

# LabVIEW

---

**Руководство пользователя**



**Техническая поддержка и информация о продукции**  
[ni.com](http://ni.com)

**Головной офис корпорации National Instruments**

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

**Региональные офисы**

Australia 61 2 9672 8846, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,  
Canada (Calgary) 403 274 9391, Canada (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949,  
Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530,  
China 86 21 6555 7838, Czech Republic 420 2 2423 5774, Denmark 45 45 76 26 00,  
Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427,  
Hong Kong 2645 3186, India 91 80 51190000, Israel 972 0 3 6393737, Italy 39 02 413091,  
Japan 81 3 5472 2970, Korea 82 02 3451 3400, Malaysia 603 9059 6711, Mexico 001 800 010 0793,  
Netherlands 31 0 348 433 466, New Zealand 64 09 914 0488, Norway 47 0 32 27 73 00,  
Poland 48 0 22 3390 150, Portugal 351 210 311 210, **Russia 7 495 238 7139**, Singapore 65 6 226 5886,  
Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00,  
Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227, United Kingdom 44 0 1635 523545

Более подробно о поддержке см. в приложении *Техническая поддержка и профессиональные услуги*. Замечания на документацию направляйте по E-mail на [info.russia@ni.com](mailto:info.russia@ni.com)

© 1992–2007 National Instruments Corporation. All rights reserved.

© 2007 Перевод на русский язык: Николаев С.В.

# Important Information

## Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

## Trademarks

CVI™, DAQPad™, DataSocket™, DIAdem™, IMAQ™, IVI™, LabVIEW™, Measurement Studio™, National Instruments™, NI™, ni.com™, NI-DAQ™, NI Developer Zone™, NI-IMAQ™, NI-VISA™, NI-VXI™, and SCXI™ are trademarks of National Instruments Corporation. Product and company names mentioned herein are trademarks or trade names of their respective companies.

## Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the **patents.txt** file on your CD, or [ni.com/patents](http://ni.com/patents).

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM

FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

---

# Содержание

Об этом руководстве .....	15
Организация этого руководства .....	16
Соглашения .....	16
Часть I.....	18
Основы LabVIEW .....	18
1. Введение в LabVIEW .....	19
Источники документации LabVIEW .....	19
ВП-шаблоны, ВП-примеры и инструменты в LabVIEW .....	22
<i>ВП-шаблоны в LabVIEW .....</i>	22
<i>ВП-примеры в LabVIEW .....</i>	23
<i>Инструменты LabVIEW .....</i>	23
2. Введение в виртуальные приборы .....	25
Лицевая панель .....	25
Блок-диаграмма .....	26
<i>Терминалы .....</i>	27
<i>Узлы .....</i>	28
<i>Проводники .....</i>	28
<i>Структуры .....</i>	28
Иконка и соединительная панель .....	29
Использование и настройка ВП и ВПП .....	30
3. Среда LabVIEW .....	31
Палитра Controls (Элементы управления) .....	31
Палитра Functions (Функции) .....	32
Навигация по палитрам Controls и Functions .....	32
Палитра инструментов .....	33
Меню и панель инструментов .....	34
<i>Меню .....</i>	34
<i>Панель инструментов (Toolbar) .....</i>	35
Окно контекстной помощи (Context Help Window) .....	35
Настройка Вашего рабочего окружения .....	36
<i>Настройка палитр Controls и Functions .....</i>	36
<i>Установка опций рабочей среды .....</i>	39
4. Построение лицевой панели .....	43
Конфигурирование объектов лицевой панели .....	43
<i>Включение и выключение видимости необязательных элементов .....</i>	44
<i>Преобразование элементов управления в индикаторы и наоборот .....</i>	44

<i>Замена объектов лицевой панели</i> .....	45
<i>Конфигурирование лицевой панели</i> .....	46
<i>Установка комбинаций клавиши для элементов управления</i> .....	46
<i>Раскрашивание объектов</i> .....	48
<i>Использование импортированной графики</i> .....	48
<i>Выравнивание и распределение объектов</i> .....	49
<i>Группировка и фиксация объектов</i> .....	50
<i>Изменение размеров объектов</i> .....	50
<i>Масштабирование объектов лицевой панели</i> .....	51
<i>Добавление свободного пространства на лицевой панели без изменения размеров окна</i> .....	53
<i>Элементы управления и индикаторы лицевой панели</i> .....	53
<i>Трехмерные (3D) и классические элементы управления и индикаторы</i> .....	53
<i>Движки, кнопки, переключатели, цифровые переключатели и временные метки</i> .....	54
<i>Графики и диаграммы (Graphs and Charts)</i> .....	57
<i>Клавиши, переключатели и лампочки (Buttons, Switches, and Lights)</i> .....	57
<i>Окна текстового ввода (Text Entry Boxes), метки (Labels) и дисплеи путей (Path Displays)</i> .....	58
<i>Элементы управления и индикаторы для массивов и кластеров</i> .....	60
<i>Списки, деревья и таблицы</i> .....	61
<i>Кольцевые и перечислительные элементы управления и индикаторы</i> .....	63
<i>Контейнерные элементы управления (Container Controls)</i> .....	65
<i>Элементы управления и индикаторы имен ввода/вывода (I/O Name)</i> .....	68
<i>Ссылки на объекты или приложения</i> .....	74
<i>Диалоговые элементы управления и индикаторы</i> .....	74
<i>Использование текстовых меток</i> .....	75
<i>Заголовки (Captions)</i> .....	76
<i>Текстовые характеристики</i> .....	77
<i>Конструирование пользовательского интерфейса</i> .....	78
<i>Использование элементов управления и индикаторов лицевой панели</i> .....	79
<i>Конструирование диалоговых окон</i> .....	80
<i>Выбор размера экрана</i> .....	80
5. Построение блок-диаграммы .....	81
<i>Соответствие между объектами лицевой панели и терминалами блок-диаграммы</i> .....	81
<i>Объекты блок-диаграммы</i> .....	81
<i>Терминалы блок-диаграммы</i> .....	81
<i>Узлы блок-диаграммы</i> .....	87
<i>Обзор функций</i> .....	88
<i>Числовые функции</i> .....	89
<i>Логические функции</i> .....	89

<i>Строковые функции</i> .....	89
<i>Функции над массивами</i> .....	90
<i>Кластерные функции</i> .....	90
<i>Функции сравнения</i> .....	90
<i>Временные и диалоговые функции</i> .....	90
<i>Функции файлового ввода/вывода</i> .....	91
<i>Функции работы с осциллографами</i> .....	91
<i>Функции управления приложениями</i> .....	92
<i>Дополнительные функции</i> .....	92
<i>Добавление терминалов у функций</i> .....	92
<i>Использование проводников для связи объектов блок-диаграммы</i> .....	93
<i>Автоматическое соединение объектов</i> .....	95
<i>Соединение объектов вручную</i> .....	95
<i>Прокладка проводников</i> .....	96
<i>Селектирование проводников</i> .....	97
<i>Исправление поврежденных проводников</i> .....	97
<i>Точки принудительного преобразования</i> .....	98
<i>Полиморфные ВП и функции</i> .....	99
<i>Полиморфные ВП</i> .....	99
<i>Полиморфные функции</i> .....	103
<i>Экспресс ВП</i> .....	103
<i>Создание ВПП из экспресс ВП</i> .....	103
<i>Динамический тип данных</i> .....	104
<i>Обработка вариантов данных</i> .....	107
<i>Числовые единицы и строгая проверка типов</i> .....	108
<i>Единицы и строгая проверка типов</i> .....	109
<i>Поток данных на блок-диаграмме</i> .....	111
<i>Зависимость по данным и искусственная зависимость по данным</i> .....	112
<i>Поток данных и управление памятью</i> .....	113
<i>Конструирование блок-диаграммы</i> .....	114
6. <i>Запуск и отладка виртуальных приборов</i> .....	116
<i>Запуск ВП</i> .....	116
<i>Настройка способа запуска ВП</i> .....	117
<i>Исправление поврежденных виртуальных приборов</i> .....	117
<i>Поиск причин повреждения ВП</i> .....	117
<i>Типичные причины повреждения ВП</i> .....	118
<i>Технология отладки</i> .....	119
<i>Подсвечивание выполнения</i> .....	121
<i>Пошаговое выполнение</i> .....	121
<i>Инструмент Probe (пробник)</i> .....	122
<i>Точки прерывания</i> .....	125
<i>Приостановка исполнения</i> .....	126

<i>Комментирование сегментов блок-диаграммы.....</i>	127
Отключение инструментов отладки.....	127
Неопределенные или неожиданные данные.....	127
<i>Данные по умолчанию в циклах.....</i>	128
<i>Значения по умолчанию в массивах.....</i>	129
<i>Предотвращение неопределенных данных.....</i>	129
Проверка и обработка ошибок.....	129
<i>Проверка на ошибки .....</i>	130
<i>Обработка ошибок.....</i>	131
7. Создание ВП и ВПП .....	134
Планирование и построение вашего проекта .....	134
<i>Разработка проектов несколькими разработчиками.....</i>	135
Шаблоны ВП .....	136
<i>Создание шаблонов ВП.....</i>	136
<i>Другие типы документов.....</i>	136
Использование встроенных ВП и функций .....	136
<i>Построение ВП и функций для управления приборами и ввода/вывода данных.....</i>	137
<i>Построение ВП, которые имеют доступ к другим ВП.....</i>	137
<i>Построение ВП, которые общаются с другими приложениями.....</i>	138
Виртуальные подприборы (ВПП).....	138
<i>Выявление однотипных операций.....</i>	139
<i>Конфигурирование соединительной панели .....</i>	140
<i>Создание иконки.....</i>	143
<i>Отображение ВПП и экспресс ВП в виде иконок либо в виде расширяемых узлов.....</i>	144
<i>Создание ВПП из фрагмента ВП.....</i>	145
<i>Конструирование ВПП.....</i>	146
<i>Просмотр иерархии ВП.....</i>	146
Сохранение ВП.....	147
<i>Преимущества сохранения каждого ВП в отдельном файле .....</i>	147
<i>Преимущества сохранения нескольких ВП в библиотеках.....</i>	148
<i>Управление виртуальными приборами в библиотеках.....</i>	149
<i>Имена ВП.....</i>	149
<i>Сохранение в формате предыдущей версии LabVIEW.....</i>	150
Распространение виртуальных приборов .....	150
Построение стандартных приложений и библиотек совместного доступа .....	151
Часть II.....	153
Построение и редактирование виртуальных приборов .....	153
8. Циклы и структуры.....	155
Структуры For Loop и While Loop.....	156
<i>Структура For Loop .....</i>	156

<i>Структура While Loop</i> .....	157
<i>Автоиндексация циклов</i> .....	159
<i>Использование циклов для построения массивов</i> .....	161
<i>Сдвигающие регистры и узел обратной связи в циклах</i> .....	161
<i>Узел обратной связи</i> .....	165
<i>Управление синхронизацией</i> .....	167
Структуры выбора и последовательности .....	167
<i>Структура Case</i> .....	168
<i>Структуры последовательности</i> .....	170
9. Событийно управляемое программирование .....	176
<i>Что такое события?</i> .....	176
<i>Зачем использовать события?</i> .....	177
<i>Компоненты структуры Event</i> .....	178
<i>Использование событий в LabVIEW</i> .....	181
<i>Пользовательские события</i> .....	191
10. Группировка данных с использованием строк, массивов и кластеров.....	195
<i>Строки</i> .....	195
<i>Строки на лицевой панели</i> .....	196
<i>Таблицы</i> .....	197
<i>Программное редактирование строк</i> .....	197
<i>Форматирование строк</i> .....	198
<i>Числа и строки</i> .....	199
<i>Конвертирование данных в XML и обратно</i> .....	200
<i>Группировка данных в массивы и кластеры</i> .....	203
<i>Массивы</i> .....	203
<i>Кластеры</i> .....	210
11. Локальные и глобальные переменные .....	213
<i>Локальные переменные</i> .....	213
<i>Создание локальных переменных</i> .....	214
<i>Глобальные переменные</i> .....	214
<i>Создание глобальных переменных</i> .....	215
<i>Чтение и запись значений переменных</i> .....	216
<i>Предосторожности при использовании локальных и глобальных переменных</i> .....	217
<i>Инициализация локальных и глобальных переменных</i> .....	217
<i>Эффект гонок</i> .....	217
<i>Соображения о памяти при использовании локальных переменных</i> .....	218
<i>Соображения о памяти при использовании глобальных переменных</i> .....	218
12. Графики и диаграммы.....	220
<i>Типы графиков и диаграмм</i> .....	220
<i>Опции индикаторов Graph и Chart</i> .....	221
<i>Несколько шкал X и Y в индикаторах Graph и Chart</i> .....	221

<i>Сглаживание линий графиков на индикаторах Graph и Chart</i> .....	221
<i>Настройка внешнего вида индикаторов Graph и Chart</i> .....	222
<i>Настройка индикаторов Graph</i> .....	223
<i>Настройка индикаторов Chart</i> .....	226
Индикаторы Waveform Graph и XY Graph.....	228
<i>Типы данных для построения одной кривой на индикаторе Waveform Graph</i> .....	229
<i>Построение нескольких кривых на индикаторе Waveform Graph</i> .....	230
<i>Типы данных для построения одной кривой на индикаторе XY Graph</i> .....	232
<i>Типы данных для построения нескольких кривых на индикаторе XY Graph</i> .....	232
.....	232
Индикаторы Waveform Chart .....	232
Индикаторы Intensity Graph и Intensity Chart.....	233
<i>Схема отображения цветов</i> .....	235
<i>Необязательные компоненты индикатора Intensity Chart</i> .....	236
<i>Необязательные компоненты индикатора Intensity Graph</i> .....	236
Индикаторы Digital Waveform Graph .....	237
<i>Маскирование данных</i> .....	239
Трехмерные графики .....	240
Тип данных waveform (осциллограмма) .....	241
Тип данных digital waveform (цифровая осциллограмма).....	242
13. Графика и звук .....	243
Использование индикатора Picture (изображение) .....	243
ВП с палитры Picture Plots (изображение кривых) .....	244
<i>Использование ВП Polar Plot в качестве ВПП</i> .....	245
<i>Использование ВП Plot Waveform и Plot XY в качестве ВПП</i> .....	245
<i>Использование ВП Smith Plot в качестве ВПП</i> .....	246
ВП с палитры Picture Functions.....	247
<i>Задание и модификация цветов с помощью ВП с палитры Picture Functions</i> .....	249
ВП с палитры Graphics Formats .....	249
ВП с палитры Sound.....	250
14. Файловый ввод/вывод .....	252
Основы файлового ввода/вывода .....	252
Выбор формата файлового ввода/вывода .....	253
<i>Когда использовать текстовые файлы</i> .....	253
<i>Когда использовать двоичные файлы</i> .....	255
<i>Когда использовать файлы протоколов данных</i> .....	256
Использование ВП высокогоуровневого файлового ввода/вывода .....	257
Использование ВП и функций низкого уровня и с подпалитры Advanced File I/O .....	258
<i>Дисковый поток</i> .....	260

Создание текстовых файлов и файлов электронных таблиц.....	260
<i>Форматирование и запись данных в файлы .....</i>	261
<i>Сканирование данных из файлов.....</i>	262
Создание двоичных файлов.....	262
Создание файлов протоколов данных .....	262
Запись в файл осциллографм.....	263
Чтение осциллографм из файлов .....	264
Потоковые параметры.....	265
Создание конфигурационных файлов .....	265
<i>Использование установок конфигурационных файлов .....</i>	266
<i>Формат конфигурационного файла в Windows .....</i>	267
Регистрация данных лицевой панели .....	268
<i>Автоматическая и интерактивная регистрация данных лицевой панели.</i>	269
<i>Интерактивный просмотр протокола данных лицевой панели.....</i>	270
<i>Программное восстановление данных лицевой панели.....</i>	272
15. Документирование и печать виртуальных приборов.....	276
Документирование ВП.....	276
<i>Установка истории изменений ВП.....</i>	277
<i>Создание описаний ВП и объектов.....</i>	278
<i>Печать документации.....</i>	278
Печать виртуальных приборов.....	281
<i>Печать активного окна.....</i>	281
<i>Программная печать ВП.....</i>	282
<i>Дополнительные способы печати .....</i>	284
16. Конфигурирование ВП .....	285
Конфигурирование внешнего вида и поведения ВП.....	285
Изменение меню.....	286
<i>Создание меню.....</i>	287
<i>Обработка пунктов меню .....</i>	287
17. Программное управление ВП.....	289
Возможности сервера виртуальных приборов.....	289
Построение приложений сервера ВП .....	290
<i>Ссылки на приложение и на ВП .....</i>	291
Манипуляция установками приложения и ВП .....	292
<i>Узлы свойств .....</i>	292
<i>Узлы вызовов.....</i>	293
<i>Манипуляция свойствами и методами класса Приложение.....</i>	294
<i>Манипуляция свойствами и методами класса Виртуальный прибор .....</i>	295
<i>Манипуляция свойствами и методами класса Приложение и класса ВП ..</i>	295
Динамическая загрузка и вызов ВП.....	296
<i>Узлы вызова по ссылке и строго типизированные ссылочные номера ВП</i>	296
Редактирование и запуск ВП на удаленных компьютерах .....	297

Управление объектами лицевой панели .....	298
<i>Строго и слабо типизированные ссылочные номера элементов управления</i>	298
18. Сетевые коммуникации в LabVIEW .....	301
Выбор между файловым вводом/выводом, сервером виртуального прибора, технологией ActiveX и работой в сети.....	301
LabVIEW в качестве сетевого клиента и сервера .....	302
Использование технологии DataSocket.....	302
<i>Задание URL .....</i>	303
<i>Форматы данных, поддерживаемые DataSocket.....</i>	305
<i>Использование DataSocket на лицевой панели .....</i>	306
<i>Чтение и запись общих данных через блок-диаграмму.....</i>	307
Опубликование виртуальных приборов на Web .....	312
<i>Опции Web сервера .....</i>	312
<i>Создание HTML документов .....</i>	313
<i>Опубликование образов лицевой панели.....</i>	313
Удаленный просмотр и управление лицевыми панелями .....	314
<i>Настройка сервера для клиентов.....</i>	315
<i>Просмотр и управление лицевыми панелями в LabVIEW или из Web броузера .....</i>	316
<i>Функциональность, не поддерживаемая при удаленном просмотре лицевых панелей и управлении ими.....</i>	318
Отправка данных из ВП по электронной почте .....	319
<i>Выбор набора символов.....</i>	320
Низкоуровневые коммуникационные приложения .....	323
<i>Протоколы TCP и UDP .....</i>	323
<i>События Apple и PPC Toolbox (Mac OS) .....</i>	324
<i>ВП с палитры Pipe (Unix) .....</i>	325
<i>Выполнение команд системного уровня (Windows и Unix) .....</i>	325
19. Связность в среде Windows.....	326
Окружение .NET .....	327
Функции и узлы с палитры .NET .....	328
LabVIEW в качестве .NET клиента .....	329
Отображение типов данных .....	330
Развертывание .NET приложений .....	331
<i>Развертывание исполняемого модуля .....</i>	331
<i>Развертывание виртуальных приборов .....</i>	331
<i>Развертывание библиотек DLL .....</i>	331
Конфигурирование приложения .NET клиента.....	331
Объекты, свойства, методы и события ActiveX .....	332
<i>Виртуальные приборы, функции, элементы управления и индикаторы, работающие с ActiveX.....</i>	332

LabVIEW в качестве клиента ActiveX .....	333
Доступ к ActiveX приложениям .....	334
Вставка объекта ActiveX на лицевую панель .....	334
Установка свойств ActiveX.....	335
LabVIEW в качестве сервера ActiveX .....	337
Поддержка настраиваемых автоматических интерфейсов ActiveX.....	338
Использование констант для установки параметров в виртуальных приборах с возможностями ActiveX.....	338
События ActiveX .....	339
Обработка событий ActiveX.....	340
20. Вызов кода из текстовых языков программирования .....	343
Узел вызова библиотечной функции .....	343
Узел кодового интерфейса.....	343
21. Формулы и уравнения.....	344
Методы использования выражений в LabVIEW .....	344
Формульные узлы.....	345
Использование формульного узла .....	345
Переменные в формульном узле .....	346
Узлы выражения .....	347
Полиморфизм в узлах выражения.....	348
Узел скриптов приложения MATLAB .....	348
Рекомендации по программированию скриптов приложения MATLAB .....	349

<b>A.</b>	Организация LabVIEW .....	351
Структура директорий LabVIEW.....	351	
Библиотеки .....	351	
Структура и поддержка.....	352	
Изучение и инструкции.....	352	
Документация.....	352	
Mac OS.....	352	
Предлагаемые места для сохранения файлов .....	353	

<b>B.</b>	Полиморфные функции .....	355
Преобразование числовых представлений.....	355	
Полиморфизм числовых функций .....	357	
Полиморфизм булевых функций .....	359	
Полиморфизм для функций обработки массивов.....	360	
Полиморфизм строковых функций.....	360	
Полиморфизм функций для конвертирования строк.....	360	
Полиморфизм дополнительных функций преобразования строк в числа...361	361	
Полиморфизм функций обработки кластеров .....	361	

Полиморфизм функций сравнения.....	361
Полиморфизм логарифмических функций .....	363
<b>C.</b>	
Функции сравнения .....	364
Сравнение булевых значений .....	364
Сравнение символьных строк.....	364
Сравнение чисел.....	365
Сравнение массивов и кластеров.....	365
<i>Массивы</i> .....	365
<i>Кластеры</i> .....	366
<b>D.</b>	
Техническая поддержка и профессиональные услуги ..	368

---

## Об этом руководстве

В данном руководстве описана среда графического программирования LabVIEW и техника создания в ней таких приложений, как тестирование и измерение, сбор данных, управление приборами, регистрация данных, анализ измерений и генерация отчетов.

Руководство пользователя можно использовать для изучения возможностей программирования в LabVIEW, включая пользовательский интерфейс и рабочее пространство, а также палитры и инструменты LabVIEW. В него не вошла специфическая информация относительно каждой палитры, инструмента, меню, диалогового окна, элемента управления и встроенных ВП и функций. Для получения дополнительной информации об этом, а также для детальных пошаговых инструкций об использовании возможностей LabVIEW и построения специальных приложений, следует пользоваться встроенной справочной системой LabVIEW (*LabVIEW Help*). Более подробно о пользовании справочной системой см. раздел *Источники документации LabVIEW* в Главе 1 *Введение в LabVIEW*.

*Руководство пользователя LabVIEW (LabVIEW User Manual)* имеется также в виде файла формата PDF. Если при инсталляции LabVIEW Вы выберете опцию **Complete**, будут установлены PDF-версии всех руководств LabVIEW, доступ к которым можно получить, выбирая в среде LabVIEW пункт главного меню **Help»Search the LabVIEW Bookshelf**.



**Примечание.** Для просмотра PDF-файлов Вам нужно установить Adobe Acrobat Reader версии 5.0.5 или выше. Чтобы скачать Acrobat Reader, воспользуйтесь Web сайтом компании Adobe Systems Incorporated [www.adobe.com](http://www.adobe.com).

Вы можете пользоваться PDF-файлами из справочной системы LabVIEW (*LabVIEW Help*), но, чтобы это работало, PDF-файлы должны быть установлены. Более подробно об использовании PDF-файлов на книжной полке LabVIEW (*LabVIEW Bookshelf*) см. в под-

## Организация этого руководства

---

*Руководство пользователя LabVIEW* состоит из двух частей. Часть I, *Основы LabVIEW*, описывает программные концепции для построения приложений в LabVIEW. Главы этой части вводят Вас в среду программирования и помогут Вам спланировать ваше приложение.

Часть II, *Построение и редактирование ВП*, описывает возможности LabVIEW, ВП и функции, которые Вы можете использовать, чтобы обеспечить работу ваших приложений в конкретных случаях. В главах этого раздела описано использование всех возможностей LabVIEW и дан обзор каждого класса ВП и функций.

## Соглашения

---

В этом руководстве приняты следующие соглашения:

» Символ » ведет нас через вложенные пункты меню и опции диалогового окна к конечному действию. Последовательность **File»Page Setup** » **Options** говорит о том, что в пункте меню **File** следует выбрать пункт **Page Setup** и затем выбрать **Options** из последнего диалогового окна.



Эта иконка обозначает совет, который привлекает вас к консультативной информации.



Эта иконка обозначает примечание, которое привлекает вас к важной информации.



Эта иконка обозначает предупреждение, которое рекомендует Вам меры предосторожности, позволяющие избежать повреждений, потерь данных или аварии системы.

**жирный** Жирный текст обозначает пункты, которые Вы должны выбрать или кликнуть в программе, такие как пункты меню и опции диалоговых окон. Жирный текст обозначает также имена параметров, элементов

управления и кнопок на лицевой панели, на диалоговых окнах, секциях диалоговых окон, названия меню и имена палитр.

*курсив*

Курсивом обозначены переменные, важные места, перекрестные ссылки или введение в ключевую концепцию. Такой шрифт также обозначает текст, который является местом-заменителем для слова или значения, которое Вы должны подставить.

**моноширинный**

Текст в таком шрифте обозначает текст или символы, которые Вам нужно ввести с клавиатуры, части кода, примеры программирования и синтаксиса. Такой шрифт используется также для имен дисковых приводов, путей, каталогов, программ, подпрограмм, имен приборов, функций, операций, переменных, имен и расширенных файлов и кодовых фрагментов.

**моноширинный жирный**

Текст, выделенный таким шрифтом, обозначает сообщения и ответы, которые компьютер автоматически выдает на экран. Такой шрифт также выделяет строки кода, которые отличаются от других примеров.

**моноширинный курсив**

Курсив в этом шрифте обозначает текст, который служит местом-заменителем для слова или значения, которое Вы должны подставить.

**Platform**

Таким шрифтом помечается конкретная платформа. Это указывает на то, что следующий за этим текст относится только к конкретной платформе (**Windows**, **UNIX** или **Mac OS**).

**щелчок правой кнопкой**

**(Mac OS)** Чтобы выполнить такое же действие, как щелчок правой кнопкой, нажмите <Command>-щелчок левой кнопкой.

# Часть I

---

## Основы LabVIEW

В этой части описаны основные программные понятия для создания приложений в LabVIEW. Главы этого раздела введут Вас в среду программирования LabVIEW и помогут Вам спланировать ваше приложение.

Часть I, *Основные понятия LabVIEW*, содержит следующие главы:

- Глава 1, *Введение в LabVIEW*, описывает LabVIEW, его обширную документацию и инструменты, которые помогут Вам спроектировать и построить виртуальные приборы.
- Глава 2, *Введение в виртуальные приборы*, описывает компоненты виртуальных приборов или ВП.
- Глава 3, *Среда LabVIEW*, описывает палитры, инструменты и меню, которыми Вы можете пользоваться для построения лицевых панелей и блок-диаграмм ВП. В этом разделе также описано, как настраивать палитры и устанавливать некоторые опции рабочего окружения.
- Глава 4, *Построение лицевой панели*, описывает, как построить лицевую панель ВП.
- Глава 5, *Построение блок-диаграммы*, описывает, как построить блок-диаграмму ВП.
- Глава 6, *Запуск и отладка ВП*, описывает, как сконфигурировать прогон ВП и определить проблемы в организации блок-диаграммы или в прохождении данных через блок-диаграмму.
- Глава 7, *Создание ВП и ВПП*, описывает, как создать ваши собственные ВП, распространять их и строить стандартные приложения и библиотеки совместного доступа.

# 1. Введение в LabVIEW

---

LabVIEW это язык графического программирования, в котором для создания приложения используются иконки вместо строк текста. В противоположность текстовым языкам программирования, где выполнение программы определяется последовательностью инструкций, LabVIEW использует потоковое программирование (dataflow programming), в котором последовательность выполнения определяется потоком данных.

В LabVIEW пользовательский интерфейс строится на основе множества инструментов и объектов. Пользовательский интерфейс называется лицевой панелью (front panel). Затем для управления объектами лицевой панели Вы добавляете код, используя графическое представление функций. Блок-диаграмма (block diagram) содержит этот код. В некоторых случаях блок-диаграмма имеет сходство со структурной схемой.

Вы можете приобрести некоторые дополнительные программные инструменты-расширения для разработки специализированных приложений. Все эти инструменты полностью интегрированы в LabVIEW. Полная информация об этих программных инструментах размещена на Web сайте компании National Instruments [ni.com](http://ni.com).

## Источники документации LabVIEW

---

LabVIEW имеет обширную документацию как для опытных пользователей LabVIEW, так и для новичков. Все руководства (manuals) по LabVIEW и примечания к приложениям (Application Notes) имеются также в виде PDF-файлов. Для просмотра PDF-файлов Вам нужно установить Adobe Acrobat Reader версии 5.0.5 или выше. Чтобы скачать Acrobat Reader, обратитесь на Web сайт Adobe Systems Incorporated [www.adobe.com](http://www.adobe.com). Для обновления документальных источников обратитесь к библиотеке руководств на продукцию National Instruments на сайте [ni.com/manuals](http://ni.com/manuals).

- **LabVIEW Bookshelf** (Книжная полка)– Используйте этот PDF-файл для поиска PDF-версий всех руководств LabVIEW и примечаний к приложениям. Для доступа к *LabVIEW Bookshelf* выберите **Help»Search the LabVIEW Bookshelf**.

- **Getting Started with LabVIEW** (Вводный курс LabVIEW) –

Используйте это руководство для самостоятельного ознакомления со средой графического программирования LabVIEW и с основными возможностями LabVIEW при построении приложений по сбору данных и управлению.

- **LabVIEW Quick Reference Card** (Карта быстрых ссылок) –

Используйте эту карту как ссылку на информацию по источникам помощи, быстрым клавишам, типам данных терминалов и по инструментам редактирования, выполнения и отладки.

- **LabVIEW User Manual** (данное руководство пользователя) –

Используйте это руководство для изучения в среде LabVIEW программных концепций, технологий, возможностей, ВП и функций, которые Вы можете использовать при создании приложений для тестирования и измерения, сбора данных, управления приборами, регистрации данных, анализа измерений и генерации отчетов.

- **LabVIEW Help** – используйте файл справки как ссылку в среде LabVIEW на информацию о палитрах, меню, инструментах, ВП и функциях. Включает также пошаговые инструкции по использованию возможностей LabVIEW. Для активизации *LabVIEW Help* нужно выбрать **Help»VI, Functions, and How-To Help**.

*LabVIEW Help* содержит ссылки на следующие ресурсы:

- *LabVIEW Bookshelf*, который содержит PDF-версии всех руководств LabVIEW и примечаний к приложениям

- Ресурсы технической поддержки на Web сайте компании National Instruments, такие как NI Developer Zone, KnowledgeBase и Product Manuals Library.

- **LabVIEW Measurements Manual** (Руководство по измерениям) –

Используйте это руководство для изучения особенностей построения в LabVIEW приложений сбора данных и управления. Если Вы пользователь-новичок в LabVIEW, то предварительно прочтите руководства *Getting Started with LabVIEW* и *LabVIEW User Manual*.

- **LabVIEW Application Builder User Guide** – Используйте этот

документ для изучения LabVIEW Application Builder (Построитель приложения LabVIEW), который имеется в составе LabVIEW Professional Development System и может также приобретаться отдельно. Это руководство содержит инструкции по инсталляции Application Builder, описывает требования к системе и перечисляет изменения между предыдущими версиями и текущей. В этом

руководстве описаны также предостережения и рекомендации относительно того, компилировать ли ВП в приложение или в совместную библиотеку.

- **LabVIEW Development Guidelines** (Руководящие принципы разработчика) – Используйте это руководство, чтобы узнать, как строить ВП легким для понимания, использования и модификации. В этом руководстве описываются способы отслеживания проекта, проектирования и документирования. В нем также описан рекомендованный стиль разработки.



**Примечание.** Руководство LabVIEW Development Guidelines в распечатанном виде имеется только в составе LabVIEW Professional Development System. PDF-версия доступна во всех пакетах поставки LabVIEW.

- **LabVIEW Analysis Concepts** (Основы анализа в LabVIEW) – Используйте это руководство, чтобы узнать о концепциях анализа, используемых в LabVIEW. Это руководство содержит информацию о генерации сигналов, быстром преобразовании Фурье (БПФ) и дискретном преобразовании Фурье (ДПФ), сглаживающих окнах, аппроксимации кривых, линейной алгебре, фундаментальных основах теории вероятности и математической статистики, поточечном анализе для обработки в реальном времени.



**Примечание.** Руководство *LabVIEW Analysis Concepts* имеется только в виде PDF-версии

- **Using External Code in LabVIEW** (Использование внешнего кода в LabVIEW) – Читайте это руководство, чтобы узнать как использовать узлы кодового интерфейса и внешние подпрограммы для импорта кода, написанного на текстовых языках программирования. Руководство включает сведения относительно вызова DLL, общих внешних процедур, библиотек функций, подпрограмм управления памятью и файлами и подпрограмм диагностики.



**Примечание.** Руководство *Using External Code in LabVIEW* имеется только в виде PDF-версии.

- **LabVIEW Release Notes** (Примечания к выпуску) – Используйте эти примечания при инсталляции и дезинсталляции LabVIEW.

Эти примечания описывают также системные требования для LabVIEW и известные версии LabVIEW.

- **LabVIEW Upgrade Notes** (Примечания к обновлениям) – Используйте эти примечания при обновлении LabVIEW на новую версию. В примечаниях описаны также новые возможности и особенности, которые Вы можете получить при обновлении.
- **LabVIEW Application Notes** (Примечания к приложению) – Используйте эти примечания, чтобы узнать о дополнительных и специализированных концепциях и приложениях LabVIEW. Для доступа к последним и обновленным примечаниям к приложению см. NI Developer Zone на сайте [ni.com/zone](http://ni.com/zone).
- **LabVIEW VXI VI Reference Manual** (Справочное руководство по VXI ВП) – Используйте это руководство, чтобы познакомиться с VXI виртуальными приборами для LabVIEW. Это руководство служит дополнением к *NI-VXI Programmer Reference Manual* (Справочное руководство программиста NI-VXI), которое поставляется с аппаратурой VXI. NI рекомендует применять технологию VISA для конфигурирования, программирования и тестирования инструментальных систем, содержащих аппаратуру VXI.



**Примечание.** Руководство *LabVIEW VXI VI Reference Manual* имеется только в виде PDF-версии.

## **ВП-шаблоны, ВП-примеры и инструменты в LabVIEW**

Используйте ВП-шаблоны, ВП-примеры и инструменты LabVIEW для облегчения проектирования и построения виртуальных приборов.

### **ВП-шаблоны в LabVIEW**

ВП-шаблоны в LabVIEW содержат виртуальные подприборы (ВПП), функции, структуры и объекты лицевой панели, которые могут помочь быстро начать построение типовых измерительных приложений. ВП-шаблоны открываются как ВП с именем **untitled**, которые Вы должны затем сохранить. Выберите **File»New**, чтобы отобразить диалоговое окно **New**, которое содержит ВП-шаблоны. Вы также можете вывести диалоговое окно **New**, нажимая кнопку **New** на стартовом диалоговом окне LabVIEW.

## ВП-примеры в LabVIEW

LabVIEW содержит сотни ВП-примеров, которые Вы можете использовать и встраивать в свои собственные ВП. Вы можете модифицировать примеры, чтобы подогнать их под ваше приложение, либо копировать и вставлять из одного или нескольких примеров в ваш собственный ВП. Для просмотра или поиска ВП-примеров выберите **Help»Find Examples**. Другие ВП-примеры можно найти в разделе NI Developer Zone сайта [ni.com/zone](http://ni.com/zone).

## Инструменты LabVIEW

LabVIEW содержит много инструментов, чтобы помочь Вам быстро сконфигурировать ваши измерительные устройства. Через меню **Tools** можно получить доступ к следующим инструментам.

- **(Windows)** Measurement & Automation Explorer (MAX) – поможет Вам сконфигурировать аппаратное и программное обеспечение National Instruments.
- **(Mac OS 9 или более ранняя версия)** NI-DAQ Configuration Utility – поможет Вам сконфигурировать аппаратуру ввода-вывода National Instruments.
- **(Mac OS 9 или более ранняя версия)** DAQ Channel Wizard – поможет Вам определить типы устройств, подсоединенных к аппаратным каналам ввода-вывода, и сохранить все их настройки.
- **(Mac OS 9 или более ранняя версия)** DAQ Channel Viewer – выдает список всех сконфигурированных каналов ввода-вывода.
- **(Mac OS 9 или более ранняя версия)** DAQ Solution Wizard – поможет Вам найти решение для типовых приложений ввода-вывода. Можно выбрать из готовых ВП-примеров или создать свои собственные ВП.

Для графического конфигурирования каналов и решения типовых измерительных задач используйте DAQ Assistant (Помощник по вводу-выводу). DAQ Assistant можно активизировать одним из следующих способов:

- Поместите экспресс-ВП DAQ Assistant на блок-диаграмму.
- Щелкните правой кнопкой элемент управления DAQmx Global Channel и выберите из контекстного меню пункт **New Channel**

**(DAQ Assistant).** Щелкните правой кнопкой элемент управления DAQmx Task Name и выберите из контекстного меню пункт **New Task (DAQ Assistant)**. Щелкните правой кнопкой элемент управления DAQmx Scale Name и выберите из контекстного меню пункт **New Scale (DAQ Assistant)**.

- Запустите Measurement & Automation Explorer (MAX) и выберите из дерева **Configuration** ветвь **Data Neighborhood** или ветвь **Scales**. Щелкните кнопку **Create New**. Сконфигурируйте NI-DAQmx канал, задачу или шкалу.

Более подробно об использовании DAQ Assistant см. в *LabVIEW Measurements Manual* (Руководство по измерениям в LabVIEW).

## **2. Введение в виртуальные приборы**

---

Программы в LabVIEW называются виртуальными приборами или ВП (Virtual Instruments - VI), поскольку их внешний вид и поведение имитируют физические приборы, такие как осциллографы и мультиметры. Каждый ВП использует функции, которые обрабатывают входные данные от пользовательского интерфейса или иных источников и отображают информацию либо перемещают ее в другие файлы или другие компьютеры.

ВП содержат три следующих компонента:

- **Лицевая панель (Front panel)** – Служит в качестве пользовательского интерфейса.
- **Блок диаграмма (Block diagram)** – Содержит графический исходный код, который определяет функционирование ВП.
- **Иконка и соединительная панель (Icon and connector pane)** – идентифицируют ВП таким образом, чтобы его можно было использовать в другом ВП. ВП внутри другого ВП называется виртуальным подприбором - ВПП (SubVI). ВПП соответствует подпрограмме (процедуре) в текстово-ориентированных языках программирования.

---

**Более подробно...**

Более подробно относительно создания ВП и ВПП см. справочную систему *LabVIEW Help*.

---

### **Лицевая панель**

---

Лицевая панель – это пользовательский интерфейс ВП. На Figure 2-1 показан пример лицевой панели.

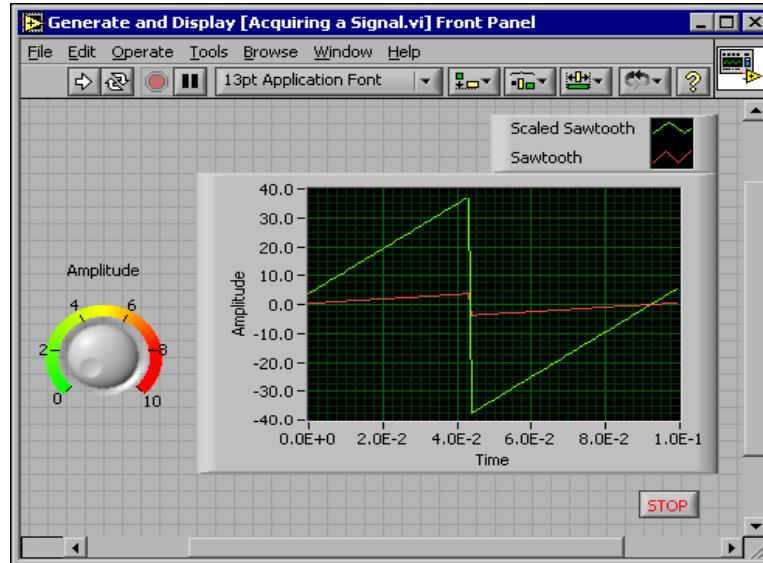


Figure 2-1. Пример лицевой панели

Вы создаете лицевую панель с элементами управления (controls) и индикаторами (indicators), которые являются интерактивными входными и выходными терминалами ВП, соответственно. Элементами управления могут быть регуляторы, кнопки, переключатели и другие входные приборы. Индикаторами могут быть графики, светодиоды и другие устройства отображения. Элементы управления имитируют входные устройства прибора и подают данные на блок диаграмму ВП. Индикаторы имитируют выходные устройства прибора и отображают данные, которые блок диаграмма получает или генерирует. Более подробно относительно лицевой панели см. в Главе 4 *Построение лицевой панели*.

## Блок-диаграмма

После создания лицевой панели нужно добавить программный код посредством графического представления функций управления объектами лицевой панели. Блок-диаграмма как раз и содержит этот графический исходный код. Объекты лицевой панели представляются на блок-диаграмме как терминалы. Более подробно относительно блок-диаграммы см. в Главе 5 *Создание блок-диаграммы*.

ВП на Figure 2-2 демонстрирует некоторые простейшие объекты блок-диаграммы.

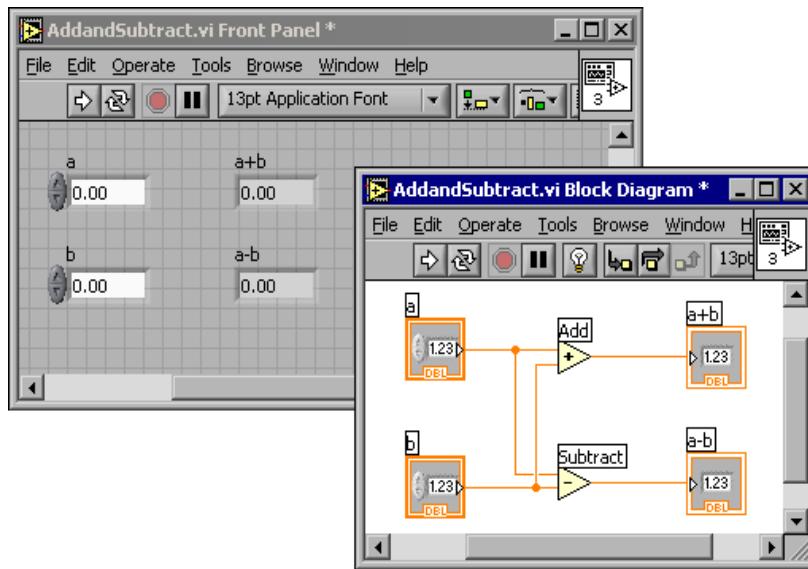


Figure 2-2. Пример блок-диаграммы и соответствующей лицевой панели

## Терминалы

Терминалы представляют типы данных элементов управления или индикаторов. Вы можете конфигурировать элементы управления и индикаторы лицевой панели так, что на блок-диаграмме они будут представлены терминалами либо в виде иконок, либо в виде терминалов типов данных. По умолчанию объекты лицевой панели имеют вид терминалов-иконок. Например, терминал-иконка кнопки, показанный слева, представляет кнопку на лицевой панели. Символы DBL внизу этой иконки обозначают числовой тип данных удвоенной точности с плавающей точкой. Терминал DBL, показанный слева ниже, представляет числовой элемент управления или индикатор удвоенной точности с плавающей точкой. Более подробно типы данных в LabVIEW и их графические представления описаны в разделе *Типы данных элементов управления и индикаторов* в Главе 5, *Построение блок-диаграммы*.



Терминалы являются входными и выходными портами, через которые идет обмен информацией между лицевой панелью и блок-

диаграммой. Данные, которые Вы вводите в элементы управления лицевой панели (**a** и **b** на Figure 2-2) поступают на блок-диаграмму через терминалы элементов управления. Данные затем поступают на функции Add (Сложение) и Subtract (Вычитание). Когда функции Add и Subtract завершат свои внутренние вычисления, на их выходах будут выработаны новые значения. Данные поступают на индикаторные терминалы, где они покидают блок-диаграмму, поступают на лицевую панель и появляются на индикаторах лицевой панели (**a+b** и **a-b** на Figure 2-2).

## Узлы

Узлы это объекты на блок-диаграмме, которые имеют входы и/или выходы и выполняют операции, когда ВП запускается. Они аналогичны командам, операторам, функциям и подпрограммам текстовых языков программирования. Функции Add и Subtract на Figure 2-2 являются узлами. Более подробно об узлах см. в разделе *Узлы блок-диаграммы* в Главе 5 *Построение блок-диаграммы*.

