform VI**. С одной стороны вы можете использовать первую функцию, поместив ее в структуру **For** и дополнив ее функцией ожидания **Wait** для согласования сбора данных во времени. Однако намного продуктивнее использовать вторую функцию, поскольку она использует таймер управления выборкой данных непосредственно в аппаратном обеспечении. Более того, временные

затраты при использовании функции **AI Acquire Waveform VI** будут такими же, что и единственный вызов функции **AI Sample Channel VI**, даже в случае если осуществляется передача достаточно большого объема данных.

Частое обновление объектов лицевой панели представляется одной из наиболее затратных операций. Это особенно касается приложений, содержащих сложные объекты, такие как графики и диаграммы. Большинство элементов индикации не перерисовываются, когда получают новые данные, равные старым. Графики – исключение из этого правила. Если перерисовка экрана становится проблемой, лучшим решением будет сократить количество объектов лицевой панели. В случае графиков чтобы увеличить скорость прорисовки, вы можете отключить автомасштабирование, сетку. Как и в случае с вводом/выводом гораздо выгоднее собрать все данные графика в массив, вместо того, чтобы прорисовывать каждую точку отдельно.

При работе с блок-диаграммой руководствуйтесь следующими правилами.

1. Убедитесь, что вы используете подходящий тип данных. Числа с плавающей запятой повышенной точности необходимо использовать лишь в тех случаях, когда требуется высокая точность. Они занимают лишнюю память, если фактически используется низший тип данных. Этот фактор становится важным при использовании больших массивов.
2. Глобальные переменные используют много памяти. Уменьшите их число, а также число раз их считывания и записи.
3. По возможности старайтесь не использовать сложные иерархические типы данных (такие как массив кластеров массивов).
4. Без необходимости старайтесь не применять ненужное приведение типов данных. Такое приведение отображается серыми точками на входах узлов. Иначе говоря, ожидаемый тип данных не соответствует тому, что подан на вход функции. На рис. 28.2 показан пример. Несмотря на то, что LabVIEW может принимать практически любые типы данных (используется полиморфизм), результатом этого всегда является потеря скорости и использование больше памяти, поскольку возникает необходимость делать копии данных. Особенно это актуально для больших массивов.
5. Обратите внимание на то, как вы обрабатываете массивы и строки. Не используете ли вы циклы там, где можно обойтись без них. Иногда имеется встроенная функция, которая выполняет эту работу или несколько циклов объединяются в один. Избегайте введения ненужных элементов в цикл, как это показано на рис. 28.3.
6. Там, где это возможно, избегайте использования функции создания массива **Build Array** внутри циклов, исключая повторных вызовов менеджера памяти.

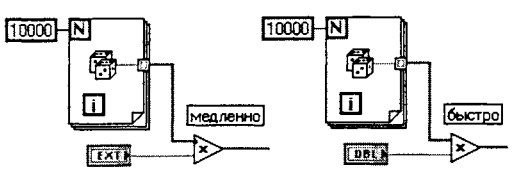


Рис. 28.2

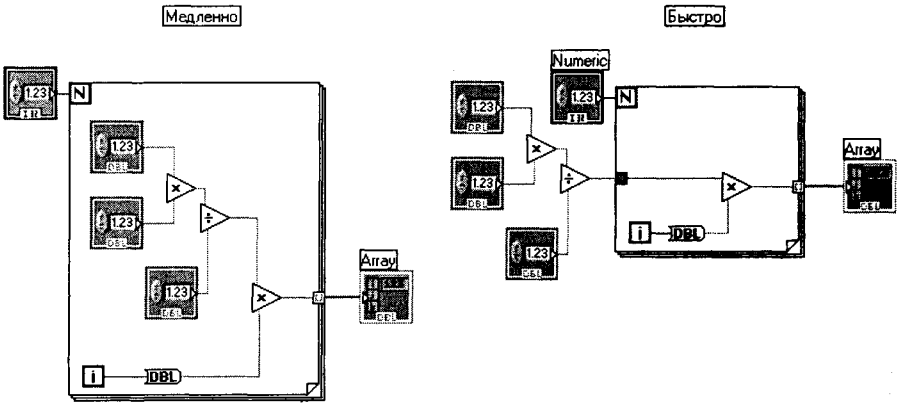


Рис. 28.3

Всякий раз, когда вы вызываете эту функцию, в памяти резервируется новое пространство для всего нового массива. Используйте вместо этого автоиндексирование или функцию замены элемента массива **Replace Array Element** с заранее установленным размером массива. Те же самые проблемы возникают с функцией «объединить строки».

7. Если это возможно, обновляйте элементы управления и индикации за пределами циклов. То есть тогда, когда не очень важно видеть значение объекта до момента окончания выполнения цикла.

Инструмент VI Metrics

Инструмент **VI Metrics** позволяет измерять сложность разрабатываемого проекта. Этот инструмент аналогичен широко используемому в текстовых языках программирования **source lines of code (SLOC) metrics**. С помощью инструмента **VI metrics** вы можете просматривать статистику ВП. Приводится информация о количестве используемых узлов, структур, диаграмм, элементов управления и индикации, о глубине вызываемых подпрограмм, о размерах блок-диаграмм и т.д. для каждой подпрограммы иерархии ВП. Иначе говоря, формируется подробный отчет о масштабах проекта. Несмотря на подробность отчета, не следует забывать, что любая статистика, включая **SLOC**, это всего лишь грубая оценка сложности проекта.

Инструмент **VI Metrics** вызывается в меню **Tools ⇒ Advanced ⇒ VI Metrics**. Работа с ним осуществляется в диалоговом окне, которое показано на рис. 28.4.

По умолчанию, диалоговое окно **VI Metrics** исключает из рассмотрения ВП, находящиеся в библиотеке LabVIEW в каталоге **vi.lib**. Вызовы к ВП из **vi.lib** рассматриваются как узлы, но информация о числе ВП, которые они вызывают и о сложности тех ВП из **vi.lib** не добавляется к общим измерениям выбранной иерархии. Таким образом, показывается статистика разработанных вами программ. Если вы хотите собрать информацию по ВП, включая ВПП из **vi.lib**, уберите флажок из **Exclude vi.lib files from statistics**.

В диалоговом окне **VI Metrics** имеются следующие элементы:

- кнопка **Select a VI** выбирает подвергаемый анализу ВП;

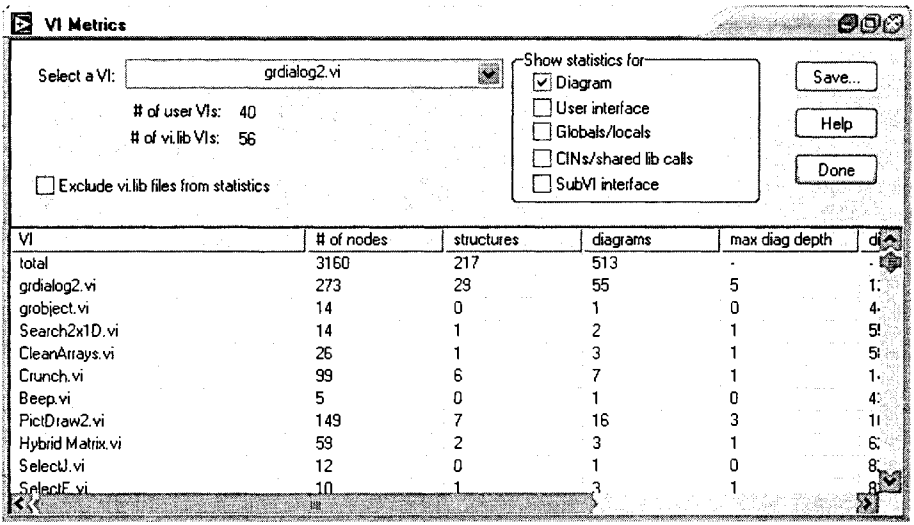


Рис. 28.4

- **# of user VIs** показывает, сколько созданных пользователем ВП относятся к выбранному ВП;
- **# of vi.lib VIs** показывает, сколько ВП взято пользователем из библиотеки **vi.lib**;
- **Exclude vi.lib files from statistics** определяет, учитывать ли взятые из библиотеки LabVIEW ВП.

Чтобы получить статистику ВП, выберите его из списка загруженных ВП вверху диалогового окна. Для каждого ВП из выбранной иерархии в таблице диалоговое окно будет показано число узлов, которое содержит каждый ВПП. Число узлов включает в себя число функций, подпрограмм, структур, терминалов объектов лицевой панели, констант, глобальных и локальных переменных, а также узлы свойств **Property Nodes**.

Число узлов не включает в себя провода, туннели или объекты, которые являются составляющими структур, такие как терминал счетчика итераций для структуры цикла **For** и цикла **While**.