## Проводники

Данные распространяются по блок-диаграмме посредством проводников. На Figure 2-2 проводники соединяют терминалы элементов управления и индикаторов с функциями Add и Subtract. Каждый проводник имеет единственный источник данных, но его можно соединять с многими ВП и функциями, которые читают эти данные. Проводники имеют разные цвета, стили и толщину в зависимости от их типа данных. Поврежденные проводники имеют вид черных пунктирных линий с крестиком  красного цвета посередине. Более подробно о проводниках см. в разделе *Проводники для соединения объектов блок-диаграммы* в Главе 5 *Построение блок-диаграммы*.

## Структуры

Структуры это графическое представление операторов цикла и операторов выбора текстовых языков программирования. Используйте структуры на блок-диаграмме для повторения блоков кода и выполнения кода условно или в специфическом порядке. Более подробно о структурах и примеры с ними см. в Главе 8 *Циклы и структуры*.

## Иконка и соединительная панель

---



После того, как Вы построите лицевую панель ВП и его блок-диаграмму, нужно создать иконку и соединительную панель, после чего этот ВП можно будет использовать как ВПП. У каждого ВП в правом верхнем углу окна лицевой панели и блок-диаграммы есть иконка, в том виде, как она показана слева. Иконка это графическое представление ВП. Она может содержать текст, картинку или их комбинацию. При использовании ВП в качестве ВПП иконка идентифицирует ВПП на блок-диаграмме другого ВП. Для настройки и редактирования иконки на ней нужно сделать двойной клик (двойной щелчок левой кнопкой мыши). Более подробно об иконке описано в разделе *Создание иконки* в Главе 7 *Создание ВП и ВПП*.



Кроме того, для использования ВП в качестве ВПП, Вам нужно создать соединительную панель, показанную слева. Соединительная панель представляет собой набор терминалов (выводов), которые соответствуют элементам управления и индикаторам такого ВП, подобно списку аргументов функции или процедуры в текстовых языках программирования. Соединительная панель определяет входы и выходы, которые могут подключаться к ВП, когда Вы используете его в качестве ВПП. Соединительная панель принимает данные от входных терминалов и передает их в блок-диаграмму через элементы управления лицевой панели, а затем принимает результаты на свои выходные терминалы от индикаторов лицевой панели.

При первом вызове соединительной панели вы увидите шаблон соединений (pattern), содержащий один терминал. При желании Вы можете выбрать любой шаблон соединений. Обычно соединительная панель имеет по одному терминалу на каждый элемент управления или индикатор лицевой панели. Можно назначить до 28 терминалов на соединительной панели. Если Вы предполагаете корректировать ВП так, что потребуется новый вход или выход, зарезервируйте дополнительные незадействованные терминалы. Более подробно о настройке соединительной панели описано в разделе *Настройка соединительной панели* в Главе 7 *Создание ВП и ВПП*.



**Примечание.** Лучше не назначать больше 16 терминалов для ВП. Большое число терминалов ухудшает внешний вид и затрудняет использование таких ВП.

## Использование и настройка ВП и ВПП

---

После того, как Вы построите ВП и создадите его иконку и соединительную панель, Вы можете использовать его в качестве ВПП. Более подробно об этом написано в разделе *ВПП* в Главе 7 *Создание ВП и ВПП*.

Вы можете сохранить ВП в отдельном файле, либо сгруппировать несколько ВП вместе и сохранить их потом в виде библиотеки ВП. Более подробно о сохранении ВП в библиотеках см. в разделе *Сохранение ВП* в Главе 7 *Создание ВП и ВПП*.

Вы можете настроить внешний вид и поведение ВП. Вы можете также создать пользовательское меню для каждого ВП, который Вы создаете, а затем конфигурировать ВП так, чтобы показывать или скрывать это меню. Более подробно о настройке ВП см. в Главе 16 *Настройка ВП*.

### **3. Среда LabVIEW**

---

Для построения лицевых панелей и блок-диаграмм ВП используйте палитры, инструменты и меню среды LabVIEW. Вы можете настраивать палитры **Controls** (Элементы управления) и **Functions** (Функции), а также устанавливать опции рабочей среды.

---

#### **Более подробно...**

Для получения более подробной информации об использовании палитр, меню, панелей инструментов и о настройке рабочей среды см. встроенную систему справки *LabVIEW Help*.

---

#### **Палитра Controls (Элементы управления)**

---

Палитра **Controls** доступна только с лицевой панели. Палитра **Controls** содержит элементы управления (controls) и индикаторы (indicators) которые Вы можете использовать при создании лицевой панели. Элементы управления и индикаторы размещены в подпалитрах в соответствии с их типами. Более подробно о типах элементов управления и индикаторов см. в разделе *Элементы управления и индикации лицевой панели* в Главе 4 *Построение лицевой панели*. Какие элементы управления и индикаторы размещены на палитре **Controls** и их внешний вид, зависит от текущей настройки внешнего вида палитры. Более подробно о внешнем виде палитры см. раздел *Создание и редактирование внешнего вида пользовательской палитры* в этой Главе.

Для отображения палитры **Controls** выберите **Window»Show Controls Palette** или нажмите правую кнопку мыши на рабочем пространстве лицевой панели. Палитру **Controls** можно переместить в любое место на экране. LabVIEW запоминает положение и размер палитры **Controls**, поэтому после следующего запуска LabVIEW палитра появится в том же месте и сохранит свои прежние размеры.

Вы можете изменять способ представления палитры **Controls**. Более подробно о настройках палитры **Controls** см. в разделе *Настройка палитр Controls и Functions* в этой Главе.

## Палитра Functions (Функции)

---

Палитра **Functions** доступна только на блок-диаграмме. Палитра **Functions** содержит ВП и функции, которые Вы можете использовать для построения блок-диаграммы. ВП и функции размещены на подпалитрах в соответствии с типами ВП и функций. Более подробно о типах ВП и функций см. в разделе *Обзор функций* в Главе 5 *Построение блок-диаграммы*. Какие ВП и функции размещены на палитре **Functions** и их внешний вид зависит от текущих настроек внешнего вида палитры. Более подробно о внешнем виде палитры см. раздел *Создание и редактирование внешнего вида пользовательской палитры* в этой Главе.

Для отображения палитры **Functions** выберите **Window»Show Functions Palette** или щелкните правой кнопкой мыши на рабочем пространстве блок-диаграммы. Палитру **Functions** можно переместить в любое место на экране. LabVIEW запоминает положение и размер палитры **Functions**, поэтому после следующего запуска LabVIEW палитра появится в том же месте и сохранит свои прежние размеры.

Вы можете изменять способ представления палитры **Functions**. Более подробно о настройках палитры **Functions** см. в разделе *Настройка палитр Controls и Functions* в этой Главе.

## Навигация по палитрам Controls и Functions

---

Когда Вы щелкаете по иконке подпалитры, вся палитра заменяется на подпалитру, которую Вы выбрали. Щелкните по объекту на палитре, чтобы захватить его курсором, после чего Вы можете поместить его на лицевую панель или блок-диаграмму. Вы можете также нажать правую кнопку мыши на иконке ВП на палитре и выбрать из контекстного меню **Open VI (Открыть ВП)**, чтобы открыть ВП.

Для навигации по палитрам, для настройки палитр и быстрого поиска элементов управления, ВП и функций используйте следующие кнопки на панели инструментов палитр **Controls** и **Functions**:



- **Up** - Перемещение вверх на один уровень иерархии палитр. Щелкнув по этой кнопке, удержите клавишу мыши нажатой, и Вы увидите список подпалитр на пути в текущую подпалитру. Выб-

рите из этого списка нужную подпалитру, чтобы быстро перейти в нее.



• **Search** (Поиск) – Переключает палитру в режим поиска, при котором ищется место размещения в палитре элемента управления, ВП или функции по текстовому названию. Когда палитра находится в режиме поиска, нажмите кнопку **Return to Palette**, чтобы выйти из режима поиска и вернуться к палитре.



• **Options** – отображает раздел **Controls/Functions Palettes** в диалоговом окне **Options**, с помощью которого можно выбрать вид и формат представления палитр.



• **Restore palette Size** (Восстановить размер палитры) – Изменяет размеры палитры к установкам по умолчанию. Эта кнопка появляется только после того, как Вы измените первоначальные размеры палитры.

## Палитра инструментов

---

Палитра инструментов (**Tools Palette**) доступна и на лицевой панели и на блок-диаграмме. Инструмент – это специальный режим действий на курсоре мыши. При этом вид курсора соответствует иконке выбранного из палитры инструмента. Используйте инструменты для работы с объектами лицевой панели и блок-диаграммы.

Для отображения палитры **Tools** выберите **Window»Show Tools Palette**. Палитру **Tools** можно поместить в любое место на экране. LabVIEW запоминает положение палитры **Tools**, поэтому после повторного запуска LabVIEW, палитра появится на том же месте.



**Совет.** Для отображения временной копии палитры **Tools** в точке фокуса курсора нажмите клавишу **<shift>** и правую кнопку мыши.



Если включен режим автоматического выбора инструмента, и Вы перемещаете курсор над объектами лицевой панели или блок-диаграммы, LabVIEW автоматически выбирает наиболее подходящий инструмент из палитры **Tools**. Вы можете отключить автоматический выбор инструмента, щелкнув по кнопке **Automation Tool Selection** на панели **Tools**, показанной слева. Нажмите клавиши **<Shift-Tab>** или щелкните кнопку, чтобы снова включить автоматический выбор инструмента. Вы можете также вручную отключить автоматический выбор инструмента.

мента, выбирая конкретный инструмент из палитры **Tools**. Чтобы вновь включить автоматический выбор инструментов нажмите клавишу **<Tab>** или щелкните кнопку **Automation Tool Selection** на палитре **Tools**.

## Меню и панель инструментов

---

Используйте пункты меню и элементы инструментальной панели для работы с объектами лицевой панели и блок-диаграммы. Используйте кнопки инструментальной панели для запуска ВП.

### Меню

Меню в верхней части окна ВП содержат общие с другими приложениями пункты, такие как **Open**, **Save**, **Copy** и **Paste**, а также другие пункты, специфичные для LabVIEW. Некоторые пункты меню имеют комбинации клавиш для быстрого запуска.

**(Mac OS)** Меню появляются в верхней части экрана.

**(Windows и UNIX)** По умолчанию в меню отображаются только часто используемые пункты меню. Щелкните стрелки внизу меню, чтобы отобразились все пункты меню. Вы можете сделать, чтобы по умолчанию отображались все пункты меню, для этого выберите **Tools»Options** и затем **Miscellaneous** из спадающего меню.



**Примечание.** Некоторые пункты меню недоступны в режиме запуска ВП.

### Контекстные меню (Shortcut menu)

Чаще всего используются контекстные меню объектов. Все объекты LabVIEW и свободное пространство на лицевой панели и блок-диаграмме имеют ассоциированные с ними контекстные меню. Используйте контекстные меню для изменения внешнего вида и поведения объектов лицевой панели и блок-диаграммы. Для вызова контекстного меню щелкните правой кнопкой мыши на объекте лицевой панели или блок-диаграммы.

### Контекстные меню в режиме запуска

Когда ВП запущен, или находится в режиме запуска, элементы лицевой панели имеют сокращенный набор пунктов в контекстном

меню. Используйте сокращенный набор пунктов контекстного меню для вырезания, копирования или вставки содержимого объектов, для установки объектов в их значения по умолчанию, или для чтения описаний объектов.

Некоторые из более сложных объектов имеют дополнительные опции. Например, контекстное меню массива включает пункты для копирования диапазона значений или перехода к последнему элементу массива.

### Панель инструментов (Toolbar)

Используйте панель инструментов, чтобы запустить и редактировать ВП. Когда Вы запускаете ВП, остаются видимыми только инструменты для отладки ВП.

### Окно контекстной помощи (Context Help Window)

---

Окно **Context Help** отображает основную информацию об объектах LabVIEW, когда Вы перемещаете курсор над каждым объектом. Контекстную справочную информацию содержат такие объекты, как ВП, функции, константы, структуры, палитры, свойства, методы, события и компоненты диалоговых окон. Вы можете также использовать **Context Help** для уточнения того, как подключать проводники к ВП и функциям. Более подробно об использовании окна **Context Help** при соединении объектов см. в разделе *Соединение объектов вручную* в Главе 5 *Построение блок-диаграммы*.



Для отображения окна **Context Help** выберите **Help»Show Context Help**. Вы также можете отобразить окно **Context Help**, щелкнув на панели инструментов кнопку **Show Context Help Window**, которая показана слева, или нажимая клавиши **<Ctrl-H>**. (**Mac OS**) Нажмите клавиши **<Command-H>**. (**UNIX**) Нажмите клавиши **<Alt-H>**.

Окно **Context Help** можно размещать в любом месте экрана. Оно меняет свои размеры, чтобы приспособится к конкретному объекту описания. Вы также можете развернуть окно **Context Help** до максимальных размеров. LabVIEW сохраняет положение и размеры окна **Context Help** таким образом, что после повторного запуска LabVIEW окно будет занимать то же положение и будет иметь тот же максимальный размер.



Вы можете заблокировать содержимое окна **Context Help** таким образом, что оно не будет изменяться при перемещении курсора над различными объектами. Для блокировки или разблокировки текущего содержимого окна **Context Help** выберите **Help»Lock Context Help**. Вы можете также блокировать или разблокировать содержимое окна, щелкнув в окне **Context Help** кнопку **Lock**, которая показана слева, или нажимая клавиши <Ctrl-Shift-L>.



Чтобы отобразить необязательные терминалы соединительной панели и полный путь (в файловой системе) к ВП щелкните в окне **Context Help** кнопку **Show Optional Terminals and Full Path**, показанную слева. Более подробно о необязательных терминалах см. в разделе *Установка обязательных, рекомендуемых и необязательных входов и выходов* в Главе 7 *Создание ВП и ВПП*.



Если для объекта, описываемого в окне **Context Help**, существует соответствующий раздел справки в LabVIEW Help, то в окне **Context Help** появится ссылка синего цвета [Click here for more help](#) (Щелкните здесь для более подробной справки). Кроме того, в окне **Context Help** доступна кнопка **More Help**, показанная слева. Для получения дополнительной информации об объекте щелкните эту ссылку или кнопку.

Сведения о создании описаний, которые отображаются затем в окне **Context Help** приведены в разделе *Создание описаний ВП и объектов* в Главе 15 *Документирование и печать ВП*.

## Настройка Вашего рабочего окружения

---

Чтобы изменить вид палитр **Controls** и **Functions** и установить опции рабочей среды, Вы можете использовать диалоговое окно **Options**.

### Настройка палитр **Controls** и **Functions**

Вы можете настраивать палитры **Controls** и **Functions** следующими способами:

- Добавлять ВП и элементы управления на палитры.
- Устанавливать различный внешний вид для различных пользователей, скрывать некоторые ВП и функции, чтобы облегчить ис-

пользование LabVIEW для одного пользователя, обеспечивая при этом полные палитры для другого пользователя.

- Переупорядочивать встроенные палитры с тем, чтобы сделать часто используемые ВП и функции более доступными.
- Преобразовывать набор элементов управления Active X в пользовательские элементы управления с добавлением их в палитры.
- Добавлять подпалитры в палитры.



**Предупреждение.** Не сохраняйте свои собственные ВП и элементы управления в директорию `vi.lib`, поскольку LabVIEW перезаписывает эти файлы, когда вы обновляете или переинсталлируете систему. Сохраняйте ваши ВП и элементы управления в директории `user.lib` для последующего добавления их в палитры **Controls** и **Functions**.

#### **Добавление ВП и элементов управления в пользовательскую подпалитру (User Subpalette) и в подпалитру приборных драйверов (Instrument Drivers Subpalette)**

Простейший метод добавления ВП и элементов управления в палитры **Functions** и **Controls** заключается в сохранении их в директории `labview\user.lib`. Когда вы перезапустите LabVIEW, палитры **User Libraries** и **User Control** будут содержать подпалитры для каждой директории в библиотеке ВП (`.lib`), или файламеню (`.mnu`) в `labview\user.lib` и иконки для каждого файла в `labview\user.lib`. После того как Вы добавите или удалите файлы из этих специальных директорий, LabVIEW будет автоматически обновлять палитры при последующем запуске.

Палитре **Instrument Drivers** соответствует директория `labview\instr.lib`. Для добавления новых приборных драйверов к палитре **Functions** сохраняйте их в этой директории.

Когда Вы добавляете ВП или элементы управления на палитры **Functions** и **Controls**, используя этот метод, Вы не можете назначить имя каждой подпалитре или точное место размещения ВП или элемента управления на палитре.

## **Создание и редактирование внешнего вида пользовательской палитры (Custom Palette)**

Чтобы управлять именем каждой подпалитры и точным расположением добавляемых элементов управления и ВП на палитрах **Functions** и **Controls**, Вы должны создать внешний вид пользовательской палитры. LabVIEW включает два встроенных вида палитр – Express и Advanced. Для создания или редактирования внешнего вида пользовательской палитры выберите **Tools»Advanced»Edit Palette Views**.



**Примечание.** Внешний вид встроенной палитры редактировать нельзя

Информацию о палитрах **Functions** и **Controls** LabVIEW сохраняет в директории `labview\menus`. Директория `menus` содержит поддиректории, соответствующие каждому виду, который Вы создаете или инсталлируете. Если Вы запускаете LabVIEW в локальной сессии, Вы можете определять индивидуальные директории `menus` для каждого пользователя, что позволяет легко подстраивать внешний вид под каждого пользователя.

Когда Вы создаете новый внешний вид палитры, LabVIEW использует копию оригинального встроенного внешнего вида, к которой Вы можете применять любые изменения. LabVIEW создает копию оригинальной встроенной палитры, размещенной в директории `labview\menus`, перед тем, как Вы сделаете любые изменения. Такая защита встроенных палитр позволяет Вам экспериментировать с палитрами без опасности разрушить их исходный внешний вид.

### **Как LabVIEW сохраняет внешний вид**

Файлы `.mni` и `.lib` содержат по одной палитре **Controls** и **Functions** каждый. Кроме того, каждый файл содержит иконку для палитр **Controls** и **Functions**. Каждую подпалитру, которую Вы создаете, нужно сохранять в отдельном `.mni`-файле.

Когда Вы выбираете внешний вид, LabVIEW проверяет директорию `menus` на наличие поддиректории, соответствующей этому внешнему виду. Затем строятся палитры верхнего уровня **Controls** и **Functions** и подпалитры из файла `root.mni` в той директории,

которую LabVIEW автоматически создает всякий раз, когда Вы создаете внешний вид.

Для каждого ВП или элемента управления LabVIEW создает на палитре иконку. Для каждой поддиректории,.mnu-файла или .lib-файла LabVIEW создает подпалитру на палитре.

### **Построение подпалитр ActiveX**

Если Вы используете на лицевой панели элементы управления ActiveX, то для преобразования набора элементов управления ActiveX в пользовательские элементы управления и последующего добавления их на палитру **Controls**, выберите **Tools»Advanced»Import ActiveX Controls**. LabVIEW по умолчанию сохранит эти элементы управления в директории `user.lib`, поскольку все файлы и поддиректории в `user.lib` автоматически появляются на палитрах.

### **Представление наборов инструментов (Toolsets) и модулей (Modules) на палитрах**

Наборы инструментов и модулей с элементами управления или ВП в директории `vi.lib\addons` появляются на палитрах **Controls** и **Functions** после того, как Вы перезапустите LabVIEW. Если используется встроенный внешний вид палитр Express, то наборы инструментов и модули инсталлируют подпалитры на подпалитрах **All Controls** и **All Functions**. Если же используется встроенный внешний вид палитр Advanced, то наборы инструментов и модули инсталлируют подпалитры на палитрах верхнего уровня **Controls** и **Functions**.

Если Вы инсталлировали наборы инструментов или модули и ВП из директорий, отличных от `vi.lib\addons`, то для добавления элементов управления и ВП на палитры, Вы можете предварительно переместить их в директорию `vi.lib\addons`.

### **Установка опций рабочей среды**

Для настройки среды LabVIEW, выберите **Tools»Options**. Чтобы установить опции для лицевой панели, блок-диаграммы, путей к директориям, рабочих качеств и дисков, выравнивающей сетки, палитр, отката, инструментов отладки, цветов, шрифтов, печати, окна

**History** и других возможностей LabVIEW, пользуйтесь диалоговым окном **Options**.

Для выбора нужной категории опций из общего списка используйте выпадающее меню на диалоговом окне **Options**.

### **Как LabVIEW сохраняет опции**

Вам нет необходимости редактировать опции вручную или знать их точный формат, поскольку диалоговое окно **Options** делает это за Вас. LabVIEW сохраняет опции по-разному для каждой платформы.

#### **Windows**

LabVIEW сохраняет опции в файле `labview.ini` в директории LabVIEW. Формат файла такой же, как и у других файлов `.ini`. Он начинается с маркера секции LabVIEW, после чего идет имя опции и ее значение. Например, `offscreenUpdates=True`.

Если Вы хотите использовать разные файлы с опциями, укажите этот файл в командной строке, с помощью которой Вы запускаете LabVIEW. Например, чтобы на вашем компьютере использовать файл с именем `lvrc` вместо `labview.ini`, щелкните правой кнопкой мыши иконку на рабочем столе и выберите **Properties** (Свойства). Щелкните кнопку **Shortcut** (Ярлык) и введите `labview -pref lvrc` в текстовое окошко командной строки.

#### **Mac OS**

LabVIEW сохраняет опции в текстовом файле `LabVIEW Preferences` в папке **System»Preferences**. Если Вы хотите использовать различные файлы с опциями, то скопируйте файл `LabVIEW Preferences` в папку **LabVIEW** и произведите изменения опций в диалоговом окне **Options**. При запуске LabVIEW вначале ищет файл с опциями в папке **LabVIEW**. Если он там не обнаружится, просматривается папка **System**. Если и в этой папке такой файл отсутствует, то создается новый файл с опциями в папке **System**. LabVIEW записывает все изменения, которые Вы сделали в диалоговом окне **Options** в первый из найденных файл `LabVIEW Preferences`.

## UNIX

LabVIEW сохраняет опции в файле `.labviewrc` в вашей домашней директории. Если Вы измените опции в диалоговом окне **Options**, то LabVIEW запишет эти изменения в файл `.labviewrc`. Вы можете создать файл `.labviewrc` в программной директории, чтобы эти изменения действовали для всех пользователей. Например, это может быть удобным для такого параметра, как путь поиска ВП. Используйте файл `.labviewrc` для сохранения опций, которые различны для разных пользователей, например установки шрифтов и цвета, поскольку параметры в файле `.labviewrc` из вашей домашней директории будут перекрывать конфликтные параметры такого же файла, но из программной директории.

Например, если Вы установили файлы LabVIEW в директорию `/opt/labview`, то опции вначале будут читаться из файла `/opt/labview/labviewrc`. Если Вы измените опцию в диалоговом окне **Option**, скажем шрифт приложения, то LabVIEW запишет это изменение в файл `.labviewrc`. Когда Вы в следующий раз запустите LabVIEW, будет использован шрифт приложения из файла `.labviewrc`, вместо шрифта приложения, установленного по умолчанию в файле `/opt/labview/labviewrc`.

Описатели опций состоят из имени опции, отделенного двоеточием, и ее значением. Имя опции содержит исполнительную часть, помещенную после точки. Имена опций чувствительны к регистру. Значение опции можно заключать в одинарные или двойные кавычки. Например, установить по умолчанию удвоенную точность (double), добавьте в файл `.labviewrc` в вашей домашней директории строку

```
labview.defPrecision : double
```

Если Вы хотите использовать различные файлы настроек, укажите имя файла в командную строку запуска LabVIEW. Например, чтобы использовать файл `lvrc` из директории `test` вместо файла `.labviewrc`, наберите в командной строке `labview -pref/test/lvrc`. LabVIEW запишет все изменения, которые Вы сделаете в диалоговом окне **Options**, в файл настроек `lvrc`. Когда Вы файл настроек в командной строке, LabVIEW все же считывает файл `labviewrc` из программной директории, однако настройки

из файла, указанного в командной строке, будут перекрывать конфликтные настройки из файла в программной директории.

## 4. Построение лицевой панели

---

Лицевая панель (front panel) – это пользовательский интерфейс ВП. Обычно Вы проектируете вначале лицевую панель, затем блок-диаграмму, которая выполняет нужные действия с входами и выходами, которые Вы создали на лицевой панели. Более подробно о блок-диаграмме см. в Главе 5 *Построение блок-диаграммы*.

Вы строите лицевую панель с элементами управления и индикаторами, которые являются интерактивными входными и выходными терминалами ВП, соответственно. Элементы управления (controls) – это кнопки, клавиши, переключатели и другие входные устройства. Индикаторы (indicators) – это графопостроители, лампочки и другие элементы отображения. Элементы управления имитируют входные устройства прибора и поставляют данные на блок-диаграмму ВП. Индикаторы имитируют выходные устройства прибора и отображают данные, которые получены в результате обработки или генерирования на блок-диаграмме.

Для отображения палитры **Controls** выберите **Windows»Show Controls Palette**, затем выберите элементы управления и индикаторы из палитры **Controls** и поместите их на лицевую панель.

---

**Более подробно...**

Более подробно о проектировании и конфигурировании лицевой панели см. в *LabVIEW Help*

---

### Конфигурирование объектов лицевой панели

---

Используйте диалоговые окна свойств или контекстное меню для конфигурирования внешнего вида и поведения элементов управления и индикаторов на лицевой панели. Используйте диалоговые окна в тех случаях, когда Вы хотите конфигурировать элементы управления или индикаторы лицевой панели через диалоговое окно, которое включает контекстную справку, или когда Вы хотите установить сразу несколько свойств объекта. Контекстные меню используйте для быстрого конфигурирования общих свойств элементов управления и индикаторов. Набор опций в диалоговом окне свойств и в контекстном меню разный для разных объектов лицевой панели. Любая опция, которую Вы можете установить с помощью контекстного меню, подавляет ту же опцию, установленную

через диалоговое окно свойств. Более подробная информация о создании и использовании пользовательских элементов управления, индикаторах и определениях типов содержится в Примечаниях к приложению (Application Note) *LabVIEW Custom Controls, Indicators, and Type Definitions*.

Для вызова диалогового окна свойств элемента управления или индикатора на лицевой панели щелкните по нему правой кнопкой мыши и выберите **Properties** (Свойства) из контекстного меню. Во время работы ВП диалоговое окно свойств недоступно.

## **Включение и выключение видимости необязательных элементов**

Элементы управления и индикаторы лицевой панели имеют необязательные элементы, которые могут быть видимыми, либо скрытыми. Установка видимости элементов для объектов лицевой панели осуществляется на вкладке **Appearance** (Внешний вид) диалогового окна свойств. Вы можете также устанавливать видимость элементов, щелкнув правой кнопкой мыши по объекту и выбрав из контекстного меню **Visible Items** (Видимые пункты). Большинство объектов имеют метки (label) и заголовки (caption). Более подробно о метках и заголовках см. в разделе *Использование текстовых меток* в настоящей Главе.

## **Преобразование элементов управления в индикаторы и наоборот**

Изначально LabVIEW конфигурирует объекты из палитры **Controls** как элементы управления либо как индикаторы, основываясь на их типичном использовании. Например, если Вы выберите переключатель (toggle switch), то он появится на лицевой панели как элемент управления, поскольку переключатель – это обычно входное устройство. Если Вы выберите светодиод (LED), то он появится на лицевой панели как индикатор, поскольку обычно светодиод – это выходное устройство.

Некоторые палитры содержат элементы управления и индикаторы для объектов одного и того же типа или класса. Например, палитра **Numeric** (Числовые) содержит цифровой элемент управления и цифровой индикатор.

Вы можете преобразовать элемент управления в индикатор, щелкнув правой кнопкой мыши по объекту и выбрав из контекстного меню **Changing to Indicator** (Преобразовать в индикатор). Аналогично

Вы можете преобразовать индикатор в элемент управления, щелкнув правой кнопкой мыши по объекту и выбирая из контекстного меню **Changing to Control** (Преобразовать в элемент управления).

## Замена объектов лицевой панели

Вы можете заменять один объект лицевой панели другим элементом управления или индикатором. Когда Вы щелкаете правой кнопкой по объекту и выбираете **Replace** (Заменить) из контекстного меню, появляется временная палитра **Controls**, даже если обычной палитры **Controls** уже открыта. Выберите элемент управления или индикатор из временной палитры **Controls**, чтобы заменить им текущий объект на лицевой панели.

Использование **Replace** из контекстного меню сохраняет, насколько это возможно, информацию об исходном объекте, такую, как его имя, описание, значение по умолчанию, направление потока данных (элемент управления или индикатор), цвет, размер и т.п. Однако, новый объект сохраняет свой собственный тип данных. Проводники от терминалов объекта или локальных переменных остаются на блок-диаграмме, но они могут быть поврежденными (broken). Например, если Вы заменяете числовой терминал строковым терминалом, то исходный проводник остается на блок-диаграмме, но становится поврежденным.

Чем больше новый объект имеет сходства с заменяемым объектом, тем больше исходных характеристик объекта могут сохраниться. Например, если Вы заменяете слайдер ( движок) одного типа на слайдер другого типа, то новый слайдер будет иметь ту же высоту, шкалу, значение, имя, описание и т.п. Если же Вы заменяете слайдер на строковый элемент управления, то LabVIEW сохранит только имя, описание и направление потока данных, поскольку слайдер имеет мало общего со строковым элементом управления.

Вы можете также вставлять (past) объекты из буфера обмена (clipboard) для замены существующих на лицевой панели элементов управления и индикаторов. Такой способ не сохраняет ни одной характеристики о старом объекте, но проводники остаются связанными с этим объектом.

## Конфигурирование лицевой панели

---

Вы можете настраивать лицевую панель, устанавливая порядок перемещения по объектам лицевой панели, используя импортированную графику и устанавливая автоматическое изменение размеров объектов лицевой панели при изменении размеров окна.

### Установка комбинаций клавиш для элементов управления

Вы можете назначить комбинации клавиш для элементов управления таким образом, что пользователь сможет перемещаться по лицевой панели без мыши. Щелкните правой кнопкой элемент управления и выберите **Advanced»Key Navigation** из контекстного меню для отображения диалогового окна **Key Navigation** (Клавиши перемещения).



**Примечание.** LabVIEW не реагирует на комбинации клавиш для скрытых (hidden) элементов управления.

Когда пользователь нажимает комбинации клавиш во время запуска ВП, связанные с ними элементы управления получают фокус. Если элемент управления текстовый или цифровой, то LabVIEW подсвечивает текст таким образом, что Вы можете редактировать его. Если элемент управления является булевым, нажмите клавишу пробела или <Enter> для изменения их значения.

Пункт **Advanced»Key Navigation** из контекстного меню для индикаторов затемнен (неактивен), поскольку вводить данные в индикатор нельзя.



**Примечание.** Вы можете также использовать событие **Key Down** (нажата клавиша), чтобы генерировать событие, когда пользователь нажмет специфическую клавишу на клавиатуре.

Более подробно о назначении комбинаций клавиш в пользовательском интерфейсе см. раздел *Key Navigation* в Главе 6 *LabVIEW Style Guidelines* руководства *LabVIEW Development Guidelines*.

## **Управление поведением кнопок с помощью клавиш перемещения**

Вы можете устанавливать соответствие между функциональными клавишами и различными кнопками, которые управляют лицевой панелью. Вы можете настроить кнопку на ВП таким образом, чтобы она действовала подобно кнопке диалогового окна, то есть, чтобы нажатие клавиши <Enter> было эквивалентно щелчку по кнопке.

Если Вы установите соответствие между клавишей <Enter> и кнопкой диалогового окна, LabVIEW автоматически изобразит вокруг этой кнопки жирную линию.

Если Вы установите соответствие клавиши <Enter> с элементом управления, пользователь не сможет ввести символ перевода строки в любой строковый элемент управления на этой лицевой панели.

Если Вы переместитесь на булев элемент управления и нажмете клавишу <Enter>, то булев элемент управления изменит свое состояние, даже если другие элементы управления используют клавишу <Enter> в качестве управляющей комбинации клавиш. Назначение клавиши <Enter> в качестве управляющей комбинации клавиш будет действовать только в том случае, если ни один из булевых элементов управления на лицевой панели не селектирован.

## **Установка порядка обхода (tabbing order) объектов лицевой панели**

Элементы управления и индикаторы на лицевой панели имеют порядок, называемый порядком обхода, который не имеет отношения к месту расположения их на лицевой панели. Элемент управления или индикатор, который Вы создаете первым на лицевой панели, будет элемент 0, вторым – 1 и т.д. Если Вы удаляете элемент управления или индикатор, порядок обхода автоматически корректируется.

Порядок обхода определяет порядок, в котором LabVIEW селектирует элементы управления или индикаторы, когда пользователь нажимает клавишу <Tab> во время работы (запуска) ВП. Порядок обхода определяет также порядок, в котором элементы управления и индикаторы появляются в записях файлов-протоколов (datalog files), которые создаются, когда Вы протоколируете данные лицевой панели. Более подробно о протоколировании данных см. в раз-

деле *Регистрация данных лицевой панели* в Главе 14 *Файловый ввод/выход*.

Порядок обхода объектов лицевой панели можно установить, выбирая **Edit»Set Tabbing Order**.

Для предотвращения доступа к элементу управления нажатием клавиши <Tab> в процессе работы ВП, поставьте птичку на пункте **Skip this control when tabbing** в диалоговом окне **Key Navigation**.

## Раскрашивание объектов

Вы можете изменять цвета не всех, но многих объектов. Например, терминалы блок-диаграммы и проводники используют специфические цвета для каждого типа и представления данных, поэтому их нельзя изменить.

Для изменения цвета объектов лицевой панели, либо рабочего пространства лицевой панели или блок-диаграммы, используйте инструмент **Coloring** (Раскрашивание), чтобы щелкнуть правой кнопкой на объекте, либо на рабочем пространстве. Вы можете также изменить цвета по умолчанию для большинства объектов, выбирая **Tools»Options** и затем категорию **Colors** из выпадающего меню.

Более подробно об использовании цветов при проектировании пользовательского интерфейса см. в разделе *Colors* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## Использование импортированной графики

Вы можете импортировать графические изображения из других приложений для использования их в качестве фона для лицевой панели, пунктов кольцевых элементов управления и отдельных частей других элементов управления и индикаторов. Более подробно об использовании импортированной графики в элементах управления см. *LabVIEW Custom Controls, Indicators, and Type Definitions Application Note* (Примечания к приложению *Настройка в LabVIEW элементов управления, индикаторов и определений типов*.)

LabVIEW поддерживает большинство стандартных графических форматов, включая, BMP, JPEG, анимационный GIF, MNG, анимационный MNG и PNG. LabVIEW поддерживает также прозрачность.

Для импорта графики, скопируйте ее в буфер обмена и вставьте на лицевую панель. Вы можете также выбрать **Edit>Import from File**.



**Примечание.** Если Вы импортируете изображение путем копирования и вставки, изображение утрачивает прозрачность.

Примеры по элементам управления с импортированной графикой см. в examples\general\controls\custom.llb. Более подробно использование графики при конструировании пользовательского интерфейса описано в разделе *Graphics and Custom Control* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## Выравнивание и распределение объектов

Для включения привязки к сетке объектов при их размещении на лицевой панели выберите **Operate>Enable Alignment Grid on Panel**. Для отключения такой привязки выберите **Operate>Disable Alignment Grid on Panel**. После этого объекты можно разместить в произвольное место, а не только по линиям сетки, хотя сама сетка на лицевой панели остается. Кроме того, включать или выключать выравнивание по сетке можно нажатием клавиш <Ctrl-#>.

**(Mac OS)** Нажмите клавиши <Command-\*>. **(Sun)** Нажмите клавиши <Meta-#>. **(Linux)** Нажмите клавиши <Alt-#>.

Аналогично Вы можете использовать привязку к сетке на блок-диаграмме.

Для отключения сетки или настройки ее параметров выберите из главного меню **Tools>Options** и затем категорию **Alignment Grid**.

Чтобы выровнять объекты после того, как Вы их разместите, селектируйте эти объекты и затем вызовите спадающее меню нажатием кнопки **Align Objects** на панели инструментов. Чтобы равномерно распределить объекты, селектируйте их и затем вызовите спадающее меню нажатием кнопки **Distribute Objects** на панели инструментов.

## Группировка и фиксация объектов

Если Вам нужно сгруппировать и заблокировать вместе несколько объектов, воспользуйтесь инструментом Positioning (позиционирование) для их селектирования. Затем щелкните на панели инструментов кнопку **Reorder** и выберите из спадающего меню **Group** (Сгруппировать) или **Lock** (Заблокировать). Сгруппированные объекты сохраняют свое относительное расположение и размеры, когда с помощью инструмента Positioning перемещаете или изменяете размеры всей группы объектов. Заблокированные объекты сохраняют свое расположение на лицевой панели, и Вы не можете их удалить, пока не разблокируете. Можно одновременно сгруппировать и заблокировать несколько объектов. Все инструменты, кроме Positioning, действуют на сгруппированные или заблокированные объекты обычным образом.

## Изменение размеров объектов

Вы можете изменить размеры большинства объектов лицевой панели. Когда Вы перемещаете инструмент Positioning над объектом, ручками или шкалами, размеры которых могут быть изменены, появляются точки, передвигая которые можно изменять размеры объектов. При изменении размеров объекта, размер шрифта остается неизменным. Изменение размеров группы объектов приводит к изменению размеров всех входящих в группу объектов.

Некоторые объекты позволяют изменять только горизонтальные или вертикальные размеры, например цифровые элементы управления и индикаторы. Другие, такие как кнопки, сохраняют неизменными пропорции размеров. Курсор в режиме Positioning для таких объектов сохраняет свой обычный вид, но пунктирная граница вокруг них изменяется только в одном направлении. При изменении размеров объекта можно вручную ограничить возможные направления изменений. Чтобы ограничиться только вертикальным, только горизонтальным направлениями или для сохранения текущих пропорций объекта, нужно удерживать клавишу **<Shift>** во время щелчка и перетаскивания ручек и шкал. Чтобы изменить размеры объекта относительно центральной точки, нужно удерживать клавишу **<Ctrl>** во время щелчка и перетаскивания ручек и шкал.

**(Mac OS)** Нажмите клавишу **<Option>**. **(Sun)** Нажмите клавишу **<Meta>**. **(Linux)** Нажмите клавишу **<Alt>**.

Чтобы изменить размеры нескольких объектов одинакового размера, селектируйте объекты и выберите спадающее меню **Resize Objects** на панели инструментов. Все селектированные объекты можно изменять по ширине или высоте наибольшего или наименьшего объекта, а также можно устанавливать размеры всех селектированных объектов в пикселях.

## Масштабирование объектов лицевой панели

Вы можете установить объекты лицевой панели масштабируемыми или автоматически изменяющими свои размеры относительно размеров окна, при изменении размеров окна лицевой панели. Можно сделать масштабируемым один объект лицевой панели или все объекты сразу. Однако, нельзя установить масштабирование для нескольких объектов на лицевой панели, если не установить их все масштабируемыми, или предварительно не сгруппировать. Чтобы установить объект масштабируемым, селектируйте объект и выберите **Edit>Scale Object with Panel**.

Если Вы установите единственный объект лицевой панели масштабируемым, этот объект будет автоматически изменять свои размеры в соответствии с любыми изменениями размеров окна лицевой панели. Остальные объекты лицевой панели изменяют свое положение таким образом, чтобы оставаться в соответствии со своим предшествующим положением на лицевой панели, но они не масштабируются соответственно новому размеру окна лицевой панели.

Как только Вы назначите одиночный объект масштабируемым, на лицевой панели вокруг него появятся серые линии, ограничивающие определенные области лицевой панели, как показано на Figure 4-1. Эти области определяют расположение других объектов лицевой панели по отношению к объекту, который установлен масштабируемым. Если Вы измените размер окна лицевой панели, то масштабируемый объект изменит свои размеры и положение относительно своего первоначального положения. Серые ограничительные линии исчезнут, если ВП запущен.

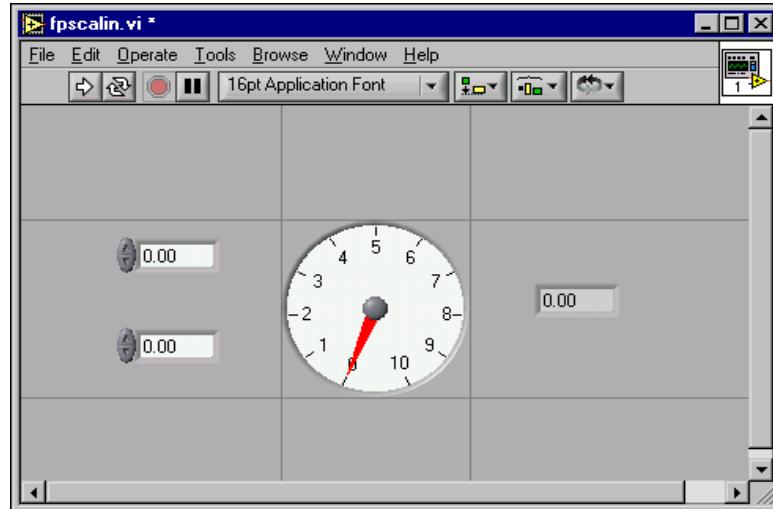


Figure 4-1. Лицевая панель с масштабируемым объектом

Когда LabVIEW масштабируются автоматически, остаются в силе те же соглашения, что и при изменении размеров объектов вручную. Например, некоторые объекты могут изменяться только по горизонтали или по вертикали, размеры шрифтов остаются без изменений.

После того, как LabVIEW сделает автоматическое перемасштабирование, при возврате к прежнему размеру окна объект может не вернуться в точности к прежним размерам и положению. Перед сохранением ВП выберите **Edit»Undo**, чтобы восстановить исходные размеры окна лицевой панели и объектов.

Вы можете установить масштабируемым массив или объект внутри массива. Если Вы установили массив масштабируемым, то нужно регулировать количество строк и столбцов, которые будут видимыми. Если Вы установили масштабируемыми объекты внутри массива, то количество видимых строк и столбцов внутри массива будет всегда одинаковым, независимо от размеров окна.

Аналогично можно установить масштабируемыми кластер или объекты внутри кластера. Если Вы установите объекты внутри кластера масштабируемыми, то кластер регулируется аналогичным образом.

## Добавление свободного пространства на лицевой панели без изменения размеров окна

Вы можете добавить свободное пространство на лицевой панели без изменения размеров окна. Для увеличения пространства между тесно расположеными объектами, нажмите клавишу **<Ctrl>** и с помощью инструмента **Positioning** щелкните по свободному пространству лицевой панели. Удерживая клавишу **<Ctrl>** нажатой, растяните выделяемую область до размеров, которые Вы хотите добавить.

**(Mac OS)** Нажмите клавишу **<Option>**. **(Sun)** Нажмите клавишу **<Meta>**. **(Linux)** Нажмите клавишу **<Alt>**.

Прямоугольник, помеченный пунктирной линией, определяет пространство, которое будет вставлено, когда Вы отпустите кнопку мыши.

## Элементы управления и индикаторы лицевой панели

---

Для построения лицевой панели используйте элементы управления и индикаторы, размещенные на палитре **Controls**. Элементами управления служат кнопки, клавиши, переключатели и другие входные устройства. Индикаторами являются графопостроители, светодиоды и другие устройства отображения. Элементы управления имитируют входные устройства и поставляют данные на входы блок-диаграммы ВП. Индикаторы имитируют выходные устройства и отображают данные, обработанные или сгенерированные блок-диаграммой.

### Трехмерные (3D) и классические элементы управления и индикаторы

Многие объекты лицевой панели выполнены в трехмерном изображении с высоким разрешением цветов (high-color). Для оптимального отображения таких объектов требуется установить монитор в режим отображения цветов, по крайней мере, 16-бит.

Трехмерные объекты лицевой панели имеют соответствующие им двумерные объекты с пониженным режимом отображения цветов. Для создания ВП для с 256 - или 16-цветных мониторов, используйте двумерные (2D) объекты, размещенные на палитре **Classic control**.

Выберите **File»VI Properties** и затем **Editor Options** из спадающего меню **Category**, чтобы изменить стиль элементов управления и индикаторов, которые LabVIEW создает, когда Вы щелкаете правой кнопкой мыши терминал и выбираете **Create»Control** или **Create»Indicator** из контекстного меню. Более подробно о конфигурировании внешнего вида и поведения ВП см. в разделе *Конфигурирование внешнего вида и поведения ВП* в Главе 16 *Конфигурирование ВП*. Выберите **Tools»Options** и затем **Front Panel** из спадающего меню, чтобы изменить стиль элемента управления или индикатора, которые LabVIEW создает в новых ВП при нажатии правой кнопки мыши на терминале и выборе из контекстного меню **Create»Control** или **Create»Indicator**.

## **Движки, кнопки, переключатели, цифровые переключатели и временные метки**

Для имитации на лицевой панели движков (slides), кнопок (knobs), переключателей (dials) и цифровых дисплеев используйте числовые элементы управления и индикаторы, размещенные на палитрах **Numeric** (Числовые) и **Classic Numeric** (Классические числовые). Эти палитры содержат также цветовые диалоговые окна для установки значений цветов и диалоговые окна временных меток для установки значений времени и даты. Используйте цифровые элементы управления и индикаторы для ввода и отображения числовых данных.

### **Движковые (Slider) элементы управления и индикаторы**

Движковые элементы управления и индикаторы включают вертикальные и горизонтальные движки, емкость (tank) и термометр (thermometer). Изменение значения движковых элементов управления и индикаторов осуществляется с помощью инструмента **Operating** (Палец) путем перетаскивания движка в новое положение, либо с помощью необязательного цифрового дисплея. Если Вы перетаскиваете движок в новое положение и ВП при этом запущен, то текущие промежуточные значения передаются в ВП по мере того, как ВП успевает считывать данные с этого элемента управления.

Движковые элементы управления или индикаторы могут отображать больше одного значения. Чтобы добавить добавочные движки, щелкните правой кнопкой мыши по элементу управления и выберите **Add Slider** из контекстного меню. Тип данных, который соответствует многодвижковому элементу управления, – это кластер,

содержащий набор числовых значений. Более подробно о кластерах см. в разделе *Кластеры* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.

### **Вращающиеся (Rotary) элементы управления и индикаторы**

Вращающиеся элементы управления и индикаторы включают ручки (knobs), переключатели (dials), циферблата (gages) и измерители (meters). Изменение значений таких объектов осуществляется щелчками мыши по вращающемуся элементу или с помощью необязательного цифрового дисплея.

Вращающиеся элементы управления или индикаторы могут отображать больше одного значения. Чтобы добавить новые стрелки, щелкните по объекту правой кнопкой мыши и выберите **Add Needle** (Добавить стрелку) из контекстного меню. Типом данных многострелочного элемента управления является кластер, который содержит каждое числовое значение. Более подробно о кластерах см. в разделе *Кластеры* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.

### **Цифровые (Digital) элементы управления и индикаторы**

Цифровые элементы управления являются простейшим способом ввода и отображения числовых данных. Вы можете изменять размеры таких объектов лицевой панели по горизонтали для подгонки их под нужное количество отображаемых цифр. Значение цифрового элемента управления или индикатора можно изменить следующими способами:

- Использовать инструмент *Operating* или *Labeling*, чтобы щелкнуть им внутри окошка цифрового дисплея и ввести числа с клавиатуры.
- Использовать инструмент *Operating*, чтобы щелкнуть им кнопки инкремента (увеличение) или декремента (уменьшение) на цифровом элементе управления.
- Использовать инструмент *Operating* или *Labeling*, чтобы поместить курсор справа от цифры, которую Вы хотите изменить, и нажимать клавиши *Up* или *Down* (стрелки) на клавиатуре.

### Числовое форматирование

По умолчанию LabVIEW отображает и хранит числа подобно калькулятору. Числовой элемент управления или индикатор отображает вплоть до шести цифр, перед тем как автоматически переключиться в показательную форму представления. Вы можете настроить количество цифр, которые LabVIEW отображает до переключения в показательную форму, на вкладке **Format and Precision** диалогового окна **Numeric Properties**, доступ к которому возможен через контекстное меню данного объекта лицевой панели.

Точность, которую Вы таки образом настроите, действует только на отображение значений. Внутренняя точность по-прежнему зависит от представления (representation).

### Элементы управления и индикаторы для временных меток (Time Stamp)

Используйте элементы управления и индикаторы для временных меток, чтобы послать значение даты в блок-диаграмму или принять его оттуда. Вы можете изменить значение временной метки в элементе управления следующими способами:

- Щелкните правой кнопкой константу и выберите пункт **Format & Precision** из контекстного меню.
- Щелкните кнопку **Time/Date Browse**, показанную слева, чтобы отобразить диалоговое окно **Time and Date**.
- Выберите **Data Operations»Set Time and Date** из контекстного меню, чтобы отобразить диалоговое окно **Set Time and Date**. Вы также можете щелкнуть правой кнопкой элемент управления временной меткой и выбрать **Data Operations»Set Time to Now** из контекстного меню.

### Цветовые боксы (Color Boxes)

Цветовой бокс отображает цвет, который соответствует специфической величине. Например, Вы можете использовать цветовые боксы для индикации различных условий, таких как выход величины за диапазон. Значение цвета выражается шестнадцатеричным числом в формате RRGGBB. Первые две цифры управляют значением красного цвета. Вторые две цифры управляют значением зеленого цвета. Последние две цифры управляют значением синего цвета.



Установите цвет цветового бокса, щелкнув инструментом Operating или Coloring, чтобы отобразить цветовую палитру.

### **Цветовой клин (Color Ramp)**

Цветовой клин использует цвет для отображения числовой величины. Вы настраиваете цветовой клин, который содержит, по крайней мере, два цветовых маркера, каждый с числовой величиной и соответствующим отображаемым цветом. Как только входная величина изменится, отображаемый цвет изменится на цвет, соответствующий этой величине. Цветовой клин обычно используется для визуального отображения диапазонов данных, таких как аварийные диапазоны, когда указатель достигает опасных значений. Например, Вы можете использовать цветовой клин, чтобы установить цветовую шкалу для самописцев или графопостроителей интенсивности. Более подробно о самописцах и графопостроителях интенсивности см. в разделе *Индикаторы Intensity Graph и Intensity Chart* в Главе 12 *Графики и диаграммы*.

Щелкните правой кнопкой цветовой клин и используйте пункты контекстного меню для настройки внешнего вида, размеров, цветов и количества цветов.

Вы можете также добавить цветовой клин к любой кнопке (knob), переключателю (dial) или указателю (gauge) на лицевой панели. Измеритель (meter) по умолчанию имеет цветовой клин видимым.

### **Графики и диаграммы (Graphs and Charts)**

Используйте графики и диаграммы для отображения зависимостей данных в графической форме.

Более подробно об использовании графиков и диаграмм в LabVIEW см. в Главе 12 *Графики и диаграммы*.

### **Клавиши, переключатели и лампочки (Buttons, Switches, and Lights)**

Используйте булевые (логические) элементы управления и индикаторы для имитации кнопок, переключателей и лампочек (светодиодов - LED), которые служат для ввода и отображения булевых значений (TRUE – истина, FALSE – ложь). Например, если Вы хотите отслеживать температуру в эксперименте, Вы можете поместить на

лицевую панель предупреждающую лампочку, которая будет указывать, когда температура превысит некоторый уровень.

Для настройки внешнего вида и поведения логических элементов используйте их контекстное меню.

## **Окна текстового ввода (Text Entry Boxes), метки (Labels) и дисплеи путей (Path Displays)**

Используйте строковые (string) и путевые (path) элементы управления и индикаторы для имитации окон текстового ввода и меток и для ввода и получения места размещения файлов или директорий.

### **Строчные элементы управления и индикаторы (String Control and Indicators)**

Ввод или редактирование текста в строковом элементе управления на лицевой панели осуществляется с помощью инструмента Operating (Управление) или Labeling (Ввод текста). По умолчанию новый или измененный текст не проходит на блок-диаграмму, пока Вы не завершите сессию редактирования. Сессия редактирования завершится, если щелкнуть где-нибудь еще на панели, перейти в другое окно, **щелкнуть кнопку Enter на панели инструментов** или нажать клавишу <Enter> на цифровой клавиатуре. Нажатие клавиши <Enter> на основной клавиатуре вводит символ перевода каретки (carriage return).

Более подробно о строковых элементах управления и индикаторах см. в разделе *Строки на лицевой панели* в Главе 10 *Группирование данных с помощью строк, массивов и кластеров*.

### **Элемент управления комбинированный бокс (Combo Box)**

Используйте элемент управления Combo Box (комбинированный бокс) для создания списка строк, которые можно циклически прокручивать на лицевой панели. Комбинированный бокс похож на элементы управления текстовое кольцо (text ring) или кольцо меню (menu ring). Однако значением и типом данных комбинированного бокса являются строки, а не числа, как в кольцевых элементах управления. Более подробно о кольцевых элементах управления см. в разделе *Кольцевые и перечислительные элементы управления и индикаторы* в настоящей Главе.

Элемент управления Combo Box можно использовать для выбора вариантов в структуре Case. Более подробно о структуре Case см. в разделе *Структура Case* в Главе 8 *Циклы и структуры*.

Нажмите правой кнопкой по элементу управления Combo Box и выберите **Edit Items** из контекстного меню для добавления строк в список, из которого можно выбирать в этом элементе управления. Порядок строк на странице **Edit Items** диалогового окна **Combo Box Properties** определяет порядок строк в элементе управления. По умолчанию Combo Box позволяет пользователю вводить строковые значения, которых нет в списке строк, определенных для этого элемента управления. Щелкните правой кнопкой по элементу управления Combo Box и выберите **Allow Undefined String** из контекстного меню, чтобы удалить метку и предотвратить ввод пользователем неопределенного строкового значения в элемент управления.

Как только Вы наберете строку в Combo Box во время исполнения, LabVIEW выберет первую наиболее короткую строку в элементе управления, которая начинается с набранных Вами символов. Если ни одна из строк в списке не совпадает с набранными символами и этот элемент управления не допускает неопределенного строкового значения, то LabVIEW не будет принимать или отображать такие вводимые Вами символы.

Когда Вы конфигурируете список строк в элементе управления Combo Box, Вы можете задать произвольное значение для каждой строки, что является полезным, если Вы хотите, чтобы строка, которая появляется в элементе управления Combo Box на лицевой панели, отличалась от строкового значения, которое выдает терминал Combo Box на блок-диаграмме. Нажмите правую кнопку на элементе управления Combo Box, выберите **Edit Items** из контекстного меню и удалите галочку с опцией **Values Match Labels** (значения совпадают с метками) на вкладке **Edit Items** диалогового окна **Combo Box Properties**. В колонке Values таблицы на этой вкладке измените значение, которое соответствует каждой строке этого элемента управления.

### **Элементы управления и индикаторы для путей к размещению файлов**

Используйте путевые элементы управления и индикаторы для ввода или отображения путей (path) к размещению файлов и директо-

рий. Путевые элементы управления и индикаторы работают подобно строковым элементам управления и индикаторам, с той лишь разницей, что LabVIEW форматирует пути с использованием стандартного синтаксиса для используемой операционной системы.

#### **Неправильные пути**

Если функция, возвращающая путь, завершается с ошибкой, она возвращает в индикатор ошибочное значение (invalid value) **<Not A Path>** (не путь). Используйте значение **<Not A Path>** в качестве значения по умолчанию для путевых элементов управления, чтобы определить ситуацию, когда пользователь отказывается от ввода пути и отображайте файловое диалоговое окно с опциями для выбора пути. Чтобы отобразить файловое диалоговое окно используйте функцию **File Dialog**.

#### **Пустые пути (Empty Paths)**

В случае операционной системы Windows пустой путь в путевом элементе управления появляется в виде пустой строки. Используйте пустые пути для сообщения пользователю о необходимости ввести путь. Когда Вы подсоединяете пустой путь к функции ввода/вывода файлов, пустой путь ссылается на список устройств, отображаемых на данном компьютере.

**(Mac OS)** Пустой путь ссылается на смонтированный том. **(UNIX)** Пустой путь ссылается на корневую директорию.

### **Элементы управления и индикаторы для массивов и кластеров**

Для создания массивов и кластеров из других элементов управления и индикаторов используйте элементы управления и индикаторы для массивов и кластеров, размещенные на палитрах **Array & Cluster** и **Classic Array & Cluster**. Более подробно о массивах и кластерах см. в разделе *Группировка данных в массивы и кластеры* Главы 10 *Группировка данных с помощью строк, массивов и кластеров*.

Палитры **Array & Cluster** содержат также элемент управления и индикатор для стандартного кластера ошибок (error cluster) и вариантный (variant) элемент управления. Подробную информацию о кластерах ошибок см. в разделе *Кластеры ошибок* в Главе 6 *Запуск и отладка ВП*. Подробную информацию о вариантном элементе

управления см. в разделе *Обработка вариантовых данных* в Главе 5 *Построение блок-диаграммы*.

## Списки, деревья и таблицы

Чтобы предоставить пользователю список пунктов, из которых можно выбирать, воспользуйтесь элементом управления listbox (списковое окно) из палитр **List & Table** и **Classic List & Table**.

### Списковое окно

Вы можете настроить списковые окна (Listboxes) так, чтобы они допускали однократный или множественный выбор. Для отображения расширенных данных о каждом пункте, таких как размер пункта и дата его создания, используйте многостолбцовое списковое окно (Multicolumn Listbox).

Когда Вы вводите символы в списковые окна во время исполнения, LabVIEW выбирает первый пункт из оконного списка, который начинается на введенный символ. Используйте клавиши «стрелка влево» и «стрелка вправо» для перехода на предыдущий или следующий пункт, который совпадает с введенными символами.

Вы можете ввести символ около пункта списка, подобно тому, как это сделано в диалоговом окне **VI Library Manager**, где директории и файлы помечены разными символами. Вы можете также вставить разделительные строки между пунктами списков.

Вы можете использовать узел свойств (Property Node) для модификации пунктов списка и сбора информации о пунктах, например, такой как определение текущего выбранного пункта или определение пункта, по которому сделан двойной щелчок (double click), если таковой имеется. Более подробно об использовании узла свойств см. в разделе *Узлы свойств* в Главе 17 *Программное управление ВП*.

### Элементы управления деревьями (Tree Controls)

Используйте элементы управления деревьями, чтобы выдать пользователю иерархический список пунктов, из которых можно выбирать. Организуйте вводимые в элемент управления пункты в группы или узлы (nodes). Щелкните по символу распахивания около узла, чтобы распахнуть его и отобразить все пункты данного узла. Вы

можете также щелкнуть по символу около узла для его сворачивания.



**Примечание.** Вы можете создать и редактировать элементы Tree Controls только в LabVIEW Full and Professional Development System. Если ВП содержит Tree Control, то Вы можете запускать ВП во всех вариантах поставки, но Вы не сможете конфигурировать такие элементы в Base Package.

Когда Вы вводите символы в элемент Tree Control во время исполнения, LabVIEW выбирает первый пункт, который начинается с введенного Вами символа. Вы можете изменять иерархию пунктов выбирая некоторый пункт и нажимая клавишу «точка» (.) для отступа вправо выбранного пункта или «запятая» (,) для перемещения его влево.

Настройка пунктов в элементе Tree Control делается также, как в элементе Listbox. Вы также можете изменить тип символа, который появляется возле каждого узла и указать, может ли пользователь перетаскивать (drag) пункты.

Вы можете использовать узел вызова (Invoke Node) для модификации пунктов в элементе Tree Control и получения сведений, таких как определение того, на каком пункте был сделан двойной щелчок, если таковой имеется. Когда Вы добавляете пункт в элементе Tree Control, LabVIEW создает уникальную метку (tag) для этого пункта. Используйте эту метку для программной модификации пунктов или для получения информации о пункте. Щелкните правой кнопкой элемент Tree Control и выберите **Edit Items** из контекстного меню для модификации меток, которые LabVIEW создает для каждого пункта. Более подробно об узлах вызова см. в разделе *Узлы вызова* Главы 17 *Программное управление ВП*.

Пример использования элемента Tree Control см. в виртуальном приборе Directory Hierarchy in Tree Control в библиотеке examples\general\controls\Directory Tree Control.llb.

### Таблицы

Для создания на лицевой панели таблиц используйте табличные элементы управления, которые находятся на палитрах **List & Table** и **Classic List & Table**.

Более подробно об использовании табличных элементов управления см. разделе *Таблицы* Главы 10 *Группировка данных с использованием строк, массивов и кластеров*.

## Кольцевые и перечислительные элементы управления и индикаторы

Для создания циклически прокручиваемых списков строк используйте кольцевые и перечислительные элементы управления и индикаторы, размещенные на палитрах **Ring & Enum** и **Classic Ring & Enum**.

### Кольцевые элементы управления (Ring Controls)

Кольцевые элементы управления это объекты, которые связывают численные значения со строками или картинками. Кольцевые элементы управления имеют вид спадающих меню, которые пользователи могут просматривать циклически для выбора нужного пункта.

Кольцевые элементы управления служат обычно для выбора взаимно исключающих пунктов, таких как режимы запуска. Например, с помощью кольцевого элемента управления пользователь может выбрать один из возможных режимов запуска: непрерывный, однократный и внешний.

Щелкните правой кнопкой по кольцевому элементу управления и выберите **Edit Items** из контекстного меню, чтобы добавить пункты в список, из которого Вы сможете выбирать. Порядок пунктов на странице **Edit Items** диалогового окна **Ring Properties** определяет порядок пунктов в элементе управления. Вы также можете настроить кольцевой элемент управления так, что пользователи смогут вводить числовые значения, не связанные ни с каким пунктом списка; для этого щелкните правой кнопкой кольцевой элемент управления и выберите из контекстного меню **Allow Undefined Values**.

Чтобы ввести неопределенное значение в кольцевой элемент управления во время исполнения, щелкните этот элемент управления, выберите **<Other>** из выпадающего меню, введите числовое значение в появляющийся цифровой дисплей и нажмите клавишу **<Enter>**. Неопределенное значение появится в кольцевом элементе управления в угловых скобках. LabVIEW не добавляет неопределенное значение к списку пунктов, из которых можно выбирать.

Когда Вы задаёте список пунктов для кольцевого элемента управления, можно назначить специфические числовые значения для каждого пункта. Если Вы не назначите специфические значения, то LabVIEW назначит последовательные значения, которые соответствуют порядку пунктов в списке, начиная с 0 для первого списка. Чтобы назначить специфические значения, щелкните правой кнопкой кольцевой элемент управления, выберите **Edit Items** из контекстного меню и удалите отметку с опции **Sequential Values** на странице **Edit Items** диалогового окна **Ring Properties**. В столбце **Values** таблицы этого диалогового окна измените числовые значения, которые соответствуют каждому пункту списка. Каждый пункт кольцевого элемента управления должен иметь уникальное числовое значение.

### **Элементы управления перечислительного типа (Enumerated Type Controls)**

Чтобы предъявить пользователю список пунктов, из которых можно выбирать, используйте элементы управления перечислительного типа. Элемент управления перечислительного типа похож на текстовый кольцевой элемент управления. Однако тип данных перечислительного элемента управления включает информацию о числовых значениях и о строковых метках. Тип же данных кольцевого элемента управления является числовым.



**Примечание.** Вы не можете позволить пользователю вводить неопределенные значения в перечислительный элемент управления и не можете назначить специфические числовые значения пунктам в перечислительном элементе управления. Если такая возможность необходима, используйте кольцевой элемент управления. Более подробно о кольцевых элементах управления см. в разделе *Кольцевые элементы управления* в настоящей Главе.

Вы можете использовать элементы управления перечислительного типа для выбора вариантов в структуре Case. Более подробно о структуре Case см. раздел *Структуры Case* в Главе 8 *Циклы и структуры*.

Числовым представлением элемента управления перечислительного типа может быть беззнаковое целое длиной 8, 16 или 32 бит. (unsigned 8, 16, 32). Чтобы изменить числовое представление щелк-

ните правой кнопкой элемент управления и выберите **Representation** из контекстного меню.

### **Тонкости использования элементов управления и индикаторов перечислительного типа**

Все арифметические функции, исключая Increment (+1) и Decrement (-1), обрабатывают данные перечислительного типа (enumerated type) как беззнаковые целые. Функция Increment преобразует последнее перечислительное значение в первое, а функция Decrement - первое в последнее. При преобразовании знакового целого к перечислительному типу все отрицательные числа преобразуются в первое значение величины перечислительного типа, а все выходящие за пределы диапазона положительные числа преобразуются к последнему значению величины перечислительного типа. Выходящие за диапазон беззнаковые целые всегда преобразуются к последнему значению величины перечислительного типа.

Если Вы подсоединяете значение с плавающей точкой (floating point) к индикатору перечислительного типа, LabVIEW преобразует значение с плавающей точкой к ближайшему числовому значению в индикаторе. С числами, выходящими за пределы диапазона, LabVIEW обращается как это описано выше. Если Вы подсоединяете перечислительный элемент управления к входу любого числового типа, LabVIEW преобразует перечислительную величину к такой числовой величине. Чтобы соединить перечислительный элемент управления к перечислительному индикатору, пункты в индикаторе должны соответствовать пунктам в элементе управления. Тем не менее, индикатор может содержать больше пунктов, чем элемент управления.

### **Контейнерные элементы управления (Container Controls)**

Контейнерные элементы управления, размещенные на палитрах **Containers** и **Classic Containers**, используйте для группировки элементов управления и индикаторов, отображения лицевой панели другого ВП на лицевой панели текущего ВП или для отображения объектов ActiveX на лицевой панели. Более подробно об использовании ActiveX см. в Главе 19 *Связность в среде Windows* настоящего руководства.

## **Элемент управления «многостраничное диалоговое окно» (Tab Control)**

Используйте элементы управления Tab Control в тех случаях, когда требуется наложение элементов управления друг на друга из-за нехватки места на лицевой панели. Элемент управления Tab Control состоит из страниц (pages) и закладок (tabs). Разместите объекты лицевой панели на каждой странице такого элемента управления, и затем пользуйтесь закладками в качестве селектора для отображения различных страниц.

Элемент управления «многостраничное диалоговое окно» обычно используется в ситуациях, когда имеется немного объектов лицевой панели, которые используются совместно или в ходе отдельной фазы работы. Например, у Вас может быть ВП, который требует от пользователя вначале установить некоторые настройки перед тем, как начнется испытание, затем позволяет пользователю модифицировать некоторые аспекты испытания по мере его выполнения и в завершение позволяет пользователю отобразить и сохранить только выбранные им данные.

На блок-диаграмме элемент управления Tab Control представляется терминалом, который по умолчанию имеет перечислительный тип. Терминалы для элементов управления и индикаторов, размещенных на страницах элемента управления Tab Control, имеют такой же вид, как и любые другие терминалы. Более подробно об элементах управления перечислительного типа см. в разделе *Элементы управления перечислительного типа* настоящей Главы.

## **Элемент управления «подпанель» (Subpanel Control)**

Используйте элемент управления «подпанель» для отображения лицевой панели другого ВП на лицевой панели текущего ВП. Например, Вы можете использовать такой элемент управления для построения пользовательского интерфейса, который ведет себя наподобие «мастера» (wizard). Поместите на лицевую панель ВП верхнего уровня кнопки **Back** (назад) и **Next** (следующий) и используйте элемент управления «подпанель» для загрузки различных лицевых панелей для каждого шага работы такого «мастера».



**Примечание.** Вы можете создавать и редактировать элемент управления «подпанель» только в LabVIEW Full and Professional Development Systems. Если ВП содержит элемент управления «подпанель», Вы можете

запускать такой ВП в LabVIEW всех пакетов поставки, но Вы не сможете конфигурировать такой элемент управления в LabVIEW Base Package.

Когда Вы размещаете элемент управления «подпанель» на лицевую панель, LabVIEW не создает соответствующий ему терминал на блок-диаграмме. Вместо этого LabVIEW создает на блок-диаграмме узел вызова (Invoke Node) с выбранным методом Insert VI. Чтобы загрузить ВП в элемент управления «подпанель», подключите ссылку (reference) на этот ВП к узлу вызова. Более подробно о ссылках на ВП и об узлах вызова см. Главу 17 *Программное управление ВП*.



**Примечание.** Поскольку у элемента управления «подпанель» нет терминала, нельзя создать массив элементов управления «подпанель» и нельзя создать определение типа для такого элемента управления. Вы можете поместить элемент управления «подпанель» в кластер, чтобы сгруппировать его с другими элементами управления, но такой кластер не может содержать только элемент управления «подпанель» (один или несколько).

Вы также не можете загружать лицевую панель ВП в удаленных копиях (instance) LabVIEW, а также нельзя загружать лицевые панели рекурсивно.

Нельзя использовать комбинации клавиш для перемещения или управления лицевой панелью в элементе управления «подпанель».

Если Вы загружаете незапущенный ВП, ВП в элементе управления «подпанель» загружается в режиме запуска.

LabVIEW отображает только видимую область лицевой панели ВП, который загружается в элемент управления «подпанель». После того как Вы остановите ВП, который содержит элемент управления «подпанель», LabVIEW очистит лицевую панель в элементе управления «подпанель». Вы также можете использовать метод Remove VI, чтобы выгрузить ВП из элемента управления «подпанель».

Пример использования элемента управления «подпанель» содержится в examples\general\controls\subpanel.llb.

## Элементы управления и индикаторы имен ввода/вывода (I/O Name)

Используйте элементы управления и индикаторы имен ввода/вывода для передачи имен DAQ-каналов, имен VISA ресурсов и логических имен IVI, которые Вы конфигурируете для ВП ввода/вывода, чтобы установить связь с прибором или DAQ-устройством.

Константы для имен ввода/вывода размещены на палитре **Functions**.



**Примечание.** Все элементы управления имен ввода/вывода и соответствующие константы доступны на всех платформах. Это позволяет вам разрабатывать ВП ввода/вывода на любых платформах, которые могут взаимодействовать с устройствами, специфичными для платформ. Однако, если Вы попытаетесь запустить ВП с такими зависимыми от платформы элементами управления вводом/выводом на платформе, которая их не поддерживает, произойдет ошибка.

**(Windows)** Для конфигурирования имен DAQ-каналов, имен VISA-ресурсов и логических имен IVI используйте Measurement & Automation Explorer, который доступен через меню **Tools**.

**(Mac OS)** Используйте утилиту NI-DAQ Configuration Utility, доступную через меню **Tools**, чтобы сконфигурировать аппаратуру ввода/вывода National Instruments. Для конфигурирования имен каналов ввода/вывода используйте программу DAQ Channel Wizard.

**(Mac OS и UNIX)** Используйте конфигурационные утилиты для вашего прибора, чтобы сконфигурировать имена VISA ресурсов и логические IVI имена. Более подробно об этом см. в документации на ваш прибор.

Элемент управления IMAQ session является уникальным идентификатором, который представляет связь с внешним оборудованием.

## Элементы управления осцилограммами (Waveform Control)

Используйте элементы управления осцилограммами для манипуляций с отдельными компонентами осцилограмм (waveform). Более

лее подробно о типе данных waveform см. в разделе *Тип данных waveform (осциллограмма)* в Главе 12 *Графики и диаграммы*.

### Элемент управления цифровая осциллограмма (Digital Waveform Control)

Используйте элемент управления цифровая осциллограмма для манипуляций с отдельными компонентами цифровой осциллограммы. Более подробно об элементе управления цифровая осциллограмма см. раздел *Тип данных digital waveform (цифровая осциллограмма)* в Главе 12 *Графики и диаграммы*.

### Элемент управления цифровыми данными (Digital Data Control)

Элемент управления цифровыми данными содержит цифровые данные, организованные по строка и столбцам. Используйте цифровой тип данных для построения цифровых осциллограмм или для отображения цифровых данных, извлеченных из цифровой осциллограммы. Подсоедините цифровую осциллограмму к индикатору цифровых данных, чтобы просмотреть отсчеты (samples) и сигналы (signals) цифровой осциллограммы. Элемент управления цифровыми данными на Figure 2-1 отображает пять отсчетов, каждый из которых содержит восемь сигналов.

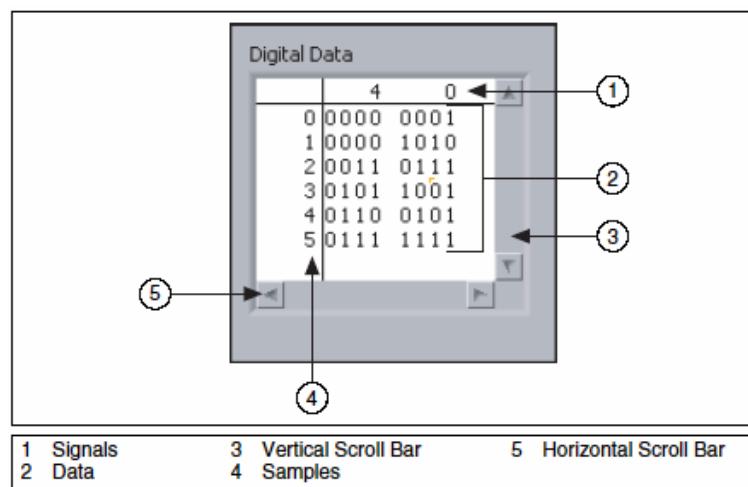


Figure 4-2. Элемент управления цифровыми данными (Digital Data Control)

Вы можете вставлять и удалять строки и столбцы в элементе управления цифровыми данными. Чтобы вставить строку, щелкните правой кнопкой отсчет в столбце (левом) номеров отсчетов и выберите **Insert Row Before** из контекстного меню. Для удаления строки, щелкните правой кнопкой отсчет в столбце номеров отсчетов и выберите **Delete Row** из контекстного меню. Чтобы вставить столбец, щелкните правой кнопкой сигнал в строке (верхней) номеров сигналов и выберите **Insert Column Before** из контекстного меню. Для удаления столбца щелкните правой кнопкой сигнал в строке номеров сигналов и выберите **Delete Column** из контекстного меню.

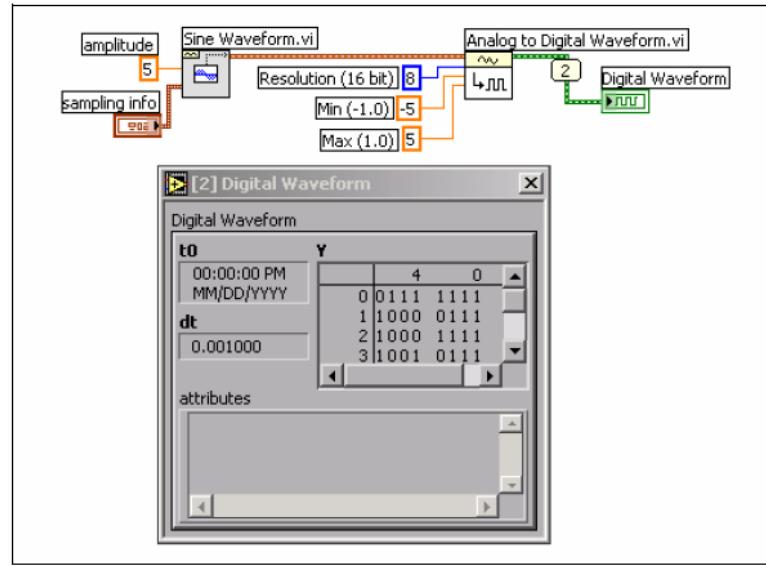
Вы также можете вырезать, копировать и вставлять цифровые данные внутри этого элемента управления. Чтобы вырезать данные, селектируйте нужную строку или столбец, щелкните правой кнопкой и выберите **Data Operations»Cut Data** из контекстного меню. Вырезать можно только целиком строку или столбец данных. Нельзя создать новую строку или столбец с цифровыми данным, которые были вырезаны. Для копирования данных выберите область, которую Вы хотите скопировать, щелкните правой кнопкой и выберите **Data Operations»Copy Data** из контекстного меню. Чтобы вставить цифровые данные, селектируйте область, в которую Вы хотите скопировать, и выберите **Data Operations»Paste Data** из контекстного меню. Вставлять можно только в область той же размерности, что и область из которой Вы вырезали или копировали. Например, если Вы скопировали четыре бита данных из одной строки, Вы можете селектировать четыре существующих бита данных чтобы вставить вместо них в той же строку или в любой другой строке. Если же Вы скопировали четыре бита данных из области две строке на два столбца, то Вы можете вставить эти данные в любую область размером две строки на два столбца.

Элемент управления цифровыми данными (digital data control) и элемент управления цифровой осциллограммой (digital waveform control) могут принимать в качестве значений только 0, 1, L, H, Z, X, T и V. Вы можете отобразить данные в элементе управления цифровыми данными в двоичном, шестнадцатеричном, восьмеричном и десятичном форматах. Цифровые состояния L, H, Z, X, T и V отображают особые состояния, которые используют некоторые измерительные устройства и которые выполняют роль вопросительных знаков (для неопределенных состояний) при отображении данных в шестнадцатеричном, восьмеричном или десятичном формате.

так. Щелкните правой кнопкой элемент управления, выберите **Data Format** и установите нужный формат представления для элемента управления.

#### **Преобразование данных к формату Digital Data**

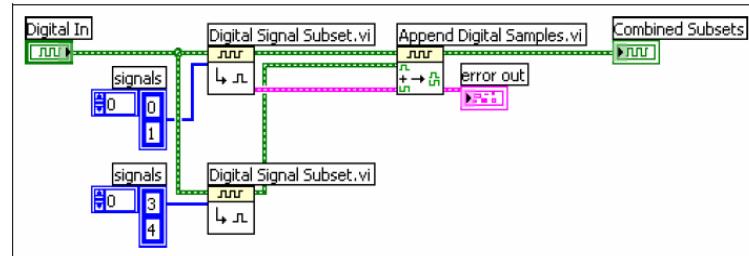
Во многих случаях измеренный сигнал вводится в виде строчки данных. Чтобы отобразить такой сигнал на индикаторе графика цифровой осциллограммы (digital waveform graph) Вы должны преобразовать введенную строчку данных к типу digital data или к типу digital waveform. Для преобразования данных к типу digital waveform используйте ВП Analog to Digital Waveform. Чтобы извлечь цифровые данные из данных типа digital waveform используйте функцию Get Waveform Components. Блок-диаграмма на Figure 4-3 имитирует ввод синусоидального сигнала с амплитудой 5, что означает, что синусоидальный сигнал может принимать значения в диапазоне от -5 до 5. ВП Analog to Digital Waveform на этой блок-диаграмме представляет каждое значение величиной из 8 бит. 8 бит могут представить минимальное значение -5 и максимальное значение 5. Пробник Digital Waveform отображает порцию результирующих значений в двоичном формате.



**Figure 4-3.** Преобразование аналоговой осциллограммы (Analog Waveform) к цифровой осциллограмме (Digital Waveform)

#### Ввод цифровых подмножеств (Digital Subset)

Для извлечения отдельных сигналов из цифровой осциллограммы используйте ВП Digital Signal Subset (подмножество цифрового сигнала). Блок-диаграмма на Figure 4-4 показывает, как можно комбинировать извлеченные отдельные сигналы с другими цифровыми данными для создания новых цифровых осциллограмм.



**Figure 4-4.** Добавление одной цифровой осциллограммы к другой

Верхний ВП Digital Signal Subset извлекает первый и второй сигналы из цифровой осциллограммы. Нижний ВП Digital Signal Subset извлекает четвертый и пятый сигналы. ВП Append Digital Samples (добавить цифровые отсчеты) добавляет первый сигнал к четвертому и второй сигнал к пятому. Затем строятся результирующие графики этих двух сигналов на графике цифровой осциллограммы.

#### **Добавление (Appending) цифровых отсчетов и сигналов**

Для добавления отсчетов одного цифрового сигнала к концу отсчетов другого цифрового сигнала используйте ВП Append Digital Signals. Вы можете добавлять сигналы с одинаковым и с разным числом отсчетов. Например, если у Вас есть два цифровых сигнала, оба из которых состоят из двух строк по 8 бит, результирующие цифровые данные будут содержать две строки по 16 бит. Если Вы комбинируете два сигнала, один из которых состоит из двух строк по 8 бит, а другой – одну строку из 8 бит, то результирующие цифровые данные будут содержать две строки по 16 бит. ВП Append Digital Signals заполняет недостающие столбцы второго отсчета значениями, которые Вы выберите на входе **default value**.

#### **Сжатие (compressing) цифровых данных**

Если Вы хотите для улучшения внешнего вида данных отобразить одинаковые битовые подмножества или одинаковые строки двух или более последовательных цифровых сигналов, используйте ВП Compress Digital Data. Например, если Вы вводите выборку из 10 цифровых осциллограмм, и десятая осциллограмма отличается от остальных девяти, то сжатие цифровых данных поможет Вам легко найти, какая осциллограмма отличается. Сжатие цифровых данных также сохраняет ресурсы памяти. Для восстановления цифровых данных после сжатия используйте ВП Uncompress Digital Data.

#### **Поиск по шаблону (Searching for a Pattern)**

Если Вы хотите произвести поиск по заданному шаблону, используйте. Например, если Вы получаете большую цифровую осциллограмму, и хотите увидеть, какая ее часть совпадает с некоторым шаблоном, подсоедините этот шаблон на вход **digital pattern** ВП Search for Digital Pattern, чтобы обнаружить хотя бы одно такое совпадение.

## Ссылки на объекты или приложения

Для работы с файлами, директориями, устройствами и сетевыми соединениями используйте элементы управления и индикаторы для ссылочных номеров (reference number), размещенные на палитрах **Refnum** и **Classic Refnum**. Для передачи информации от объекта лицевой панели к ВПП используйте ссылочный номер (refnum) элемента управления. Более подробно о ссылочных номерах элементов управления см. в разделе *Управление объектами лицевой панели* в Главе 17 *Программное управление ВП*.

Ссылочный номер (reference number или refnum) это уникальный идентификатор для таких объектов как файл, устройство или сетевое соединение. Когда Вы открываете файл, устройство или сетевое соединение, LabVIEW создает refnum, связанный с этим файлом, устройством или сетевым соединением. Все операции, выполняемые над открытыми файлами, устройствами или сетевыми соединениями используют refnum для идентификации каждого объекта. Для передачи ссылочных номеров из Вашего ВП или в него используйте элемент управления или индикатор Refnum. Например, используйте элемент управления или индикатор Refnum для модификации содержимого файла, на который ссылается Refnum без закрытия или повторного открытия этого файла.

Поскольку Refnum является временным указателем на открытый объект, он будет правильным только на период, в течение которого объект открыт. Если Вы закроете объект, LabVIEW устранит связь между ссылочным номером и объектом, и этот ссылочный номер становится недействительным. Если Вы откроете тот же объект снова, LabVIEW создаст новый ссылочный номер, отличающийся от предыдущего ссылочного номера.

LabVIEW запоминает информацию, связанную с каждым ссылочным номером, такую как текущее место записи в или чтения из данного объекта и степень доступа пользователя, так что Вы можете выполнять параллельные, но независимые операции на одном объекте. если ВП открывает объект много раз, то каждая операция его открытия возвращает различные ссылочные номера.

## Диалоговые элементы управления и индикаторы

Используйте диалоговые элементы управления в создаваемых вами диалоговых окнах. Диалоговые элементы управления и индикаторы

разработаны специально для использования в диалоговых окнах и включают кольцевые и прокручиваемые элементы управления, чистовые движки и перемещаемые полосы, списочные окна, таблицы, строковые и путевые элементы управления, элемент управления «многостраничное диалоговое окно» (tab control), элемент управления деревьями (tree control), кнопки, проверочные окна (checkboxes), радио кнопки и непрозрачная метка, цвет которой автоматически совпадает с фоновым цветом его предшественника. Эти элементы управления отличаются от тех, которые появляются на лицевой панели только с точки зрения их внешнего вида. Эти элементы управления имеют цвета, которые Вы установили для вашей системы.

Поскольку диалоговые элементы управления изменяют свой вид в зависимости от платформы (операционной системы), на которой Вы запускаете ВП, внешний вид элементов управления в созданном Вами ВП будет совместимым со всеми платформами, совместимыми с LabVIEW (Windows, Unix, Mac OS). Когда Вы запускаете ВП на другой платформе, диалоговые элементы управления адаптируют свой цвет и внешний вид так, чтобы соответствовать стандартным элементам управления в диалоговых окнах этой платформы.

Выберите **File»VI Properties** и затем **Window Appearance** из выпадающего меню **Category**, чтобы скрыть строку меню и полосы прокрутки и создать ВП, который выглядит и ведет себя как стандартное диалоговое окно в каждой из платформ. Выберите **Editor Options** из выпадающего меню **Category**, чтобы изменить стиль элементов управления или индикаторов, которые LabVIEW создает, когда Вы щелкаете правой кнопкой по терминалу и выбираете из контекстного меню **Create»Control** или **Create»Indicator**. Выберите **Tools»Options** и затем **Front Panel** из верхнего выпадающего меню, чтобы изменить стиль элемента управления или индикатора, которые LabVIEW создает в новых ВП, когда Вы щелкаете правой кнопкой терминал и выбираете **Create»Control** или **Create»Indicator** из контекстного меню. Более подробно о конфигурировании вида и поведения ВП см. в разделе *Конфигурирование внешнего вида и поведения ВП* в Главе 16 *Конфигурирование ВП*.

## Использование текстовых меток

---

Для идентификации объектов на лицевой панели и блок-диаграмме используйте метки (labels).

В LabVIEW имеется два типа меток – собственные метки (owned labels) и свободные метки (free labels). Собственные метки принадлежат отдельному объекту, перемещаются вместе с ним и характеризуют только его. Вы можете перемещать собственную метку независимо, но если Вы переместите помеченный объект, то метка переместится вместе с таким объектом. Можно скрыть (hide) собственные метки, но нельзя копировать или удалять их независимо от объекта. Для числовых элементов и индикаторов Вы можете также отобразить метку единиц измерения (unit label), выбирая из контекстного меню **Visible Items»Unite Label**. Более подробно о числовых метках см. в разделе *Числовые единицы и строгая проверка типов* в Главе 5 *Построение блок-диаграмм*.

Свободные метки не присоединены ни к одному из объектов, и их можно создавать, перемещать, поворачивать или удалять независимо. Используйте их для пояснения лицевых панелей и блок-диаграмм.

Для создания или редактирования свободных меток сделайте двойной щелчок по свободному пространству или воспользуйтесь инструментом Labeling. Собственные метки редактируются также, но в пределах выделенного для них места.

Для получения более подробной информации о создании описательных меток для пользовательского интерфейса воспользуйтесь разделом *Labels* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

### **Заголовки (Captions)**

Объекты лицевой панели могут иметь также заголовки. Для отображения заголовка щелкните объект правой кнопкой и выберите из контекстного меню **Visible Items»Caption**. В отличие от метки заголовок не является именем объекта, но Вы можете использовать заголовок в качестве расширенного описания объекта. Заголовок не виден на блок-диаграмме.

Если Вы назначите объект терминалу соединительной панели, то заголовок появится на блок-диаграмме в тот момент, когда Вы, используя инструмент Wiring, будете перемещать курсор над терминалом ВПП. Заголовок появляется также около терминала ВПП в окне **Context Help**. Более подробно о терминалах соединительной

панели см. в разделе *Конфигурирование соединительной панели* в Главе 7 *Создание ВП и ВПП*.

## Текстовые характеристики

---

LabVIEW использует шрифты уже установленные на Ваш компьютер. Для изменения атрибутов текста используйте спадающее меню **Text Settings** на панели инструментов. Если Вы селектируете объекты или текст перед тем, как Вы сделаете выбор из спадающего меню **Text Settings**, то изменения коснутся того, что Вы селектировали. Если ничего не было селектировано, то изменения применяются к шрифту по умолчанию. Изменения шрифта по умолчанию не изменят шрифта уже существующих меток. Они затронут только метки, которые Вы будете создавать уже после этих изменений.

Для применения специфических стилей к селектированному тексту на лицевой панели выберите **Font Dialog** из спадающего меню **Text Settings**. Если Вы не селектировали никакой текст, то будет помечена птичкой опция **Panel Default**. Если Вы выберите **Text Settings»Font Dialog** из блок-диаграммы, не селектируя ни одного объекта, то будет помечена птичкой опция **Diagram Default**. Вы можете установить различные шрифты для лицевой панели и для блок-диаграммы. Например, у Вас может быть маленький шрифт на блок-диаграмме и большой – на лицевой панели.

Спадающее меню **Text Settings** содержит следующие встроенные шрифты:

- **Application Font** – шрифт, используемый по умолчанию для панелей Controls и Functions и для текста внутри новых элементов управления.
- **System Font** – Используется для меню.
- **Dialog Font** – Используется для текста в диалоговых окнах.

При переносе ВП, содержащего один из этих встроенных шрифтов, на другую платформу, будут подбираться максимально близкие к ним шрифты.

Спадающее меню **Text Settings** содержит также подпункты **Size**, **Style** и **Color**.

Изменения шрифта, которые Вы делаете из этих подменю, применяются к объектам, которые Вы до этого селектировали. Например, если Вы выбрали новый шрифт, когда были селектированы кнопка и график, то все метки, шкалы и цифровые дисплеи этих объектов изменят свой шрифт на новый. Когда Вы делаете изменения, LabVIEW старается сохранить, насколько это возможно, старые атрибуты. Например, если Вы измените в некоторых объектах шрифт на Courier, то объекты сохранят по возможности прежними размер и стиль шрифта. Когда Вы используете диалоговое окно **Font**, то LabVIEW изменит у селектированных объектов выбранные текстовые характеристики. Если Вы выбираете один из встроенных шрифтов или текущий шрифт (Current Font), то LabVIEW применит этот шрифт и размер к селектированным объектам.