Кроме подсчета числа узлов в ВП программное средство **VI Metrics** может рассчитать и другие данные, характеризующие сложность одного или нескольких ВП. Для вывода на экран дополнительной информации по проекту поместите флажок в соответствующую категорию справа вверху. Доступны следующие дополнительные оценки:

- **Block Diagram Metrics** – метрики блок-диаграммы.
- **User Interface Metrics** – метрики пользовательского интерфейса.
- **Globals/Locals Metrics** – метрики глобальных и локальных переменных.
- **CINs/Shared Library Metrics** – метрики разделяемых библиотек и библиотек CIN.
- **SubVI Interface Metrics** – метрики интерфейса ВПП.

Метрики блок-диаграммы содержат информацию о сложности блок-диаграммы ВП, его строении и проводах. Число **Structures** соответствует количеству структур цикла **For**, цикла **While**, варианта, открытой и многослойной последовательности. В каждой структуре имеется своя внутренняя диаграмма, которая вносит вклад в количество диаграмм ВП в столбце **Diagrams**. При этом в ряде объектов (можно выделить структуры варианта, открытой и многослойной последовательности) имеется несколько внутренних диаграмм. Общее количество таких диаграмм и представлено в этом столбце. Следующий столбец **Maximum diagram depth** показывает максимальное число вложенных уровней блок-диаграммы. Если ВП не имеет структур, его глубина составляет 0. В колонках **Diagram width** и **Diagram height** представлены ширина и высота блок-диаграмм в пикселях. Число **Wire** sources характеризует общее число источников в ВП. Под источником здесь понимается начало каждого провода. Каждый провод имеет один источник, который может иметь несколько назначений. Если провод проходит через структуру, то туннель, который обеспечивает передачу данных во внутреннюю диаграмму структуры, считается новым источником.

Метрики пользовательского интерфейса содержат информацию об объектах пользовательского интерфейса ВП. В колонках **Controls** и **Indicators** показывается число элементов управления и индикации на лицевой панели главного ВП. Массивы и кластеры рассматриваются как один элемент. Столбцы **Property reads** и **Property writes** характеризуют число считываемых и записанных блок-диаграммой свойств. Следует отметить, что в общей статистике узлов (**# of nodes**) несколько узлов свойств на блок-диаграмме для одного объекта считаются за один узел. В описываемых же столбцах учитывается каждое свойство.

Метрики глобальных и локальных переменных содержат информацию о глобальных и локальных переменных. Числа **Global reads** и **Global writes** соответствуют числам считываемых и записываемых на блок-диаграмме глобальных переменных. Числа **Local reads** и **Local writes** соответствуют числам считываемых и записываемых на блок-диаграмме локальных переменных.

Метрики интерфейса ВПП содержат информацию о ВПП, их элементах управления и индикации. В столбцах **Connector inputs** и **Connector outputs** показывается число входов и выходов соединительных панелей ВПП.

Нажав на кнопку **Save**, сохраните всю информацию о структуре ВП в текстовый файл. Данные этого файла впоследствии можно отформатировать в таблицу или анализировать, используя LabVIEW.

Выводы

Оптимизация кода большого проекта начинается с оценки размеров, времени выполнения и использования памяти программой и отдельных ее узлов. Инструмент **VI Performance Profiler** позволяет оценить временные затраты и используемую память для ВП и всей его иерархии. Непроизводительные затраты памяти, как правило, следует искать в неправильном использовании ввода-вывода при работе с файлами, реальным сигналом, сетью; излишнем нагромождением лицевой панели элементами индикации (графиками); неэффективном использовании структур, типов данных. Инструмент **VI Metrics** выводит полную статистику о ВП и каждом его узле. Эта статистика дает представление о числе узлов, структур, диаграмм, объектов лицевой панели и др.

Лекция 29

Обеспечение готовых проектов LabVIEW документацией

Изучаются инструменты, помогающие обеспечить проект всей необходимой документацией.

Создание любого проекта должно сопровождаться документированием его возможностей, файлами помощи. Обычно программное обеспечение сопровождается документами двух категорий:

- техническое описание, включающее требования, спецификации, рабочий план, план испытаний и историю исправлений;
- пользовательская документация, объясняющая как использовать данное программное обеспечение.

Стиль каждого из упомянутых документов различен. Аудитория документов технического описания обычно владеет обширными знаниями о документируемых возможностях. У аудитории пользовательских документов, как правило, меньше опыта работы с программным обеспечением. Кроме этого конечного пользователя вашего продукта можно рассматривать как конечного пользователя виртуального подприбора и как конечного пользователя виртуального прибора или приложения. В случае если программное обеспечение разрабатывается в виде библиотеки ВП с целью дальнейшего использования при создании проектов, вам следует составить документы аналогичные файлам помощи LabVIEW. Предполагается, что потребители вашей библиотеки такие же разработчики, как и вы. То есть они используют ваши ВП в качестве ВПП. Для каждого ВПП составьте обзорную информацию, например включающую имя файла, назначение, иконку соединительной панели, описание типа входных и выходных данных. Если вы разрабатываете программу для пользователей, которые незнакомы с LabVIEW, к документам следует приложить вводный материал. Вы можете создать документ, в котором содержатся системные требования, основная информация по установке и обзор возможностей написанной программы. Если пакет использует ввод/вывод, включите требования к аппаратуре и инструкции по конфигурации системы, которые пользователь должен выполнить перед использованием приложения.

В зависимости от проекта размер каждого документа может различаться. Для простых программ, предназначенных для домашнего использования, возможно, не требуется большого количества документов. Если вы планируете продать продукт, вам необходимо разработать подробную пользовательскую документацию. Если же вы желаете получить знак качества, вам следует убедиться, что выполнены все требования к техническому описанию.

В LabVIEW включены функции, которые упрощают процесс создания документации к проектируемому ВП. Инструмент **Revision History** позволяет записывать все изменения по мере написания программы. Страница **Documentation** диалогового окна **VI Properties** предназначена для создания некоторых из приведенных выше документов. Менеджер подсказок **Description and Tip** формирует описание и всплывающие подсказки объектов ВП. Диалоговое окно **Print** позволяет распечатывать лицевую панель, блок-диаграмму, соединительную панель и описания ВП, названия и описания элементов управления и индикации, имена и пути ВПП. Имеется возможность, как распечатать эту информацию на принтере, так и экспортировать ее в текстовый, HTML или RTF файл, который впоследствии можно конвертировать в файл помощи. Рассмотрим подробнее каждый из упомянутых инструментов.

Окно VI History

Чтобы вызвать окно **VI History**, выберите **Tools** ⇒ **VI Revision History**. Появится диалоговое окно, показанное на рис. 29.1.

Для добавления нового комментария введите в поле Comment описание проделанных вами изменений. После этого нажмите **Add**. Пока вы редактируете ВП, оставляйте окно **History** открытым, чтобы при каждом изменении вводить информацию о нем. Если в процессе работы вы записали комментарий в поле **Comment**, но не добавили его в историю, то при сохранении ВП LabVIEW автоматически

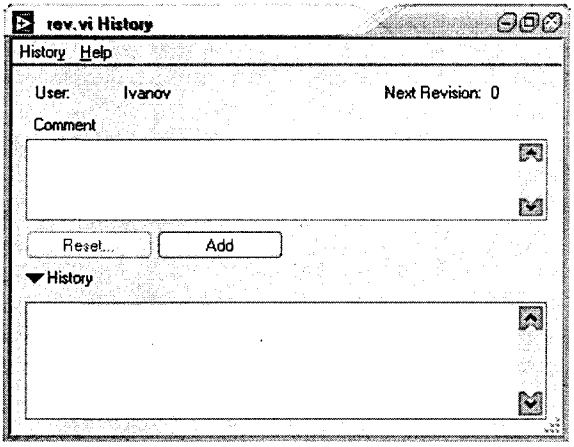


Рис. 29.1

добавит комментарий в историю. Следует учесть, что после того, как вы внесли комментарий в архив, вы не сможете потом отредактировать его. В заголовке записанного комментария указывается номер, дата и время изменения, а также имя пользователя. Номер изменения ВП соответствует числу внесенных вами исправлений. Он учитывает все сохранения ВП. По умолчанию имя пользователя определяется автоматически при запуске LabVIEW. Можно изменить, выбрав **Tools ⇒ User Name**.

Настройка инструмента **Revision History** осуществляется на страницах **Revision History** в разделах **File ⇒VI Properties** и **Tools ⇒ Options**. Обе страницы содержат одинаковые настройки. Различие состоит в том, что на первой странице можно установить желаемые параметры для текущего ВП, а на второй – желаемые параметры также и для новых ВП. Чтобы появилась возможность устанавливать параметры инструмента **Revision History** из раздела **File ⇒ VI Properties**, снимите флажок с **Use the Default History Settings from the Options Dialog**.