Когда Вы работаете с объектами, содержащими много текстовых зон, подобных движкам (slide), изменения шрифтов, которые Вы делаете, действуют на объекты или на текст, который Вы к настоящему времени селектировали. Например, если Вы селектируете весь движок и выберите **Style»Bold** из спадающего меню **Text Settings**, то шкала, цифровой дисплей и метка будут выводиться жирным шрифтом. Если Вы селектируете только метку и выберите **Bold**, то только метка будет выводиться жирным шрифтом.

Более подробная информация об использовании шрифтов и текстовых характеристик для конструирования пользовательского интерфейса приведена в разделе *Fonts and Text Characteristics* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## **Конструирование пользовательского интерфейса**

---

Если ВП служит в качестве пользовательского интерфейса или диалогового окна, то весьма важны вид и формат лицевой панели. Дружественный дизайн лицевой панели помогает пользователям легко определить необходимые действия. Вы можете конструировать лицевые панели, которые выглядят как приборы или устройства.

Более подробно о конструировании пользовательского интерфейса описано в руководстве *LabVIEW Development Guidelines*.

Относительно использования событий для улучшения функциональности пользовательского интерфейса см. в разделе *Структуры выбора и последовательности* в Главе 8 *Циклы и структуры*.

## Использование элементов управления и индикаторов лицевой панели

Элементы управления и индикаторы являются основными компонентами лицевой панели. Когда Вы конструируете лицевую панель, рассмотрите логически, как пользователь взаимодействует с ВП и группой элементов управления и индикаторов. Если некоторые элементы управления связаны, добавьте декоративную границу вокруг них или разместите их в кластере. Используйте элементы оформления, размещенные в палитре **Decorations**, чтобы сгруппировать или разделить объекты на лицевой панели с помощью боксов, линий или стрелок. Эти элементы служат только для оформления и не отображают никаких данных.

Не помещайте объекты на лицевой панели слишком близко друг к другу. Пытайтесь оставить некоторое пустое пространство, чтобы сделать лицевую панель более легкой для восприятия. Пустое пространство также предохраняет пользователей от случайных щелчков по элементам управления или кнопкам. Назначайте специфические имена кнопкам и пользуйтесь общепринятой терминологией. Используйте имена вроде Старт, Стоп или Сохранить вместо ОК. Специфические имена более понятны пользователям.

Пользуйтесь имеющимися в LabVIEW шрифтами и цветами по умолчанию. LabVIEW замещает встроенные шрифты на соответствующие семейства шрифтов на различных платформах. Если Вы выберите другой шрифт, то LabVIEW подставит наиболее подходящий, если именно этого шрифта на компьютере не окажется. Цветами LabVIEW манипулирует так же, как и шрифтами. Если нужный цвет не доступен на компьютере, то LabVIEW заменит его на более подходящий. Вы можете также использовать системные цвета, чтобы адаптировать вид лицевой панели к системным цветам любого компьютера, на котором запускается ВП.

Избегайте размещения одних объектов поверх других объектов. Размещение метки или любого другого объекта поверх или с частичным перекрытием элемента управления или индикатора снижа-

ет скорость обновления экрана и может привести к мерцанию изображения элемента управления или индикатора.

Более подробная информация об использовании формата, шрифтов и цвета для конструирования пользовательского интерфейса приведена в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## Конструирование диалоговых окон

Если ВП содержит последовательность диалоговых окон, которые появляются на одном и том же месте экрана, организуйте их так, чтобы кнопки на первом диалоговом окне непосредственно не совпадали с кнопками на следующем диалоговом окне. Пользователи могут сделать двойной щелчок по кнопке на первом диалоговом окне и неосознанно щелкнуть кнопку на следующем диалоговом окне.

Более подробно о диалоговых элементах управления см. в разделе *Диалоговые элементы управления и индикаторы* в настоящей Главе. Более подробно об использовании диалоговых окон в пользовательском интерфейсе см. в разделе *Dialog Boxes* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## Выбор размера экрана

Когда Вы конструируете ВП, рассмотрите, будет ли лицевая панель отображаться на компьютерах с различным разрешением экрана. Чтобы установить окно лицевой панели пропорциональным экранному разрешению, выберите **File»VI Properties**, затем выберите **Window Size** в спадающем меню **Category** и поместите галочку на опции **Maintain Proportions of Window for Different Monitor Resolutions**.

Более подробно о выборе размера экрана для пользовательского интерфейса см. в разделе *Sizing and Positioning* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## 5. Построение блок-диаграммы

---

После того как Вы построите лицевую панель, Вы добавляете код, использующий графическое представление функций для управления объектами лицевой панели. Блок-диаграмма содержит графический исходный код.

---

**Более подробно...**

Более подробно относительно конструирования и конфигурирования блок-диаграммы см. справочную систему *LabVIEW Help*.

---

### Соответствие между объектами лицевой панели и терминалами блок-диаграммы

---

Объекты лицевой панели на блок-диаграмме имеют вид терминалов. Сделайте двойной щелчок по терминалу блок-диаграммы, чтобы высветить соответствующий ему элемент управления или индикатор на лицевой панели. Данные, которые Вы вводите в элементы управления лицевой панели, поступают на блок-диаграмму через терминалы элементов управления. В ходе выполнения выходные данные поступают на терминалы индикаторов, через которые они покидают блок-диаграмму, вводятся на лицевую панель и появляются на индикаторах лицевой панели.

### Объекты блок-диаграммы

---

Объекты блок-диаграммы включают терминалы (terminals), узлы (nodes) и функции (functions). Вы строите диаграмму, соединяя объекты проводниками (wires).

### Терминалы блок-диаграммы



Вы можете настраивать так, чтобы элементы управления или индикаторы лицевой панели отображались на блок-диаграмме либо как терминалы в виде иконок либо в виде терминалов типов данных. По умолчанию, объекты лицевой панели отображаются в виде иконок. Иконка кнопки, показанная слева, представляет кнопку на лицевой панели. Буквы DBL внизу терминала представляют тип данных: числовой с плавающей точкой с удвоенной точностью.



DBL-терминал, показанный слева, представляет числовой элемент управления или индикатор двойной точности с плавающей точкой. Щелкните правой кнопкой терминал и выберите **Display Icon** из контекстного меню, чтобы удалить птичку и отобразить терминал в виде значка типа данных. Для отображения на блок-диаграмме не только типа данных терминала, но и типа объекта лицевой панели, используйте терминал в виде иконки. Если же нужно сэкономить место на блок-диаграмме, используйте терминалы в виде значка типа данных.



**Примечание.** Терминалы-иконки крупнее терминалов-типов данных, поэтому Вы можете ненамеренно затемнить другие объекты блок-диаграммы, когда преобразуете терминалы-типы данных в терминалы-иконки.

Терминал это любая точка, к которой Вы можете присоединить проводник, не являющаяся другим проводником. В LabVIEW имеются терминалы элементов управления и индикаторов, узловые терминалы (node terminal), константы и специализированные терминалы на управляющих структурах, такие как входные и выходные терминалы на формульном узле. Для подсоединения к терминалам и передачи данных к другим терминалам используются проводники (wires). Щелкните правой кнопкой объект блок-диаграммы и выберите **Visible Items»Terminals** из контекстного меню, чтобы сделать терминалы видимыми. Щелкните правой кнопкой объект блок-диаграммы и еще раз выберите **Visible Items»Terminals**, чтобы скрыть терминалы.

### Типы данных элементов управления и индикаторов

В Табл. 5-1 показаны обозначения для различных типов терминалов элементов управления и индикаторов. Цвета и обозначения для каждого терминала указывают тип данных элементов управления и индикаторов. Терминалы элементов управления ограничены более толстыми линиями, чем индикаторы. Кроме того, для отличия терминалов элементов управления от терминалов индикаторов используются стрелочки. У терминалов элементов управления они справа, а у терминалов индикаторов – слева.

Табл. 5-1. Терминалы элементов управления и индикаторов

Control	Indicator	Data Type	Color	Default Values
 		Числовой однократной точности с плавающей точкой	Оранже-вый	0.0
 		Числовой двойной точности с плавающей точкой	Оранже-вый	0.0
 		Числовой расширенной точности с плавающей точкой	Оранже-вый	0.0
 		Числовой комплексный однократной точности с плавающей точкой	Оранже-вый	0.0+i 0.0
 		Числовой комплексный двойной точности с плавающей точкой	Оранже-вый	0.0+i 0.0
 		Числовой комплексный расширенной точности с плавающей точкой	Оранже-вый	0.0+i 0.0
 		Числовой знаковый целый 8-бит	Синий	0
 		Числовой знаковый целый 16-бит	Синий	0
 		Числовой знаковый целый 32-бит	Синий	0
 		Числовой беззнаковый целый 8-бит	Синий	0
 		Числовой беззнаковый целый 16-бит	Синий	0
 		Числовой беззнаковый целый 32-бит	Синий	0
 		Временная метка <64.64>- бит	Коричне-вый	Дата и время (местное)
 		Перечислительный тип	Синий	–
 		Логический	Зеленый	FALSE
 		Строчковый	Розовый	пустая строка

Control	Indicator	Data Type	Color	Default Values
		Массив – в квадратных скобках включен тип данных его элементов, принимает цвет, соответствующий этому типу данных	Изменяется	–
		Кластер – включает несколько типов данных. Имеет коричневый цвет, если все элементы кластера числовые или розовый, если элементы кластера имеют различный тип.	Коричневый или розовый	–
		Путь	Морской волны	<Not A Path>
		Динамический (изменяемый)	Темно синий	–
		Осциллограмма (Waveform) – Кластер элементов, которые содержат дату, начальное время и шаг дискретизации. Более подробно о типе данных Waveform см. раздел <i>Тип данных waveform (осциллограмма)</i> Главы 12 <i>Графики и диаграммы</i> .	Коричневый	–
		Цифровая осциллограмма	Темно зеленый	–
		Цифровые данные	Темно зеленый	–
		Ссылочный номер (refnum)	Морской волны	–
		Вариантный (Variant) – содержит имя элемента управления или индикатора, информацию о типе данных, которые Вы конвертируете, и сами данные. Более подробно о вариантом типе данных см. в разделе <i>Обработка вариантов данных</i> в этой Главе.	Фиолетовый	–

Control	Indicator	Data Type	Color	Default Values
		Имя ввода/вывода – передает имена каналов, ресурсные имена VISA и логические имена IVI, которые Вы конфигурируете для ВП ввода/вывода, чтобы осуществить связь с прибором или DAQ-устройством. Более подробно о типе данных имя ввода/вывода см. в разделе <i>Элементы управления и индикаторы имен ввода/вывода</i> в Главе 4 <i>Построение лицевой панели</i> .	Фиолетовый	–
		Рисунок – отображает рисунок, который может содержать линии, окружности, текст и другие типы графических элементов. Более подробно о типе данных рисунок см. в разделе <i>Использование индикатора Picture (изображение)</i> в Главе 13 <i>Графика и звук</i> .	Синий	–

Многие типы данных имеют соответствующие наборы функций, которые могут манипулировать такими данными. Более подробную информацию относительно того, какие функции используют какой тип данных, можно найти в разделе *Обзор функций* в настоящей Главе.

Более подробно о выборе типа данных для оптимизации используемой памяти см. в разделе *Memory and Speed Optimization* в Главе 6 *LabVIEW Style Guide* в руководстве *LabVIEW Development Guidelines*.

### Константы

Константы – это терминалы на блок-диаграмме, которые поставляют фиксированные значения данных. Универсальные константы – это константы с фиксированными значениями, такими как  $\pi$  ( $\pi$ ) и

бесконечность ( $\infty$ ). Пользовательские (user defined) – это константы, которые Вы определяете и редактируете перед запуском ВП.

Установите метку у константы, щелкнув правой кнопкой константу и выбрав **Visible Items»Label** из контекстного меню. Универсальные константы имеют заранее определенные значения меток, которые Вы можете потом редактировать, используя инструмент **Operating** или инструмент **Labeling**.

Большинство констант находятся вверху или внизу соответствующих палитр.

#### **Универсальные константы**

Используйте универсальные константы для математических вычислений и при форматировании строк и путей. LabVIEW включает следующие типы универсальных констант:

- **Универсальные числовые константы** – Набор широко используемых математических и физических величин с высокой точностью, таких как основание натурального логарифма (е) и скорость света. Универсальные числовые константы размещены на палитре **Additional Numeric Constant**.
- **Универсальные строковые константы** – Набор широко используемых неотображаемых строковых символов, таких как перевод строки и возврат каретки. Универсальные строковые константы размещены на палитре **String** (Строки).
- **Универсальные файловые константы** – Набор широко используемых значений путей к файлам, таких как Not A Path (Не путь), Not A Refnum (Не ссылочный номер) и Default Directory (Директория по умолчанию). Универсальные файловые константы размещены на палитре **File Constant**.

#### **Пользовательские константы (User-Defined Constants)**

Палитра Functions содержит константы, размещенные по типам. Это логические, числовые, кольцевые, перечислительные, цветовые и строковые константы, константы-массивы, константы-кластеры и путевые константы.

Для создания пользовательской константы щелкните правой кнопкой по входному терминалу ВП или функции и выберите **Create**

**Constant** из контекстного меню. Нельзя изменять пользовательские константы при запущенном ВП.

Вы также можете создать константу перетаскиванием элемента управления на блок-диаграмму. При этом LabVIEW создает константу, которая содержит значение, взятое из элемента управления лицевой панели, на момент перетаскивания. Элемент управления остается на лицевой панели. В последствии изменение значения в элементе управления не влияет на значение константы и наоборот.

Используйте инструмент *Operating* или *Labeling*, чтобы щелкнуть константу и затем редактировать ее значение. Если включен автоматический выбор инструментов (зеленая кнопка), то сделайте двойной щелчок по константе, чтобы переключиться на инструмент *Labeling* и затем редактировать значение константы.

## Узлы блок-диаграммы

Узлы – это объекты блок-диаграммы, которые имеют входы и/или выходы и выполняют операции, когда ВП запущен. Они аналогичны операторам, функциям и подпрограммам в текстовых алгоритмических языках. LabVIEW включает следующие типы узлов:

- **Функции** – встроенные исполнительные элементы, сравнимые с оператором или функцией. Более подробно о функциях, доступных в LabVIEW см. в разделе *Обзор функций* настоящей Главы.
- **ВПП** – ВП, используемые на блок-диаграмме другого ВП, сравнимы с подпрограммами. Более подробно об использовании ВПП на блок-диаграмме см. в разделе *Виртуальные подприборы (ВПП)* в Главе 7 *Создание ВП и ВПП*.
- **Структуры** – Элементы управления процессами, такие как структуры последовательности Flat Sequence и Stacked Sequence, структура выбора Case, циклы For Loops и While Loops. Более подробно об использовании структур см. в Главе 8 *Циклы и структуры*.
- **Формульные узлы (Formula Nodes)** – Структуры с изменяемыми размерами для введения выражений непосредственно в блок-диаграмму. Более подробно об использовании формульных узлов см. в разделе *Формульные узлы* в Главе 21 *Формулы и уравнения*.

- **Узлы выражений (Expression Nodes)** – Структуры для вычисления выражений, или уравнений, которые содержат одну переменную. Более подробно об использовании узлов выражений см. в разделе *Узлы выражений* в Главе 21 *Формулы и уравнения*.
- **Узлы свойств (Property Nodes)** – Структуры для установки или определения свойств класса. Более подробно об использовании узлов свойств см. в разделе *Узлы свойств* в Главе 17 *Программное управление ВП*.
- **Узлы вызовов (Invoke Nodes)** – Структуры для выполнения методов классов. Более подробно об использовании узлов вызова см. в разделе *Узлы вызовов* в Главе 17 *Программное управление ВП*.
- **Узлы кодовых интерфейсов (Code Interface Nodes - CIN)** – Структуры для вызова кода из тестовых языков программирования. Более подробно о них см. в разделе *Узел кодового интерфейса* в Главе 20 *Вызов кода из текстовых языков программирования*.
- **Узлы вызова по ссылке (Call by Referenced Nodes)** – Структуры для вызова динамически загружаемых ВП. Более подробно о них см. в разделе *Узлы вызова по ссылке и строго типизированные ссылочные номера* в Главе 17 *Программное управление ВП*.
- **Узлы библиотечного вызова (Call Library Nodes)** – Структуры для вызова стандартных совместных библиотек или DLL. Более подробно о них см. в разделе *Узел вызова библиотечных функций* в Главе 20 *Вызов кода из текстовых языков программирования*.

## Обзор функций

---

Функции – это основные операционные блоки LabVIEW. Иконки функций на палитре **Functions** имеют светло желтый фон и черные линии. Функции не имеют лицевых панелей или блок-диаграмм, но имеют соединительную панель. Функции нельзя ни открывать, ни редактировать.

Палитра **Functions** содержит также ВП, которые поставляются вместе с LabVIEW. Используйте эти ВП как ВПП, когда Вы строите свои ВП для сбора данных, управления приборами, связи и т.п. Более подробно об использовании встроенных ВП см в разделе *Использование встроенных ВП и функций* в Главе 7 *Создание ВП и ВПП*.

## Числовые функции

Используйте числовые (Numeric) функции для создания и выполнения арифметических, тригонометрических, логарифмических и комплексных математических операций над числами и для преобразования чисел из одного типа данных к другому.

## Логические функции

Используйте логические (Boolean) функции для выполнения логических операций над одиночными булевыми величинами или массивами булевых величин, для решения задач:

- Преобразование значения TRUE (истина) в значение FALSE (ложь) и наоборот.
- Определение булевой величины, которую нужно возвращать, если получены две или более булевые величины.
- Преобразование булевой величины в число (1 или 0).
- Выполнение сложных операций над двумя и более булевыми величинами.

## Строковые функции

Используйте строковые (String) функции для выполнения следующих задач:

- Конкатенация (сцепление) двух или более строк.
- Извлечение подмножества строк из строки.
- Поиск и замещение символов или их наборов в строке.
- Преобразование числовых данных в строки.
- Форматирование строк для использования в приложениях- текстовых процессорах или в приложениях - электронных таблицах.

Более подробно об использовании строковых функций см. разделе *Строки* в Главе 10 *Группировка данных с помощью строк, массивов и кластеров*.

## Функции над массивами

Используйте функции над массивами (Array) для создания массивов и манипуляции с ними при решении таких задач как:

- Извлечение отдельных элементов данных из массива.
- Добавление отдельных элементов данных в массив.
- Разбиение массива на его отдельные элементы данных.

Более подробно об использовании функций над массивами см. в разделе *Массивы* в Главе 10 *Группировка данных с помощью строк, массивов и кластеров*.

## Кластерные функции

Используйте кластерные (Cluster) функции для создания кластеров и манипуляции с ними при решении таких задач как:

- Извлечение отдельных элементов данных из кластера.
- Добавление отдельных элементов данных в кластер.
- Разбиение кластера на его отдельные элементы данных.

Более подробно об использовании кластерных функций см. в разделе *Кластеры* Главы 10 *Группировка данных с помощью строк, массивов и кластеров*.

## Функции сравнения

Используйте функции сравнения (Comparison) для сравнения булевых величин, строк, чисел, массивов и кластеров.

Более подробно об использовании функций сравнения см. в Приложении С *Функции сравнения*.

## Временные и диалоговые функции

Используйте временные (Time) и диалоговые (Dialog) функции для выполнения следующих задач:

- Манипуляция со скоростью выполнения операций.

- Извлечение информации о времени и дате из часов вашего компьютера.
- Создание диалоговых окон для выдачи пользователям инструкций.

Палитра Time & Dialog содержит также ВП для обработки ошибок. Более подробно об использовании ВП для обработки ошибок см. в разделе *Проверка и обработка ошибок* в Главе 6 *Запуск и отладка виртуальных приборов*.

## Функции файлового ввода/вывода

Используйте функции файлового ввода/вывода (File I/O) для выполнения следующих задач:

- Открытие и закрытие файлов.
- Чтение из и запись в файлы.
- Создание директорий и файлов, заданных в путевом элементе управления.
- Извлечение информации о директории.
- Запись строк, числе, массивов и кластеров в файлы.

Палитра **File I/O** содержит также ВП, которые выполняют общие задачи файлового ввода/вывода. Более подробно об использовании ВП и функций файлового ввода/вывода см. в Главе 14 *Файловый ввод/вывод*.

## Функции работы с осциллограммами

Используйте функции для работы с осциллограммами (Waveform) при решении следующих задач:

- Построение осциллограмм, которые включают значения осциллограмм, информацию о канале и временной привязке.
- Извлечение отдельных элементов данных из осциллограммы.
- Редактирование отдельных элементов осциллограммы.

Более подробно о создании и использовании осциллограмм в ВП см. в Главе 5 *Создание типовых измерительных приложений* в Руководстве по программированию.

*кководстве по измерениям в LabVIEW (LabVIEW Measurements Manual).*

## Функции управления приложениями

Используйте функции управления приложениями (Application Control) для программного управления ВП и LabVIEW-приложениями на Вашем локальном компьютере или через компьютерную сеть. Более подробно об использовании функций управления приложениями см. в Главе 17 *Программное управление ВП*.

## Дополнительные функции

Используйте дополнительные (Advanced) функции для вызова программного кода из библиотек, таких как динамически связываемые библиотеки (dynamic link libraries – DLL), чтобы манипулировать данными LabVIEW для использования их в других приложениях, для создания и манипуляции регистрационными ключами Windows и для вызова сегментов кода на текстовых языках программирования. Более подробно об использовании дополнительных функций см. в руководстве *Using External Code in LabVIEW*.

## Добавление терминалов у функций

Вы можете изменять количество терминалов у некоторых функций. Например, для построения массива из 10 элементов, Вы можете увеличить до 10 количество терминалов у функции Build Array.

Вы можете добавлять терминалы для растягиваемых (expandable) ВП и функций, используя инструмент Positioning для оттягивания нижней или верхней границы функции вниз или вверх, соответственно. Вы можете также использовать инструмент Positioning для удаления терминалов у растягиваемых ВП или функций, но Вы не сможете удалить уже подсоединеные терминалы.

Вы можете также добавлять или удалять терминалы, щелкнув правой кнопкой по терминалам функции и выбрав **Add Input** (добавить вход), **Add Output** (добавить выход), **Remove Input** (удалить вход) или **Remove Output** (удалить выход) из контекстного меню. Для разных функций Вы можете добавлять терминалы для входов, выходов или для элементов управления ссылочными номерами (ref-num). Пункты контекстного меню **Add Input** и **Add Output** добавляют терминалы сразу же после того, как Вы щелкните правой

кнопкой по терминалу. Пункты контекстного меню **Remove Input** и **Remove Output** удаляют терминалы после того, как Вы щелкните правой кнопкой по терминалу. Если для удаления подсоединеных терминалов Вы используете пункты контекстного меню, то LabVIEW удаляет терминалы и разрывает проводники.

## Использование проводников для связи объектов блок-диаграммы

---

Данные передаются через объекты блок-диаграммы посредством проводников (wires). Каждый проводник имеет единственный источник данных, но Вы можете подсоединять его ко многим ВП или функциям, которые читают эти данные. Вы должны подсоединить все обязательные (required) терминалы блок-диаграммы. В противном случае, ВП оказывается поврежденным (broken) и не будет запускаться. Отобразите окно **Context Help**, чтобы увидеть, какие терминалы узлов блок-диаграммы являются обязательными. Обязательные терминалы помечены в окне **Context Help** метками в жирном шрифте. Более подробно об обязательных терминалах см. в разделе *Установка обязательных, рекомендуемых и необязательных входов и выходов* в Главе 7 *Создание ВП и ВПП*. Более подробно о поврежденных ВП см. в разделе *Исправление поврежденных виртуальных приборов* в Главе 6 *Запуск и отладка ВП*.

Проводники имеют различные цвета, стили и толщину, в зависимости от их типа данных. Поврежденные проводники имеют вид пунктирных черных линий с красным символом **✗** посередине. Стрелки с боков красного символа **✗** указывают направление потока данных, а цвет стрелок указывает тип данных, проходящих по проводнику. Более подробно о типе данных проводников см. в *LabVIEW Quick Reference Card* (Карта быстрых ссылок LabVIEW).

Когда Вы перемещаете инструмент Wiring над узлом ВП или функции, около не присоединенных терминалов появляются обрывки проводников. Они показывают тип данных каждого терминала. Появляются также краткие подсказки (tip strip) с именами терминалов. После того, как Вы подсоедините терминал, обрывок проводника для такого терминала перестанет появляться при перемещении инструмента Wiring над его узлом.

Сегмент (segment) проводника это одна горизонтальная или вертикальная часть проводника. Изгиб (bend) проводника – это место,

где сливаются два сегмента. Точка, в которой два или более сегментов сливаются, есть соединение (junction). Одна ветвь (branch) проводника содержит все сегменты проводника от соединения до соединения, от терминала до соединения или от терминала до терминала, если между ними нет соединений. На Figure 5-1 показаны сегмент, изгиб и соединение проводника.

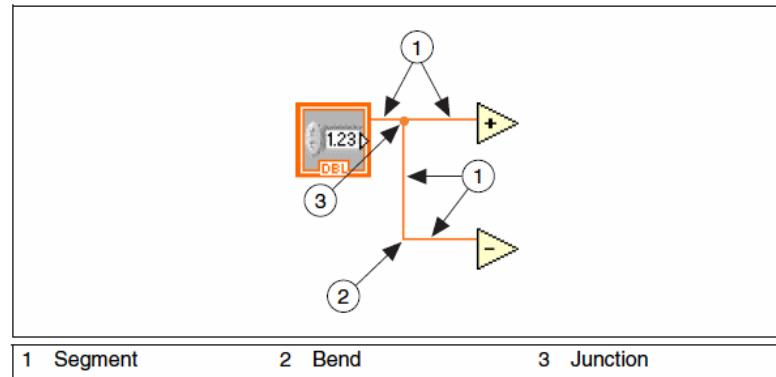


Figure 5-1. Сегмент (Segment), изгиб (Bend) и соединение (Junction) проводника

Во время соединения терминалов может быть сделан однократный изгиб проводника на угол 90 градусов перемещением курсора в вертикальном или горизонтальном направлении. Чтобы сделать несколько изгибов на проводнике, щелкните клавишей мыши, чтобы зафиксировать сегмент проводника и затем перемещайте курсор в новом направлении. Вы можете многократно фиксировать сегменты проводника и перемещаться в новых направлениях. Чтобы отменить последнюю точку, в которой Вы зафиксировали сегмент проводника, нажмите клавишу **<Shift>** и щелкните еще раз на блок-диаграмме.

Когда Вы пересекаете проводники, появляется небольшой промежуток в проводнике, который Вы рисовали первым, чтобы показать, что первый проводник размещается под вторым проводником.



**Предупреждение.** Пересекающиеся проводники создают беспорядок на блок-диаграмме и делают ее трудной для отладки.

Более подробно о рекомендациях и способах соединения проводников см. в разделе *Wiring Techniques* в Главе *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## Автоматическое соединение объектов

LabVIEW автоматически соединяет объекты, когда Вы их только помещаете на блок-диаграмму. Вы также можете автоматически соединять объекты, которые уже помещены на блок-диаграмму. LabVIEW соединяет терминалы, которые лучше подходят, и оставляет несоединенными терминалы, которые не подходят друг к другу.

Если Вы переместите селектированный объект близко к другому объекту на блок-диаграмме, то LabVIEW начертит временные проводники, чтобы показать Вам правильные соединения. Когда Вы отпускаете кнопку мыши, чтобы поместить объект на блок-диаграмму, LabVIEW автоматически подсоединит проводники.

Включение/выключение автоматического соединения проводников осуществляется нажатием клавиши пробела при перемещении объекта с помощью инструмента **Positioning**.

По умолчанию автоматическое соединение активно, когда Вы выбираете объект из палитры **Function**, или, когда Вы копируете объект уже имеющийся на блок-диаграмме, нажимая клавишу **<Ctrl>** и перемещая этот объект. Автоматическое соединение по умолчанию отключено, когда Вы используете инструмент **Positioning** для перемещения уже имеющегося на блок-диаграмме объекта.

Вы можете отключить автоматическое соединение, выбирая **Tools»Options** и затем **Block Diagram** из выпадающего меню. В диалоговом окне нужно снять птичку с опции **Enable auto wiring**.

## Соединение объектов вручную

Используйте инструмент **Wiring** для соединения вручную терминалов одного узла блок-диаграммы с терминалами другого узла блок-диаграммы. Курсорной точкой этого инструмента является конец распущенной катушки с проводом. Когда Вы перемещаете инструмент **Wiring** над терминалом, этот терминал мерцает. При этом появляется также подсказка с указанием имени терминала. Если подключение к терминалу будет давать поврежденный проводник

(broken wire), то курсор перестанет иметь вид катушки и превратится в текстовое предупреждение. Вы можете создавать поврежденные проводники, но их нужно исправить прежде, чем Вы сможете запускать ВП. Более подробно об исправлении поврежденных проводников см. в разделе *Исправление поврежденных проводников* в этой Главе.

Для точного определения того, как правильно подсоединять проводники, пользуйтесь окном контекстной помощи **Context Help**. Когда Вы перемещаете курсор над ВП или функцией, в окне **Context Help** появляется перечень всех терминалов ВП или функции. Окно **Context Help** не показывает терминалы для растягиваемых (expandable) ВП и функций, таких, например, как функция Build Array. Чтобы отобразить необязательные терминалы соединительной панели, щелкните кнопку **Show Optional Terminals and Full Path** в окне **Context Help**. Более подробно о необязательных терминалах см. в разделе *Установка обязательных, рекомендуемых и необязательные входов и выходов* в Главе 7 *Создание ВП и ВПП*.

## Прокладка проводников

LabVIEW автоматически прокладывает маршрут для проводника в процессе его подсоединения, обходя при этом существующие объекты на блок-диаграмме, такие как циклы и структуры. LabVIEW прокладывает проводники таким образом, чтобы сократить число изгибов на проводнике. Когда это возможно, автоматически прокладываемые проводники от терминалов элементов управления выходят с правой стороны от терминала, а автоматически прокладываемые проводники к терминалам индикатора входят с левой стороны терминала.

Чтобы произвести автоматическую прокладку уже существующего проводника, щелкните правой кнопкой и выберите **Clean Up** (упорядочить) из контекстного меню.

Для временного отключения автоматической прокладки и включения ручной прокладки после того, как Вы уже начали соединение, нажмите клавишу **<A>**. Для возобновления режима автоматической прокладки проводника, нажмите эту же клавишу **<A>** еще раз. После того, как Вы завершите соединение, LabVIEW вновь переходит в режим автоматической прокладки. Вы можете также временно отключить автоматическую прокладку после того, как Вы щелкните, чтобы начать или установить проводник, путем удерживания

нажатой клавиши мыши, пока Вы не подсоедините к другому терминалу или не установите точку и затем отпустите кнопку мыши. После отпускания кнопки мыши LabVIEW восстановит режим автоматической прокладки.

Вы можете отключить автоматическую прокладку маршрутов проводников для всех новых проводников, выбирая **Tools»Options** и затем **Block Diagram** из спадающего меню.

Если Вы отключили автоматическую прокладку проводников, то Вы сможете соединять терминалы вертикально или горизонтально в зависимости от направления, в котором Вы сделаете первое движение инструментом **Wiring**. Проводники подсоединяются к центру терминала, независимо от того, где Вы щелкните терминал. После того, как Вы щелкните терминал, нажмите пробел для переключения между горизонтальным и вертикальным направлением.

Вы можете также нажимать клавишу пробела для переключения между горизонтальным и вертикальным направлением, если автоматическая прокладка включена. Если LabVIEW находит маршрут для прокладки в новом направлении, этот проводник переключится на это направление.

## **Селектирование проводников**

Селектирование проводников осуществляется с помощью однократного (single-click), двойного (double-click) или тройного (triple-click) щелчка с помощью инструмента **Positioning**. Однократный щелчок проводника селектирует один сегмент проводника. Двойной щелчок проводника селектирует ветвь. Тройной щелчок проводника селектирует весь проводник.

## **Исправление поврежденных проводников**

Поврежденные проводники имеют вид пунктирных черных линий с красным символом  посередине. Поврежденные проводники возникают по различным причинам, например, когда Вы пытаетесь соединить два объекта с несовместимыми типами данных. Подвигайте инструментом **Wiring** над поврежденным проводником, чтобы отобразить подсказку с причиной повреждения проводника. Эта информация появляется также в окне **Context Help**, когда Вы перемещаете инструмент **Wiring** над поврежденным проводником. Щелкните правой кнопкой и выберите **List Errors** (выдать список

ошибок) из контекстного меню, чтобы отобразить окно **Errors list**. Щелкните кнопку **Help** для более подробной информации о причине повреждения проводника.

Для удаления конкретного поврежденного проводника сделайте по нему тройной щелчок инструментом **Positioning** и затем нажмите клавишу **<Delete>**. Вы можете также щелкнуть проводник правой кнопкой и выбрать из контекстного меню такие опции как **Delete Wire Branch** (удалить ветвь проводника), **Create Wire Branch** (создать ветвь проводника), **Remove Loose Ends** (удалить потерянные концы), **Clean Up Wire** (упорядочить проводник), **Change to Control** (заменить на элемент управления), **Change to Indicator** (заменить на индикатор), **Enable Indexing at Source** (разрешить индексацию источника) и **Disable Indexing at Source** (запретить индексацию источника). Эти опции изменяются в зависимости от причины повреждения проводника.

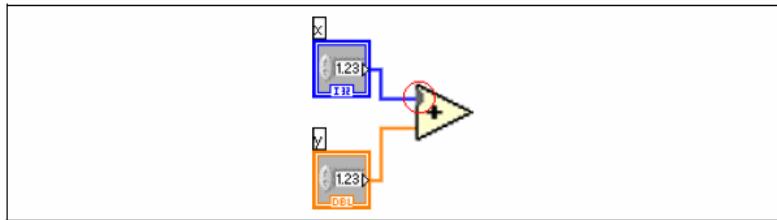
Вы можете удалить все поврежденные проводники, выбирая **Edit»Remove Broken Wires** или нажимая клавиши **<Ctrl-B>**.



**Предупреждение.** Будьте осторожны, удаляя все поврежденные проводники. Иногда проводники выглядят поврежденными из-за того, что Вы еще не завершили соединения на блок-диаграмме.

### Точки принудительного преобразования

Точки принудительного преобразования (coercion dots) появляются на блок-диаграмме, чтобы предупредить Вас о том, что Вы соединяете вместе два различных числовых типа данных. Точка означает, что LabVIEW конвертирует величину, поступающую в данный узел, к другому представлению (representation). Например, функция **Add** (сложение) ожидает на обоих своих входах величины удвоенной точности с плавающей точкой. Вы можете изменить один из входов к целому типу, и тогда на функции **Add** появится точка принудительного преобразования типа, как это показано на Figure 5-2.



**Figure 5-2.** Точка принудительного преобразования (Coercion Dot)

Точка принудительного преобразования помещается на границе терминала, для индикации того, что имело место автоматическое числовое преобразование. Поскольку ВП и функции могут иметь много терминалов, точка принудительного преобразования может появляться внутри иконки, если Вы соединяете один терминал с другим терминалом.

Точки принудительного преобразования появляются также на терминалах, когда Вы подключаете любой тип данных к терминалу вариантового (variant) типа, за исключением случая, когда соединяются два терминала вариантового типа. Более подробно о вариантовом типе данных см. разделе *Обработка вариантовых данных* в данной Главе.

Точки принудительного преобразования вынуждают ВП использовать больше памяти и снижают быстродействие. Страйтесь быть последовательными при выборе типов данных в ваших ВП.

## Полиморфные ВП и функции

Полиморфные ВП и функции могут подстраиваться к входным данным различного типа. Большинство структур LabVIEW являются полиморфными, таковыми являются некоторые ВП и функции.

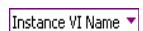
### Полиморфные ВП

Полиморфные ВП допускают различные типы данных хотя бы для одного входного или выходного терминала. По сути полиморфный ВП есть набор ВПП с одной и той же конфигурацией соединительных панелей. Каждый ВПП является представителем (instance) полиморфного ВП.

Например, ВП Read Key (прочесть клавишу) является полиморфным. Его терминал **default value** может принимать данные следующих типов: булевые, числовые двойной точности с плавающей точкой, числовые 32-битовые знаковые целые, путевые, строковые и числовые 32-битовые беззнаковые целые.

Для большинства полиморфных ВП типы данных, которые Вы подсоединяете к их входам, определяют конкретного представителя (instance) для использования. Если полиморфный ВП не содержит ВПП, совместимого с таким типом данных, появится поврежденный проводник. Если типы данных, которые Вы подсоединяете к входам полиморфного ВП, не определяют никакого представителя для использования, Вы можете выбрать такого представителя вручную. Если Вы вручную выбираете представителя полиморфного ВП, то такой ВП перестает вести себя как полиморфный ВП, поскольку он теперь может принимать и возвращать только те типы данных, которые присущи выбранному Вами представителю.

Чтобы выбрать конкретного представителя вручную, щелкните правой кнопкой полиморфный ВП, выберите **Select Type** из контекстного меню и затем выберите конкретного представителя для использования. Вы также можете использовать инструмент Operating,

 чтобы щелкнуть селектор полиморфного ВП, показанный слева, и выбрать представителя из выпадающего меню. Щелкните правой кнопкой полиморфный ВП на блок-диаграмме и выберите **Visible Items»Polymorphic VI Selector** из контекстного меню, чтобы отобразить такой селектор. Для изменения полиморфного ВП так, чтобы он вновь мог принимать все поддерживаемые типы данных, щелкните такой полиморфный ВП правой кнопкой и выберите **Select Type»Automatic** из контекстного меню или воспользуйтесь инструментом Operating, чтобы щелкнуть селектор этого полиморфного ВП и выбрать **Automatic** из выпадающего меню.

### Построение полиморфных ВП

Создавайте полиморфные ВП в тех случаях, когда требуется выполнять однотипные операции над различными типами данных.



**Примечание.** Вы можете создавать и редактировать полиморфные ВП только в LabVIEW Professional Development System.

Например, Вы хотите выполнять одну и ту же математическую операцию над числом однократной точности с плавающей точкой, над массивом чисел или над осциллограммой. Вы можете создать три отдельных ВП – Compute Number, Compute Array и Compute Waveform. Когда потребуется выполнить операцию, Вы помещаете на блок-диаграмму один из этих ВП, в зависимости от типа данных, который используется на входе.

Вместо того, чтобы вручную размещать версию ВП на блок-диаграмму, Вы можете создать и использовать единственный полиморфный ВП. Полиморфный ВП Compute включает трех представителей ВП, как показано на Figure 5-3.

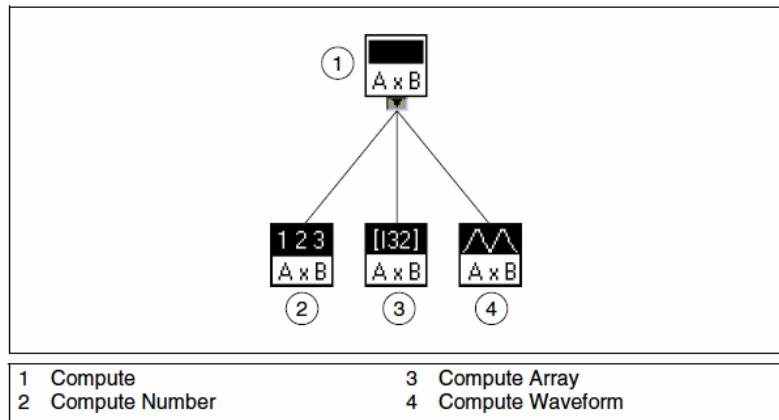


Figure 5-3. Пример полиморфного ВП

ВП Compute статически подключает правильного представителя ВП, основываясь на типе данных, которые подсоединенны к ВП Compute на блок-диаграмме, как показано на Figure 5-4.

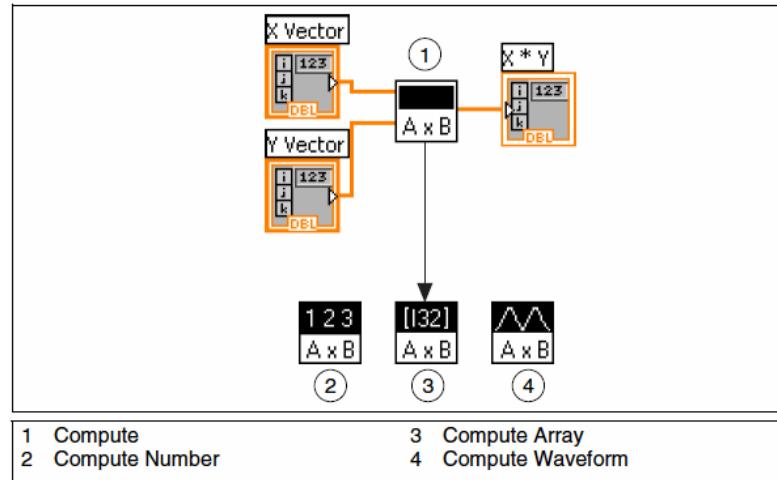


Figure 5-4. Полиморфный ВП,зывающий ВПП

Полиморфные ВП отличаются от обычных ВП тем, что они не имеют блок-диаграммы или лицевой панели.

При создании полиморфных ВП обратите внимание на следующие моменты:

- Все ВП, которые входят в полиморфный ВП, должны иметь одинаковую конфигурацию (pattern) соединительных панелей, поскольку все они должны совпадать с соединительной панелью итогового полиморфного ВП.
- Входы и выходы на соединительной панели каждого представителя ВП должны соответствовать входам и выходам на соединительной панели полиморфного ВП.
- ВП, которые Вы используете для построения полиморфного ВП, не должны состоять из одних и тех же ВПП и функций.
- Каждая из лицевых панелей таких ВП не обязана иметь одинаковое число объектов. Однако каждая лицевая панель должна иметь, по крайней мере, столько же элементов управления и индикаторов, сколько образуют соединительную панель полиморфного ВП.
- Вы можете создать иконку для полиморфного ВП.

- Нельзя использовать полиморфные ВП в составе других полиморфных ВП.

Когда Вы генерируете полную документацию для ВП, содержащего полиморфные ВПП, то и полиморфные ВП и их представители появляются в разделе документации «перечень ВПП» («list of subVI»).

## Полиморфные функции

Функции являются полиморфными в различной степени – ни один, некоторые или все их входы могут быть полиморфными. Некоторые входы функций допускают числовые или булевые значения. Некоторые допускают числа или строки. Некоторые допускают не только скалярные числа, но также и массивы чисел, кластеры чисел, массивы кластеров чисел и т.д. Некоторые допускают только одномерные массивы, хотя элементы массивов могут быть любого типа. Некоторые функции допускают все типы данных, включая комплексные числа. Более подробно о полиморфных функциях см. в Приложении В *Полиморфные функции*.

## Экспресс ВП

---

Используйте экспресс ВП (Express VI) для реализации типовых измерительных задач. Экспресс ВП – это функциональный узел, который требует минимума соединений, поскольку он конфигурируется посредством диалоговых окон. Входы и выходы экспресс ВП зависят от того, как Вы его настроите. На блок-диаграмме экспресс ВП имеет вид растягиваемого узла с иконкой, окруженной голубым фоном.

Более подробно о растягиваемых узлах см. в разделе *Отображение ВПП и экспресс ВП в виде иконок либо в виде расширяемых узлов* в Главе 7 *Создание ВП и ВПП*. Более подробно об экспресс ВП см. в руководстве *Getting Started with LabVIEW*.

## Создание ВПП из экспресс ВП

Можно создать ВПП из сконфигурированного экспресс ВП. Например, Вы можете захотеть сохранить сконфигурированный экспресс ВП Write LabVIEW Measurement File (записать в измерительный файл) для использования его в качестве ВПП в других ВП, которые Вы создаете, вместо того, чтобы каждый раз перенастраивать такой экспресс ВП. Чтобы создать ВПП из экспресс ВП, щелкните

этот экспресс ВП и выберите **Open Front Panel** из контекстного меню.

В процессе создания ВПП из экспресс ВП появляется лицевая панель ВПП. Затем можно редактировать ВП и потом сохранить его. Чтобы сохранились значения, введенные в элементы управления, выберите **Operate»Make Current Value Default** или щелкните правой кнопкой каждый элемент управления и выберите из контекстного меню **Make Current Value Default**. Новый ВПП появится как растягиваемый узел, замещающий экспресс ВП на блок-диаграмме.

После того, как Вы создадите ВП из экспресс ВП, его уже нельзя будет конвертировать обратно в экспресс ВП.