Доступны следующие параметры:

- **Add an entry every time the VI is saved** – Автоматически генерирует комментарий, если вы изменили и сохранили ВП. Если вы не ввели комментарий в поле **Comment**, LabVIEW добавляет только заголовок.
- **Prompt for a comment when the VI is closed** – При каждом закрытии ВП запрашивает добавить комментарий. LabVIEW не запрашивает добавить комментарий, если вы не сделали никаких изменений. LabVIEW не запрашивает добавление комментария, если вы изменили только архив исправлений. Эта опция полезна, если вы включили автосохранение и не желаете, чтобы LabVIEW запрашивала комментарий каждый раз, когда ВП автоматически сохраняется.
- **Prompt for a comment when the VI is saved** – При каждом сохранении проекта LabVIEW запрашивает добавить комментарий. Эта опция полезна, если вы предпочитаете сразу же комментировать изменения, когда вы их сделали. LabVIEW не запрашивает комментария, если вы не сделали никаких изменений или если вы меняли только архив изменений.
- **Record comments generated by LabVIEW** при каждом сохранении ВП автоматически создает комментарий LabVIEW, (например, такое как преобразование ВП в новую версию LabVIEW).
- Кроме того, на вкладке **Revision History** в настройках **Options**, можно настроить порядок использования LabVIEW имени пользователя.
- **Login automatically with the LabVIEW registration name** устанавливает имя пользователя тем, на которое зарегистрирована система LabVIEW.
- **Login automatically with the system user name** устанавливает системное имя пользователя.
- **Show the login prompt at LabVIEW startup time** при каждой загрузке LabVIEW запрашивает имя пользователя.

В любой момент времени историю изменения ВП можно удалить нажатием на **Reset**. При этом все без исключения записи в архиве будут удалены. Одновременно с удалением архива LabVIEW запрашивает, обнулять ли номер изменений.

Использовать архив изменений можно при печати ВП, также можно экспортировать архив в файлы формата HTML, RTF и текстовый файл. Последнее выполняется при выборе пункта меню **File ⇒ Export to File** в главном меню диалогового окна **History**.

В заключение отметим, что инструмент **Revision History** это инструмент исключительно разработки проектов. Поэтому окно History не работает в режиме выполнения ВП. Поэтому при сохранении ВП со специальными настройками, а точнее, вырезая блок-диаграмму, удаляется весь архив комментариев.

Страница Documentation Properties

Чтобы вызвать страницу документации, выберите в свойствах ВП **File ⇒ VI Properties** страницу **Documentation** (см. рис. 29.2). С ее помощью можно создать описание к ВП, которое позволит описать функции ВП и дать пользователям инструкции по его применению.

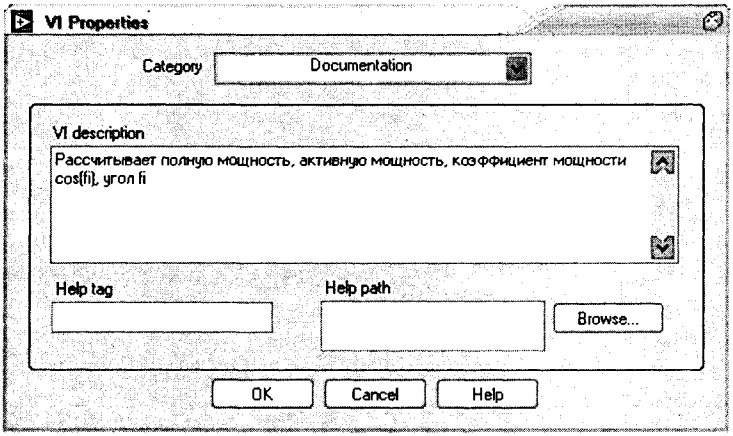


Рис. 29.2

Сначала следует ввести или отредактировать описание ВП в поле **VI description**. Необходимо включить в описание ВП следующие элементы: обзор ВП, инструкции по использованию ВП, описание входных и выходных данных. Используйте тэги B и /B для выделения жирным любой надписи в описании. Это описание появляется в окне контекстной справки **Context Help**, как показано на рис. 29.3.

Описание объекта позволит ориентироваться другим разработчикам при использовании ВП в качестве ВПП. Каждый элемент управления и индикации нуждается в описании, которое включает в себя следующую информацию:

- назначение;
- тип данных;
- допустимые значения;
- значения по умолчанию;

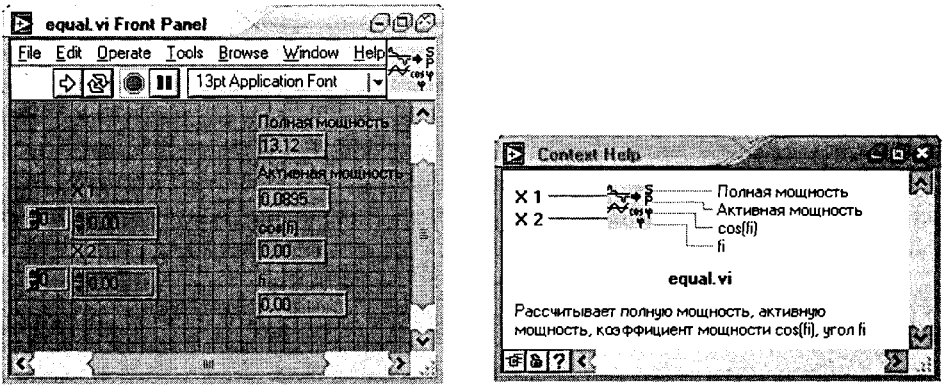


Рис. 29.3.

- характер изменения (например, 0, пустой массив, пустая строка и т.п.);
- дополнительная информация.

Опишите, какие требуются, рекомендуются входные и выходные данные, значения каких входных данных устанавливать необязательно. Это объяснит пользователю способ подключения к ВПП проводов. Чтобы выделить в окне контекстной справки требуемые, рекомендуемые и необязательные входные данные выберите в контекстном меню соединительной панели **This Connection is ⇒ Required, Recommended** или **Optional**.

На рис. 29.3 показано, как в контекстной справке выглядит добавленное ранее описание в сочетании с иконкой и перечисленными входами и выходами ВП.

Если у вас имеется HTML файл или файл помощи, его можно связать с ВП. Для этого на странице Documentation нажмите кнопку **Browse** и откройте ваш файл. Доступны форматы .chm, .hlp, .htm и .html. В поле **Help Path** появится путь к выбранному файлу. Ссылка на этот файл появляется в нескольких местах: внизу окна контекстной справки возникает надпись **«click here for more help»**, в контекстном меню иконки ВПП появляется пункт Help, становится активным пункт меню **Help ⇒ Help for This VI**, и, наконец, справка вызывается клавишей F1, когда окно лицевой панели ВП активно.

При использовании файлов помощи с расширением .chm or .hlp имеется возможность привязать ВП к конкретной теме этого файла. При использовании файла .chm в поле **Help Tag** следует ввести имя соответствующего отдельного .htm или .html файла в HTML проекте либо ключевое слово. При использовании файла .hlp в поле **Help Tag** следует ввести ключевое слово.

Чтобы сохранить все изменения, нажмите **OK**.

Сами файлы помощи можно создать, используя соответствующие средства. Полноценная справочная система создается с помощью программных средств обработки текстов, компилятора HTML и компилятора файлов помощи. При создании исходного материала используйте диалоговое окно **Print**, речь о котором пойдет в конце этой главы.

Окно Description and Tip

Добавить объяснение к каждому элементу управления и индикации или любому объекту лицевой панели и блок-диаграммы можно с помощью окна **Description and Tip**, которое вызывается в контекстном меню любого объекта выбором пункта **Description and Tip**. Описание, введенное в поле **Description**, появляется в окне контекстной справки, когда вы перемещаете курсор по объекту, а также в любой создаваемой вами документации к ВП. При использовании такого объекта в другом ВП его описание сохраняется.

Запишите комментарии в поле **Tip** для каждого элемента управления и индикации. При наведении курсора на элемент в режиме выполнения программы появится всплывающая подсказка с введенным вами текстом. Это позволит пользователям проще ориентироваться с вашей программой. Пример всплывающей подсказки с введенной в поле **Tip** фразой «Отключить все источники тока» показан на рис. 29.4.

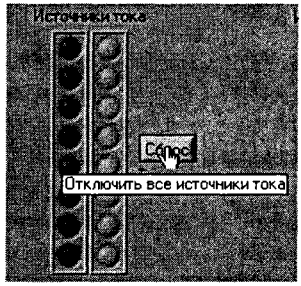


Рис. 29.4

Распечатка ВП с помощью инструмента Print VI

Выберите **File ⇒ Print** для печати ВП на принтер или в HTML, RTF или текстовый файл. Появится мастер печати, первая вкладка **Select VIs** которого показана на рисунке 29.5.

Здесь вы можете выбрать необходимые ВП для печати, иерархию ВП, добавить или удалить файлы. Если печать уже полностью настроена под ваши нужды, распечатать ВП можно, нажав на кнопку **Print** на любом из этапов мастера. Это позволит сократить время на получение нескольких типовых распечаток различных ВП. Если вы используете этот мастер впервые или существующие настройки вас не удовлетворяют, перейдите к следующему этапу выбора содержимого печати **Print contents** (рис. 29.6). Он позволяет выбрать стандартные шаблоны или создать пользовательскую конфигурацию элементов печати. Документация может включать следующие пункты:

- Иконка и соединительная панель;
- Лицевая панель и блок-диаграмма;
- Элементы управления и индикации и типы данных терминалов;
- Описание ВП и его объектов;
- Иерархия ВП;
- Список ВПП;
- История изменений.

Встроенные шаблоны позволяют распечатывать документы с уже готовым набором элементов для стандартных целей.

Доступны следующие шаблоны:

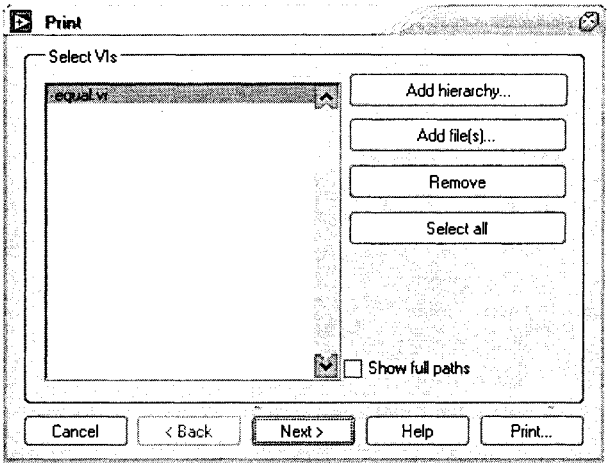


Рис. 29.5

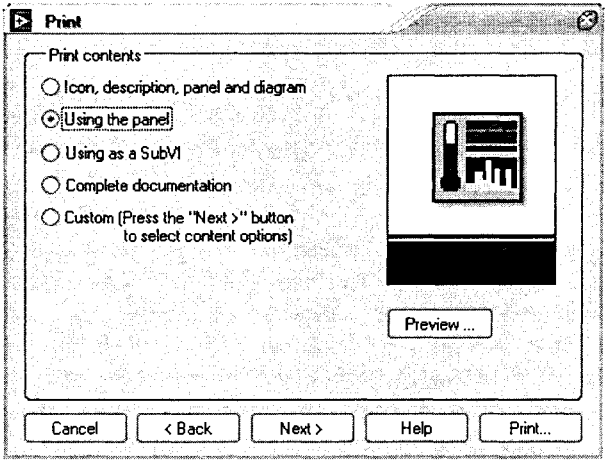


Рис. 29.6

- **Icon, description, panel and diagram** печатает имя ВП, его описание, иконку, соединительную и лицевую панели, а также блок-диаграмму.
- **Using the panel** печатает имя ВП, описание, лицевую панель, элементы управления и индикации, включая тип данных, названия и описания. Печать элементов управления и индикации производится в установленном порядке табуляции, который редактируется в **Edit ⇒ Set Tabbing Order**. Именно этот шаблон предназначен для создания справочного описания ВП, о котором упоминалось ранее. Он позволяет сформировать страничку с описанием лицевой панели, с которой пользователь будет взаимодействовать.

- **Using as a SubVI** печатает имя ВП, описание, иконку и соединительную панель, элементы управления и индикации, включая тип данных, названия и описания. Печать элементов управления и индикации также производится в установленном порядке табуляции. Именно этот шаблон предназначен для создания справочного описания ВПП, о котором упоминалось ранее. Он позволяет сформировать страничку с описанием ВПП.
- **Complete documentation** печатает полную информацию ВП, которая включает в себя имя ВП; описание; иконку и соединительную панель; лицевую панель; элементы управления и индикации, включая их тип данных, названия и описания; блок-диаграмму; список ВПП, включающий иконки, имена и пути; архив изменений, иерархию ВП.
- **Custom** позволяет печатать пункты, определяемые пользователем.

Остановимся на последнем пункте подробнее. При выборе пользовательской конфигурации на вкладке **Custom details** следует уточнить, какие элементы вы желаете распечатать (см. рисунок 29.7). Выберите материал из следующих компонентов:

- **Icon and description** печатает иконку ВП и его описание. Этот пункт разделяется на подпункты:
- **VI connector and icon** печатает иконку ВП, а также его входы и выходы;
- **Description** печатает описание ВП.
- **Front panel** печатает лицевую панель.
- **Controls and Indicators** печатает список элементов управления и индикации в порядке табуляции. Также этот пункт позволяет печатать элементы управления и индикации из массивов, кластеров и ссылок. Выберите **All controls**, если вы желаете распечатать все элементы управления и индикации.

Выберите **Connected Controls**, если вам необходимо вывести на печать только те элементы управления и индикации, которые подсоединены к соединительной панели ВП. Этот пункт разделяется на подпункты:

- **Descriptions** печатает описание элементов управления и индикации.
- **Include data type information** включает в распечатку тип данных каждого элемента управления и индикации.
- **Block diagram** печатает блок-диаграмму.
- **Hidden frames** печатает блок-диаграмму с видимыми поддиаграммами структур варианта, события и многослойной последовательности. Затем на печать выводятся все скрытые поддиаграммы каждой из структур.
- **Ordered (Repeat from higher level if nested)** печатает блок-диаграмму с видимыми поддиаграммами структур. Затем на печать выводятся все поддиаграммы для каждой структуры – видимые и скрытые поддиаграммы в порядке их следования. Такой способ распечатки дублирует поддиаграммы на каждом уровне, однако он позволяет лучше представить иерархию вложенных структур ВП.
- **VI hierarchy** печатает иерархию ВП.

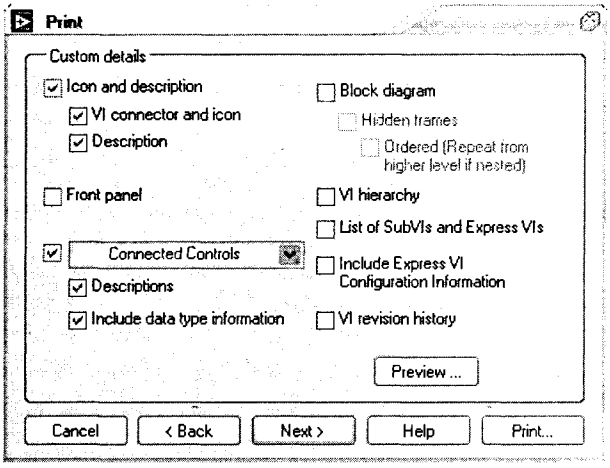


Рис. 29.7

- **List of SubVIs and Express VIs** печатает иконку, имя и путь всех используемых ВПП и экспресс-ВП.
- **Include Express VI Configuration Information** печатает информацию о конфигурации для каждого экспресс-ВП на блок-диаграмме.
- **VI revision history** печатает историю ВП (в случае если она существует).

Далее следует страница (см. рис. 29.8) с общими настройками печати, в которых указывается, печатать ли заголовок на каждой странице с именем ВП, датой последнего изменения и номером страницы (**Print header (name/date/page #)**). Также на этой вкладке можно указать поля страницы. По умолчанию проставляются значения полей, установленные на странице **Printing** диалогового окна **Options** или в

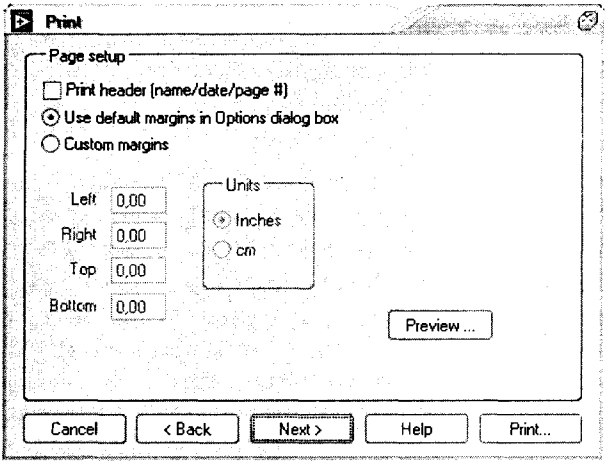


Рис. 29.8

настройках **File ⇒ Page Setup (Use default margins in Options dialog box)**. Имеется возможность установить другие значения (**Custom margins**).

На следующем этапе **Destination** можно выбрать тип документа для распечатки (см. рис. 29.9). Можно вывести печать на принтер, создавать HTML и RTF файлы, а также экспортировать текст распечатки в текстовый файл.