## Динамический тип данных



Динамический тип данных отображается в виде терминала темно-синего цвета, показанного слева. Многие экспресс ВП принимают и/или выдают данные динамического типа. Вы можете подсоединять данные динамического типа к любому индикатору или входу, который принимает числовые, булевые данные или данные типа осциллограммы. Подсоединяйте данные динамического типа к индикатору, который может наилучшим образом отобразить такие данные. Таким индикатором может быть график, диаграмма или числовой индикатор.

Динамический тип данных предназначен для использования в экспресс ВП. Большинство остальных ВП и функций, поставляемых вместе с LabVIEW, не допускают такой тип данных. Чтобы использовать встроенные ВП и функции для анализа и обработки данных динамического типа, такие данные необходимо конвертировать. Более подробно о конвертировании из динамического типа см. в разделе *Конвертирование из динамических данных* в настоящей Главе.

В дополнение к данным, связанным с самим сигналом, динамический тип данных включает атрибуты, которые представляют такую информацию о сигнале, как имя сигнала, или дата и время его ввода. Атрибуты определяют, как сигнал будет отображаться на графике или диаграмме. Например, если Вы используете экспресс ВП DAQ Assistant (помощник ввода/вывода) для ввода сигнала и отображения его на графике, то имя сигнала появляется в виде надписи (plot legend) на графике, а шкала X подстраивается так, чтобы

отобразить временную информацию, связанную с сигналом, в относительном или абсолютном времени, основываясь на атрибутах сигнала. Если же Вы используете экспресс ВП Spectral Measurements (спектральные измерения) для анализа сигнала с помощью алгоритма БПФ и отображения результирующих величин на графике, то шкала X автоматически подстраивается для отображения графика сигнала в частотной области, основываясь на атрибутах сигнала. Щелкните правой кнопкой выходной терминал динамического типа ВП или функции на блок-диаграмме и выберите **Create»Graph Indicator** для отображения этих данных на графике или выберите **Create»Numeric Indicator** для отображения данных в числовом форме.

В Табл. 5-2 приведен перечень индикаторов, которые принимают динамический тип данных и описано, как индикаторы обрабатывают такие данные.

**Табл. 5-2. Индикаторы динамического типа данных**

Данные в динамическом типе данных	Индикатор	Результат
Одно числовое значение	График	Рисует график одной величины, включая временную метку и атрибуты
Один канал		Рисует график всей осциллограммы, включая временную метку и атрибуты
Много каналов		Рисует график всех данных, включая временную метку и атрибуты
Одно числовое значение	Числовой индикатор	Отображает одно числовое значение.
Один канал		Отображает последнее значение данных в этом канале
Много каналов		Отображает последнее значение данных первого канала
Одно числовое значение	Булев индикатор	Отображает значение TRUE, если значение числовой величины больше или равно 0.5
Один канал		Отображает значение TRUE, если последнее значение данных в канале больше или равно 0.5

Много каналов		Отображает значение TRUE, если последнее значение данных в первом канале больше или равно 0.5
---------------	--	---

### Конвертирование из динамических данных

При совместном использовании с другими ВП и функциями, используйте экспресс ВП Convert from Dynamic Data для конвертирования данных динамического типа к данным числового типа, к данным типа осциллограмма (waveform) и к данным типа массив (array). Когда Вы помещаете экспресс ВП Convert from Dynamic Data на блок-диаграмму, появляется диалоговое окно **Configure Convert from Dynamic Data**. В нем отображаются опции, которые позволяют Вам указать ваши пожелания по форматированию данных, которые возвращает экспресс ВП Convert from Dynamic Data.

Например, если Вы вводите от устройства ввода синусоидальный сигнал, выберите опцию **Single Waveform** в диалоговом окне **Configure Convert from Dynamic Data**. Соедините выход **Waveform** экспресс ВП Convert from Dynamic Data с функцией или ВП, которые принимают тип данных waveform. Если Вы вводите набор значений температуры от различных каналов с помощьюDAQ-устройства, выберите опции **Most recent values from each channel** и **Floating point numbers (double)**. Затем соедините выход **Array** экспресс ВП Convert from Dynamic Data с функцией или ВП, которые принимают числовой массив в качестве входа.

Когда Вы соединяете данные динамического типа с массивом индикаторов, LabVIEW автоматически помещает на блок-диаграмму экспресс ВП Convert from Dynamic Data. Сделайте двойной щелчок по экспресс ВП Convert from Dynamic Data, чтобы открыть диалоговое окно **Configure Convert from Dynamic Data** и установить желаемый способ отображения данных в массиве.

### Конвертирование в динамические данные

При использовании различных экспресс ВП нужно переходит к динамическому типу данных. Для конвертирования числовых и булевых данных, а также данных типа осциллограмма или массив к динамическому типу данных используйте экспресс ВП Convert to Dynamic Data. Когда Вы помещаете экспресс ВП Convert to Dynamic Data на блок-диаграмму, появляется диалоговое окно **Configure**

**Convert to Dynamic Data.** Используйте это диалоговое окно для выбора способа конвертирования к динамическому типу данных.

Например, если Вы вводите синусоидальный сигнал, используя ВП Analog Input Traditional NI-DAQ, и хотите использовать для анализа сигнала экспресс ВП Signal Analysis, выберите на диалоговом окне **Configure Convert to Dynamic Data** опцию Single Waveform. Затем соедините выход **Dynamic Data Type** с экспресс ВП, который принимает данные динамического типа на входе.

## Обработка вариантовых данных

---

Вариантные данные (Variant data) не соответствуют никакому специфическому типу данных и могут содержать атрибуты. LabVIEW представляет вариантные данные с помощью различных типов данных. Вариантный тип данных отличается от других типов данных, поскольку он хранит имя элемента управления или индикатора, информацию о типе данных, из которого он конвертирован, и собственно данные. Это позволяет LabVIEW корректно конвертировать данные варианного типа к любому желаемому типу данных. Например, если Вы конвертируете строковые данные к данным варианного типа, то вариантный тип данных будет содержать текст и показывает, что этот текст является строкой.

Для обработки вариантовых данных используйте вариантные функции (Variant functions). Вы можете конвертировать любые типы данных LabVIEW к варианту типу, чтобы использовать вариантные данные в других ВП и функциях. Некоторые полиморфные функции возвращают вариантный тип данных.

Используйте вариантный тип данных, когда важно манипулировать данными независимо от их типа данных, как, например, при передаче или хранении данных; чтении и/или записи в неизвестные устройства; хранении данных в стеке, в очереди или в накопителе уведомлений (notifier) или при выполнении операций над множеством разнородных элементов управления.

Вы также можете использовать функцию Flatten to String (преобразование к выровненной строке) для конвертирования любого типа данных к строковому типу данных, чтобы представлять данные независимо от их типа. Преобразование данных к выровненным (flatten) строкам является обычным при использовании для передачи

данных протокола TCP/IP, поскольку этот протокол воспринимает только строки.

Однако, использование выровненных данных имеет ограничения, поскольку LabVIEW не может принудительно преобразовать выровненные данные, когда исходный тип данных не совпадает с типом, к которому Вы хотите конвертировать. Кроме того, попытка обратного конвертирования выровненного целого к числу расширенной точности с плавающей точкой приведет к ошибке. Более подробно о выравнивании данных и обратного их конвертирования см. в разделе *Flatten Data* руководства (Application Note) *LabVIEW Data Storage*.

Другим преимуществом использования вариантного типа данных является его способность сохранять атрибуты данных. Атрибуты – это дополнительная информация о данных, которую вариантный тип данных сохраняет. Например, если Вам нужно знать время, когда порция данных была создана, Вы можете сохранить такие данные в виде вариантных данных и добавить атрибут, называемый **Time**, чтобы сохранить время в виде строки. Сами атрибутивные данные могут иметь любой тип. Используйте вариантные атрибуты, когда Вы хотите сортировать данные по отдельным атрибутам, идентифицировать устройство или приложение, которое генерирует данные, или фильтровать данные по отдельным атрибутам.

## Числовые единицы и строгая проверка типов

---

Вы можете ассоциировать единицы измерения физических величин, такие как метр или километры в секунду, с любым числовым элементов управления или индикатором, которые имеют представление с плавающей точкой.

Единицы для элемента управления появляются в отдельной собственной метке, называемой меткой единиц (unit label). Чтобы отобразить метку единиц, щелкните правой кнопкой этот элемент управления и выберите из контекстного меню **Visible Items»Unit Label**. Щелкните правой кнопкой метку единиц и выберите из контекстного меню **Build Unit String**, чтобы ее редактировать.

Когда LabVIEW отображает метку единиц, Вы можете ввести единицу, используя стандартные аббревиатуры, такие как m для метра, ft для фута, s для секунд и т.п.

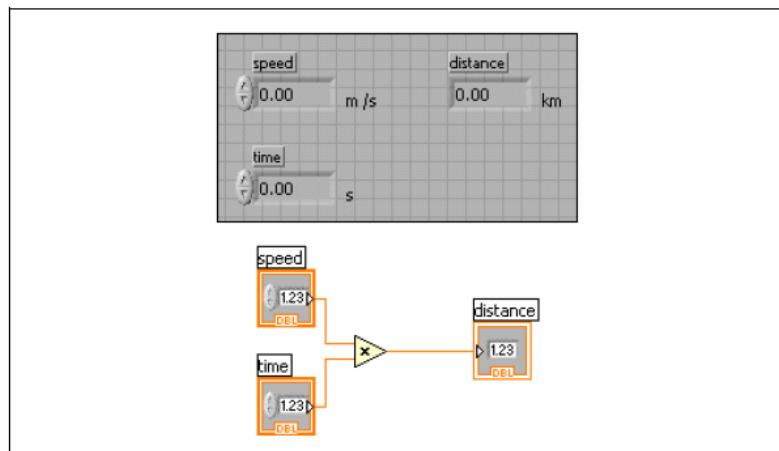


**Примечание.** Нельзя использовать единицы в формульном узле (Formula Node).

## Единицы и строгая проверка типов

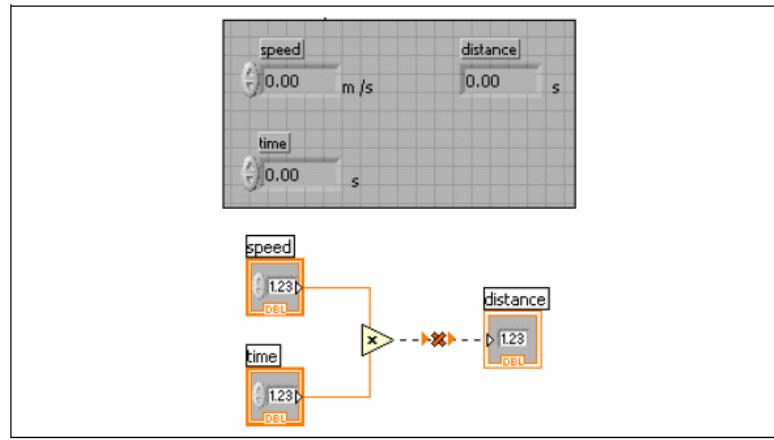
Если Вы ассоциируете единицы с объектом, то Вы можете соединять только объекты, имеющие совместимые единицы. LabVIEW использует строгую проверку типов (strict type checking), чтобы убедиться, что единицы совместимы. Если Вы соедините два объекта с несовместимыми единицами, LabVIEW вернет ошибку. Например, LabVIEW возвращает ошибку, если Вы соедините объект с типом единиц mile (миля) с объектом с типом единиц liter (литр), поскольку миля это единица расстояния, а литр – единица объема.

На Figure 5-5 показано соединение объектов с несовместимыми единицами. В этом случае LabVIEW автоматически масштабирует индикатор **distance** для отображения километров вместо метров, поскольку единицей для данного индикатора является километр.



**Figure 5-5.** Соединение объектов с совместимыми единицами

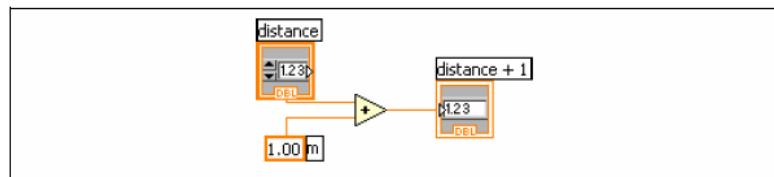
На Figure 5-6 имеет место ошибка, поскольку индикатор **distance** имеет в качестве типа единиц секунды. Для исправления этой ошибки нужно сменить тип единиц секунды на любую единицу длины, например километры, как показано на Figure 5-5.



**Figure 5-6.** Соединение объектов с несовместимыми единицами дает поврежденные проводники

Некоторые ВП и функции неоднозначны по отношению к единицам. Нельзя использовать такие ВП и функции с другими терминалами, которые имеют единицы. Например, функция Increment (приращение) неоднозначна по отношению к единицам. Если Вы используете единицы длины, функция Increment не может определить, что Вы хотите добавить один метр, один километр или один фут. Ввиду такой неоднозначности нельзя использовать функцию Increment и другие функции приращения или уменьшения (decrement) значений с данными, которые имеют ассоциированные единицы.

Чтобы исключить неоднозначность в этом примере, используйте числовую константу с походящей единицей и функцию Add (сложить) для создания своей собственной функции приращения на одну единицу, как показано на Figure 5-7.



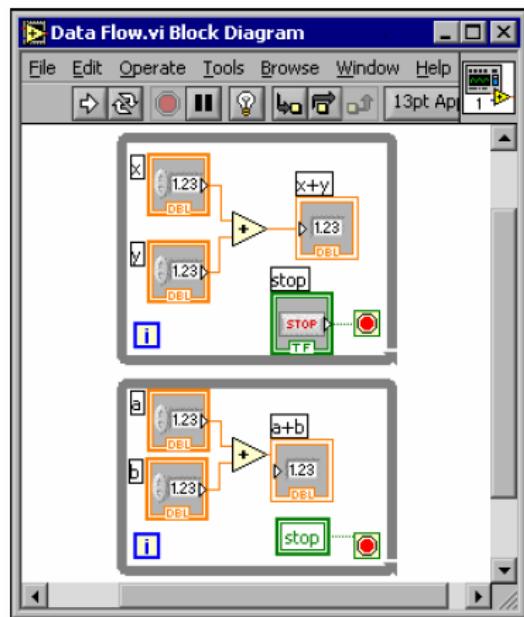
**Figure 5-7.** Создание собственной функции приращения на одну единицу

## Поток данных на блок-диаграмме

При запуске ВП LabVIEW следует модели потока данных (dataflow). Узел блок-диаграммы начинает исполнение, когда данные на всех его входах станут доступными. Когда узел завершает исполнение, данные поступают на его выходные терминалы и затем поступают к узлу, который является следующим на маршруте потока данных.

Visual Basic, C++, JAVA и большинство других текстовых языков программирования в процессе исполнения программы следуют модели потока управления (control flow). В модели потока управления порядок выполнения программы определяется последовательным порядком программных элементов.

Поскольку в LabVIEW порядок исполнения элементов блок-диаграммы определяет поток данных, а не последовательный порядок команд, можно создавать блок-диаграммы с параллельными операциями. Например, Вы можете одновременно запустить два цикла While Loops и отображать результаты их выполнения на лицевой панели.



LabVIEW является многозадачной и многопоточной (multithreaded) системой, которая выполняет одновременно несколько исполняемых потоков и виртуальных приборов. Более подробно об одновременном выполнении задач см. в руководстве (Application Note) *Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability*.

## **Зависимость по данным и искусственная зависимость по данным**

Порядок исполнение на основе модели потока управления задается последовательностью инструкций. Порядок исполнения на основе модели потока данных задается данными, то есть является зависимым по данным. Узлы, которые получают данные от других узлов, всегда выполняются только после того, как все эти предшествующие узлы завершат свое исполнение.

Узлы блок-диаграммы, которые не соединены проводниками, могут исполняться в любом порядке. Хотя в руководстве *LabVIEW Development Guidelines* рекомендуется использовать схему соединения блоков слева направо и сверху вниз, но узлы вовсе не обязательно исполняются в порядке слева направо и сверху вниз.

Вы можете использовать структуры последовательности (sequence) для управления порядком исполнения, когда естественная зависимость по данным отсутствует. Более подробно о структуре последовательности см. в разделе *Структуры последовательности* в Главе 8 *Циклы и структуры*. Для управления порядком исполнения можно также использовать параметры прохождения потока (flow-through parameters). Более подробно об этих параметрах см. в разделе *Потоковые параметры* в Главе 14 *Файловый ввод/вывод*.

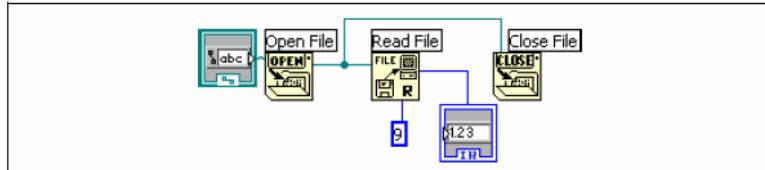
Вы также можете создать искусственную зависимость по данным, при которой принимающие узлы фактически не используют принятые данные для обработки. Вместо этого они используют поступившие данные только для запуска своего исполнения. В качестве примера использования искусственной зависимости по данным см. ВП *Timing Template (data dep)* в библиотеке `examples\general\ctructs.llb`.

### **Потеря зависимости по данным**

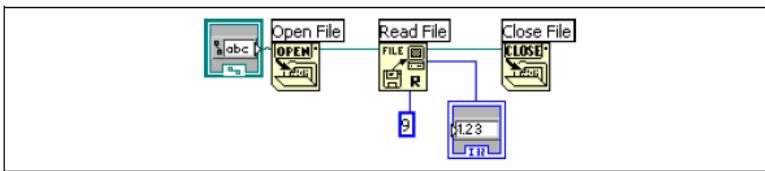
Не предполагайте порядок исполнения слева направо и сверху вниз, когда имеет место отсутствие зависимости по данным. Убе-

дитесь, что Вы явно определили последовательность событий, когда это необходимо, с помощью подсоединения нужного потока данных.

В следующем примере имеет место отсутствие зависимости между функцией Read File (читать файл) и функцией Close File (закрыть файл), поскольку функция Read File не соединена с функцией Close File. Этот пример, скорее всего, не будет работать, поскольку нет способа определить, какая из функций должна исполняться первой. Если функция Close File выполнится первой, то функция Read File не будет работать



На следующей блок-диаграмме установлена зависимость путем соединения выхода функции Read File с входом функции Close File. Здесь функция Close File уже не может запуститься, пока он не примет данные с выхода функции Read File.



### Поток данных и управление памятью

Выполнение в соответствии с моделью потока данных делает управление памятью более легким, чем при выполнении в соответствии с моделью потока управления. В LabVIEW Вы не можете размещать переменные или назначать им значения. Вместо этого Вы создаете блок-диаграмму с проводниками, которые представляют перемещения данных.

Виртуальные приборы и функции, которые генерируют данные, автоматически выделяют память под данные. Когда ВП или функция больше не использует данные, LabVIEW освобождает соответст-

вующую память. Когда Вы добавляете новые данные в массив или в строку, LabVIEW выделяет память, необходимую для размещения новых данных.

Поскольку LabVIEW автоматически управляет памятью, Вам нет необходимости управлять процессами выделения и освобождения памяти. Если ваш ВП работает с большими массивами данных, Вам следует понимать, когда имеет место выделение памяти. Понимание используемых принципов поможет Вам писать собственные ВП со значительно меньшими затратами памяти. А минимизация расхода памяти позволит увеличить быстродействие ваших ВП.

Более подробно о выделении памяти см. в руководстве (Application Note) *LabVIEW Performance and Memory Management*. Более подробно об оптимизации расхода памяти в процессе разработки см. в разделе *Memory and Speed Optimization* в Главе 6 *LabVIEW Style Guide* в руководстве *LabVIEW Development Guidelines*.

## Конструирование блок-диаграммы

---

Придерживайтесь следующих рекомендаций при конструировании блок-диаграмм:

- Используйте стиль слева направо и сверху вниз. Хотя положение элементов на блок-диаграмме не определяет порядок их выполнения, отсутствие соединений справа налево делает блок-диаграмму более организованной и легкой в понимании. Только соединения и управляющие структуры определяют порядок выполнения.
- Избегайте создания блок-диаграмм, занимающих более одного или двух экранов. Если блок-диаграмма становится большой и сложной, это создает трудности для ее понимания и отладки.
- Определите, не можете ли Вы повторно использовать некоторые компоненты блок-диаграммы в других ВП или объединить сегмент блок-диаграммы в качестве логически законченного компонента. Если да, то разделите блок-диаграмму на ВПП, которые выполняют отдельные задачи. Использование ВПП поможет Вам управлять изменениями и сделает отладку блок-диаграммы быстрой. Более подробно о ВПП см. в разделе *Виртуальные подприборы* в Главе 7 *Создание ВП и ВПП*.

- Используйте обработку ошибок ВП, функций и параметров для управления ошибками на блок-диаграмме. Более подробно об обработке ошибок см. в разделе *Проверка и обработка ошибок* в Главе 6 *Запуск и отладка ВП*.
- Улучшайте внешний вид блок-диаграммы за счет эффективных соединений. Плохая организация соединений может и не порождать ошибки, но может сделать блок-диаграмму трудной для чтения и отладки или создать видимость того, что ВП делает какие-то вещи, которых он на самом деле не делает.
- Избегайте соединений под границами структур или между перекрывающимися объектами, поскольку LabVIEW может скрыть некоторые сегменты результирующего соединения.
- Избегайте размещения объектов поверх проводников. Проводники соединяют только те объекты, по которым Вы щелкнули. Размещение терминалов или иконок поверх проводников создает иллюзию соединений, хотя на самом деле их нет.
- Используйте свободные метки для пояснения программного кода на блок-диаграмме.
- Увеличьте пространство между плотно или тесно сгруппированными объектами путем нажатия клавиши **<Ctrl>** и использования инструмента **Positioning**, чтобы щелкнуть им свободное пространство на блок-диаграмме. Удерживая нажатой клавишу, расширьте выделенную область до размеров, необходимых для вставки. Прямоугольник, помеченный пунктирной линией, определит пространство, которое будет вставлено. Для вставки этого пространства просто отпустите клавишу.

Более подробно о конструировании блок-диаграммы см. в разделе *Block-Diagram Style* в Главе 6 *LabVIEW Style Guide* в руководстве *LabVIEW Development Guidelines*.

## 6. Запуск и отладка виртуальных приборов

Чтобы запустить ВП Вы должны соединить все ВПП, функции и структуры с типами данных, которые ожидают терминалы. Иногда ВП выдают данные или запускаются способом, которого Вы не ожидали. Вы можете использовать LabVIEW для настройки способа запуска ВП, и для разрешения проблем с организацией блок-диаграммы или с прохождением данных через блок-диаграмму.

---

### Более подробно...

Более подробно относительно запуска и отладки ВП см. справочную систему *LabVIEW Help*.

---

### Запуск ВП

---

При запуске ВП выполняются операции, для которых Вы его создали. Вы можете запустить ВП если кнопка **Run** на панели инструментов имеет вид сплошной белой стрелки.



Белая непрерывная стрелка означает также, что Вы можете использовать ВП в качестве ВПП, если Вы создадите для него соединительную панель. В то время, когда ВП запущен, кнопка **Run** превращается в темную стрелку, показанную слева, что означает, что ВП запущен. Вы не сможете редактировать ВП в то время, когда он запущен.

ВП запускается, если Вы нажмете кнопки **Run** или **Run Continuously** или кнопки пошагового запуска на панели инструментов блок-диаграммы. Щелчок по кнопке **Run** запускает ВП однократно.

ВП останавливается, когда он завершает свой поток данных. Щелчок по кнопке **Run Continuously** запускает ВП непрерывно (много-кратно), до тех пор, пока Вы не остановите его вручную. Щелчок по кнопкам пошагового запуска запускает очередной шаг выполнения ВП. Более подробно об использовании кнопок пошагового запуска для отладки ВП см в разделе *Пошаговое выполнение* в настоящей Главе.



**Примечание.** Избегайте использования кнопки **Abort Execution**  для остановки ВП. Либо дождитесь завершения потока данных, либо создайте программный способ остановки ВП. Делая так, Вы всегда будете

иметь ВП в известном состоянии. Например, поместите на лицевую панель кнопку, с помощью которой можно остановить ВП.

## Настройка способа запуска ВП

Чтобы настроить способ запуска ВП выберите **File»VI Properties** и затем **Execution** из спадающего меню **Category**. Например, можно настроить так, чтобы ВП запускался немедленно после его открытия, или чтобы приостанавливался, когда происходит вызов ВП. Можно также настроить ВП так, чтобы он выполнялся с различным приоритетом. Например, если важно, чтобы ВП выполнялся без задержки на выполнение других операций, настройте ВП так, чтобы он запускался с наивысшим приоритетом. Более подробно о создании многопоточных ВП см. в руководстве (Application Note) *Using LabVIEW to Create Multithreaded VIs for Maximum Performance and Reliability*. Более подробно о настройке способа запуска ВП см. в Главе 16 *Конфигурирование ВП*.



**Примечание.** Некорректная настройка ВП для запуска с различными приоритетами может вызвать неожиданные эффекты. Для большинства ВП нет необходимости перенастраивать приоритеты запуска.

## Исправление поврежденных виртуальных приборов



Если ВП не запускается, он является поврежденным или неиспользованным. Кнопка **Run** часто имеет поврежденный вид, показанный слева, когда ВП, который Вы создаете или редактируете, содержит ошибки. Если она остается поврежденной даже после того, как Вы завершите соединение блок-диаграммы, то такой ВП является поврежденным, и он не сможет быть запущен.

### Поиск причин повреждения ВП

Щелкните поврежденную клавишу **Run** или выберите **Window»Show Error List**, чтобы определить причину повреждения ВП. В окне **Error List** отображается список всех ошибок. В разделе **VI List** приводится список имен всех ВП, находящихся в памяти и имеющих ошибки. В разделе **errors and warnings** приводится список ошибок и предупреждений для ВП, который Вы выбрали в разделе **VI List**. В разделе **Details** эти ошибки подробно описываются и, в некоторых случаях, даются рекомендации как их исправить или как найти дополнительную информацию об этом. Чтобы открыть

встроенный файл справки, который отображает список ошибок LabVIEW и их описания, нажмите кнопку **Help**.

Щелкните кнопку **Show Error** или сделайте двойной щелчок по описанию ошибки, чтобы отобразить соответствующую блок-диаграмму или лицевую панель и высветить объект, который вызвал ошибку.



На панели инструментов появляется кнопка **Warning** (предупреждение), показанная слева, если ВП имеет предупреждение, и Вы предварительно поставили галочку на опции **Show Warning** в окне **Error List**.

Чтобы настроить LabVIEW на отображение предупреждений в окне **Error List**, выберите **Tools»Options**, затем категорию **Debugging** (отладка) из выпадающего меню и поставьте галочку на опции **Show warnings in error box by default**.

Наличие предупреждений не препятствует запуску ВП. Они лишь помогают Вам избавиться от потенциальных проблем в ВП.

## Типичные причины повреждения ВП

В процессе редактирования ВП может оказаться поврежденным по следующим причинам:

- Блок-диаграмма содержит поврежденные проводники из-за несовпадения типов данных или свободные (не присоединенные) концы. Более подробно о соединении объектов блок-диаграммы см. в разделе *Использование проводников для связи объектов блок-диаграммы* в Главе 5 *Построение блок-диаграммы*.
- Обязательные терминалы блок-диаграммы не подсоединенены. Более подробно об обязательных терминалах см. в разделе *Установка обязательных, рекомендуемых и необязательных входов и выходов* в Главе 7 *Создание ВП и ВПП*.
- Поврежден ВПП, или Вы изменили его соединительную панель после размещения его иконки на блок-диаграмму вашего ВП. Более подробно о ВПП см. в разделе *Виртуальные подприборы* Главы 7 *Создание ВП и ВПП*.

## Технология отладки

---

Если ВП не поврежден, но Вы получаете неверные результаты, можно воспользоваться следующими способами поиска и устранения проблем с ВП или с потоком данных на блок-диаграмме:

- Подключите параметры **errort in** (ошибка на входе) и **errort out** (ошибка на выходе), имеющиеся у большинства встроенных ВП и функций. Эти параметры обнаруживают ошибки, появляющиеся в каждом узле, и показывают место их возникновения. Можно также использовать эти параметры в ВП, которые Вы создаете сами. Более подробно об использовании этих параметров см. в разделе *Обработка ошибок* в данной Главе.
- Чтобы избавиться от всех предупреждений, выберите **Windows»Show Error List** и поместите птичку на опцию **Show Warning**. При этом Вы сможете видеть все предупреждения для Вашего ВП. Выявите их причины и устраните.
- Сделайте тройной щелчок инструментом Operating по проводнику, чтобы высветить весь его маршрут и убедиться, что проводник соединен с нужными терминалами.
- Используйте окно **Context Help** для проверки значений по умолчанию для каждой функции и ВП на блок-диаграмме. ВП и функции получают значения по умолчанию, если рекомендованные или необязательные входы не подсоединенны. Например, булев вход может устанавливаться в состояние TRUE, если он не подключен.
- Используйте диалоговое окно **Find** для поиска ВПП, текста или других объектов для корректировки всего ВП.
- Выберите **Brows»Show VI Hierarchy** для выявления неподключенных ВПП. В отличие от не подсоединеных функций, не подсоединеные ВП не всегда порождают ошибки, если Вы не настроите его входы как обязательные. Если Вы ошибочно поместите неподключенный ВПП на блок-диаграмму, он выполнится вместе со всей блок-диаграммой. Следовательно, такой ВП может совершить непредусмотренные действия.
- Используйте подсвечивание в процессе выполнения, чтобы проследить прохождение данных через блок-диаграмму.

- Для просмотра каждого действия ВП выполните его в пошаговом режиме.
- Используйте инструмент *Probe*, для наблюдения за текущими значениями данных и проверки выходов *error out* у ВП и функций, особенно у тех, которые выполняют ввод/вывод.
- Используйте точки прерывания (breakpoints) для приостановки выполнения, после чего Вы можете продолжить пошаговое исполнение или вставить пробники (probes).
- Производите временную приостановку (suspend) процесса выполнения ВП, чтобы изменять значения элементов управления и индикаторов, чтобы управлять количеством повторных запусков или чтобы вернуться в начало выполнения ВП.
- Снабжайте комментариями сегмент блок-диаграммы, чтобы проверить, не работает ли этот ВП лучше без него.
- Проверьте, не являются ли данные, которые выдает одна из функций или ВПП неопределенными. Это часто случается с числами. Например, в одной точке ВП может иметь место деление на ноль, что дает результат *Inf* (бесконечность), тогда как последующие функции или ВПП будут ожидать число.
- Если ВП работает медленнее, чем ожидалось, убедитесь, что Вы не включили режим подсвечивания исполнения (highlighting execution) в ВПП. Кроме того, закройте лицевые панели и блок-диаграммы ВПП, когда Вы их не используете, поскольку открытые окна влияют на скорость выполнения.
- Проверьте представление (representation) элементов управления и индикаторов, чтобы увидеть, не происходит ли переполнения из-за конвертирования чисел с плавающей точкой в целые или целых чисел в меньшие целые. Например, Вы можете присоединить 16-битовое целое к функции, которая принимает только 8-битовое целое. Это заставляет функцию конвертировать 16-битовое целое к 8-битовому представлению, что потенциально может привести к потере данных.
- Проверьте, нет ли цикла *For Loops*, у которого по ошибке задано нулевое число итераций и который выдает пустые массивы.
- Убедитесь, что Вы правильно задали начальные значения сдвигающих регистров, даже если они предназначены только для сохранения данных от одной итерации цикла до следующей.

- Проверьте порядок элементов кластера в исходной и конечной точках. LabVIEW проверяет на соответствие типов данных и размера кластера во время редактирования, но несовпадение элементов одного и того же типа не обнаруживается.
- Проверьте порядок выполнения узлов.
- Проверьте, не содержит ли ВП скрытых ВПП. Вы можете неумышленно сделать ВПП скрытым, помещая поверх него другой узел или уменьшая размер структуры, оставляя при этом ВПП вне поля зрения.
- Сверьте перечень ВПП данного ВП с результатами, которые получите, выбирая **Browse This VI's SubVI** и **Browse Unopened SubVIs**, чтобы увидеть, нет ли лишних ВПП. Кроме того, откройте окно **Hierarchy**, чтобы посмотреть ВПП данного ВП. Чтобы избавиться от возможности некорректных результатов из-за наличия скрытых ВПП, делайте все входы ВПП обязательными.

## Подсвечивание выполнения



Для просмотра анимации выполнения блок-диаграммы щелкните кнопку **Highlight Execution**, показанную слева. Подсвечивание выполнения показывает перемещение данных на блок-диаграмме от одного узла к другому с помощью пузырька, который перемещается вдоль проводников. Для того, чтобы видеть, как данные перемещаются через ВП от узла к узлу, используйте подсвечивание выполнения совместно с пошаговым исполнением.



**Примечание.** Подсвечивание выполнения очень сильно замедляет работу ВП.

Если кластер **error out** сообщает об ошибке, то значение ошибки появляется в красной рамке рядом с выводом **error out**. Если ошибка отсутствует, то рядом с выводом **error out** появляется **OK** в зеленой рамке. Более подробно о кластерах ошибок см. в разделе *Кластеры ошибок* в этой Главе.

## Пошаговое выполнение



Применяйте пошаговое выполнение ВП, чтобы просмотреть каждое действие на блок-диаграмме в процессе ее исполнения. Кнопки пошагового исполнения, показанные слева, действуют только в пошаговом режиме. Войдите в пошаговый режим, щелкнув клавишу

**Step Over** (шаг через, перешагнуть) или **Step Into** (шаг внутрь, войти) на панели инструментов блок-диаграммы. Перемещайте курсор над кнопками **Step Over**, **Step Into** или **Step Out**, чтобы прочесть подсказки, которые описывают, каким будет следующий шаг, если Вы щелкните эту кнопку. Вы можете выполнить в пошаговом режиме весь ВП или запустить его затем в обычном режиме.



Если Вы осуществляете пошаговое выполнение при включенном подсвечивании, то на иконке ВПП, который в данный момент исполняется, появится значок выполнения (стрелка).

## Инструмент Probe (пробник)



Используйте инструмент Probe, показанный слева, чтобы при запуске ВП проверить текущие значения на проводниках. Используйте инструмент Probe, если Вы имеете сложную блок-диаграмму с последовательностью операций, каждая из которых может давать неверные данные. Используйте инструмент Probe совместно с подсвечиванием выполнения, пошаговым выполнением и установкой точек прерывания (breakpoints) для выявления имеются ли и в каком месте неверные данные. Если данные доступны, то пробник немедленно обновляется в течение одного шага или во время паузы в точке прерывания. Когда выполнение приостанавливается на каком-то узле в связи с пошаговым выполнением или из-за точки прерывания, Вы также можете установить пробник на проводник, который только что выполнился, чтобы просмотреть значение, которое через него прошло.

### Типы пробников

Вы можете проверять текущие значения на проводниках при запуске ВП, используя родовой пробник (Generic probe), используя для просмотра данных индикатор с палитры **Controls**, используя типовой пробник (supplied probe), используя настраиваемый типовой пробник или создавая новый пробник.

#### Родовой пробник

Используйте родовой пробник (generic probe) для просмотра данных, которые проходят через проводник. Чтобы воспользоваться родовым пробником щелкните правой кнопкой проводник и выберите из контекстного меню **Custom Probe»Generic**.

Родовой пробник только отображает данные. Нельзя конфигурировать реакцию пробника на данные.

LabVIEW устанавливает родовой пробник, когда Вы щелкаете правой кнопкой проводник и выбираете **Probe**, при условии, что Вы еще не специфицировали настраиваемый или типовой пробник для данного типа данных.

#### **Использование индикаторов для просмотра данных**

Вы также можете использовать индикаторы для просмотра проходящих через проводник данных. Например, если Вы смотрите числовые данные, то для просмотра данных внутри пробника можно использовать индикатор диаграмма (Chart). Щелкните правой кнопкой проводник, выберите **Custom Probe»Controls** контекстного меню и выберите индикатор, который Вы хотите использовать. Можно также щелкнуть иконку **Select a Control** на палитре **Control** и выбрать любой пользовательский элемент управления или определитель типа, сохраненный на компьютере или в совместном каталоге на сервере. LabVIEW обращается с определителями типов как с пользовательскими элементами управления, когда вы используете их для просмотра данных в пробниках.

Если тип данных индикатора, который Вы выбрали, не соответствует типу данных проводника, который Вы щелкнули правой кнопкой, то LabVIEW не поместит такой индикатор на этот проводник.

#### **Типовой пробник**

Типовые пробники (supplied probes) это ВП, которые отображают исчерпывающую информацию о данных, проходящих через проводник. Например, VI Refnum Probe возвращает имя ВП, путь, и ссылочный номер в шестнадцатеричной форме. Можно также использовать типовой пробник, чтобы реагировать на поток данных через проводник. Например, используйте пробник Еттог на кластере ошибок для получения статуса, кода, источника и описания данной ошибки и описания условия прерывания при возникновении ошибки.

Типовые пробники появляются в верхней части контекстного меню **Custom Probe**. Щелкните правой кнопкой проводник и выберите из контекстного меню **Custom Probe**, чтобы затем выбрать типовой пробник. В контекстном меню появятся только те пробники,

которые соответствуют типу данных проводника, который Вы щелкнули правой кнопкой.

В качестве примера использования типовых пробников см. ВП Using Supplied Probes из библиотеки examples\ general\ probes.llb.

### Настраиваемый пробник

Используйте мастер настраиваемого пробника (Custom Probe Wizard) для создания пробника, основываясь на уже существующем пробнике, или для создания нового пробника. Чтобы запустить мастер настраиваемого пробника, щелкните правой кнопкой проводник и выберите из контекстного меню **Custom Probe»New**. Создайте пробник, когда нужно иметь больше контроля над тем, как LabVIEW исследует данные, которые проходят через проводник. Когда Вы создаете новый пробник, тип данных пробника совпадает с типом данных проводника, который вы щелкнули правой кнопкой. Если Вы хотите изменить создаваемый пробник, Вы должны открыть его из директории, в которой Вы его сохранили.

Вы можете делать отладку настраиваемого пробника также как ВП. Однако пробник нельзя установить ни на своей собственной блок-диаграмме, ни на блок-диаграмме любого из его ВПП. При отладке блок-диаграммы пробников используйте родовой пробник.

После того, как Вы выбрали пробник из контекстного меню **Custom Probe**, его можно найти, используя кнопку **Select Control** палитры **Control**, или можно создать новый пробник с помощью мастера настраиваемого пробника. Этот пробник становится пробником по умолчанию для данного типа данных, и LabVIEW загрузит именно этот пробник, когда Вы щелкните правой кнопкой проводник и выберите **Probe** из контекстного меню. LabVIEW загрузит только те пробники, которые точно соответствуют типу данных проводника, который Вы щелкаете правой кнопкой. То есть, пробник двойной точности с плавающей точкой не может использоваться для проводника с 32-битовым целым, несмотря на то, что LabVIEW может конвертировать данные.



**Примечание.** Если Вы хотите, чтобы настраиваемый пробник стал пробником по умолчанию для некоторого типа данных, сохраните пробник в директории user.lib\\_probes\default. Не сохраняйте пробники

в директории `vi.lib\probes`, поскольку LabVIEW перезаписывает эти файлы при обновлении или при повторной инсталляции.

Более подробно о предосторожностях, которые нужно соблюдать при использовании и создании настраиваемых пробников, см. в справочной системе *LabVIEW Help* (раздел *Caveats and Recommendations when Using Custom Probes*).

## Точки прерывания



Используйте инструмент Breakpoint (точка прерывания), показанный слева, чтобы поместить точки прерывания на ВП, узлах и проводниках блок-диаграммы с целью приостановки выполнения в этих точках. Если Вы установили на проводнике точку прерывания, то выполнение будет приостановлено сразу после того, как данные пройдут через этот проводник. Размещайте точку прерывания на блок-диаграмму для приостановки выполнения только после того, как все узлы на блок-диаграмме выполняются.

Когда ВП приостанавливается в точке прерывания, LabVIEW переносит блок-диаграмму на передний план (делает ее видимой), при этом для выделения узлов, помеченных точкой прерывания, используется красная кайма, а для выделения проводников с точкой прерывания – красная точка. Когда Вы перемещаете курсор над существующими точками прерывания, черная область внутри курсора инструмента Breakpoint становится белой.

Когда во время исполнения достигается точка прерывания, ВП приостанавливается, и кнопка **Pause** становится красной. Вы можете выбрать следующие действия:

- Произвести одношаговый запуск, используя клавиши пошагового исполнения.
- Поставить пробники на проводники, чтобы проверить текущие значения.
- Изменить значения элементов управления лицевой панели.
- Щелкнуть кнопку **Pause**, чтобы продолжить исполнение до следующей точки прерывания или до завершения работы ВП.

LabVIEW сохраняет точки прерывания вместе с ВП, но они становятся активными только после запуска ВП. Можно просмотреть все точки прерывания, выбирая **Brows»Breakpoints**.

## Приостановка исполнения

Делайте приостановку выполнения (suspend execution) ВПП для изменения значений элементов управления и индикаторов, для контроля над количеством запусков ВПП перед возвратом в точку вызова или чтобы вернуться к началу выполнения ВПП. Вы можете устанавливать так, чтобы все вызовы ВПП были с приостановкой или чтобы приостанавливались только отдельные вызовы ВПП.

Чтобы сделать все вызовы ВПП с приостановкой, откройте ВПП и выберите **Operate»Suspend when Called**. Такой ВПП будет автоматически приостанавливаться, когда его будет вызывать другой ВП. Если Вы выберите этот же пункт меню при пошаговом выполнении, то ВПП не будет приостановлен немедленно. Этот ВПП будет приостановлен, только когда его вызовет другой ВП.

Чтобы приостановить конкретный вызов некоторого ВПП, щелкните правой кнопкой узел этого ВПП на блок-диаграмме и выберите из контекстного меню **SubVI Node Setup**. Поместите птичку на опцию **Suspend when called**, чтобы приостановить выполнение только этого представителя (instance) ВПП.

Окно **Hierarchy**, которое Вы отображаете, выбирая **Brows»Show VI Hierarchy**, показывает, находится ли ВП в состоянии приостановки исполнения (pause) или в состоянии приостановленного вызова (suspend). Пометка стрелкой означает, что ВП запущен как обычно или в пошаговом режиме. Пометка значком паузы означает, что ВП приостановлен (paused) или приостановлен его вызов (suspended). Зеленый значок паузы или пустота в белом или в черном означает, что ВП приостановлен во время его вызова. Красный значок паузы или сплошной значок в белом или черном означает, что ВП временно приостановлен. Изображение восклицательного знака означает, что приостановлен вызов ВП. ВП может быть одновременно приостановленным (paused) и с приостановленным вызовом (suspended).

## Определение текущих представителей ВПП

Когда Вы приостанавливаете ВП, спадающее меню **Call list** (список вызовов) на панели инструментов отображает цепочку вызовов от ВП верхнего уровня до текущего ВПП. Этот список отличается от такого же списка, который Вы видите, когда выбираете **Browse»This VI's Callers**. В нем отображены все вызовы ВП, независимо от того, являются ли они запущенными. Используйте меню **Call list** для определения текущего представителя ВПП, если на блок-диаграмме их несколько. Когда Вы выбираете ВП из меню **Call list**, его блок-диаграмма открывается и LabVIEW высвечивает текущего представителя данного ВПП.

## Комментирование сегментов блок-диаграммы

Вы можете запускать ВП с отключенным сегментом блок-диаграммы, подобно тому, как комментируется сегмент кода в текстовых языках программирования. Отключайте сегмент блок-диаграммы, чтобы узнать, не будет ли ВП работать без него лучше. Поместите сегмент, который Вы хотите отключить, внутрь структуры Case (Выбор) и используйте булевые константы для запуска обоих вариантов. Более подробно о структуре Case см. в разделе *Структуры варианта* в Главе 8 *Циклы и структуры*. Кроме того, Вы можете создать копию ВП и удалить этот сегмент из блок-диаграммы в этой копии. Удалите потом эту версию ВП, если Вы решите, что она не нужна.

## Отключение инструментов отладки

Вы можете отключить отладочные инструменты для уменьшения памяти и некоторого увеличения эффективности. Щелкните правой кнопкой соединительную панель и выберите **VI Properties**. В меню **Category** выберите **Execution** и уберите птичку с опции **Allow Debugging**.

## Неопределенные или неожиданные данные

Неопределенные данные, которыми являются `NaN` (not a number – не число) или `Inf` (infinity – бесконечность), делают недействительными все последующие операции. Операции с плавающей точкой возвращают следующие две символические величины, которые указывают на неправильные вычисления или бессмысленные результаты:

- `NaN` (not a number – не число) – представляет значение с плавающей точкой, которое вырабатывают неправильные операции, такие как взятие квадратного корня от отрицательного числа.
- `Inf` (infinity – бесконечность) – представляет значение с плавающей точкой, которое вырабатывают такие операции, как деление на ноль.