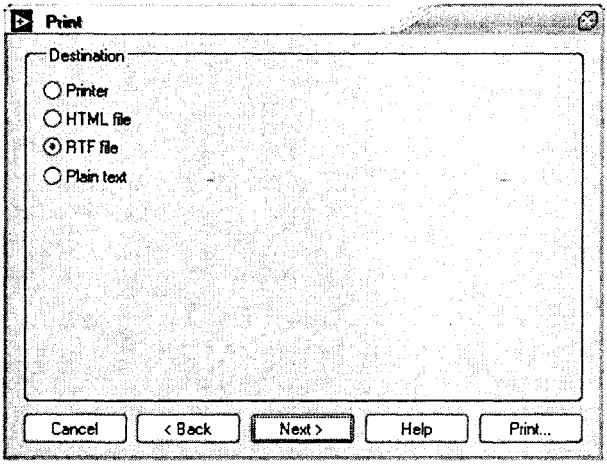


Рис. 29.9

В зависимости от типа печати следующий этап позволяет настроить для него распечатку. При распечатке всех данных на принтере можно выбрать некоторые дополнительные опции (см. рис. 29.10).

Чтобы занять минимальное число страниц, следует указать **Scale front panel to fit**. Этот пункт масштабирует лицевую панель до минимального вписываемого в страницу размера (если этот размер составляет не менее четверти исходного размера). Если этот пункт неактивен, это означает, что не выбран пункт печати лицевой панели. Аналогично этой опции опция **Scale block diagram to fit** позволяет сократить место для печати блок-диаграммы. Эта опция затемнена (этот случай показан на рисунке), если пользователь решил не печатать блок-диаграмму. **Page breaks between sections** осуществляет разрыв страниц для следующих секций: соединительной панели и иконки с описанием, лицевой панели, списка объектов лицевой панели и их описаний, блок-диаграммы, различных деталей блок-диаграммы, иерархии ВП и списка ВПП. **Print section headers** определяет печатать ли заголовок для каждой секции, например заголовок **VI Revision History** перед секцией истории изменений. **Surround panel with border** позволяет печатать границу лицевой панели. Чтобы распечатать документ, нажмите **Print**.

При экспортировании данных ВП в HTML и RTF файлы следует точно знать, какой именно документ вам нужно получить. Определитесь, желаете ли вы создать файл с возможностью его конвертирования в файл помощи (HTML-файл) или

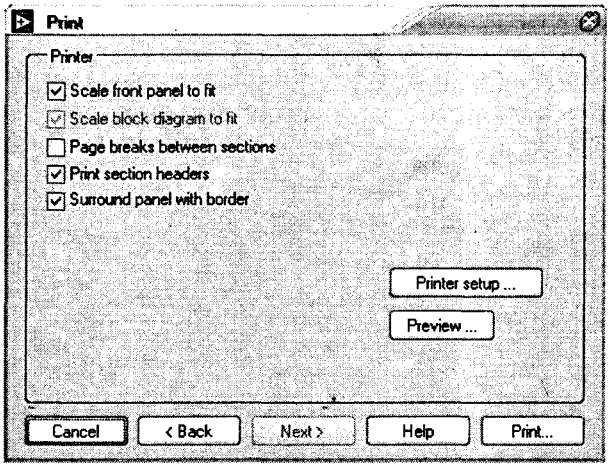


Рис. 29.10

файл для последующего его редактирования в системах обработки текста (RTF-файл). Различие при экспортировании данных ВП состоит в способе сохранения рисунков иконок, лицевой панели, блок-диаграммы и др., генерируемых LabVIEW. В первом случае LabVIEW сохраняет рисунки в виде внешних растровых изображений форматов JPEG, PNG или GIF. Во втором случае LabVIEW позволяет выбрать, внедрять графику в сам документ или создать внешние BMP файлы. Настройки сохранения рисунков производятся на следующем шаге мастера.

Экспорт данных распечатки в файл HTML дополняется выбором формата рисунка и его качества (рис. 29.11). LabVIEW позволяет получить файлы PNG без потери качества, сжатые файлы JPEG и несжатые файлы GIF. Имеется возмож-

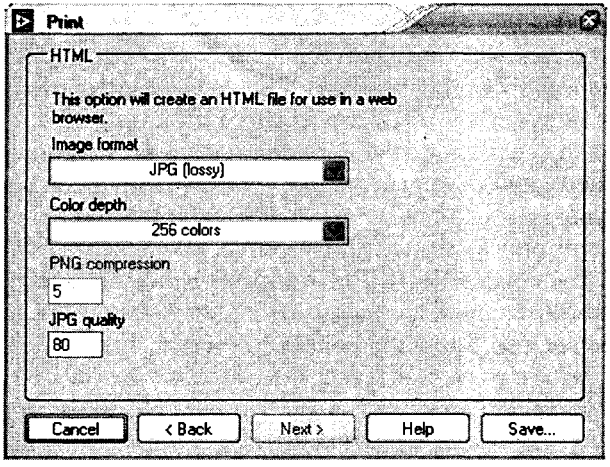


Рис. 29.11

ность установить степень сжатия изображения в формате PNG установкой **PNG compression**. Для изображений в формате JPEG есть дополнительная настройка качества изображения **JPG quality**. В дальнейшем изображения, созданные LabVIEW, можно конвертировать в изображения любого другого формата и степени сжатия. С помощью опции насыщенности цвета **Color depth** установите качество цветопередачи изображений. При настройке типа изображения следует учитывать конечную цель создания документации ВП. При публикации этих документов в сети Интернет предполагается, что изображения сохранены в общепринятых форматах JPG и GIF, поддерживаемых большинством браузеров. При этом качество этих изображений не должно быть очень высоким, чтобы размер рисунков при просмотре документов позволял использовать медленную модемную связь. При формировании файлов помощи также не следует увлекаться высоким качеством изображений. Нажмите **Print**, чтобы экспортировать данные в HTML файл.

Когда вы сохраняете данные о ВП в RTF файл, вам предоставляется выбор сохранять изображения во внешних файлах формата BMP или внедрять их внутри документа (рис. 29.12). Здесь вы также можете выбрать степень насыщенности цвета рисунков. Нажмите **Print**, чтобы получить готовый RTF файл.

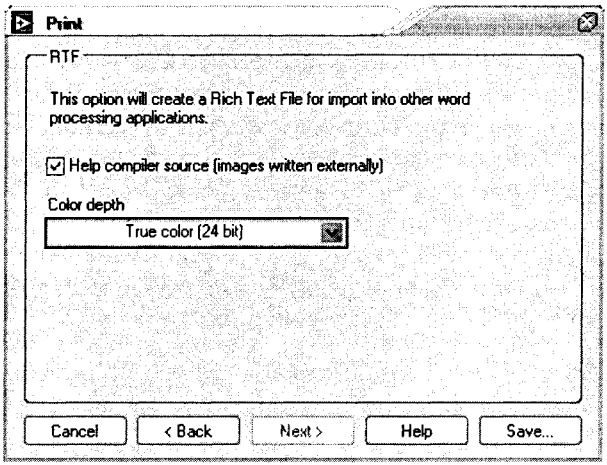


Рис. 29.12

Когда вы экспортируете текст (например, описания или историю изменений) в текстовый файл, дополнительная настройка сохранения текста (см. рис. 29.13) заключается в возможности ограничить число символов в строке, например до 60.

Выводы

В LabVIEW предусмотрены четыре основных средства для обеспечения вашего проекта документацией. Инструмент **Revision History** позволяет вести список произво-

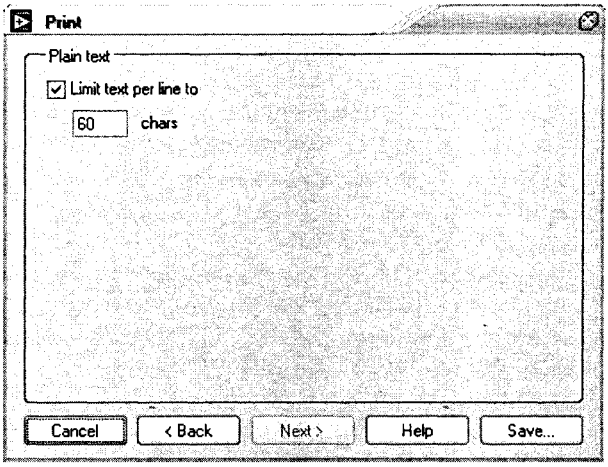


Рис. 29.13

димых изменений и дополнений. Страница **Documentation** диалогового окна **VI Properties**, предназначена для ввода описания ВП. Менеджер подсказок **Description and Tip** позволяет формировать описание и всплывающие подсказки к элементам управления и индикации. ВП, а также его элементы управления и индикации, целесообразно снабдить описанием, когда планируется его распространение в качестве ВПП. Всплывающие подсказки к элементам управления и индикации предпочтительно использовать при разработке пользовательского интерфейса для облегчения работы пользователя с программой. Инструмент **Print** предназначен для распечатки ВП и различного рода документации к нему. Использование этого инструмента позволит получить готовую документацию, предназначенную как для других разработчиков, так и для конечных пользователей программы.

Лекция 30

Создание автономно выполняемого приложения при помощи инструмента Application Builder

Изучается инструмент создания автономного приложения, запускаемого без установки LabVIEW.

Программное средство **Application Builder** представляет собой дополнительный программный пакет, предназначенный для создания автономно выполняемых приложений. Это означает, что для запуска скомпилированного ВП не требуется установки LabVIEW. Пользователи вашего продукта необязательно должны быть пользователями LabVIEW. Поэтому они смогут использовать созданные вами программы, используя свободно распространяемый **LabVIEW Run-Time Engine**, который включает в себя необходимые для запуска приложений LabVIEW библиотеки. В случае, если пользователи вашей программы являются разработчиками в среде LabVIEW, они смогут запустить ее, но не смогут редактировать и просматривать блок-диаграммы ВП и ВПП.

Мастер компилирования ВП в отдельно запускаемую программу состоит из пяти независимых вкладок с различными настройками: файлы приложения (**Target**), исходные файлы (**Source Files**), настройки ВП (**VI Settings**), настройки приложения (**Application Settings**), настройки инсталлятора (**Installer Settings**). Первые две вкладки служат для определения исходных файлов ВП и конечных файлов программы. Остальные вкладки позволяют установить настройки всех ВП проекта, конечных файлов приложения, а также настройки инсталлятора. Остановимся на этом подробнее.

Вкладка файлов приложения (Target)

Эта страница позволяет определить тип конечного файла, его имя, местоположение на диске (рис. 30.1). Во всплывающем меню **Build Target** производится выбор компиляции программы **Application (EXE)** или библиотеки совместного доступа **Shared Library (DLL)**. Библиотеки DLL используются, когда необходимо вызвать ВПП из текстовых языков программирования. Они обеспечивают возможность для различных языков программирования взаимодействовать с кодом, разработанным в

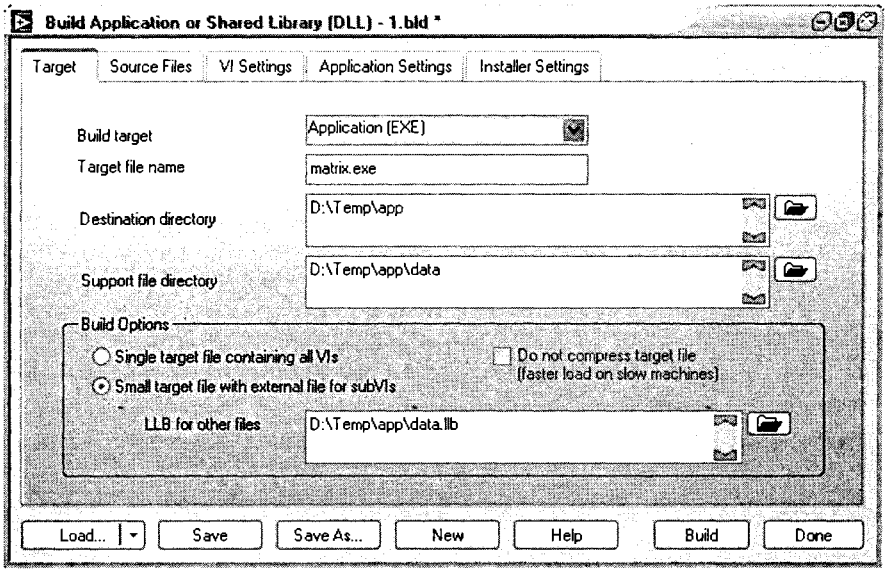


Рис. 30.1

LabVIEW. В дальнейшем будем рассматривать только создание программ, т.е. файлов приложения EXE.

В поле **Target file name** вводится имя приложения. Поля **Destination directory** и **Support file directory** определяют расположение файлов самой программы, а также файлов ее поддержки. Последние по умолчанию помещаются в подкаталог data основного каталога размещения программы. Внизу страницы можно установить дополнительные опции компиляции. Имеется возможность выбрать, вместить ли все ВП в один исполняемый файл EXE (**Single target file containing all VIs**) либо разделить проект на небольшой исполняемый файл EXE и внешний файл библиотеки для ВПП (**Small target file with external file for subVIs**). Чтобы ускорить загрузку создаваемого приложения, воспользуйтесь опцией **Do not compress target file**. Это позволит уменьшить время загрузки приложения, что может быть важным при использовании программы на медленных машинах.

Вкладка исходных файлов (Source Files)

Страница исходных файлов предназначена для управления файлами, составляющими проект. Они могут включать не только файлы главного ВП и его ВПП, но и различные файлы, имеющие отношение к проекту, например динамические ВП, файлы помощи, файлы кодов ошибок и т.д. (рис. 30.2).

На левой половине страницы представлен список исходных файлов. Ниже в поле **Source file path** приводится путь к выделенному файлу. Справа осуществляется управление списком файлов. **Add Top-Level VI** добавляет главный ВП (необходимо указать хотя бы один ВП верхнего уровня, хотя их может быть несколько).

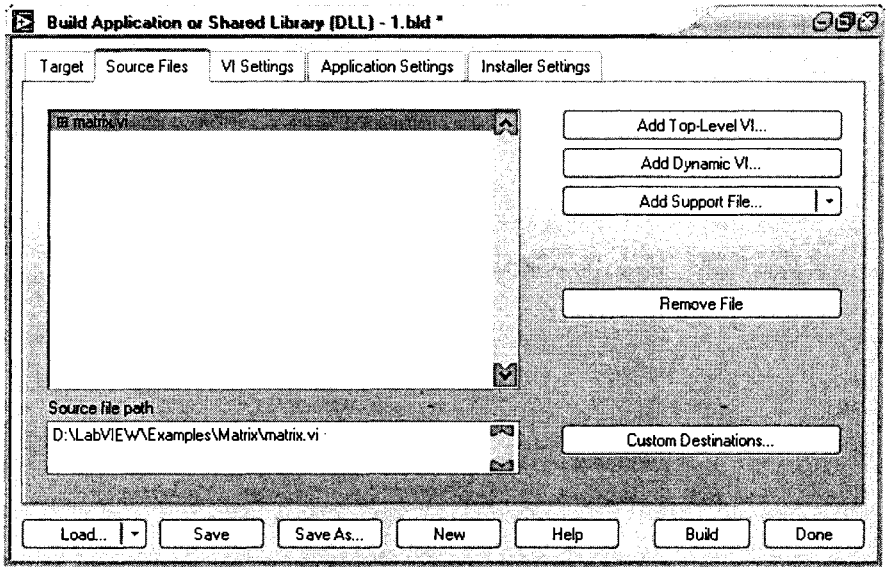


Рис. 30.2

При добавлении главного ВП LabVIEW включает все его ВПП, а также относящиеся к нему файлы, такие как файлы пользовательского меню. LabVIEW не добавляет файлы, вызываемые посредством **VI Server**. Чтобы включить и эти файлы, используйте **Add Dynamic VI**. **Add Support File** позволяет добавить файлы, не являющимися файлами LabVIEW. К ним относятся драйверы, текстовые файлы, файлы помощи и т.д. **Remove Files** удаляет файл из списка. С помощью **Custom Destination** можно указать путь размещения каждого конкретного файла или библиотеки вашего проекта.

Вкладка настройки ВП (VI Setting)

Страница настроек ВП позволяет указать режим выполнения каждого ВПП проекта (рис. 30.3).

На этой странице представлена таблица свойств ВП, добавленных в проект. В ней рассматриваются следующие свойства:

- **Remove Panel** убирает лицевую панель выбранного ВП. Удаление лицевой панели сокращает размер приложения. Поэтому если пользователю не придется взаимодействовать с лицевой панелью какого-либо ВПП, разумнее установить значение **No**.
- **Run When Opened** запускает ВП, когда он открыт. Это свойство имеет значение **Yes** только для ВП верхнего уровня.
- **Show Abort Button** показывает кнопку **Abort**.
- **Allow User to Close** разрешает пользователю закрыть лицевую панель во время выполнения программы.

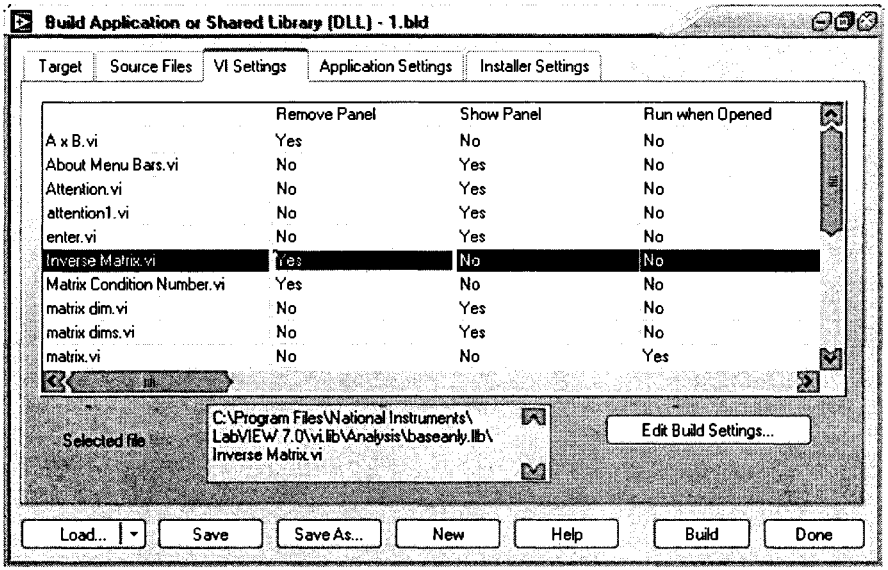


Рис. 30.3

- **Window is Modal** показывает, что лицевая панель работает в качестве диалогового окна. Если установлено значение **Yes**, то при появлении диалогового окна (т.е. вызове этого ВПП), пользователь не сможет работать с другими окнами.
- **Window Has Title Bar** показывает строку заголовка лицевой панели.
- **Show Menu Bar** отображает меню лицевой панели.
- **Show Tool Bar** показывает панель инструментов.
- **Show Scroll Bars** отображает полосы прокрутки.
- **Auto-Center** автоматически центрирует лицевую панель на экране.
- **Size the Front Panel to the Width and Height of the Entire Screen** автоматически меняет размеры лицевой панели, размещая ее на весь экран.