LabVIEW не проверяет условия переполнения и потери значимости над целыми числами. Переполнение и потеря значимости для чисел с плавающей точкой определяются в соответствии с IEEE 754, Standard for Binary Floating-Point Arithmetic (Стандарт для двоичной арифметики с плавающей точкой).

Операции с плавающей точкой надежно распространяют `NaN` и `Inf`. Когда Вы явно или неявно конвертируете `NaN` или `Inf` в целые или булевые значения, эти значения становятся бессмысленными. Например, деление 1 на ноль дает `Inf`. Конвертирование `Inf` в 16-битное целое дает значение 32767, которое выглядит как нормальное значение. Более подробно о конвертировании числовых значений см. в разделе *Преобразование числовых представлений* в Приложении В *Полиморфные функции*.

Перед конвертированием данных к целому типу, используйте инструмент Probe для проверки промежуточных значений на правильность. Проверьте на значение `NaN` путем подсоединения функции сравнения `Not A Number/Path/Refnum?` к значению, которое Вы подозреваете на неправильность.

## Данные по умолчанию в циклах

Циклы For Loop возвращают данные по умолчанию, когда счетчик итераций равен нулю.

Более подробно о значениях по умолчанию для типов данных см. в разделе *Типы данных элементов управления и индикаторов* в Главе 5 *Построение блок-диаграммы*.

### Циклы For Loop

Циклы For Loop вырабатывают значения данных по умолчанию, если Вы присоедините 0 к терминалу счетчика циклов или если Вы присоедините к циклу пустой массив на вход с включенной автоиндексацией. Такой цикл не выполняется, и любой выходной тун-

нель с отключенной автоиндексацией содержит значение по умолчанию для типа данных туннеля. Используйте сдвигающие регистры для передачи значений через цикл, независимо от того, будет цикл выполняться или нет.

Более подробно о циклах For Loop, автоиндексации и сдвигающих регистрах см. в разделе *Структуры For Loop и While Loop* в Главе 8 *Циклы и структуры*.

### **Значения по умолчанию в массивах**

Индексация за пределами границ массива вырабатывает значение по умолчанию для элемента массива. Вы можете использовать функцию Array Size для определения фактического размера массива. Более подробно о массивах см. в разделе *Массивы* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*. Более подробно об индексации см. в разделе *Автоиндексация циклов* в Главе 8 *Циклы и структуры*. Вы можете нечаянно индексировать за границы массива из-за индексирования прошлого массива последним элементом, используя цикл While Loop, или подавая слишком большое значение на вход **indexing** функции Index Array, или подавая пустой массив на функцию Index Array.

### **Предотвращение неопределенных данных**

Не полагайтесь на специальные значения вроде NaN, Inf или пустые массивы для того, чтобы определить, что ВП вырабатывает неопределенные данные. Вместо этого либо подтвердите, что этот ВП вырабатывает определенные данные, либо делайте сообщение об ошибках, если имеет место ситуация, похожая на получение неопределенных данных.

Например, если Вы создаете ВП, который использует входящий массив для автоиндексации цикла For Loop, определите, что Вы хотите, чтобы делал ВП, когда массив будет пустым. Либо выработайте код ошибки на выходе, либо подставьте определенные данные для значений, которые определяют поведение цикла.

### **Проверка и обработка ошибок**

---

Каждая ошибка имеет числовой код и соответствующее сообщение об ошибке. По умолчанию LabVIEW автоматически обрабатывает любые ошибки, когда ВП запущен в режиме suspend execution (вы-

полнение с приостановкой при вызове), подсвечивая ВПП или функцию, в которых произошла ошибка, и отображая диалоговое окно **Error**.

Выберите **File»VI Properties** и затем **Execution** из спадающего меню **Category**, чтобы отключить автоматическую обработку ошибок для конкретного ВП (снимите птичку с опции **Enable automatic error handling**). Выберите **Tools»Options** и затем **Block Diagram** из спадающего меню, чтобы отключить автоматическую обработку ошибок для всех новых ВП, которые Вы после этого будете создавать (снимите птичку с опции **Enable automatic error handling for new VIs**).

Чтобы игнорировать любую ошибку, в ВПП или в функции, подсоедините выход **error out** этого ВПП или функции к входу **error in** ВП **Clear Errors**. Чтобы отключить автоматическую обработку ошибок для ВПП или функции, соедините их параметр **error out** к параметру **error in** другого ВПП или функции или к индикатору **error out**.

Для управления ошибками в LabVIEW используйте ВП, функции и параметры для обработки ошибок. Например, если LabVIEW обнаруживает ошибку, Вы можете отобразить сообщение об ошибке в диалоговом окне. Используйте обработку ошибок совместно с инструментами отладки для поиска и устранения ошибок. National Instruments настоятельно рекомендует использовать обработку ошибок.

## Проверка на ошибки

Важно понять, что если Вы находитесь внутри ВП, который Вы создаете, то Вы не можете предсказать все проблемы, с которыми может столкнуться пользователь. Без механизма проверки на ошибки Вы сможете узнать только то, что ВП работает неправильно. Проверка ошибок говорит Вам, почему и где произошли ошибки.

Когда Вы выполняете любые операции ввода/вывода, рассмотрите вероятность того, что ошибки могут возникнуть. Почти все функции ввода/вывода возвращают информацию об ошибках. Включите проверку ошибок в ВП, особенно для операций ввода/вывода (файловых, последовательных, от измерительных устройств ввода/вывода).

да/вывода, от коммуникационных устройств) и обеспечьте механизм для соответствующей обработки ошибок.

Проверка на ошибки в ВП может помочь Вам определить следующие проблемы:

- Некорректная инициализация коммуникаций или запись неверных данных во внешнее устройство.
- Внешнее устройство, лишившись питания, становится неисправным или работает неправильно.
- Произошло обновление операционной системы, в результате чего изменились пути к файлам или библиотекам. Вы же будете считать, что проблема в ВП или в системной программе.

## Обработка ошибок

По умолчанию LabVIEW автоматически обрабатывает ошибки посредством приостановки выполнения. Вы можете выбрать другой метод обработки ошибок. Например, если ВП ввода/вывода на блок-диаграмме находится в ожидании, Вы можете не захотеть останавливать все приложение. Вы можете пожелать, чтобы ВП сделал повторную попытку через некоторое время. В LabVIEW можно реализовать на блок-диаграмме ВП такой вариант обработки ошибки.

ВП и функции возвращают ошибки одним из двух способов – в виде цифрового кода ошибки или в виде кластера ошибки. Как правило, функции используют числовой код ошибки, а ВП – кластер ошибки, обычно с ошибками на входах и на выходах. Более подробно о кластерах ошибок см. в разделе *Кластеры ошибок* в настоящей Главе.

Обработка ошибок в LabVIEW следует модели потока данных. Подобно тому, как поток данных проходит через ВП, может проходить и информация об ошибках. Передайте информацию об ошибках от начала ВП к его концу. Включите в конце этого ВП обработчик ошибок для определения того, что данный ВП работает без ошибок. Для передачи через ВП информации об ошибках используйте в каждом ВП, который Вы используете или строите, кластеры **error in** и **error out**.

Как только такой ВП запустится, LabVIEW будет делать проверку на ошибки в каждом исполняемом узле. Если ошибки не обнаружены, значит, данный узел выполнился нормально. Если же в узле будет обнаружена ошибка, то этот узел передаст ошибку следующему узлу без исполнения некоторой части своего кода. Следующий узел поступает таким же образом и т.д. В конце исполненного потока данных будет сообщение об этой ошибке.

### Кластеры ошибок

Кластеры **error in** и **error out** содержат следующие компоненты:

- **status** – булева величина, которая принимает значение TRUE, если имеется ошибка. Большинство ВП, функций и структур, которые принимают булевые данные, могут использоваться для распознавания этого параметра. Например, Вы можете соединить кластер ошибок к булевым входам функций Stop, Quit LabVIEW или Select. Если имеет место ошибка, то кластер ошибок выдаст значение TRUE на входы этих функций.
- **code** – 32-битовой целое со знаком, которое идентифицирует ошибку ее числовым кодом. Ненулевое значение кода ошибки, сопровождаемое значением **status** FALSE, сигнализирует о предупреждении, а не о фатальной ошибке.
- **source** – строка, которая идентифицирует место возникновения ошибки.

### Использование для обработки ошибок циклов While Loop

Вы можете подсоединить кластер ошибок к терминалу условия цикла While Loop, чтобы остановить итерации этого цикла. Когда Вы подсоединяете кластер ошибок к терминалу условия, то фактически на этот терминал проходят только значения TRUE или FALSE параметра **status** из кластера ошибок. При возникновении ошибки цикл While Loop остановится.

Когда кластер ошибок присоединяется к терминалу условия (Condition terminal), пункты контекстного меню **Stop if True** и **Continue if True** изменяются на **Stop on Error** и **Continue while Error**.

### Использование для обработки ошибок структур Case

Когда Вы подсоединяете кластер ошибок к селекторному терминалу структуры Case, метки селектора отображают два случая: **Error**

и `No Error`, а рамка структуры `Case` изменяет свой цвет на красный для варианта `Error` и зеленый для варианта `No Error`. Если имеет место ошибка, структура `Case` выполняет поддиаграмму `Error`. Более подробно об использовании структуры `Case` см. в разделе *Структура Case* в Главе 8 *Циклы и структуры*.

## 7. Создание ВП и ВПП

---

После того, как Вы познакомились с построением лицевой панели и блок-диаграммы, Вы можете создавать свои собственные ВП и ВПП, распространять ВП и строить стандартные исполняемые приложения и совместные библиотеки.

Дополнительные сведения о планировании вашего проекта, включая информацию о ловушках и инструментах, которые Вы можете использовать при разработке своих проектов, содержатся в руководстве *LabVIEW Development Guidelines*.

---

**Более подробно...**

Более подробно относительно создания и использования ВПП, о сохранении ВП и построении стандартных приложений и совместных библиотек см. в справочной системе *LabVIEW Help*.

---

### Планирование и построение вашего проекта

---

Перед разработкой ваших собственных ВП создайте перечень задач, которые ваши пользователи должны будут выполнять. Определите компоненты пользовательского интерфейса, количество и типы элементов управления и индикаторов, которые потребуются для анализа данных, отображения результатов и т.п. Продумайте и обсудите с предполагаемыми пользователями или коллегами по проекту, как и когда пользователям потребуется доступ к функциям и возможностям. Создайте простую лицевую панель, чтобы продемонстрировать ее пользователям или коллегам и определить, помогает ли такая лицевая панель решать их задачи. Используйте такой процесс взаимодействия для улучшения пользовательского интерфейса.

Разделите ваше приложение на логические части легко управляемого размера. Начните с блок-диаграммы верхнего уровня, которая включает основные компоненты вашего приложения. Например, блок-диаграмма может включать блок конфигурации, блок ввода данных, блок обработки данных, блок отображения результатов, блок сохранения данных на диск и блок обработки ошибок.

После создания блок-диаграммы верхнего уровня определите входы и выходы. Затем создавайте ВПП, которые реализуют основные компоненты блок-диаграммы верхнего уровня. Использование ВПП делает блок-диаграмму верхнего уровня легкой для чтения, отладки, понимания и сопровождения. Вы также можете создавать ВПП для типовых и часто используемых операций, которые используются в нескольких местах. Сразу после создания ВПП проверяйте их. Вы можете создавать тестовые процедуры высокого уровня, однако выловить ошибки в небольшом модуле гораздо легче, чем тестировать иерархию из нескольких ВП. Вы можете обнаружить, что начальное проектирование блок-диаграммы высокого уровня является неполным. Использование ВПП для реализации сложных задач обеспечивает легкость модификации или реорганизации вашего приложения. Более подробно о ВПП см. в разделе *Виртуальные подприборы* в этой Главе.

Выберите **Help»Find Examples** для ознакомления с примерами блок-диаграмм и ВПП.

## **Разработка проектов несколькими разработчиками**

Если над одним проектом работают несколько разработчиков, то с самого начала определяйте обязанности по программированию, стандарты на интерфейсы и кодирование, чтобы гарантировать успешность процесса проектирования и хорошую совместную работу приложения. Более подробно о стандартах кодирования см. в разделе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

Сохраняйте оригиналы проекта ВП на одном компьютере и установите политику управления исходными кодами. Рассмотрите использование пакета LabVIEW Professional Development System, который включает инструменты управления исходными кодами, упрощающие совместное пользование файлами. Эти инструменты включают также утилиту для сравнения виртуальных приборов и просмотра различий между версиями. Более подробно об управлении исходными кодами см. в разделе *Source Code Control* в Главе 2 *Incorporating Quality into the Development Process* в руководстве *Development Guidelines*.

## Шаблоны ВП

---

Шаблоны виртуальных приборов (VI templates) включают ВПП, функции, структуры и объекты лицевой панели, которые могут потребоваться Вам, чтобы начать строить типовые измерительные приложения. Шаблоны ВП открываются как виртуальные приборы с именем «безымянный» (untitled), которые затем нужно сохранить. Выберите **File»New** для отображения диалогового окна **New**, которое содержит имеющиеся шаблоны ВП. Вы также можете отобразить диалоговое окно **New**, выбирая пункт **New** из выпадающего меню **New** главного диалогового окна **LabVIEW**.

Если Вы используете шаблон ВП в качестве ВПП, LabVIEW предложит Вам сохранить шаблон как ВП перед его закрытием.

### Создание шаблонов ВП

Создавайте пользовательские шаблоны ВП, чтобы исключить необходимость помещать одни и те же компоненты на лицевые панели или блок-диаграммы всякий раз, когда Вы создаете новые ВП. Чтобы создать пользовательский шаблон ВП, постройте ВП и сохраните его как шаблон (template).

### Другие типы документов

Выберите соответствующий пункт раздела **Other Documents Types** в перечне **Create New** диалогового окна **New**, чтобы начать создавать глобальные переменные, пользовательские элементы управления, меню времени исполнения, полиморфные ВП или шаблоны для глобальных переменных и элементов управления.

## Использование встроенных ВП и функций

---

Чтобы помочь Вам создавать свои специфические приложения, LabVIEW содержит ВП и функции, обеспечивающие ввод/вывод, доступ к другим ВП и связь с другими приложениями. Вы можете использовать эти и ВП и ВПП в ваших приложениях для ускорения их разработки. Более подробно о ВПП см. в разделе *Виртуальные подприборы (ВПП)* в настоящей Главе.

## Построение ВП и функций для управления приборами и ввода/вывода данных

LabVIEW включает сотни примеров ВП, которые Вы можете использовать и встраивать в создаваемые вами ВП. Вы можете или модифицировать ВП из примера и адаптировать его к вашему приложению, или копировать фрагменты из одного или нескольких примеров и вставлять их в свой ВП.

Вы можете использовать встроенные ВП и функции для управления внешними приборами, такими как осциллографы, или осуществлять сбор данных, например, снимая данные с термопары.

Используйте ВП и функции с палитры **Instrument I/O** для управления внешними приборами. Чтобы управлять приборами в LabVIEW, Вы должны иметь оборудование, правильно установленное, включенное и работающее на вашем компьютере. ВП и функции, которые Вы используете для управления приборами, зависят от протоколов связи, которые поддерживает ваше оборудование. Более подробно о построении ВП для управления приборами см. в руководстве *LabVIEW Measurements Manual*.

Используйте ВП и функции с палитры **Data Acquisition** для ввода/вывода данных через DAQ-устройства. Чтобы использовать эти ВП, Вы должны иметь установленными программный драйвер NI-DAQ и оборудование DAQ. Более подробно об инсталляции программного драйвера NI-DAQ и оборудования DAQ и о построении ВП для сбора данных см. в руководстве *LabVIEW Measurements Manual*. После того, как Вы введете данные, Вы можете использовать встроенные ВП и функции с палитр **Analyze**, **Report Generation** и **Mathematics** для обработки, генерации отчетов и выполнения математических операций над этими данными. Более подробно об основах математики и обработки данных в LabVIEW см. в руководстве *LabVIEW Analysis Concepts*.

## Построение ВП, которые имеют доступ к другим ВП

Используйте ВП и функции из подпалитры **Application Control** (управление приложениями) для управления поведением ВП в тех случаях, когда он вызывается как ВПП или запускается пользователем. Вы можете использовать эти ВП и функции для одновременного конфигурирования нескольких ВП. Кроме того, если Вы находитесь в локальной сети с другими пользователями LabVIEW,

то можете использовать эти ВП и функции для удаленного доступа и управления ВП. Более подробно об удаленном управлении ВП см. в Главе 17 *Программное управление ВП*.

## Построение ВП, которые общаются с другими приложениями

Используйте ВП и функции с подпалитры **File I/O** (файловый ввод/вывод) для чтения данных из или записи данных в другие приложения, такие, например, как Microsoft Excel. Вы можете использовать эти ВП и функции для создания отчетов или внедрения данных из других приложений в ВП. Более подробно о передаче данных в другие приложения и из них см. в Главе 14 *Файловый ввод/вывод*.

Используйте ВП и функции из подпалитры **Communication** (связь) для передачи данных через сеть, используя протокол обмена вроде FTP, и построение на основе этого протокола приложений клиент-сервер. Более подробно о связи с другими приложениями через локальную или глобальную сеть см. в Главе 18 *Сетевые коммуникации в LabVIEW*.

**(Windows)** Используйте функции **ActiveX**, чтобы добавить объекты ActiveX в ВП или для управления ActiveX-совместимыми приложениями. Более подробно об использовании технологии ActiveX см. в Главе 19 *Связность в среде Windows*.

## Виртуальные подприборы (ВПП)

---

После того, как Вы построите ВП и создадите для него иконку и соединительную панель, его можно использовать внутри другого ВП. Виртуальный прибор, вызываемый с блок-диаграммы другого виртуального прибора, называется виртуальным подприбором (ВПП). ВПП соответствует подпрограмма в тестовых языках программирования. Узлу ВПП соответствует вызов подпрограммы. Этот узел не является самим ВПП, также как оператор вызова подпрограммы не является самой подпрограммой. Блок-диаграмма, которая содержит несколько одинаковых узлов ВПП, несколько раз вызывает один и тот же ВПП.

Элементы управления и индикаторы ВПП получают данные от блок-диаграммы вызывающего ВП и возвращают в нее результаты. Щелкните на палитре **Functions** иконку **Select a VI**, найдите нуж-

ный файл ВП, сделайте по нему двойной щелчок и поместите этот ВП на блок-диаграмму, чтобы создать ВПП, который вызывается из данного ВП.



**Примечание.** Перед тем, как использовать ВП в качестве ВПП, Вы должны установить соединительную панель. Более подробно о конфигурировании соединительной панели см. в разделе *Конфигурирование соединительной панели* в настоящей Главе.

Вы можете открыть ВПП для редактирования, делая двойной щелчок по узлу ВПП на блок-диаграмме с помощью инструмента **Operating** или **Positioning**. Если Вы сохраняете ВПП, то внесенные изменения будут действовать на все вызовы данного ВПП, а не только на текущий его представитель.

Обычно, когда LabVIEW вызывает ВПП, он запускается без отображения его лицевой панели. Если Вы хотите, чтобы конкретный представитель некоторого ВПП отображал свою лицевую панель при вызове, щелкните правой кнопкой иконку ВПП и выберите из контекстного меню **SubVI Node Setup**. Если же Вы хотите, чтобы все представители ВПП отображали свою лицевую панель при вызове, выберите **File»VI Properties**, затем из спадающего меню **Category** выберите **Window Appearance** и щелкните кнопку **Customize**.

## Выявление однотипных операций

При создании ВП Вы можете обнаружить, что некоторые операции выполняются многократно. Рассмотрите возможность использования ВПП или циклов для повторного выполнения таких операций. Например, блок-диаграмма на Figure 7-1 содержит две одинаковые операции.

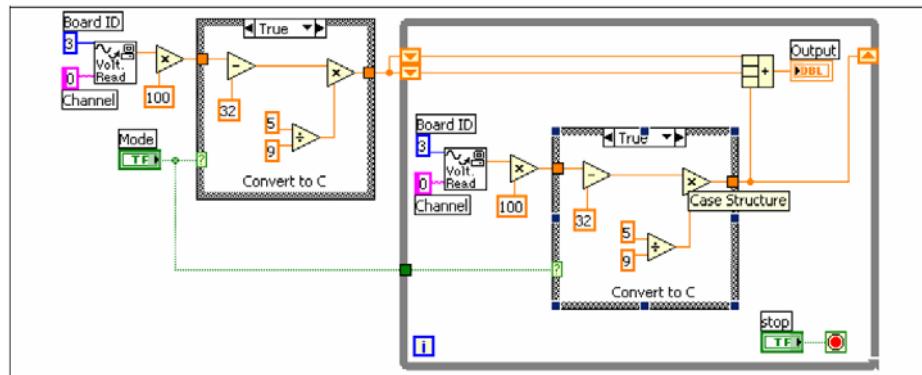


Figure 7-1. Блок-диаграмма с двумя одинаковыми операциями

Вы можете создать ВПП, который выполняет эту операцию, и затем дважды вызвать этот ВПП, как это показано на Figure 7-2.

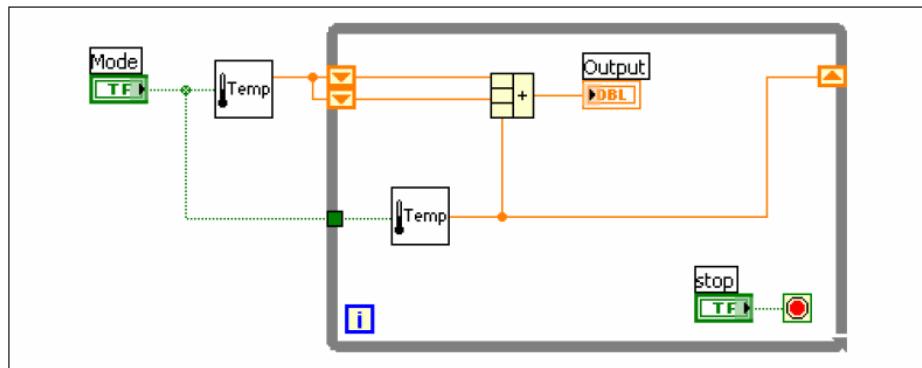


Figure 7-2. Двукратный вызов ВПП

Кроме того, Вы можете использовать этот ВПП в других ВП. Более подробно об использовании циклов для объединения однотипных операций см. в Главе 8 *Циклы и структуры*.

### Конфигурирование соединительной панели



Чтобы использовать ВП в качестве ВПП, Вы должны построить соединительную панель, показанную слева. Соединительная панель представляет собой набор терминалов, которые соответствуют элементам управления и индикаторам ВП, подобно списку параметров

в операторе вызова функций в текстовых языках программирования. Соединительная панель определяет входы и выходы, которые Вы можете подсоединять к такому ВП, в результате чего появляется возможность использовать его в качестве ВПП. Более подробно о соединительной панели см. в разделе *Иконка и соединительная панель* в Главе 2 *Введение в виртуальные приборы*.

Определите связи, назначая элемент управления или индикатор каждому терминалу соединительной панели. Для определения соединительной панели щелкните правой кнопкой иконку в правом верхнем углу окна лицевой панели и выберите **Show Connector** из контекстного меню, чтобы отобразить соединительную панель. Соединительная панель появится на месте иконки. Каждый прямоугольник на соединительной панели представляет терминал. Используйте эти прямоугольники для назначения входов и выходов. Число терминалов, которые отображаются на соединительной панели, зависит от числа элементов управления и индикаторов на лицевой панели.

Соединительная панель может содержать не более 28 терминалов. Если ваша лицевая панель содержит более 28 элементов управления и индикаторов, которые Вы хотите использовать для программного доступа, сгруппируйте некоторые из них в кластеры и назначьте этому кластеру один терминал соединительной панели. Более подробно о группировке данных в виде кластеров см. в разделе *Кластеры* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.



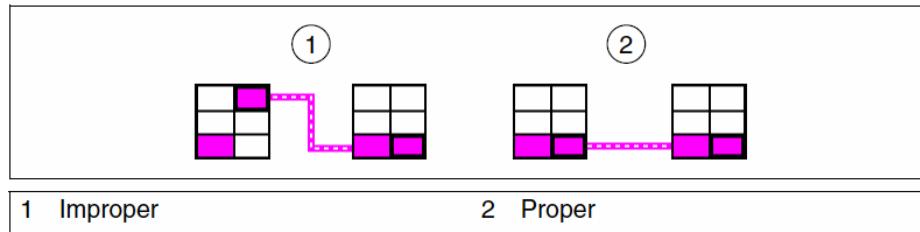
**Примечание.** Назначение более 16 терминалов для ВП может ухудшить его читабельность и затруднить его использование.

Чтобы выбрать нужную конфигурацию (pattern) соединительной панели, щелкните по ней правой кнопкой и выберите из контекстного меню **Patterns**. Выберите конфигурацию соединительной панели с нужным числом дополнительных терминалов. Вы можете оставить эти дополнительные терминалы не присоединенными, пока они Вам не потребуются. Такая гибкость позволяет Вам делать изменения с минимальным эффектом на всю иерархию ВП.

Если Вы создаете группу ВПП, которые часто используются совместно, то согласуйте вид соединительных панелей отдельных ВПП

таким образом, чтобы однотипные входы размещались на одном и том же месте, что поможет Вам запомнить расположение каждого входа. Если Вы создаете ВПП, который вырабатывает выход, который другие ВПП используют в качестве входа, располагайте такие входной и выходной терминалы на одном уровне, чтобы упростить конфигурацию соединительных проводников. В частности, размещайте кластеры **error in** в левом нижнем углу соединительной панели, а кластеры **error out** - в правом нижнем углу.

Пример правильного и неправильного выравнивания терминала кластера ошибок показан на Figure 7-3.



**Figure 7-3.** Неправильное (improper) и правильное (proper) выравнивание терминала кластера ошибок

Рекомендации по стилю использования настроек соединительной панели можно найти в разделе *Connector Panes* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

#### **Установка обязательных, рекомендуемых и необязательных входов и выходов**

Вы можете назначить, какие входы и выходы являются обязательными(required), рекомендуемыми (recommended) и необязательными (optional), чтобы предохранить пользователей от забывчивости при подсоединении проводников к терминалам ВПП.

Щелкните правой кнопкой терминал на соединительной панели и выберите из контекстного меню **This Connection Is**. Текущая установка отображается птичкой. Выберите **Required**, **Recommended** или **Optional**.

Для входного терминала опция **Required** означает, что блок-диаграмма, на которой находится данный ВПП, будет неисправным

(broken), если хотя бы один из обязательных входов не будет присоединен. Для выходных терминалов опция **Required** недоступна. Для входных и выходных терминалов опция **Recommended** или **Optional** означает, что блок-диаграмма, на которой находится данный ВПП, может выполняться, даже если имеются неподключенные рекомендуемые или необязательные терминалы. При этом ВП не будет выдавать никаких предупреждающих сообщений.

Входы и выходы ВП из библиотеки `vi.lib` уже помечены как **Required**, **Recommended** или **Optional**. Входы и выходы ВП, которые Вы создаете, LabVIEW по умолчанию устанавливает в состояние **Recommended**. Устанавливайте состояние терминала **Required** только в том случае, когда для правильной работы ВП этот терминал обязательно должен быть подключен.

В окне **Context Help** метки обязательных терминалов пишутся жирным шрифтом, метки рекомендуемых терминалов – обычным шрифтом, а метки необязательных терминалов – тусклым шрифтом. Метки необязательных терминалов вообще не будут отображаться, если Вы щелкните кнопку **Hide Optional Terminals and Full Path** в окне **Context Help**.

## Создание иконки



Каждый ВП имеет в верхнем правом углу окна лицевой панели и блок-диаграммы иконку, показанную слева. Иконка это графическое представление ВП. Она может содержать текст, картинку или их комбинацию. Если ВП используется в качестве ВПП, то иконка представляет этот ВПП на блок-диаграмме другого ВП.

Исходная иконка (по умолчанию) содержит номер, который показывает, сколько новых ВП Вы открыли с начала запуска LabVIEW. Чтобы создать собственную иконку вместо иконки по умолчанию, щелкните правой кнопкой иконку в правом верхнем углу лицевой панели или блок-диаграммы и выберите из контекстного меню **Edit Icon**. Аналогичный результат будет, если сделать двойной щелчок по этой иконке.

Вы можете также перетащить (drag) любую графику из вашей файловой системы и вставить (drop) ее в верхний правый угол лицевой панели или блок-диаграммы. LabVIEW конвертирует графику в иконку размером 32×32 пикселя.

В зависимости от типа используемого монитора можно создать отдельные иконки для монохромного, 16-цветного и 256-цветного режимов. Для вывода на печать LabVIEW использует монохромную иконку, если у Вас нет цветного принтера.

Более подробно о построении иконки см. раздел *Icons* в Главе 6 *LabVIEW Style Guide* в руководстве *LabVIEW Development Guidelines*.

## Отображение ВПП и экспресс ВП в виде иконок либо в виде расширяемых узлов

Вы можете отображать ВП и экспресс ВП как в виде иконок, так и в виде расширяемых (expandable) узлов. Расширяемые узлы имеют вид иконок, окруженных цветным полем. ВПП имеет поле желтого цвета, а экспресс ВП – синего. Используйте иконки, если Вы хотите сэкономить место на блок-диаграмме. Используйте расширяемые узлы, чтобы облегчить процесс соединения проводниками и улучшить документирование блок-диаграммы. По умолчанию ВПП появляются на блок-диаграмме в виде иконок, а экспресс ВП – в виде расширяемых узлов. Чтобы отобразить ВПП или экспресс ВП в виде расширяемого узла, щелкните правой кнопкой и выберите из контекстного меню пункт **View as Icon**, чтобы снять с него птичку.



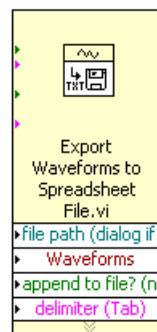
**Примечание.** Если ВПП или экспресс ВП имеет вид расширяемого узла, то для такого узла нельзя сделать видимыми терминалы и нельзя активизировать доступ к базе данных (пункт контекстного **Enable Database Access** становится недоступным).

Когда Вы изменяете размер расширяемого ВПП или экспресс ВП, их входные и выходные терминалы появляются ниже иконки. Необязательные терминалы имеют серый фон. Рекомендуемые или обязательные входные или выходные терминалы, которые Вы не отобразили снизу, появляются в виде входных или выходных стрелок в цветном поле, которое окружает иконку. Если Вы подсоединили проводник к необязательному терминалу, когда ВПП или экспресс ВП находился в растянутом состоянии, и затем изменили размер ВПП или экспресс ВП так, что необязательный терминал исчез из растягиваемого поля, то необязательный терминал появится в виде входной или выходной стрелки на цветном поле. Однако

если Вы отсоедините проводник от такого необязательного терминала, то соответствующая ему стрелка исчезнет.

По умолчанию при растягивании расширяемого узла входы появляются сверху. Чтобы выбрать нужный вход или выход, отображаемый на данном терминале, щелкните правой кнопкой этот терминал в расширяемой области узла и выберите из контекстного меню пункт **Select Input/Output**. Линия в контекстном меню отделяет входы от выходов. Метки для расширяемых ВПП или экспресс ВП появляются в цветном поле, окружающем иконку. Чтобы подогнать размер расширяемого узла к наиболее длинному наименованию терминала, щелкните правой кнопкой ВПП или экспресс ВП и выберите из контекстного меню пункт **Size to Text**.

Показанный ниже расширяемый ВПП отображает четыре из 10 входных и выходных терминалов.



Более подробно об экспресс ВП см. в руководстве *Getting Started with LabVIEW*.

## Создание ВПП из фрагмента ВП

Чтобы преобразовать фрагмент ВП в ВПП, нужно с помощью инструмента **Positioning** селектировать фрагмент блок-диаграммы, который Вы хотите использовать повторно, и выбрать из главного меню пункт **Edit»Create SubVI**. Селектированный фрагмент блок-диаграммы заменится на иконку нового ВПП. LabVIEW создаст для нового ВПП элементы управления и индикаторы и подсоединит его к существующим проводникам.

Создание ВПП из селектированной области удобно, но все же требует определенного планирования, чтобы получилась логичная иерархия из ВП. Рассмотрите каждый объект на предмет включения его в селектируемую область и правильного функционирования resulting BП.

## Конструирование ВПП

Если пользователям нет необходимости видеть лицевую панель ВПП, то можно сэкономить время на создание ее внешнего вида, включая цвета и шрифты. Однако, организация лицевой панели все же важна, поскольку Вам может потребоваться просматривать лицевую панель в процессе отладки ВП.

Размещайте элементы управления и индикаторы в том порядке, как они расположены на соединительной панели. Размещайте элементы управления слева лицевой панели, а индикаторы – справа. Размещайте кластеры **error in** в левой нижней части лицевой панели, а кластеры **error out** – в правой нижней части. Более подробно о конфигурировании соединительной панели см. в разделе *Конфигурирование соединительной панели* в настоящей Главе.

Если элемент управления имеет значение по умолчанию, поместите это значение в скобках как часть имени (метки) элемента управления. Также включите в имя элемента управления единицы измерения, если они имеются. Например, если элемент управления устанавливает верхний предел температуры и имеет значение по умолчанию 75°F, присвойте ему имя **верхний предел температуры (75°F)**. Если Вы будете использовать ВПП на разных платформах, исключите использование специальных символов в именах элементов управления. Например, **degF** вместо **°F**.

Называйте булевые элементы управления так, чтобы пользователи могли определить, что этот элемент управления делает, когда он находится в состоянии истина (TRUE). Используйте имена вроде **Отказаться**, **Сброс** или **Инициализация**, чтобы пояснить выбранное действие.

## Просмотр иерархии ВП

В окне **Hierarchy** (иерархия) отображается графическое представление иерархии вызовов для всех ВП, загруженных в память, включая определения типов и глобальные переменные. Чтобы отобра-

зить окно **Hierarchy** выберите **Browse»Show VI Hierarchy**. Используйте это окно для просмотра всех ВПП и узлов, которые образуют текущий ВП.

При перемещении инструмента **Operating** над объектами окна **Hierarchy** LabVIEW отображает имя каждого ВП. Вы можете использовать инструмент **Operating**, чтобы перетащить ВП из окна **Hierarchy** на блок-диаграмму для использования этого ВП в качестве ВПП внутри другого ВП. Вы можете также селектировать и копировать в буфер узел или несколько узлов, чтобы потом вставить их в другие блок-диаграммы. Сделайте двойной щелчок по ВП в окне **Hierarchy**, чтобы отобразить его лицевую панель.

ВП, которые содержат ВПП, имеют стрелку на своей нижней границе. Щелкните эту стрелку, чтобы отобразить скрытые ВПП. Стрелка будет красной, когда все ВПП являются скрытыми, и черной, когда все ВПП являются видимыми.

## Сохранение ВП

---

Вы можете сохранять каждый ВП в отдельном файле, либо можете группировать несколько ВП и сохранять их в библиотеке ВП. Файлы библиотек ВП имеют расширение .11b. National Instruments рекомендует вам сохранять ВП в отдельных файлах, организованных в директории, особенно когда над одним проектом работают много разработчиков. Более подробно об организации ВП в директории см. в разделе *Organizing VIs in Directories* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

### Преимущества сохранения каждого ВП в отдельном файле

Имеются следующие причины для сохранения каждого ВП в отдельном файле:

- Вы можете использовать файловую систему для манипуляций с каждым отдельным файлом.
- Вы можете использовать поддиректории.
- Хранение всех ВП и элементов управления в отдельных файлах более надежно, чем хранение целого проекта в одном файле.

- Вы можете использовать встроенные в Professional Development System инструменты для управления исходным кодом (source code control tools).

## Преимущества сохранения нескольких ВП в библиотеках

Имеются следующие причины для сохранения нескольких ВП в виде библиотек:

- Вы можете использовать имена файлов длиной более 255 символов.
- Преобразовать библиотеку ВП под другую платформу гораздо легче, чем преобразовывать множество отдельных файлов. При этом гарантируется, что пользователь получит все необходимые файлы.
- Вы можете несколько сократить размер файлов для вашего проекта, поскольку библиотека ВП компрессируется для уменьшения дискового пространства.
- Вы можете помечать некоторые ВП в вашей библиотеке как ВП верхнего уровня. При открытии такой библиотеки LabVIEW автоматически откроет все ВП верхнего уровня из этой библиотеки.



**Примечание.** Множество встроенных в LabVIEW ВП и примеров хранятся в виде библиотек ВП, что гарантирует правильное их размещение на всех платформах.

Если Вы используете библиотеки ВП, предусмотрите разбиение вашего приложения на несколько библиотек ВП. Помещайте ВП верхнего уровня в одну библиотеку и заведите другие библиотеки, чтобы хранить группы ВП отдельно по их функциям. Сохранение изменений в ВП, размещенном в библиотеке, требует большего времени, чем для размещенного в отдельном файле, поскольку операционной системе приходится вносить изменения в большой библиотечный файл. Сохранение изменений в большом библиотечном файле может потребовать больших затрат памяти и снизить эффективность. Страйтесь ограничивать размер библиотечных файлов около 1 МВ.

## Управление виртуальными приборами в библиотеках

Чтобы упростить копирование, переименование и удаление файлов внутри библиотек ВП, используйте VI Library Manager (менеджер библиотеки ВП), для активизации которого выберите **Tools»VI Library Manager**. Это инструмент можно использовать также для создания новых библиотек ВП и директорий, а также для преобразования библиотек ВП в директории и обратно. Создание новых библиотек ВП и директорий и преобразование их друг в друга важно в тех случаях, когда Вы манипулируете вашими ВП с помощью инструментов управления исходным кодом (source code control tools).

Чтобы исключить файловые операции над ВП, размещенными в памяти, перед использованием менеджера библиотек ВП закройте все ВП, которые могут оказывать влияние.

**(Windows 2000/XP/Me/98)** Можно сделать двойной щелчок по .lib файлу в проводнике Windows, что приведет к открытию содержания этой библиотеки. После этого компоненты библиотеки можно открывать, копировать, перемещать, переименовывать и удалять обычным образом.

## Имена ВП

Когда Вы сохраняете ВП, используйте описательные имена. Такие описательные имена, как Temperature Monitor.vi и Serial Write & Read.vi, позволяют легко различать ВП и получать представление об их использовании. Если же Вы используете неоднозначные имена вроде VI#1.vi, то они могут затруднить идентификацию разных ВП, особенно если Вы сохранили много таких ВП.

Продумайте, будут ли ваши пользователи запускать ВП на других платформах. Исключите использование символов, которые некоторые операционные системы резервируют для специальных целей, например такие, как \ : / ? \* < > #.

Ограничьте имена ВП 31 символом, если ваши пользователи могут запускать их в среде Mac OS 9.x или более ранних версий.

## Сохранение в формате предыдущей версии LabVIEW

Вы можете сохранить ВП в формате предыдущей версии LabVIEW, чтобы сделать удобным обновление LabVIEW и помочь, при необходимости, поддерживать такие ВП в двух версиях LabVIEW. Если Вы обновляете LabVIEW до новой версии, то сохраняется возможность вернуться к последней версии этих ВП.

Когда Вы сохраняете ВП в формате предыдущей версии, LabVIEW конвертирует не только этот конкретный ВП, но и все ВП из его иерархии, включая файлы vi.llb.

Часто ВП использует функциональность, не поддерживаемую предыдущими версиями LabVIEW. В таких случаях LabVIEW сохраняет ВП в максимально возможной степени и выдает отчет с перечислением того, что не удалось конвертировать. Этот отчет сразу появляется в диалоговом окне **Warnings** (сообщения). Для подтверждения этих сообщений и закрытия диалогового окна нажмите кнопку **OK**. Чтобы сохранить эти сообщения в текстовом файле для последующего просмотра, нажмите кнопку **Save**.

## Распространение виртуальных приборов

---

Если Вы намерены распространять свои ВП на другие компьютеры или другим пользователям, продумайте, хотите ли Вы включить исходный код блок-диаграммы, который пользователи смогут просматривать и редактировать, или же Вы хотите скрыть или удалить блок-диаграмму.

Если Вы намерены распространять ВП другим разработчикам LabVIEW вместе с исходным кодом блок-диаграммы, то можно назначить защиту блок-диаграммы с помощью пароля. Такая блок-диаграмма останется все еще доступной, но пользователь должен ввести пароль, чтобы просмотреть ее или внести изменения.

Если Вы намерены распространять ВП разработчикам на других алгоритмических языках, то можно построить стандартное приложение (stand-alone application) или библиотеку совместного доступа (shared library). И стандартное приложение и библиотека совместного доступа удобны в тех случаях, когда Вы не предполагаете, что ваши пользователи будут редактировать ваши ВП. При этом пользователи смогут запускать ваше приложение или использовать ва-

шу совместную библиотеку, но они не смогут просматривать и вносить изменения в их блок-диаграммы.



**Примечание.** Создавать стандартные приложения или библиотеки совместного доступа можно только в пакете LabVIEW Professional Development System, либо используя дополнительную программу Application Builder.

Другой возможностью для распространения ваших ВП является удаление исходного кода блок-диаграммы, в результате чего другие пользователи не смогут редактировать такие ВП. Чтобы сохранить ВП без блок-диаграммы с целью сокращения размера файла и предотвращения внесения изменений другими пользователями, выберите в главном меню пункт **File»Save With Options**. Если Вы сохраните ВП без блок-диаграммы, то пользователи не смогут переносить такой ВП на другие платформы или обновлять такой ВП до новой версии LabVIEW.



**Предупреждение.** Если Вы сохраняете ваши ВП без блок-диаграмм, не записывайте их в файлы с оригинальными версиями этих же ВП. Сохраняйте их в других директориях или под другими именами.

Более подробно о перенесении ВП с одной платформы на другую и их локализации см. в заметках к приложению (Application Notes) *Porting and Localizing LabVIEW VIs*.

## Построение стандартных приложений и библиотек совместного доступа

---

Чтобы воспользоваться программой Application Builder для создания стандартных приложений и инсталляторов или библиотек совместного доступа (DLLs) для ваших ВП, выберите из главного меню пункт **Tools»Build Application or Shared Library (DLL)**. Используйте совместные библиотеки в тех случаях, когда Вы хотите вызывать ваши ВП из совместной библиотеки, используя текстовые языки программирования, такие как LabWindows™/CVI™, Microsoft Visual C++, и Microsoft Visual Basic.

Совместные библиотеки удобны, когда Вы хотите поделиться функциональными возможностями ваших ВП с другими разработ-

чиками. Использование совместных библиотек дает способ, с помощью которого языки программирования отличные от LabVIEW получают доступ к коду, разработанному в среде LabVIEW.

Другие разработчики смогут запускать такое стандартное приложение или смогут воспользоваться такой совместной библиотекой, но они не смогут просматривать и редактировать соответствующую блок-диаграмму.



**Примечание.** Пакет LabVIEW Professional Development System содержит программу Application Builder. Если же Вы используете LabVIEW Base Package или Full Development System, то программу Application Builder можно приобрести отдельно, посетив интернет сайт National Instruments [ni.com/info](http://ni.com/info) с последующим введением информационного кода `rdlv21`. Для конфигурирования параметров создаваемого приложения или совместной библиотеки воспользуйтесь вкладками диалогового окна **Build Application or Shared Library (DLL)**. После задания всех установок сохраните их в скрипте, что позволит Вам, при необходимости, быстро осуществить повторное построение этого же приложения.

Более подробно об инсталляции программы Application Builder см. в руководстве *LabVIEW Application Builder User Guide*.

# Часть II

---

## Построение и редактирование виртуальных приборов

В этой части рассматриваются возможности LabVIEW, ВП и функции, которые Вы можете использовать, чтобы разными способами обеспечить работу ваших приложений. Главы этой части описывают полезность каждой функциональной возможности LabVIEW и очерчивают каждый класс ВП и функций.

Часть II, *Построение и редактирование виртуальных приборов*, содержит следующие главы:

- Глава 8, *Циклы и структуры*, описывает, как использовать на блок-диаграмме циклы и структуры, чтобы осуществить повтор блоков кода и выполнять код в нужной последовательности и в зависимости от условий.
- Глава 9, *Событийно управляемое программирование*, описывает, как динамически и статически регистрировать события и как создавать и обозначать свои собственные события.
- Глава 10, *Группировка данных с использованием строк, массивов и кластеров*, описывает, как применять строки, массивы и кластеры для группировки данных.
- Глава 11, *Локальные и глобальные переменные*, описывает, как применять локальные и глобальные переменные для передачи информации между точками вашего приложения, которые не могут быть соединены проводником.
- Глава 12, *Графики и диаграммы*, описывает, как применять графики и диаграммы для наглядного графического представления зависимостей данных.