Как правило, все перечисленные свойства устанавливаются в процессе работы над ВП и его ВПП. Однако если имеется необходимость изменить какие-либо параметры определенного ВП, выделите его в таблице (в поле **Selected file** появится путь к нему) и нажмите **Edit Build Settings**. В появившемся диалоговом окне вы сможете установить все необходимые значения свойств.

Вкладка настроек приложения (Application Settings)

Эта страница позволяет изменить параметры компилируемого приложения (рис. 30.4).

С помощью нее вы можете использовать свою иконку для файла EXE вашей программы. LabVIEW может импортировать черно-белые и цветные иконки в двух разрешениях: 16Ч16 и 32Ч32 в целом четыре возможные иконки. В случае, если

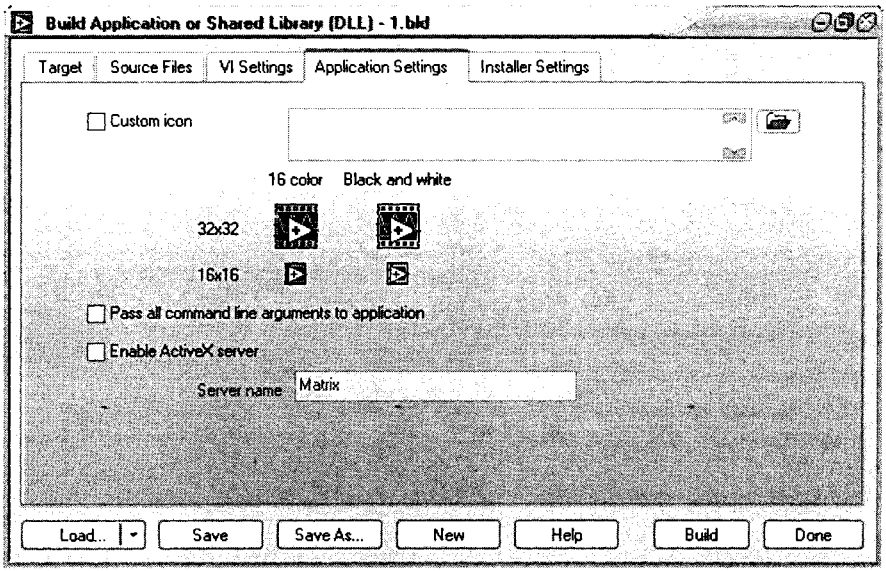


Рис. 30.4

файл иконки не содержит все четыре варианта, LabVIEW для пропущенной иконки использует иконку по умолчанию.

Pass all command line arguments to application передает все параметры запуска программы из командной строки.

Enable ActiveX server активирует сервер **ActiveX**.

Вкладка настроек инсталлятора (Installer Settings)

Эта страница предназначена для тонкой настройки инсталлятора вашего приложения (рис. 30.5).

По умолчанию LabVIEW не создает инсталлятор. Создание инсталлятора для программы будет реализовано, если вы выберите опцию **Create Installer**. После этого станет возможным управлять настройками инсталлятора. В общие настройки входят следующие:

- **Installation name** устанавливает имя программы, которое появляется в диалоговом окне инсталлятора.
- **Installer directory** определяет место, куда поместить инсталлятор во время компиляции программы.
- **Start menu program group** – назначает имя группы в меню «Пуск». Для того, чтобы установить пункты, которые появятся в этом меню, используйте диалоговое окно **Installation Destination Settings** (оно появится при нажатии на кнопку **Files**).
- **Default Installation directory** определяет директорию установки программы по умолчанию. Пользователи смогут выбрать другую директорию.

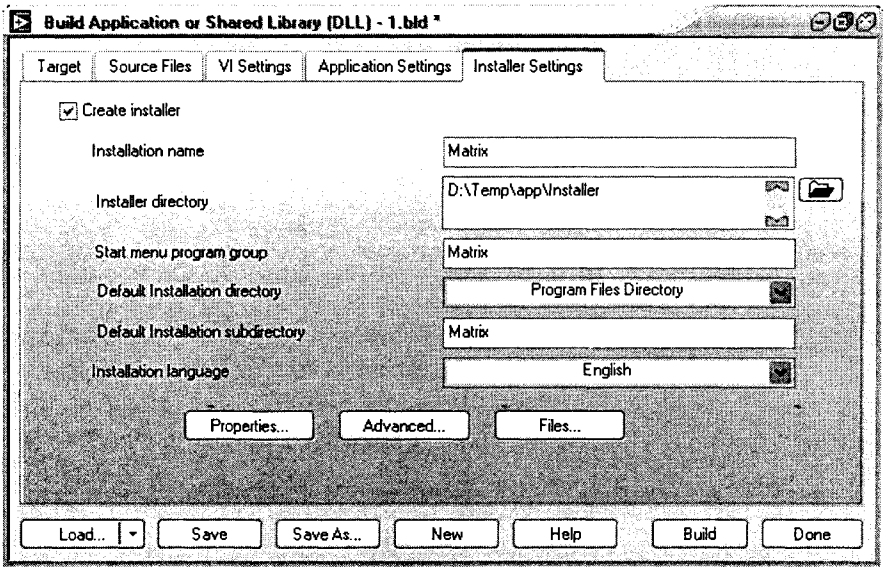


Рис. 30.5

- **Default Installation subdirectory** определяет название подкаталога установки по умолчанию.
- **Installation language** устанавливает язык инсталлятора. Имеется возможность выбрать четыре языка: английский, французский, немецкий или японский.

Далее следует обратить внимание на свойства инсталлятора. Нажмите **Properties** и заполните данные о производителе, авторе и версии продукта. Если необходимо, измените имена установочных файлов install.msi и data.cab. Введите информацию в поля **Subject**, **Keywords** и **Comments**. Данные об авторе, названии, теме, ключевых словах и комментариях будут показываться проводником при выборе свойств инсталлятора.

Чтобы инсталлятор включал в себя все необходимое, и при этом в нем не было лишних составляющих, следует вызвать диалоговое окно дополнительных свойств инсталлятора **Advanced**. Кроме этого в дополнительных настройках при необходимости можно указать параметры запуска программы после завершения установки. Включается опция запуска программы после установки флажком **Run executable after installation**. В поле **Executable** нужно ввести имя файла программы. Этот файл должен быть одним из тех, которые вы поместили в директорию инсталлятора (**Installer directory**). В поле **Command line arguments** введите параметры запуска программы. При записи параметров каждый путь следует заключать в кавычки. Опция **Wait until done** позволяет запустить программу после завершения установки, но до завершения работы инсталлятора.

Инсталлятор может включать в себя **LabVIEW Run-Time Engine**, необходимый для запуска программы на любом компьютере. Имеет смысл сопровождать программу этим блоком, если пользователи вашей программы заведомо не являются

пользователями LabVIEW и если они не работают с другими программами, написанными в LabVIEW. В других случаях лучше исключить средство **LabVIEW Run-Time Engine** из инсталлятора из-за его внушительных размеров (более 15 мегабайт), в то время как серьезный проект может занять менее 1-2 мегабайт. Если вы решили собрать полноценный инсталлятор и вас не беспокоит место, занимаемое им на диске, то не убирайте флажок **LabVIEW Run-Time Engine**. Дополнительно можно добавить или убрать из инсталлятора следующие элементы средства **LabVIEW Run-Time Engine**:

- **NI Reports Support** устанавливает поддержку создания отчетов;
- **3D Graph Support** устанавливает поддержку для обзора и управления 3D графикой;
- **DataSocket Support** устанавливает поддержку **DataSocket**.

Кроме этого имеется возможность включить в инсталлятор поддержку для обзора и управления удаленными лицевыми панелями Web-сервера LabVIEW (**Remote Panel License Support**), поддержку **serial portion NI-VISA**, необходимую для доступа к **serial instruments or serial port (Serial Port Support)**, поддержку для доступа к параллельным портам и использования ВП **In Port и Out Port (Port I/O Support)**.

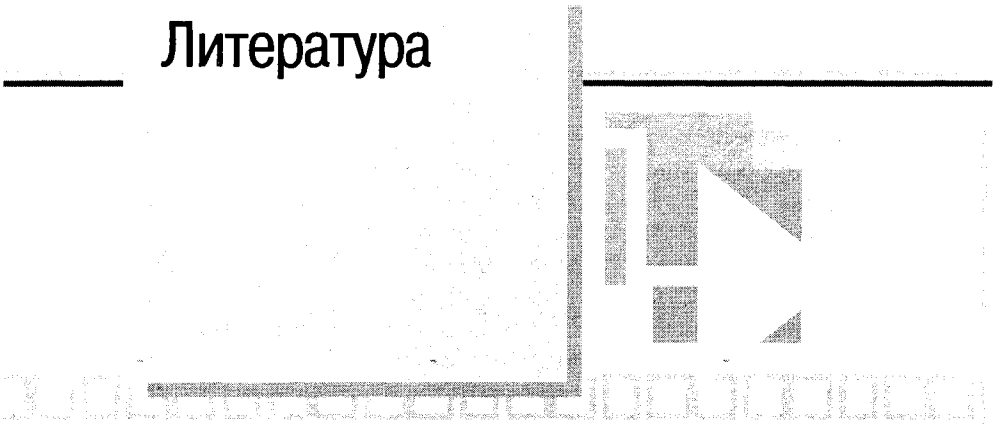
Выделите **Hardware Configuration** для включения в инсталлятор также информации о конфигурации оборудования в **Measurement & Automation Explorer**. Нажмите кнопку **Configure**, чтобы запустить мастер экспорта **Measurement & Automation Explorer** для создания файла конфигурации оборудования.

Внимательная обработка всех настроек инсталлятора позволит настроить установку лишь необходимых библиотек для запуска и последующей работы программы.

Выводы

Программное средство **Application Builder** предназначено для сборки программы, запуск которой не требует установки LabVIEW. Для сборки программы необходимо указать файлы проекта и место для файлов программы. При этом имеются дополнительные возможности, например, установить для запускаемого файла свою иконку или снабдить программу полноценным инсталлятором.

Литература



Основная

1. Тревис Д. LabVIEW для всех. – М.: ДМК Пресс; ПриборКомплект, 2004. – 544 с.
2. LabVIEW™ 7 Express. Базовый курс 1. Издательство National Instruments, 2003.
3. Жарков Ф. П., Каратаев В. ВА., Никифоров В. Ф., Панов В. С. Использование виртуальных инструментов LabVIEW. – М.: Радио и связь, 1999. – 268 с.

Дополнительная

1. П.А. Бутырин, И.С. Козьмина, И.В. Миронов. Основы компьютерных технологий электротехники. – М.: Издательство МЭИ, 2000. – 112 с.
2. П.А. Бутырин, Т.А. Васьковская. Диагностика электрических цепей по частям. Теоретические основы и компьютерный практикум. – М.: Издательство МЭИ, 2003. – 112 с.
3. Б.С. Мельников. Сравнительное моделирование в среде LabVIEW. – М.: Издательство МЭИ, 2003. – 52 с.
4. Б. Патон. LabVIEW: Основы аналоговой и цифровой электроники. Издательство National Instruments, 2002. – 192 с.
5. Н Эртугрул. Лабораторное исследование электрических цепей и машин. Издательство National Instruments, 2002. – 102 с.

Типы данных LabVIEW

Графическое представление	Тип данных			Значение по умолчанию	Диапазон значений	Приблизительное количество десятичных цифр	Размер занимаемой памяти, байт
I8	целое число	со знаком	8 бит	0	–128 ÷ 127	-	1
I16			16 бит	0	32 768 ÷ 32 767	-	2
I32 *			32 бита	0	–2 147 483 648 ÷ 2 147 483 647	-	4
U8		положительное	8 бит	0	0 ÷ 255	-	1
U16			16 бит	0	0 ÷ 65 535	-	2
U32			32 бита	0	0 ÷ 4 294 967 295	-	4
SGL	число с плавающей запятой		одинарная точность	0,0	±1.40e–45 ÷ ±3.40e+38	6	4
DBL *			двойная точность	0,0	±4.94e–324 ÷ ±1.79e+308	15	8
EXT			повышенная точность	0,0	±6.48e–4966 ÷ ±1.19e+4932	18**	10** 16***
CSG	комплексное число		одинарная точность	0,0+i0,0	такие же, как и у соответствующей точности, для каждой (действительной и мнимой) части		8
CDB *			двойная точность	0,0+i0,0			16
CXT			повышенная точность	0,0+i0,0			20** 32***
TS	метка времени time stamp			Время создания	5.42e–20 ÷ 9,22e18 секунд	15	16
TF	Логический Boolean			ложь (F)	может принимать только два значения: ложь (FALSE) или правда(TRUE)		1
abc	строка символов String			пустая строка	содержит текст в ASCII формате		
	Путь к файлу Path			пустой путь	близок к строковому типу, однако, LabVIEW форматирует его, используя стандартный синтаксис для используемой платформы.		

* используется по умолчанию

** для Microsoft Windows

*** при записи на диск

	Тип данных	
	**Перечисление Enumerated**	Беззнаковый численный тип, но каждому числу назначается символьная метка.
	**Массив Array**	Массивы включают типы данных составляющих элементов и принимают соответствующий им цвет.
	**Кластер Cluster**	Упорядоченная совокупность элементов различного типа. Отображается коричневым цветом, если все его элементы численные, иначе отображается розовым.
	**Осциллограмма Waveform**	Является кластером, содержащим массив данных **Y**, начальное значение времени **t0** и интервал времени между измерениями **dt**.
	**Цифровая осциллограмма Digital waveform**	Аналогично Waveform, но массив данных **Y** имеет тип **Digital.**
	**Digital Table**	Таблица двоичных данных.
	**Динамический Dinamic**	Кроме данных сигнала, динамический тип содержит дополнительную информацию, например, название сигнала или дату и время его получения. Большинство экспресс-ВП принимают и/или возвращают данные динамического типа. Данные динамического типа можно направлять к любому элементу отображения или полю ввода, принимающему данные численного, логического или сигнального типа.
	**Ссылка Reference number (refnum)**	Указатель на уникальный объект, например, файл, элемент управления или отображения, ВП.
	**Имя устройства ввода/вывода I/O name**	Позволяет выбирать и передавать имя устройства ввода/вывода.
	**Картинка Picture**	Используется для вывода рисунков.
	**Вариант Variant**	Может содержать любые данные.

Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)

Н. А. Бутырин, Е. А. Васьковская, В.В. Каратаев, С. В. Материкин

Книга состоит из курсо лекций, которые содержат как информацию о возможностях новой версии пакета LabView 7.0, так и практические задания, выполнение которых необходимо для овладения этим прикладным инструментом исследования физических процессов и управления ими.

Каждая глава рассчитана на одно занятие за компьютером и может быть использована как при обучении группы учащихся преподавателем, ток и при самообучении.

Для большей доступности большинство практических занятий ограничивается в книге исследованием чисто виртуальных приборов, что не требует приобретения специальной материальной части (АЦП и др.)

Материал книги разбит на три части. В первой из них даются основные сведения о среде LabVIEW и ее возможностях, об исследовании виртуальных объектов при помощи математического моделирования. Вторая часть посвящена построению виртуальных приборов для проведения измерений в реальных физических устройствах, в частности, дано описание лабораторной установки ELVIS. В третьей части описывается техника составления больших проектов в LabVIEW.

Издание предназначено для инженеров и студентов технических вузов.

СВЕДЕНИЯ ОБ АВТОРАХ

Настоящее издание разработано коллективом сотрудников кафедры Теоретических основ электротехники (ТОЭ) Московского энергетическаго института (МЭИ). На кафедре ТОЭ МЭИ с 2000 года работает учебно-научная лаборатория «Виртуальные приборы электротехники» с 18 рабочими местами, оснащенными аппаратно-программным комплексом LabVIEW. Ежегодно в этой лаборатории около 400 студентов энергетического и электротехнического институтов (факультетов) МЭИ под руководством сотрудников кафедры изучают основы LabVIEW. Сотрудниками кафедры выпущено несколько книг по LabVIEW и его использованию в практике преподавания электротехнических дисциплин.

УРОВЕНЬ ПОЛЬЗОВАТЕЛЯ

- ✓ начинающий
- ✓ средний
- опытный
- профессиональный

NATIONAL INSTRUMENTS

ПриборКомплект

Internet-магазин
www.dmk.ru
www.abook.ru

Книга – почтой*
Россия, 123242,
Москва, а/я 20
e-mail: post@abook.ru

Оптовая продажа:
Альянс-книга
тел./факс: (095) 258-9195
e-mail: abook@abook.ru

* Подробнее см. в конце книги