- Глава 13, *Графика и звук*, описывают, как отображать или модифицировать графические изображения и звук в ВП.
- Глава 14, *Файловый ввод/вывод*, описывает, как выполняются операции файлового ввода/вывода.
- Глава 15, *Документирование и печать виртуальных приборов*, описывает, как документировать и распечатывать ВП.
- Глава 16, *Конфигурирование ВП*, описывает, как конфигурировать ВП и ВПП, чтобы они работали так, как этого требует ваше приложение.
- Глава 17, *Программное управление ВП*, описывает, как установить связь с виртуальными приборами и другими копиями LabVIEW, чтобы можно было программно управлять этими виртуальными приборами и средой LabVIEW.
- Глава 18, *Сетевые коммуникации в LabVIEW*, описывает, как использовать виртуальные приборы для обмена или организации сети с другими процессами, включая запуск из других приложений или с удаленного компьютера.
- Глава 19, *Связность в среде Windows*, описывает, как обеспечить внешний доступ к объектам, командам и функциям для доступа к ним из других приложений Windows.
- Глава 20, *Вызов кода из текстовых языков программирования*, описывает, как вызвать код из текстовых языков программирования и использовать библиотеки DLL.
- Глава 21, *Формулы и уравнения*, описывает, как применять уравнения в виртуальных приборах.

## 8. Циклы и структуры

---

Структуры – это графическое представление операторов цикла и выбора из текстовых языков программирования. Используйте структуры на блок-диаграмме для повторения фрагментов кода либо для выполнения их последовательно один за другим, либо в другом заданном порядке.

Подобно другим узлам, структуры имеют терминалы, которые позволяют соединять их с другими узлами блок-диаграммы, они выполняются автоматически при поступлении входных данных и выдают данные на выходе после того, как их выполнение завершится.

Каждая структура имеет четкую границу с изменяемыми размерами. Эта граница охватывает фрагмент блок-диаграммы, который выполняется в соответствии с правилами данной структуры. Такой фрагмент блок-диаграммы внутри границ структуры называется поддиаграммой (subdiagram). Терминалы, через которые данные поступают внутрь структуры и выходят из нее наружу, называются туннелями. Туннель – это точка на границе структуры, к которой может быть произведено соединение.

---

**Более подробно...**

Более подробно относительно использования структур см. справочную систему *LabVIEW Help*.

---

Для управления процессом исполнения блок-диаграммы используйте следующие структуры, размещенные на палитре **Structures**:

- **For Loop** (цикл с заданным числом итераций) – Повторяет выполнение поддиаграммы заданное число раз.
- **While Loop** (цикл по условию) – Повторяет выполнение поддиаграммы, пока не выполнится некоторое условие.
- **Case Structure** (структура выбора) – Содержит много поддиаграмм, но только одна из них выполняется в зависимости от значения входной величины, поступившей на вход этой структуры.

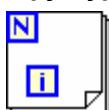
- **Sequence structure** (строктура последовательности) – Содержит одну или несколько поддиаграмм, которые выполняются последовательно одна за другой.
- **Formula node** (формульный узел) – Выполняет математические операции над числовой величиной, поступившей на вход. Более подробно об использовании формульного узла см. в разделе *Формульные узлы* в Главе 21 *Формулы и уравнения*.
- **Event structure** (событийная структура) – Содержит одну или несколько поддиаграмм, которые выполняются в зависимости от того, как пользователь взаимодействует с этим ВП.

Чтобы отобразить контекстное меню структуры, щелкните правой кнопкой по ее границе.

## Структуры For Loop и While Loop

Используйте структуры For Loop и While Loop для управления повторяющимися операциями.

### Структура For Loop



Структура For Loop (цикл с заданным числом итераций), показанная слева, выполняет поддиаграмму заданное число раз.



Значение, поступающее на терминал числа повторений – count terminal (это входной терминал), показанный слева, определяет количество повторов выполнения поддиаграммы. Число повторений можно задать явно, путем подключения проводника с нужным значением снаружи цикла к левой или верхней стороне терминала числа повторений, либо неявно, путем задания числа повторений с помощью автоиндексации (auto-indexing). Более подробно о неявном задании числа повторений см. в разделе *Автоиндексация для установки числа повторений в цикле For Loop* в настоящей Главе.



Терминал итерации – iteration terminal (выходной терминал), показанный слева, содержит значение, равное количеству завершенных итераций (повторов). Подсчет итераций всегда начинается с нуля. При этом в процессе выполнения первой итерации на выходе терминала итераций будет значение **0**.

И терминал числа повторений, и терминал итераций имеют тип 32-битного целого со знаком. Если Вы подсоедините к терминалу чис-

ла повторений число с плавающей точкой, LabVIEW округлит его и принудительно приведет его к диапазону представления целых чисел. Если к терминалу числа повторений присоединить **0** или отрицательное число, то цикл не будет исполняться ни разу, а выходы будут содержать значения по умолчанию, принятые для каждого типа данных.

Для передачи данных от текущей итерации к следующей итерации, добавьте сдвигающие регистры. Более подробно о создании сдвигающих регистров в циклах см. в разделе *Сдвигающие регистры и узел обратной связи в циклах* в настоящей Главе.

## Структура While Loop



Структура While Loop, показанная слева, повторно выполняет поддиаграмму до тех пор, пока удовлетворяется некоторое условие. Структура While Loop подобна конструкциям Do Loop или Repeat-Until Loop в текстовых языках программирования.



Цикл While Loop выполняет поддиаграмму до тех пор, пока терминал условия – condition terminal (входной терминал) не получит заданное булево значение. По умолчанию терминал условия появляется в виде **Stop if True** (остановиться, если Истина), показанный слева. Если терминал условия имеет вид **Stop if True**, то поддиаграмма цикла While Loop будет выполняться до тех пор, пока на терминал условия не поступит значение TRUE. Вы можете изменить поведение и внешний вид терминала условия. Для этого щелкните правой кнопкой этот терминал или границу цикла While Loop и выберите из контекстного меню пункт **Continue if True** (продолжить, если Истина). Терминал условия примет вид, показанный слева. Если терминал условия имеет вид **Continue if True**, то поддиаграмма цикла While Loop будет выполняться до тех пор, пока на терминал условия не поступит значение FALSE. Вы можете также использовать инструмент Positioning и щелкнуть им терминал условия, чтобы изменить его вид.



Кроме того, с помощью терминала условия цикла While Loop Вы можете произвести типовую обработку ошибок. Если Вы подсоедините кластер ошибок (error cluster) к терминалу условия, то фактически из этого кластера на терминал поступит только параметр **status**, который может принимать значения TRUE и FALSE. Кроме того, пункты контекстного меню **Stop if True** и **Continue if True** заменятся на **Stop if Error** (остановиться, если ошибка) и **Continue**

**while Error** (продолжить, пока ошибка). Более подробно о кластере ошибок и его обработке см. в разделе *Проверка и обработка ошибок* в Главе 6 *Запуск и отладка виртуальных приборов*.

- i** Терминал итерации – iteration terminal (выходной терминал), показанный слева, содержит значение, равное количеству завершенных итераций (повторов). Подсчет итераций всегда начинается с нуля. При этом в процессе выполнения первой итерации на выходе терминала итераций будет значение **0**.

Для передачи данных от текущей итерации к следующей итерации, добавьте сдвигающие регистры. Более подробно о создании сдвигающих регистров в циклах см. в разделе *Сдвигающие регистры и узел обратной связи в циклах* в настоящей Главе.

### Недопущение бесконечного числа итераций в циклах While Loop

Если Вы поместите булев элемент управления снаружи цикла While Loop, как показано на Figure 8-1, и этот элемент управления будет установлен в значение FALSE перед активизацией цикла, а терминал условия находится в состоянии **Stop if True**, то Вы получите бесконечный цикл. Вы также можете получить бесконечный цикл, если установите в элементе управления значение TRUE, а терминал условия при этом находится в состоянии **Continue if True**.

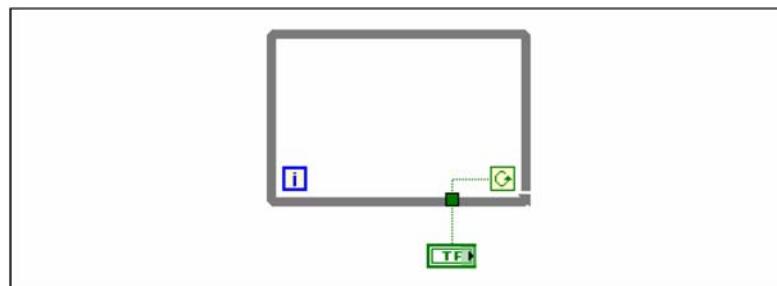


Figure 8-1. Бесконечный цикл While Loop

Изменение значения в элементе управления в процессе исполнения не остановит бесконечный цикл, поскольку это значение считывается только один раз, перед началом активизации цикла. Чтобы остановить бесконечный цикл Вам не останется ничего другого, как

прекратить исполнение ВП нажатием кнопки **Abort** на панели инструментов.

## Автоиндексация циклов

Если Вы подсоедините массив к входному туннелю циклов For Loop или While Loop, то сможете прочитать и обработать каждый элемент этого массива, активизировав автоиндексацию (auto-indexing). Более подробно о массивах см. в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.

Если Вы подсоедините массив к входному туннелю на границе цикла и активизируете автоиндексацию на этом туннеле, то элементы этого массива будут поступать в цикл по одному в каждой итерации цикла, начиная с первого элемента. Если же автоиндексация отключена, то массив поступит в цикл целиком до начала первой итерации. Когда Вы используете автоиндексацию для выходного туннеля, то выходной массив получает по одному новому элементу в каждой итерации цикла. Таким образом, размер автоиндексированного выходного массива всегда совпадает с числом повторений цикла.

Чтобы активизировать (enable) или отключить (disable) автоиндексацию, щелкните правой кнопкой туннель на границе цикла и выберите **Enable Indexing** или **Disable Indexing**. По умолчанию автоиндексация для цикла While Loop отключена.

В режиме автоиндексации цикл индексирует (извлекает) скалярные элементы из одномерного массива, одномерные массивы из двумерного массива и т.д. Обратная картина имеет место для выходных туннелей. Скалярные элементы накапливаются последовательно в одномерный массив, одномерные массивы накапливаются в двумерный массив и т.д.

## Автоиндексация для установки числа повторений в цикле For Loop

Если Вы активизируете автоиндексацию для массива, подключенного к входному терминалу цикла For Loop, то LabVIEW присвоит терминалу числа повторений значение равное размеру массива. В этой ситуации Вам нет необходимости подсоединять терминал числа повторений. Поскольку обычно циклы For Loop используются для поэлементной обработки массивов, то LabVIEW по умолчанию

нию активизирует автоиндексацию для всех массивов, которые Вы подключаете к циклу For Loop. Если Вам не нужна поэлементная обработка массивов, отключите автоиндексацию.

Если автоиндексация активна более чем для одного входного туннеля, или если терминал числа повторений подключен, то в качестве числа повторений берется наименьшее значение из возможных. Например, если в цикл входят два массива с автоиндексацией – из 10 и 20 элементов, соответственно, и Вы подаете значение 15 на терминал числа повторений, то цикл выполнится 10 раз, и из второго массива будут извлечены только 10 первых элементов. Если Вы строите кривые зависимостей по данным из двух источников на одном графике, и хотите отобразить первые 100 элементов, то подсоедините к терминалу числа повторений значение 100. Если один из источников данных выдаст только 50 элементов, то цикл выполнится 50 раз и извлечет только первые 50 элементов. Для определения числа элементов в массиве используйте функцию Array Size.

Когда Вы применяете автоиндексацию для выходного туннеля, выходной массив получает по одному новому элементу в каждой итерации цикла. Следовательно, длина автоиндексированного массива на выходе всегда равна числу повторений цикла. Например, если цикл выполняется 10 раз, то выходной массив будет содержать 10 элементов. Если отключить автоиндексацию выходного туннеля, то на следующий за этим циклом узел блок-диаграммы поступит только один элемент с последней итерации. Признаком активного состояния автоиндексации служат квадратные скобки внутри квадратиков, обозначающих туннели на границе цикла. Толстый проводник от выходного туннеля к следующему узлу также указывает на наличие автоиндексации в данном цикле. В случае автоиндексации проводник имеет большую толщину, поскольку по нему предается массив, а не скаляр.

### **Автоиндексация с циклами While Loop**

Если Вы активизируете автоиндексацию для массива, поступающего в цикл While Loop, то массив индексируется точно так же, как и в цикле For Loop. Однако, число повторений цикла While Loop не ограничивается размером массива, поскольку цикл While Loop повторяется до тех пор, пока удовлетворяется условие. Когда в процессе индексации входного массива цикл While Loop доходит до последнего элемента массива, то в качестве последующих значений в цикл будут поступать значения, принятые по умолчанию для типа

данных входного массива. Вы можете предотвратить поступление в цикл While Loop значений по умолчанию, используя функцию Array Size. Функция Array Size определяет количество элементов в массиве. Сконфигурируйте цикл While Loop так, чтобы он остановил выполнение, когда наступит итерация, номер которой совпадет с размером входного массива.



**Предупреждение.** Поскольку заранее определить размер выходного массива нельзя, использование автоиндексации на выходе для цикла For Loop является более эффективным, чем для цикла While Loop. Слишком большое число итераций может привести к переполнению памяти вашей системы.

## Использование циклов для построения массивов

В дополнение к применению циклов для чтения и обработки элементов массивов, циклы For Loop и While Loop можно использовать для построения массивов. Подсоедините выход ВП или функции, помещенных внутри цикла, к границе этого цикла. Если Вы используете цикл While Loop, щелкните правой кнопкой образовавшийся туннель и выберите из контекстного меню пункт **Enable Indexing**. В цикле For Loop автоиндексация активна по умолчанию. Выходом туннеля будет массив, каждый элемент которого будет содержать выходное значение ВП или функции, полученные на соответствующей итерации цикла.

Примеры построения массивов можно найти в библиотеке examples\general\arrays.llb.

## Сдвигающие регистры и узел обратной связи в циклах

Для передачи значений из одной итерации цикла For Loop или While Loop к следующим итерациям используйте либо сдвигающий регистр (shift register), либо узел обратной связи (Feedback Node).



### Сдвигающие регистры

Используйте сдвигающие регистры в тех случаях, когда Вам нужно передать данные из предыдущих итераций цикла. Сдвигающий регистр имеет вид пары терминалов, как показано слева, каждый точно друг против друга на противоположных вертикальных сторонах границы цикла. Правый терминал содержит стрелку, направленную вверх, и принимает для хранения данные по завершению текущей

итерации. К началу следующей итерации LabVIEW сдвигает эти данные в сдвигающем регистре, и они появляются на выходе терминала на левой границе цикла. Чтобы создать сдвигающий регистр, щелкните правой кнопкой левую или правую границу цикла и выберите из контекстного меню **Add Shift Register**.

Сдвигающий регистр может передавать данные любого типа и автоматически адаптируется к типу данных объекта, который подключается к нему первым. Данные, которые Вы подсоединяете к входам одного сдвигающего регистра, должны быть одного и того же типа.

Инициализируйте сдвигающий регистр, присоединяя к терминалам сдвигающего регистра на левой стороне цикла элементы управления или константы. Инициализация сдвигающего регистра устанавливает начальные значения, которые будут находиться в сдвигающем регистре при первой итерации цикла после запуска ВП. Если регистр не инициализировать, то на первой итерации цикла в качестве значений регистра будут использоваться значения, оставшиеся после завершения этого цикла при предыдущем запуске ВП, либо значения по умолчанию для данного типа данных, если ВП до этого ни разу не запускался.

Используйте цикл с неинициализированным сдвигающим регистром, для многократного запуска ВП таким образом, чтобы при каждом запуске ВП начальное состояние сдвигающего регистра совпадало с последним значением из предыдущего выполнения. Используйте неинициализированный сдвигающий регистр, чтобы сохранить информацию о состоянии между последовательными запусками ВП. После завершения цикла последнее значение, сохраненное в сдвигающем регистре, появляется на выходе правого терминала.

Вы можете создать несколько сдвигающих регистров для одного цикла. Если имеется несколько операций, которые используют значения с предыдущих итераций, то создайте несколько сдвигающих регистров, чтобы сохранить данные от каждого из этих процессов в данной структуре.

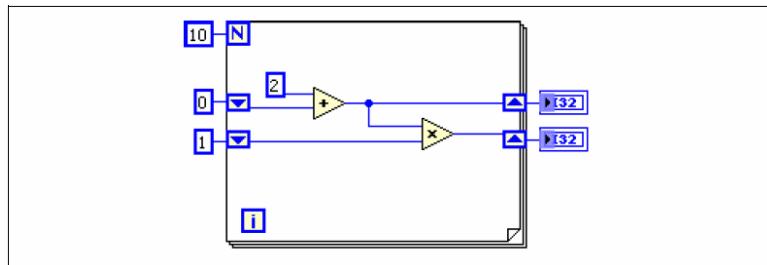


Figure 8-2. Несколько сдвигающих регистров

### Многоступенчатые сдвигающие регистры

Многоступенчатые (stacked) сдвигающие регистры дают доступ к данным от нескольких предыдущих итераций. Чтобы создать многоступенчатый сдвигающий регистр щелкните правой кнопкой левый терминал и выберите из контекстного меню **Add Element**. Многоступенчатый сдвигающий регистр запоминает значения от нескольких предыдущих итераций и передает эти значения к следующим итерациям.

Многоступенчатый сдвигающий регистр может быть только на левой стороне цикла, поскольку правый терминал только получает данные от текущей итерации.

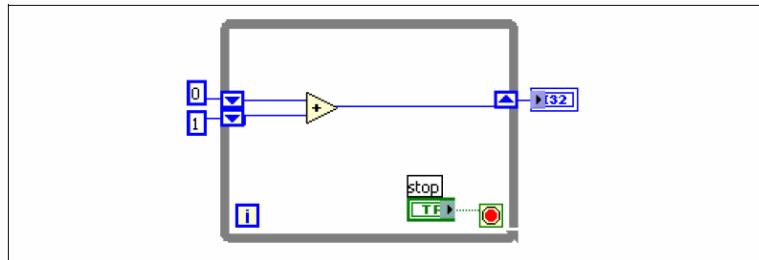


Figure 8-3. Многоступенчатый сдвигающий регистр

Если Вы добавите один элемент к левому терминалу, то значения от двух последних итераций будут передаваться к следующей итерации, причем значение с самой последней итерации будет храниться в верхнем терминале сдвигающего регистра. Нижний терминал хранит данные, поступившие в него из предыдущей итерации.

### **Замещение сдвигающих регистров туннелями**

Для замещения сдвигающего регистра туннелем, когда отпадает необходимость в передаче данных от одной итерации цикла к другой, щелкните правой кнопкой сдвигающий регистр и выберите из контекстного меню **Replace with Tunnels**.

Если Вы замещаете туннелем выходной терминал сдвигающего регистра в цикле For Loop, то соединение с любым узлом снаружи цикла разорвется, поскольку в цикле For Loop индексация по умолчанию активна. Щелкните правой кнопкой туннель и выберите из контекстного меню **Disable Indexing at Source**, чтобы отключить индексацию и автоматически исправить разорванный проводник. Если же Вы хотите, чтобы индексация оставалась активной, удалите неисправный проводник и соединенный с ним индикаторный терминал, щелкните правой кнопкой туннель и выберите **Create Indicator**.

Более подробно об индексации в циклах см. раздел *Автоиндексация циклов* в настоящей Главе.

### **Замещение туннелей сдвигающими регистрами**

Для замещения туннеля сдвигающим регистром, при необходимости передавать данные из одной итерации в следующую, щелкните правой кнопкой туннель и выберите из контекстного меню **Replace with Shift Register**. Если на противоположной стороне границы цикла туннель отсутствует, то LabVIEW создаст пару терминалов сдвигающего регистра. Если же туннель на противоположной стороне границы цикла существует, то LabVIEW заместит туннель, который Вы щелкнули, терминалом сдвигающего регистра, а курсор превратится в иконку сдвигающего регистра. Щелкните туннель на противоположной стороне цикла, чтобы заместить туннель на сдвигающий регистр, или щелкните блок диаграмму, чтобы поместить сдвигающий регистр на границе строго напротив другого терминала сдвигающего регистра. Если терминал сдвигающего регистра появляется позади туннеля, сдвигающий регистр окажется не присоединенным.

Если Вы преобразуете туннель с активной индексацией в сдвигающий регистр на границе цикла While Loop, то соединение с любым узлом снаружи цикла разорвется, поскольку сдвигающий регистр не может быть автоиндексированным. Удалите неисправный проводник, соедините такой выходной проводник с права от сдви-

гающего регистра с другим туннелем, щелкните его правой кнопкой, выберите из контекстного меню **Enable Indexing** и соедините туннель с узлом.

Более подробно об индексации в циклах см. раздел *Автоиндексация циклов* в настоящей Главе.

## Узел обратной связи



Узел обратной связи (Feedback Node), показанный слева, появляется автоматически в циклах For Loop или While Loop только тогда, когда Вы подсоединяете выход ВПП, функции или группы ВПП или функций к входу того же ВП, функции или группы. Подобно сдвигающему регистру узел обратной связи запоминает данные, когда цикл завершает итерацию, посыпает это значение на следующую итерацию цикла. Таким образом могут передаваться данные любого типа. Стрелка на узле обратной связи показывает направление потока данных.

Вы также можете выбрать узел обратной связи из подпалитры **Structures** и поместить его только внутри цикла For Loop или While Loop. Если Вы помещаете узел обратной связи на проводник перед ответвлением к туннелю, то этот узел обратной связи будет передавать каждое значение в туннель. Если же Вы помещаете узел обратной связи на проводник после ответвления к туннелю, то этот узел обратной связи будет передавать каждое значение обратно к входу ВП или функции и затем передает последнее значение в туннель.

Например, цикл For Loop на Figure 8-4 повторяется 10 раз. Узел обратной связи передает значение от предыдущей итерации цикла к туннелю до того, как это значение попадет на вход функции Add. Значение в этом туннеле всегда является значением из предыдущей итерации. На последней итерации цикла узел обратной связи содержит последнее значение, которое в данном случае равно 45, но это значение не передается в туннель или в числовой индикатор. Когда этот ВП завершит работу, в числовом индикаторе будет значение 36, которое является значением не с последней итерации, а с предпоследней итерации цикла.

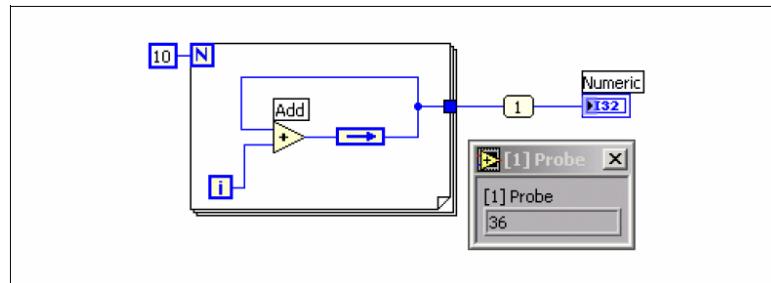


Figure 8-4. Прохождение предпоследнего значения за пределы цикла For Loop

Цикл на Figure 8-5 также повторяется 10 раз. Однако, узел обратной связи передает в каждой итерации цикла значение от предыдущей итерации только на вход функции Add. На последней итерации цикла узел обратной связи передает значение (36) от предыдущей итерации только на вход функции Add. Функция Add прибавляет к значению (9), которое генерирует терминал итераций, значение (36) поступившее из узла обратной связи и отправляет результат в туннель. Когда этот ВП завершит работу в числовом индикаторе будет **45**.

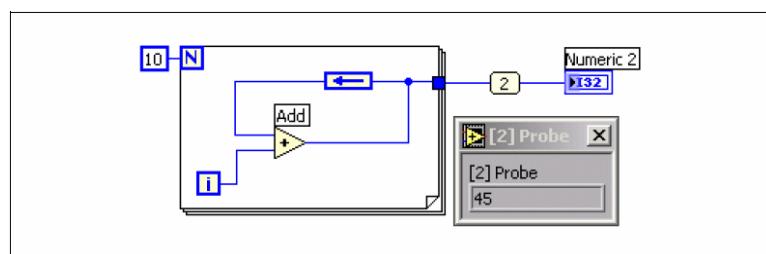


Figure 8-5. Прохождение последнего значения за пределы цикла For Loop

#### Инициализация узлов обратной связи

Чтобы добавить инициирующий терминал на границе цикла для его инициализации, щелкните правой кнопкой узел обратной связи и выберите из контекстного меню пункт **Initializer Terminal**. Когда Вы берете узел обратной связи с палитры или если Вы преобразуете инициированный сдвигающий регистр в узел обратной связи, то инициирующий терминал на границе цикла появится автоматически. Инициализация узла обратной связи устанавливает начальное значение, которое появляется на выходе узла обратной связи на первой итерации цикла после запуска ВП. Если Вы не инициализи-

руете узел обратной связи, то на его выход проходит последнее записанное в него значение или значение по умолчанию, принятое для типа данных, если цикл до этого еще ни разу не исполнялся. Если Вы оставите вход инициирующего терминала неподключенными, то при каждом запуске ВП начальным значением узла обратной связи будет его последнее значение в предыдущем запуске.

### **Замещение сдвигающего регистра узлом обратной связи**

Чтобы заместить сдвигающий регистр узлом обратной связи, щелкните правой кнопкой сдвигающий регистр и выберите из контекстного меню пункт **Replace with Feedback Node**. Чтобы заместить узел обратной связи сдвигающим регистром, щелкните правой кнопкой узел обратной связи и выберите из контекстного меню пункт **Replace with Shift Register**.

## **Управление синхронизацией**

Может возникнуть необходимость в управлении скоростью выполнения некоторых процессов, таких, например, как построение графиков. Вы можете использовать в цикле функцию ожидания (**Wait (ms)**). Эта функция приостанавливает переход к следующей итерации цикла на время, значение которого в миллисекундах подано на ее вход.

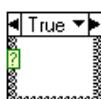
Более подробно об использовании функции ожидания в цикле для оптимизации затрат памяти см. раздел *Memory and Speed Optimization* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## **Структуры выбора и последовательности**

---

Структуры Case (вариант), Stacked Sequence (сложенная последовательность), Flat Sequence (развернутая последовательность) и Event (событие) могут содержать несколько поддиаграмм. Структура Case выполняет одну из нескольких поддиаграмм в зависимости от значения, поданного на ее вход. Структуры Stacked Sequence и Flat Sequence выполняют все свои поддиаграммы последовательно одна за другой. Структура Event выполняет свои поддиаграммы в зависимости от того, как пользователь взаимодействует с ВП.

## Структура Case



Структура Case (вариант), показанная слева, имеет две или больше поддиаграмм или вариантов. Единовременно только одна поддиаграмма является видимой и исполняется только один из вариантов. Входная величина структуры определяет, какой из вариантов (поддиаграмма) будет исполняться. Структура Case подобна операторам `select case` или `if...then...else` текстовых алгоритмических языков.



Селектор варианта на верхней границе структуры Case, показанный слева, содержит в центре имя управляющего значения, соответствующего данному варианту, и по бокам – стрелочки увеличения и уменьшения этого значения. Щелкните по этим стрелочкам, чтобы прокрутить все доступные варианты. Вы можете также щелкнуть стрелочку «вниз» на имени варианта и выбрать нужный вариант из спадающего меню.



Для определения варианта, который должен выполняться, подсоедините входную управляющую величину к входу селекторного терминала, показанного слева. К нему можно подсоединить целое число, булеву величину, строку или значение перечислительного типа. Селекторный терминал можно переместить на любое место в пределах левой границы структуры Case. Если к селекторному терминалу подключена булева величина, то структура имеет два варианта: вариант TRUE и вариант FALSE. Если к селекторному терминалу подключено целое число, строка или величина перечислительного типа, то такая структура может иметь любое количество вариантов.

Определяйте один из вариантов по умолчанию (default case), чтобы вылавливать ситуацию выхода управляющего значения за заданный диапазон. В противном случае Вам необходимо явно указать список всех возможных значений на селекторном терминале. Например, если селектор является целым числом, и Вы определяете варианты для значений 1, 2 и 3, то необходимо задать вариант по умолчанию, который будет выполняться, если входная величина будет равна 4 или любому другому правильному целому значению.

### Значения и типы данных селектора варианта

В метку селектора выбора Вы можете ввести одно значение или их список и интервалы значений. В качестве разделителя в списках

используется запятая. Для числовых величин задание интервала в виде 10..20 означает все числа от 10 до 20 включительно. Кроме того, можно использовать открытые интервалы. Например, ..100 представляет все числа, меньшие или равные 100, а 100.. представляет все числа большие или равные 100. Можно также комбинировать списки и интервалы, например ..5, 6, 7..10, 12, 13, 14. Когда Вы вводите значения, которые содержат перекрывающиеся интервалы в селекторной метке одного и того же варианта, то структура Case сама преобразует такую метку к более компактной форме. Предыдущий пример приведет к более компактной форме ..10, 12..14. Для строковых интервалов интервал a..c включает все a и b, но не c. Интервал a..c, c включает и последнюю величину c.

Когда Вы вводите в метку селектора варианта строку или значения перечислительного типа, то эти значения отображаются в кавычках, например "red", "green" и "blue". Однако, нет необходимости набирать кавычки при вводе, за исключением случаев, когда строка или перечислительное значение содержат запятую или символ интервала (", " или ".."). В строковых значениях можно использовать специальные коды с обратным слэшем для неалфавитных символов, такие как \r для возврата каретки, \n для перевода строки и \t для табуляции. Список специальных кодов с обратным слэшем можно найти в справочной системе *LabVIEW Help*.

Если Вы измените тип данных проводника, подсоединенного к селекторному терминалу структуры Case, то структура Case автоматически конвертирует значения селектора варианта к новому типу данных, если это возможно. Если Вы конвертируете строку в чистовое значение, например 19, в строку, то значением строки будет "19". Если же Вы конвертируете строку в чистовое значение, то LabVIEW конвертирует только те строковые значения, которые являются представлением числа. Другие значения остаются строками. Если Вы конвертируете число к булевому значению, то LabVIEW конвертирует 0 в FALSE, 1 в TRUE, а все остальные чистовые значения становятся строками.

Если Вы вводите селекторное значение, не соответствующее типу объекта, присоединенного к селекторному терминалу, то это значение будет иметь красный цвет, чтобы указать, что Вы должны удалить или изменить это значение перед исполнением этой структуры.

ры, и что ВП становится неисполняемым. Кроме того, поскольку возможны ошибки округления, присущие арифметике с плавающей точкой, числа с плавающей точкой нельзя использовать в качестве селекторных значений. Если же Вы присоедините величину с плавающей точкой к структуре Case, то LabVIEW округлит эту величину к ближайшему целому. Если Вы введете значение с плавающей точкой в метку селектора варианта, то это значение станет красным, что означает, что Вы должны удалить или изменить это значение прежде, чес структура сможет исполниться.

### **Входные и выходные туннели**

Для структуры Case можно создавать несколько входных и выходных туннелей. Входы доступны для всех вариантов, но не обязательно все они должны использоваться в каждом варианте. Однако Вы должны определить каждый выходной туннель для каждого варианта. Когда Вы создаете выходной туннель в одном варианте, туннели появляются в том же месте на границе структуры во всех других вариантах. Если, по крайней мере, один туннель не подсоединен, все выходные туннели на этой структуре будут иметь вид белых квадратиков. Вы можете определить различные источники данных для одного и того же выходного туннеля в каждом варианте, но их типы данных должны быть совместимыми. Кроме того, Вы можете щелкнуть правой кнопкой некоторый выходной туннель и выбрать из контекстного меню пункт **Default If Unwired**, чтобы использовать для всех неподключенных туннелей значение по умолчанию, соответствующее типу данных данного туннеля.

### **Использование структур Case для обработки ошибок**

Когда Вы присоединяете кластер ошибок к селекторному терминалу структуры Case, метка селектора варианта отображает два варианта **Error** и **No Error**, а граница структуры Case изменяет цвет - на красный для **Error** и на зеленый для **No Error**. Структура Case выполняет соответствующую каждому варианту поддиаграмму в зависимости от состояния ошибки. Более подробно об обработке ошибок см. в разделе *Обработка ошибок* в Главе 6 *Запуск и отладка виртуальных приборов*.

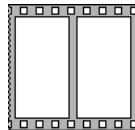
## **Структуры последовательности**

Структура последовательности содержит одну или несколько поддиаграмм, или кадров (frames), которые выполняются последовательно один за другим. Структуры последовательности редко ис-

пользуются в LabVIEW. Более подробно об использовании структур последовательности см. в разделах *Использование структур последовательности* и *Предостережение от чрезмерного использования структур последовательности* в настоящей Главе.

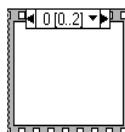
Имеется два типа структур последовательности: Flat Sequence (развернутая последовательность) и Stacked Sequence (сложенная последовательность).

### Структура Flat Sequence

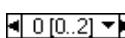


Структура Flat Sequence (развернутая последовательность), показанная слева, отображает все кадры одновременно и исполняет последовательно слева направо, пока не выполнится последний кадр. Используйте структуру Flat Sequence, чтобы исключить использование локальных переменных последовательности (sequence locals) и улучшить читаемость блок-диаграммы. Когда Вы добавляете или удаляете кадры в структуре Flat Sequence, размеры границ структуры автоматически изменяются. Для переупорядочивания кадров в структуре Flat Sequence используйте команды редактирования **cut** (вырезать) и **paste** (вставить), чтобы перенести содержимое из одного кадра в другой.

### Структура Stacked Sequence



Структура Stacked Sequence (сложенная последовательность), показанная слева, располагает кадры один над другим, поэтому одновременно Вы можете видеть только один кадр. Кадры исполняются в последовательном порядке: сначала кадр 0, затем кадр 1 и т.д., пока не выполнится последний кадр. Структура Stacked Sequence возвращает данные только после того, как выполнится последний кадр. Используйте структуру Stacked Sequence в тех случаях, когда Вы хотите сэкономить место на блок-диаграмме.



Идентификатор селектора последовательности, показанный слева, расположен на верхней границе структуры Stacked Sequence. Он содержит номер текущего кадра и интервал номеров существующих кадров. Используйте идентификатор селектора последовательности для перемещения между кадрами и переупорядочивания кадров. Метка кадра в структуре Stacked Sequence подобна метке селектора варианта в структуре Case. Метка кадра содержит номер кадра в центре и стрелочки инкремента и декремента по бокам. Щелкайте эти стрелки для прокрутки доступных кадров. Вы можете также щелкнуть стрелочку «вниз» в поле метки кадра и выбрать

нужный кадр из спадающего меню. Чтобы изменить порядок кадров в структуре Stacked Sequence, щелкните правой кнопкой границу кадра, выберите из контекстного меню пункт **Make This Frame** и выберите нужный номер для этого кадра.

В отличие от метки селектора варианта, нельзя вести свои обозначения для меток кадров. Когда Вы добавляете, удаляете или переставляете кадры в структуре Stacked Sequence, LabVIEW автоматически устанавливает в метках кадров нужные номера.

### Использование структур последовательности

Используйте структуры последовательности для управления порядком выполнения, когда естественная зависимость по данным отсутствует. Узел, который получает данные от другого узла, зависит от этого узла по данным и всегда выполняется после того, как этот узел завершит свое выполнение.

Внутри каждого кадра структуры последовательности, как и на основной блок-диаграмме, порядок выполнения узлов определяется зависимостью по данным. Более подробно о зависимости по данным см. в разделе *Зависимость по данным и искусственная зависимость по данным* в Главе 5 *Построение блок-диаграммы*.

Туннели на структурах Stacked Sequence могут иметь только один источник данных, в отличие от структур Case. Выход такого источника может поступать от любого кадра, но данные покидают структуру Stacked Sequence только тогда, когда будут выполнены все кадры, а не только этот конкретный кадр. Также как и в структуре Case данные от входных туннелей доступны во всех кадрах.

Чтобы передать данные от одного кадра к любому другому кадру структуры Stacked Sequence, используйте терминал переменной по-



следовательности (sequence local terminal), показанный слева. Стрелка, направленная наружу, появляется на терминале локальной переменной кадра, который содержит источник данных. Терминал на всех последующих кадрах содержит входящую стрелку, показывающую, что этот терминал служит источником данных для этих кадров. На всех кадрах, предшествующих кадру с источником данных подключенному к терминалу локальной переменной, использовать этот терминал нельзя.

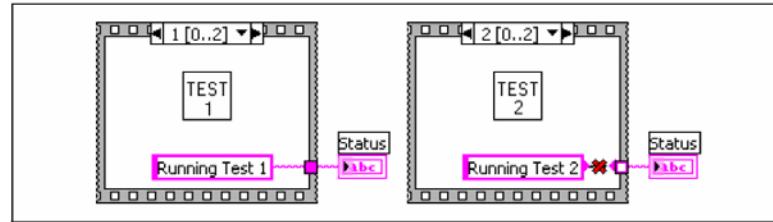
Примеры использования структур последовательности можно найти в библиотеке examples\general\structs.llb.

### **Предостережение от чрезмерного использования структур последовательности**

Чтобы получить преимущества присущего LabVIEW параллелизма, избегайте неоправданного использования структур последовательности. Структуры последовательности гарантируют порядок выполнения и запрещают параллельные операции. Например, асинхронные задачи, использующие устройства ввода/вывода, такие как PXI, GPIB, последовательные порты и DAQ-устройства, могут выполняться независимо от других операций, если не предотвратить этого с помощью структур последовательности. Структуры последовательности также скрывают фрагменты блок-диаграммы и изменяют естественный порядок потока данных слева на право.

Если Вам необходимо управлять порядком выполнения, рассмотрите уже существующую зависимость по данным между узлами. Например, Вы можете использовать ошибку ввода/вывода (errort I/O) для управления порядком выполнения ввода/вывода. Более подробно об ошибке ввода/вывода см. в разделе *Обработка ошибок* в Главе 6 *Запуск и отладка виртуальных приборов*.

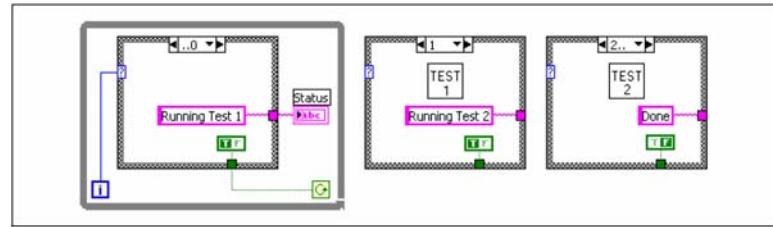
Кроме того, не используйте структуры последовательности для обновления состояния индикаторов из нескольких кадров структуры последовательности. Например, ВП используемый для реализации тестов, может иметь индикатор **Status**, который отображает наименование текущего исполняемого теста. Если каждый тест выполняется путем вызова в отдельном кадре соответствующего ВПП, то Вы не сможете обновлять содержимое этого индикатора из каждого кадра, что демонстрируется неисправным проводником в структуре Stacked Sequence на Figure 8-6.



**Figure 8-6.** Обновление состояния индикатора из разных кадров структуры Stacked Sequence

Поскольку все кадры структуры Stacked Sequence выполняются до того, как данные покинут эту структуру, только один кадр может назначить значение в индикатор **Status**.

Вместо этого используйте структуру Case и цикл While Loop, как показано на Figure 8-7.



**Figure 8-7.** Обновление состояния индикатора из разных вариантов структуры Case

Каждый вариант в структуре Case эквивалентен кадру структуры последовательности. Каждая итерация цикла While Loop выполняет следующий вариант. Индикатор **Status** отображает состояние ВП для каждого варианта. Индикатор **Status** обновляется в варианте, который предшествует варианту, в котором вызывается соответствующий ВПП, поскольку данные покидают структуру Case после каждого ее выполнения на очередной итерации цикла.

В отличие от структуры последовательности, структура Case может выдать данные для завершения цикла While Loop во время исполнения любого варианта. Например, если в процессе выполнения первого теста возникла ошибка, структура Case может выдать на терминал условия FALSE, чтобы завершить цикл. В противопо-

ложность этому, структура последовательности вынуждена выполнить все ее кадры, даже если произойдет ошибка.

### **Замещение структур последовательности**

Чтобы преобразовать структуру Flat Sequence в структуру Stacked Sequence, щелкните правой кнопкой структуру Flat Sequence и выберите из контекстного меню пункт **Replace with Stacked Structure**. Чтобы преобразовать структуру Stacked Sequence в структуру Flat Sequence, щелкните правой кнопкой структуру Stacked Sequence и выберите из контекстного меню пункт **Replace with Flat Sequence**.

## 9. Событийно управляемое программирование

---

LabVIEW это среда потокового программирования, в которой порядок выполнения элементов блок-диаграммы определяется потоком данных. Наличие событийно управляемого программирования расширяет потоковую среду LabVIEW, что делает доступным прямое взаимодействие с пользователем через лицевую панель либо через другие асинхронные механизмы для влияния на выполнение блок-диаграммы.



**Примечание.** Возможность событийно управляемого программирования доступна только в пакетах LabVIEW Full Development System и LabVIEW Professional Development System. Вы сможете запускать ВП, построенные с этой возможностью, в LabVIEW Base Package, но не сможете реконфигурировать событийно управляемые компоненты.

---

**Более подробно...**

Более подробно относительно использования событий в вашем приложении см. справочную систему *LabVIEW Help*.

---

### Что такое события?

---

Событие (event) – это асинхронное уведомление о том, что нечто произошло. События могут порождаться пользовательским интерфейсом, внешними устройствами ввода/вывода или другими частями этой же программы. События пользовательского интерфейса включают щелчки мыши, нажатия клавиш и т.д. События внешнего ввода/вывода включают аппаратные таймеры и триггеры, которые сигнализируют, когда завершается сбор данных или когда возникают ошибочные ситуации. Другие события могут генерироваться программно и используются для взаимодействия между различными частями программы. LabVIEW поддерживает события, генерируемые пользовательским интерфейсом и генерируемые программно, но не поддерживает события внешнего ввода/вывода.

В событийно управляемой программе события, происходящие в системе, оказывают непосредственное влияние на исполняемый по-

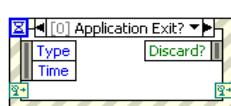
ток. В противоположность этому, процедурная программа выполняется в заранее определенном последовательном порядке. Событийно управляемые программы обычно включают цикл, который ожидает наступление следующего события. Реакция программы на каждое событие определяется кодом, написанным для каждого специфического события. Порядок выполнения событийно управляемой программы определяется тем, какие события происходят и в каком порядке они появляются. Некоторые фрагменты такой программы могут выполняться часто, потому что события, которые они обрабатывают, происходят часто, а другие фрагменты программы могут вообще не выполняться, поскольку соответствующие события никогда не происходят.

### **Зачем использовать события?**

Используйте события пользовательского интерфейса LabVIEW для синхронизации действий пользователя на лицевой панели с выполнением блок-диаграммы. События позволяют выполнить определенный управляемый событием вариант всякий раз, когда пользователь выполнит заданное действие. Без событий блок-диаграмма должна была бы опрашивать объекты лицевой панели в цикле и проверять, не произошли ли какие-либо изменения. Опрос лицевой панели требует больших затрат процессорного времени и может пропустить обнаружение некоторых изменений, если они происходят слишком часто. Используя события для отклика на указанные действия пользователя, Вы избавляетесь от необходимости опрашивать лицевую панель, чтобы определить, какое действие совершил пользователь. Взамен LabVIEW активно уведомляет блок-диаграмму всякий раз, когда происходит указанное взаимодействие. Использование событий уменьшает требования программы к процессорному времени, упрощает код блок-диаграммы и гарантирует, что блок-диаграмма сможет откликнуться на все действия, которые сделает пользователь.

Используйте программно генерируемые события для взаимодействия между различными частями программы, которые не имеют зависимости по потоку данных. Программно генерируемые события, как и события пользовательского интерфейса, обладают многими достоинствами, и могут разделять между собой части событийно управляемого кода, делая его легким для реализации продвинутых архитектур, таких как автоматы с очередью состояний.

## Компоненты структуры Event



Используйте структуру Event (событие), показанную слева, для обработки событий в ВП. Структура Event работает подобно структуре Case с встроенной функцией Wait On Notification (ожидать при уведомлении). Структура Event

может иметь несколько вариантов, каждый из которых является отдельной процедурой обработки события. Вы можете конфигурировать каждый вариант для обработки одного или более событий, но только одно из них может наступать единовременно. Когда структура Event выполняется, она ожидает наступления хотя бы одного из указанных событий, затем выполняется вариант, соответствующий этому событию. Структура Event заканчивает выполнение после обработки в точности одного события. При этом не возникает зацикливания при обработке нескольких событий. Подобно функции Wait On Notification структура Event задает временной интервал (time out) в течение которого ожидается уведомление о событии. Когда это наступает, выполняется специфический вариант Timeout.



Метка селектора события в верхней части структуры Event, показанная слева, указывает, какие события вызывают выполнение отображаемого в данный момент варианта. Для просмотра других вариантов щелкните стрелочку «вниз», расположенную сразу за именем события, и выберите из выпадающего меню нужный вариант. Вы можете также установить курсор над меткой селектора и, удерживая клавишу <Ctrl>, вращать колесико мыши.



Терминал Timeout в левом верхнем углу структуры Event, показанный слева, задает количество миллисекунд, в течение которых ожидается наступление события. По умолчанию эта величина задается равной (-1), что означает неограниченное время ожидания наступления события. Если Вы подсоедините значение к терминалу Timeout, то Вы должны определить вариант Timeout.



Узел данных события (Event Data Node), показанный слева, внешне напоминает функцию Unbundle By Name (разобрать кластер по именам). Этот узел прикреплен изнутри к левой границе каждого варианта структуры Event. Через этот узел поступают данные о событии, когда оно происходит. Вы можете изменять размеры этого узла по вертикали, чтобы добавить выходы, и можете назначить каждому выходному пункту этого узла нужный элемент данных о

событий. Этот узел выдает разные элементы данных в разных вариантах структуры Event в зависимости от того, для обработки какого события (событий) вы конфигурируете данный вариант. Если Вы настраиваете один вариант для обработки нескольких событий, то узел данных события выдает только те элементы данных, которые являются общими для всех этих событий.



Узел фильтра события (Event Filter Node), показанный слева, подобен узлу данных события. Этот узел прикреплен изнутри к правой границе вариантов для событий с фильтром (в названии таких событий присутствует вопросительный знак). Узел фильтра события определяет подмножество данных, доступных в узле данных события и которые можно изменять в процесс обработки события. Этот узел отображает различные данные в зависимости от того, для обработки какого события (событий) вы конфигурируете данный вариант. По умолчанию (если не подключены) элементы данных узла фильтра событий замещаются соответствующими элементами данных из узла данных события. Если Вы не подсоедините значения к входам узла фильтра событий, то эти элементы данных останутся неизмененными. Более подробно о фильтре событий см. в разделе *Уведомление и фильтр событий* в настоящей Главе.



Терминалы динамических событий (Dynamic Event Terminals), один из которых показан слева, отображаются, если щелкнуть правой кнопкой структуру Event и выбрать из контекстного меню пункт **Show Dynamic Event Terminals**. Эти терминалы используются только при наличии регистрации динамической событий. Более подробно об этих терминалах см. в разделах *Динамическая регистрация событий* и *Динамическая модификация регистрации* в настоящей Главе.



**Примечание.** Подобно структуре Case, структура Event может иметь туннели. Однако по умолчанию Вы не должны подсоединять выходной туннель в каждом варианте структуры Event. Все не подсоединеные туннели используют значения по умолчанию для типа данных этих туннелей. Чтобы вернуться к поведению по умолчанию, принятому для структуры Case, при котором туннели должны быть обязательно подсоединены для всех вариантов, щелкните правой кнопкой туннель и снимите птичку с пункта контекстного меню **Use Default If Unwired**.

Более подробно о значениях по умолчанию для разных типов данных см. в справочной системе *LabVIEW Help*.

### **Уведомляющие и фильтруемые события**

Существуют два типа событий пользовательского интерфейса – уведомляющие и фильтруемые.

Уведомляющие события показывают, что пользователь уже совершил некоторое действие, например, изменил значение элемента управления. Используйте уведомляющие события для реагирования на событие после того, оно произошло и обработано в LabVIEW. Вы можете настроить любое количество структур Event для реакции на одно и то же уведомляющее событие для одного и того же объекта. Когда событие наступает, LabVIEW параллельно отправляет копии этого события в каждую структуру Event, настроенную для обработки данного события.

Фильтруемые события информируют Вас о том, что пользователь выполнил некоторое действие, до того, как LabVIEW обработает их. Это позволяет Вам настроить, каким образом программа будет реагировать на действия пользователя. Используйте фильтруемые события, чтобы участвовать в их обработке, вероятно изменяя существующую по умолчанию реакцию на данное событие. В варианте структуры Event для фильтруемого события Вы можете подтвердить или изменить данные о событии перед тем, как LabVIEW завершит его обработку, либо Вы можете полностью удалить (discard) событие, чтобы предотвратить его воздействие на ВП. Например, Вы можете настроить структуру Event на удаление события **Panel Close?**, что не позволит пользователю интерактивно закрыть лицевую панель ВП. Фильтруемые события содержат в конце имени знак вопроса как, например **Panel Close?**, чтобы помочь Вам отличать их от уведомляющих событий. Большинству фильтруемых событий соответствуют уведомляющие события с тем же именами, но без знака вопроса, который LabVIEW генерирует после фильтруемого события, если в структуре Event отсутствует вариант, в котором данное событие удаляется (discard).

Как и для уведомляющих событий, Вы можете настроить любое количество структур Event для реагирования на одно и то же фильтруемое событие от некоторого элемента. Однако LabVIEW посыпает фильтруемые события последовательно к каждой из структур Event, настроенных на данное событие. Порядок, в кото-

ром LabVIEW рассыпает такие события в цепочку структур Event, зависит от порядка, в котором эти события были зарегистрированы. Более подробно о регистрации событий см. в разделе *Использование событий в LabVIEW* в настоящей Главе. Каждая структура Event должна завершить свою обработку события до того, как LabVIEW сможет уведомить следующую структуру Event. Если обработка события в структуре Event изменяет какие-либо данные этого события, то к следующей в цепочке структуре Event будут переданы измененные данные. Если какая-нибудь структура Event в такой цепочке удалит (discard) некоторое событие, то это событие не будет передаваться во все последующие структуры Event этой цепочки. LabVIEW завершает обработку действия пользователя, запускающего событие, только после того, как все настроенные на него структуры Event обработают это событие без его удаления.



**Примечание.** National Instruments рекомендует использовать фильтруемые события только тогда, когда Вы хотите принять участие в обработке действий пользователя либо путем удаления события, либо путем изменения данных о событии. Если Вы хотите только узнать, что пользователь выполнил некоторое действие, используйте уведомляющие события.

Варианты структуры Event, которые настроены на обработку фильтруемых событий, имеют узел фильтра события, как это описано в разделе *Компоненты структуры Event* настоящей Главы. Вы можете изменить данные о событии подсоединяя, новые значения к этим терминалам. Если вход не подключен, то соответствующий ему элемент данных останется без изменений. Вы можете полностью удалить событие, подав значение TRUE на терминал **Discard?**.



**Примечание.** Один вариант (case) структуры Event не может обрабатывать и уведомляющие и фильтруемые события. В одном варианте всегда можно обрабатывать несколько уведомляющих событий, но несколько фильтруемых событий можно обрабатывать только в том случае, если элементы данных о событии одинаковы для всех этих событий.

## Использование событий в LabVIEW

LabVIEW может вырабатывать множество различных событий. С целью исключения выработки нежелательных событий используйте

регистрацию событий, чтобы указать события, относительно которых Вы хотите получать уведомления. LabVIEW поддерживает две модели регистрации событий – статическую и динамическую.

Статическая регистрация позволяет указывать, какие события на лицевой панели ВП Вы хотите обрабатывать в каждом варианте структуры Event на блок-диаграмме этого ВП. LabVIEW автоматически регистрирует эти события при запуске ВП. Каждое события связано с элементом управления на лицевой панели ВП, с окном лицевой панели как с единым целым или с приложением. Нельзя статически конфигурировать структуру Event для лицевой панели другого ВП. Конфигурирование является статическим, поскольку в процессе работы ВП Вы не можете изменять, какие события будут обрабатываться структурой Event. Более подробно о статической регистрации см. в разделе *Статическая регистрация событий* в настоящей Главе.

Динамическая регистрация избавлена от ограничений статической регистрации за счет объединения регистрации событий с сервером ВП, который позволяет использовать ссылки на приложение, на ВП и на элемент управления, чтобы указать во время выполнения на объекты, для которых Вы хотите вырабатывать события. Динамическая регистрация обеспечивает большую гибкость в управлении тем, какие события должны вырабатываться и когда. Однако, динамическая регистрация сложнее статической, поскольку она требует использования ссылок сервера ВП с функциями блок-диаграммы для явной регистрации и отмены регистрации событий вместо автоматической регистрации на основе информации, заданной при конфигурировании структуры Event. Более подробно о динамической регистрации см. в разделе *Динамическая регистрация событий* в настоящей Главе.



**Примечание.** Вообще-то LabVIEW вырабатывает события пользовательского интерфейса только как результат прямого взаимодействия пользователя с активной лицевой панелью. LabVIEW не генерирует такие события, как Value Change (изменение значения), когда Вы используете сервер ВП, глобальные переменные, локальные переменные, DataSocket и т.п. Более подробно о программном генерировании события Value Change см. разделе *Value (Signaling) property* справочной системы *LabVIEW Help*. Во многих случаях Вы можете использовать программно генерируемые события вместо очередей и уведомлений.

Данные о событии (event data) всегда включают временную метку (time stamp), перечислительный номер (enumeration) произошедшего события и ссылку сервера ВП (VI Server reference) на объект, вызвавший это событие. Временная метка – это отсчет времени в миллисекундах, который можно использовать для вычисления времени между двумя событиями или для определения порядка возникновения событий. Ссылка на объект, вызвавший событие, относится к классу сервера ВП для этого объекта. События группируются в классы в соответствии с типом объекта, вызвавшего событие. Таким объектом может быть приложение, ВП или элемент управления. Если один вариант обрабатывает несколько событий для объектов из различных классов сервера ВП, то ссылка будет иметь тип родительского класса, общего для всех объектов. Например, если создается один вариант структуры Event для обработки событий от цифрового числового элемента управления (digital numeric control) и от цветового клина (color ramp control), то типом ссылки на эти элементы управления как источники события будет Numeric, поскольку и цифровой числовой элемент управления и цветовой клин принадлежат классу Numeric. Более подробно об использовании классов сервера ВП см. в Главе 17 *Программное управление ВП*.



**Примечание.** Если Вы одно и тоже событие регистрируете и в класс VI (ВП) и в класс Control (элемент управления), то LabVIEW генерирует первым событие VI. LabVIEW генерирует события класса Controls для объектов-контейнеров, таких как кластеры, до того, как будут сгенерированы события для объектов этих контейнеров. Если вариант структуры Event для события VI или для события Control объекта-контейнера удаляет это событие, то последующие события не генерируются.

Каждая структура Event и каждый узел Register For Events (регистрация событий) на блок-диаграмме имеют свою очередь, которую LabVIEW использует для запоминания событий. Когда событие происходит, LabVIEW помещает копию этого события в каждую очередь, зарегистрированную для этого события. Структура Event обрабатывает все события из своей очереди и события из очередей всех узлов Register For Events, которые присоединены к терминалам динамических событий этой структуры Event. LabVIEW использует эти очереди, чтобы гарантировать надежное поступление событий ко всем зарегистрированным структурам Event в порядке их возникновения.

По умолчанию LabVIEW блокирует лицевую панель, которая содержит объекты, генерирующие события, пока все структуры Event не завершат обработку события. Пока лицевая панель заблокирована, LabVIEW не обрабатывает действия на лицевой панели, но по-мешает запросы на эти действия в буфер и обрабатывает их после разблокирования лицевой панели. Блокирование лицевой панели не действует на некоторые действия, такие как перемещение окна, взаимодействие с полосой прокрутки и щелчок по кнопке **Abort**. Вы можете отключить блокировку лицевой панели для уведомляющих событий. Для этого нужно удалить птичку с соответствующей опции в диалоговом окне **Edit Events**, которое вызывается с помощью контекстного меню структуры Event. Блокировка лицевой панели для фильтруемых событий должна быть активной. Это гарантирует неизменность внутреннего состояния LabVIEW до завершения обработки уже поступивших событий.

LabVIEW может генерировать события, даже если нет структуры Event, ожидающей его обработку. Поскольку структура Event в каждый момент времени может обрабатывать только одно событие, для надежной обработки всех наступивших событий поместите структуру Event в цикл While Loop.



**Предупреждение.** Если структура Event для обработки событий отсутствует и блокировка лицевой панели активна, то пользовательский интерфейс такого ВП перестает откликаться на действия пользователя. В этом случае для остановки ВП щелкните кнопку **Abort**. Вы можете отключить блокировку лицевой панели, для чего следует щелкнуть правой кнопкой структуру Event и снять птичку с опции **Lock front panel until the event case for this event complete** в диалоговом окне **Edit Events**. Для фильтруемых событий отключить блокировку лицевой панели нельзя.

Более подробно о предосторожностях при работе со структурой Event и о доступных событиях см. в справочной системе *LabVIEW Help*.

### Статическая регистрация событий

Статическая регистрация возможна только для событий пользовательского интерфейса. Для конфигурирования структуры Event, которая обрабатывает статически регистрируемые события, используйте диалоговое окно **Edit Events**. Выберите источник события,

которым может быть приложение, ВП или отдельный элемент управления. Выберите конкретное событие из списка генерируемых выбранным источником, например, Panel Resize, Value Change и т.п. Выберите номер варианта, который будет обрабатывать данное событие в соответствии с требованиями Вашего приложения. Более подробно о диалоговом окне **Edit Events** и о том, как статически зарегистрировать события, см. в справочной системе *LabVIEW Help*.

LabVIEW статически регистрирует события автоматически и прозрачно, когда Вы запускаете ВП, содержащий структуру Event. LabVIEW генерирует события для ВП только тогда, когда ВП запускается или когда другой запущенный ВП вызывает данный ВП в качестве ВПП.

Когда Вы запускаете ВП, то LabVIEW устанавливает этот ВП верхнего уровня и всю иерархию ВПП, которые он вызывает со своей блок-диаграммы, в состояние исполнения, которое называется защищенным (reserved). Вы не можете редактировать ВП или щелкать кнопку **Run**, пока ВП находится в защищенном состоянии, поскольку такой ВП может вызываться в качестве ВПП в любой момент времени, пока его родительский ВП запущен. Когда LabVIEW устанавливает ВП в защищенное состояние, автоматически регистрируются события, которые Вы статически сконфигурировали во всех структурах Event на блок-диаграмме этого ВП. Когда ВП верхнего уровня завершает свое исполнение, LabVIEW устанавливает его и всю иерархию его ВПП в состояние простого (idle execution state) и автоматически снимает с регистрации его события.

Примеры использования статической регистрации событий см. в библиотеке `examples\general\uievents.llb`.

### **Динамическая регистрация событий**

Динамическая регистрация событий дает Вам полный контроль над тем, какие события генерирует LabVIEW и когда это происходит. Используйте динамическую регистрацию событий, если Вы хотите, чтобы генерация событий происходила в части приложения или если Вы хотите изменять какие ВП или элементы управления должны генерировать события в процессе выполнения вашего приложения. При динамической регистрации Вы можете обрабатывать события в ВПП, а не в ВП, который порождает эти события.

Обработка динамически регистрируемых событий включает следующие основные шаги:

1. Получите ссылку сервера ВП на объекты, для которых Вы хотите обрабатывать события.
2. Зарегистрируйте события для объектов, путем записи ссылок сервера ВП в узел регистрации событий (Register For Events node).
3. Заключите структуру Event в цикл While Loop для обработки событий от этих объектов, пока не наступит условие завершения цикла.
4. Используйте функцию Unregister For Events, чтобы завершить регистрацию событий.

Для динамической регистрации события от объекта сначала получите ссылку сервера ВП на объект. Для получения ссылок на приложение или на ВП используйте функции Open Application Reference или Open VI Reference. Для получения ссылки на элемент управления, используйте узел свойств (Property Node), чтобы запросить у ВП ссылки на его элементы управления, или щелкните правой кнопкой элемент управления и выберите из контекстного меню пункт **Create»Reference**, чтобы создать константу ссылки на элемент управления. Более подробно об использовании ссылок сервера ВП см. в разделе *Ссылки на приложение и на ВП* в Главе 17 *Программное управление ВП*.

Используйте функцию Register For Events для динамической регистрации событий. Вы можете изменить размер узла Register For Events, чтобы показать входы одного или больше источников событий. Подсоедините на каждый вход источника событий ссылку на приложение, ВП или элемент управления. Щелкните правой кнопкой любой вход и выберите из контекстного меню в пункте **Events** событие, которое Вы желаете зарегистрировать. Перечень событий, из которых можно выбирать, зависит от типа ссылки сервера ВП, которую Вы подсоедините к входу источника событий. В пункте контекстного меню **Events** перечисляются те же события, которые можно увидеть в диалоговом окне **Edit Events**, когда Вы регистрируете события статически. Когда узел Register For Events выполняется, LabVIEW регистрирует указанные события для объектов, свя-

занных с каждым входом источника событий. Как только Вы зарегистрируете события, LabVIEW будет ставить их в очередь по мере возникновения, пока структура Event не обработает их. Действия, совершаемые до выполнения этого узла, не вырабатывают событий, если другой объект предварительно не зарегистрирован для этого.



**Примечание.** В отличие от узла свойств (Property Node) узел Register For Events не требует, чтобы Вы подсоединяли верхний левый вход. Используйте этот вход, если только Вы хотите модифицировать существующую регистрацию. Более подробно о перерегистрации см. в разделе *Динамическая модификация регистрации* в настоящей Главе.

Терминалы динамических событий, для появления которых следует щелкнуть правой кнопкой структуру Event и выбрать из контекстного меню пункт **Show Dynamic Event Terminals**, ведут себя подобно сдвигающим регистрам. Левый терминал получает регистрационную ссылку на событие или кластер таких ссылок. Если Вы не подсоедините внутренний правый терминал, то он будет нести те же данные, что и левый терминал. Однако, Вы можете подсоединить регистрационную ссылку на событие или кластер таких ссылок к внутреннему правому терминалу через узел Register For Events и модифицировать регистрацию события динамически. Более подробно об использовании терминалов динамических событий см. в разделе *Динамическая модификация регистрации* в настоящей Главе.

На выходе узла Register For Events появляется ссылка на зарегистрированное событие, которая имеет строгий тип данных, который LabVIEW использует для передачи информации о регистрации событий в структуру Event. Вы можете просмотреть зарегистрированные события в окне **Context Help**, перемещая курсор над ссылкой на зарегистрированное событие. После того, как Вы настроите узел Register For Events, подсоедините его выход **event registration refnum** к терминалу динамического события на левой стороне структуры Event и сконфигурируйте эту структуру Event на обработку зарегистрированных событий. Исключите ответвления от проводника ссылки на зарегистрированное событие, поскольку это позволит нескольким структурам Event извлекать события из одной очереди и может привести к эффекту гонок, который может вызвать непредсказуемое поведение.

Чтобы сконфигурировать структуру Event для обработки динамически регистрируемых событий, используйте диалоговое окно **Edit Events**. Раздел этого диалогового окна **Events Source** содержит подзаголовок **Dynamic**, в котором перечислены все динамически регистрируемые источники событий. Имена этих источников событий те же самые, что и имена ссылок, которые подключены к узлу Register For Events, связанному с этой структурой Event, и перечисляются в том же порядке. Выберите нужный вам источник события из списка **Dynamic**. Заметьте, что имена событий, которые Вы зарегистрировали с помощью узла Register For Events, будут подсвечены в разделе **Events**. После того, как Вы выберите событие, создайте блок-диаграмму варианта для обработки данных этого события в соответствии с требованиями вашего приложения.

Чтобы остановить генерацию событий, подключите терминал динамического события на правой стороне структуры Event к входу **event registration refnum** функции Unregister For Events снаружи цикла While Loop, содержащего эту структуру Event. Когда функция Unregister For Events выполнится, LabVIEW снимет с регистрации все события, на которые указывает регистрационная ссылка событий, уничтожит очередь событий, связанную с этой же ссылкой, и удалит все события, оставшиеся в этой очереди. Если Вы не снимите события с регистрации, а пользователь выполнит действия, генерирующие события, после того, как цикл While Loop, содержащий структуру Event, завершится, то LabVIEW будет неопределенно ставить события в очередь. Это может привести к тому, что ВП перестанет откликаться на действия пользователя, если Вы сконфигурировали эти события для блокировки лицевой панели. В этом случае LabVIEW уничтожит очередь событий, когда ВП перейдет в ожидание.

LabVIEW автоматически снимает с регистрации все события, когда ВП верхнего уровня завершает свою работу. Однако, National Instruments рекомендует явно снимать события с регистрации, особенно в долго работающих приложениях, чтобы сохранить ресурс памяти.

#### Пример динамического события

Блок-диаграмма на Figure 9-1 демонстрирует использование динамической регистрации на примере события Mouse Enter в строковом элементе управления.

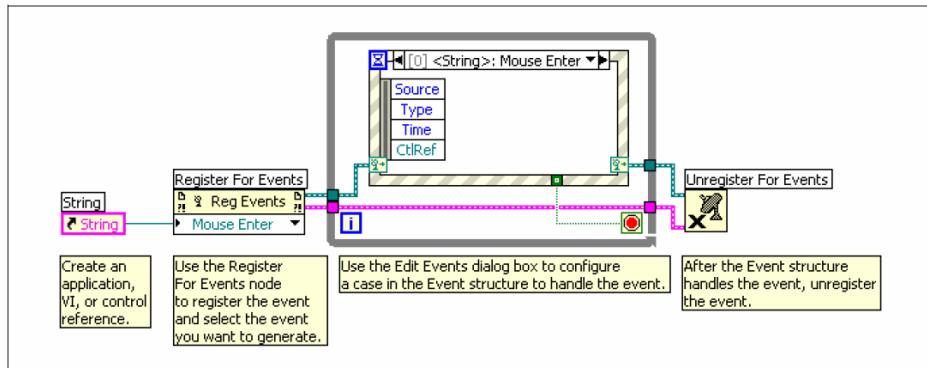


Figure 9-1. Динамически регистрируемые события

Более подробно о том, как динамически регистрировать события см. в справочной системе *LabVIEW Help*.

Примеры динамической регистрации событий приведены в библиотеке examples\general\dynamicevents.llb.

### Динамическая модификация регистрации

Если Вы регистрируете события динамически, то можете модифицировать информацию о регистрации во время исполнения, чтобы изменить объекты LabVIEW, генерирующие события. Подсоедините верхний левый вход **event registration refnum** узла Register For Events, если Вы хотите не создавать новую регистрацию, а собираетесь модифицировать уже существующую, связанную с присоединенной ссылкой. Когда Вы присоединяетесь к входу **event registration refnum**, узел автоматически изменяет размер, чтобы показать те же события на тех же типах ссылок, что Вы указали в узле Register For Events, который изначально создал регистрационную ссылку событий. Вы не сможете вручную изменять размер или перенастраивать эту функцию, пока вход **event registration refnum** подключен.

Если Вы присоединяете ссылку на объект к входу **event source** узла Register For Events и уже подключили вход **event registration refnum**, то этот узел заменится независимо от того, какая ссылка была раньше зарегистрирована через соответствующий вход **event source** исходного узла Register For Events. Вы можете присоединить константу Not a Refnum к входу **event source**, чтобы снять с регистрации отдельное событие. Если Вы оставляете вход **event source** не

присоединенным, то LabVIEW не изменяет регистрацию данного события. Используйте функцию Unregister For Events, если хотите снять с регистрации все события, связанные с данной регистрационной ссылкой.

Пример на Figure 9-2 показывает, как можно динамически во время исполнения изменить объекты LabVIEW, которые генерируют события. При исполнении этой блок-диаграммы LabVIEW регистрирует ссылку Numeric и ожидает события на элементе управления **Numeric**, связанном с этой ссылкой. Когда LabVIEW генерирует событие Value Change для элемента управления **Numeric**, вариант структуры Event, помеченный Numeric Value Change, выполняет узел Register For Events, чтобы изменить числовой элемент управления, зарегистрированный для события Value Change, с **Numeric** на **Numeric 2**. Если пользователь в последствии будет изменять значение в элементе управления **Numeric**, то LabVIEW не будет генерировать события Value Change. Однако, элемент управления **Numeric 2** будет генерировать события Value Change. Каждый раз, когда LabVIEW будет генерировать событие Value Change для элемента управления **Numeric 2**, будет исполняться узел Register For Events, но это не будет оказывать действия, поскольку элемент управления **Numeric 2** уже зарегистрирован для события Value Change.

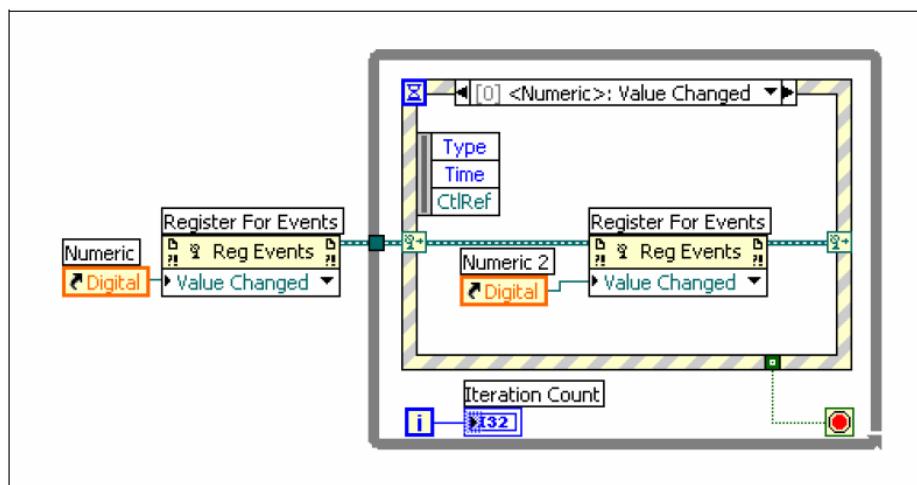


Figure 9-2. Динамически модифицируемая регистрация событий



**Примечание.** Нельзя динамически модифицировать события, зарегистрированные статически.

## Пользовательские события

Вы можете программно создавать свои собственные события, называемые пользовательскими событиями, чтобы передавать определенные пользователем сведения. Подобно очередям и уведомлениям, пользовательские события позволяют асинхронно взаимодействовать различным частям приложения. Вы можете обрабатывать в одной и той же структуре Event и события пользовательского интерфейса и программно генерируемые пользовательские события.

### Создание и регистрация пользовательских событий

Чтобы определить пользовательское событие, подсоедините объект блок-диаграммы, такой как терминал лицевой панели или константу, к функции Create User Event. Тип данных объекта определит тип данных пользовательского события. Если тип данных – кластер, то имя и тип каждого поля кластера определяет данные, которые переносятся с помощью пользовательского события. Если тип данных – не кластер, то пользовательское событие переносит одно значение того же типа, а метка объекта становится именем пользовательского события и единственного элемента данных.

Выход **user event out** функции Create User Event является ссылкой строгого типа, которая несет имя и тип данных пользовательского события. Подсоедините выход **user event out** функции Create User Event к входу **event source** узла Register For Events.

Обработка пользовательского события осуществляется тем же способом, что и обработка динамически регистрируемых событий пользовательского интерфейса. Подсоедините выход **event registration refnum** узла Register For Events к терминалу динамического события на левой стороне структуры Event. Используйте диалоговое окно **Edit Events** для конфигурирования варианта структуры Event для обработки этого события. Имя пользовательского события появляется под заголовком **Dynamic** в разделе **Event Source** диалогового окна.

Пункты данных пользовательского события появляются в узле Even Data Node на левой границе структуры Event. Пользовательские события являются уведомляющими событиями и могут совместно использовать тот же вариант структуры Event, что используют события пользовательского интерфейса или другие пользовательские события.

Можно подсоединять к узлу Register For Events комбинацию из пользовательских событий и событий пользовательского интерфейса.

### Генерирование пользовательских событий

Используйте функцию Generate User Event, чтобы доставить пользовательское событие и связанные с ним данные к другим частям приложения через структуру Event, сконфигурированную для обработки данного события. Функция Generate User Event получает ссылку на пользовательское событие и значение сопутствующих данных. Значения данных должны соответствовать типу данных этого пользовательского события.

Если пользовательское событие не зарегистрировано, то функция Generate User Event не даст эффекта. Если же пользовательское событие зарегистрировано, но нет структуры Event, ожидающей его, то LabVIEW поместит такое пользовательское событие и сопутствующие данные в очередь, пока не освободится структура Event, способная обработать это событие. Вы можете зарегистрировать одно и то же пользовательское событие несколько раз, используя отдельные узлы Register For Event. В этом случае каждая очередь, связанная со своей регистрационной ссылкой на событие (event registration refnum), получает свою собственную копию пользовательского события и связанных с ним данных, всякий раз, когда выполняется функция Generate User Event.



**Примечание.** Чтобы имитировать взаимодействие пользователя с лицевой панелью, Вы можете создать пользовательское событие, которое имеет элементы данных с теми же именами и типами, что и существующее событие пользовательского интерфейса. Например, Вы можете создать пользовательское событие с именем MyValChg, используя кластер из двух булевых полей с именами OldVal и NewVal, которые имеют те же имена, что и имена элементов события пользовательского интерфейса Value Change, связанного с булевым элементом управ-

ления. Вы можете использовать один и тот же вариант структуры Event как для имитированного пользовательского события `MyValChg`, так и для реального события `Value Change` от булева элемента управления. Структура Event выполнит этот вариант, либо если узел `Generate User Event` сгенерирует пользовательское событие, либо если пользователь изменит значение в данном элементе управления на лицевой панели.

### **Снятие с регистрации пользовательских событий**

Пользовательские события снимаются с регистрации, когда они больше не нужны. Вдобавок к этому уничтожьте пользовательское событие, подключив ссылку на пользовательское событие к входу `user event` функции `Destroy User Event` (разрушить пользовательское событие). Подключите выход `error out` функции `Unregister For Events` к входу `error in` функции `Destroy User Event`, чтобы убедиться, что эти функции выполнились должным образом. LabVIEW снимает с регистрации все события и разрушает существующие пользовательские события автоматически, когда ВП верхнего уровня завершает свою работу. Однако, National Instruments рекомендует Вам снимать с регистрации и разрушать события явно, особенно для долго работающих приложений, чтобы сохранить ресурсы памяти.

### **Пример пользовательского события**

Блок-диаграмма на Figure 9-3 показывает, как использовать пользовательские события. Кластер констант на блок-диаграмме задает имя пользовательского события `My Data` и тип данных для события в виде строки с именем `string`. Узел `Register For Events` регистрирует пользовательское событие на заданную ссылку пользовательского события. Структура Event в цикле `While Loop` ожидает наступление события. Параллельно с циклом `While Loop` функция `Generate User Event` посылает событие, которое приводит к выполнению варианта `User Event` в структуре Event. Когда цикл `While Loop` завершится, ВП снимет с регистрации все события и разрушит пользовательское событие.

Постройте ВП, показанный на Figure 9-3, а затем используйте исполнение с подсвечиванием, чтобы рассмотреть, как данные о событии перемещаются от узла к узлу.

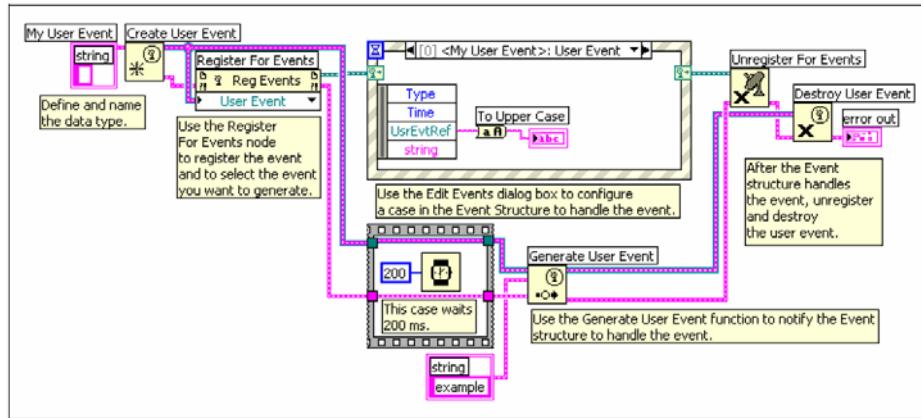


Figure 9-3. Генерирование пользовательских событий

Примеры на динамическую регистрацию событий см. в библиотеке examples\general\dynamicevents.llb.

## 10. Группировка данных с использованием строк, массивов и кластеров

---

Используйте строки, массивы и кластеры для группирования данных. Строки представляют собой последовательность ASCII символов. Массивы объединяют элементы одного и того же типа. Кластеры группируют элементы разных типов.

---

**Более подробно...**

Более подробно о группировании данных с использованием строк, массивов и кластеров см. справочную систему *LabVIEW Help*.

---

### Строки

---

Строка это последовательность отображаемых и неотображаемых ASCII символов. Строки предоставляют независимый от платформы формат для информации и данных. Наиболее общими применениями строк являются:

- Создание простых текстовых сообщений.
- Передача числовых данных в виде строк символов к приборам и затем преобразование этих строк в числа.
- Хранение числовых данных на диске. Чтобы сохранить числа в ASCII файле, Вы должны перед записью конвертировать числа в строки.
- Выдача пользователю инструкций или предупреждений через диалоговое окно.

На лицевой панели строки появляются в таблицах, в окошках ввода текста и в метках. Редактируйте и манипулируйте строками на блок-диаграмме с помощью строковых функций (подпалитра **String**). Форматирование строк может потребоваться для использования в других приложениях, вроде текстовых редакторов и электронных таблиц, или для использования внутри других ВП и функций.

## Строки на лицевой панели

Используйте строковые элементы управления и индикаторы, чтобы имитировать окна текстового ввода и метки. Более подробно о строковых элементах управления и индикаторах см. в разделе *Строковые элементы управления и индикаторы* в Главе 4 *Построение лицевой панели*.

### Способы отображения строк

Щелкните правой кнопкой элемент управления или индикатор на лицевой панели, чтобы выбрать нужный способ отображения из приведенных в Table 10-1. В этой таблице показаны также примеры текстовых сообщений с использованием каждого способа отображения.

Table 10-1. Способы отображения строк

Способ отображения	Описание	Пример сообщения
Normal Display	Отображаются печатаемые символы с использованием установленного для элемента управления шрифта. Непечатаемые символы появляются в виде пустых квадратиков.	There are four display types. \ is a backslash.
'\' Codes Display	Все непечатаемые символы отображаются в виде слэш-кода.	There\ sare\sfour\sdisplay\stypes.\n\\sis\sa\sbslash.
Password Display	Каждый вводимый символ заменяется звездочкой (*).	***** *****
Hex Display	Каждый символ отображается в виде ASCII кода в шестнадцатеричной форме.	5468 6572 6520 6172 6520 666F 7572 2064 6973 706C 6179 2074 7970 6573 2E0A 5C20 6973 2061 2062 6163 6B73 6C61 7368 2E

## Таблицы

Используйте элемент управления Table, чтобы создать таблицу на лицевой панели. Каждый элемент в таблице является символьной строкой и находится в определенном столбце и в определенной строке таблицы. Таким образом, таблица есть отображение двумерного массива строк.

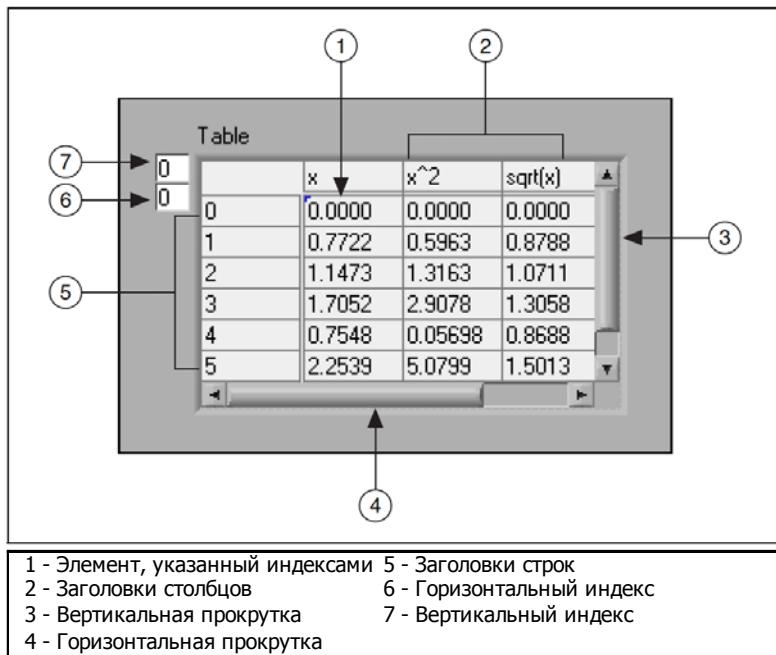


Figure 10-1. Составные части элемента управления Table

## Программное редактирование строк

Используйте строковые функции с подпалилты **String**, чтобы редактировать строки в следующих случаях:

- Поиск с целью извлечь или заменить символы или подстроку внутри строки.
- Изменение всего текста в строке к верхнему или к нижнему регистру.

- Обнаружение и извлечение совпадающих по шаблону совпадений внутри строки.
- Извлечение одной текстовой строчки из всей символьной строки.
- Поворот и сдвиг текста внутри строки.
- Конкатенация (слияние) двух или более строк.
- Удаление символов из строки.

Примеры использования функций с подпалитры **String** приведены в библиотеке examples\general\string.llb. Более подробно о минимизации используемой памяти при программном редактировании строк см. раздел *Memory and Speed Optimization* в Главе 6 *LabVIEW Style Guide* в руководстве *LabVIEW Development Guidelines*.

## Форматирование строк

Чтобы использования данные в другом ВП, функции или приложении, часто требуется конвертировать данные к символьной строке и затем форматировать ее к виду, в котором ВП, функция или приложение сможет ее прочитать. Например, Microsoft Excel принимает строки, которые включают такие разделители, как символы табуляции, запятые или пробелы. Excel использует эти разделители для разделения чисел или слов внутри ячеек.

Например, чтобы записать одномерный массив чисел в электронную таблицу с использованием функции Write File, Вы должны переформатировать массив в строку и отделить каждое число таким разделителем, как символ табуляции. Чтобы записать массив чисел в электронную таблицу с использованием ВП Write to Spreadsheet File, Вы должны переформатировать этот массив с помощью функции Array to Spreadsheet String, задав формат и разделитель.

Используйте строковые функции с подпалитры **String** для решения следующих задач:

- Конкатенация (слияние) двух или более строк.
- Извлечение подстроки из строки.
- Конвертирование данных в строки.
- Форматирование строки для использования в текстовом редакторе или в электронных таблицах.

Для сохранения строк в виде текстовых файлов или файлов электронных таблиц используйте ВП и функции файлового ввода/вывода с подпалитры **File I/O**.

### **Форматные спецификаторы**

Во многих случаях для форматирования строки требуется вводить один или несколько форматных спецификаторов на вход **format string** строковой функции. Форматный спецификатор – это код, который указывает, каким образом следует конвертировать числовые данные в строку или обратно. LabVIEW использует специальные форматные коды для задания формата в виде текстовой строки, передаваемой указанный вход строковых функций.

Функции Format Into String и Scan From String могут использовать много форматных спецификаторов на своем входе **format string**, по одному на каждый вход или выход растягиваемой функции.

Функции Array To Spreadsheet String и Spreadsheet String To Array используют только один форматный спецификатор на входе **format string**, поскольку эти функции имеют только один вход для конвертирования. Любые дополнительные спецификаторы, которые Вы подадите на эти функции, LabVIEW трактует как обычные символьные строки, не имеющие специального значения.

### **Числа и строки**

Числовые и строковые данные отличаются, поскольку строковые данные это ASCII символы, а числовые данные таковыми не являются. Текстовые файлы и файлы электронных таблиц принимают только строки. Чтобы подать числовые данные в текстовый файл или в файл электронной таблицы, вначале следует конвертировать числовые данные в строку.

Чтобы добавить числа в уже существующую строку, конвертируйте числовые данные в строку и используйте функцию Concatenate Strings или другую подходящую строковую функцию, чтобы добавить эту новую строку к уже существующей. Для конвертирования чисел в строки используйте функции с подпалитры **String/Number Conversion**.

Строки могут включать набор чисел, которые Вы отображаете на графике или диаграмме. Например, Вы можете читать текстовый

файл, который включает набор чисел, на основании которых Вы хотите построить график. Хотя эти числа заданы в виде ASCII текста, Вы можете прочитать эти числа как строку и затем форматировать эту строку в набор чисел, перед тем как строить кривую на графике.

На Figure 10-2 показана строка, содержащая набор чисел, которая конвертируется в числа, из них строится массив и на основании этого массива на графике строится кривая.

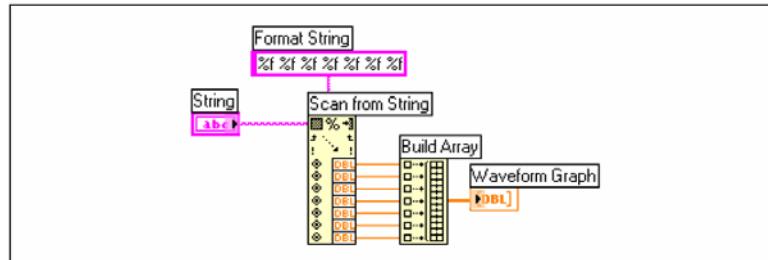


Figure 10-2. Конвертирование строки в числа

### Конвертирование данных в XML и обратно

Расширяемый язык разметки (eXtensible Markup Language – XML) является стандартом по форматированию, который использует теги (метки) для описания данных. В отличие от тегов в HTML, теги в XML идентифицируют часть данных.

Например, предположим, что Вы продавец книг, который продает книги через сеть Internet. Вы хотите классифицировать каждую книгу в вашей библиотеке по следующим критериям:

- Тип книги (художественная (fiction) или нет (nonfiction)).
- Заглавие (Title)
- Автор (Author)
- Издатель (Publisher)
- Цена (Price)
- Жанр (Genre)
- Аннотация (Synopsis)

- Количество страниц (Pages)

Вы можете создать XML файл для каждой книги. Такой XML файл для книги с названием *Touring Germany's Great Cathedrals* (Экскурсия по величим соборам Германии) будет иметь примерно следующий вид:

```
<nonfiction>
<Title>Touring Germany's Great Cathedrals</Title>
<Author>Tony Walters</Author>
<Publisher>Douglas Drive Publishing</Publisher>
<PriceUS>$29.99</PriceUS>
<Genre>Travel</Genre>
<Genre>Architecture</Genre>
<Genre>History</Genre>
<Synopsis>This book fully illustrates twelve of
Germany's most inspiring cathedrals with full-color
photographs, scaled cross-sections, and time lines of
their construction.</Synopsis>
<Pages>224</Pages>
</nonfiction>
```

Аналогично Вы можете классифицировать данные в LabVIEW по имени, значению и типу. Строку с именем User Name можно представить в XML следующим образом:

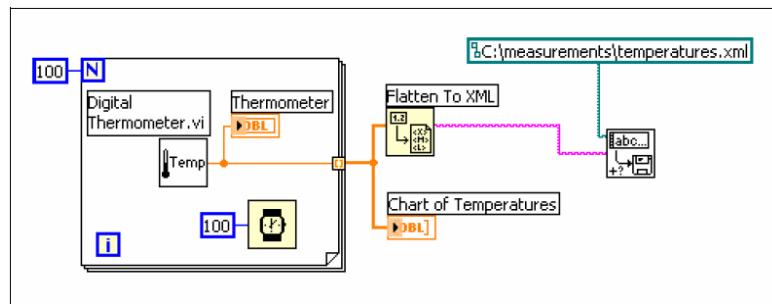
```
<String>
<Name>User Name</Name>
<Value>Reggie Harmon</Value>
</String>
```

### **Использование типов данных, основанных на XML**

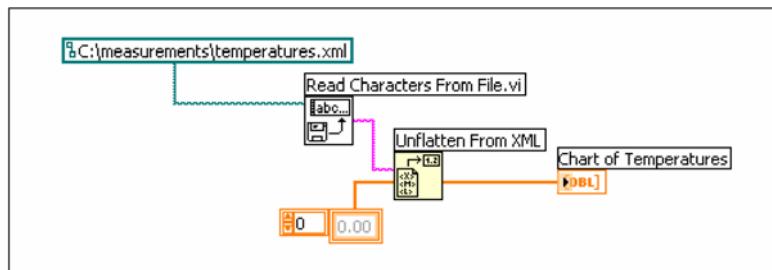
Преобразование данных LabVIEW в формат XML происходит так, что, сохранив эти данные в файл, Вы сможете легко определить значение (значения), имя (имена) и тип данных по тегам, которые описывают эти данные. Например, если Вы конвертируете массив значений температуры в XML и сохраните эти данные в текстовом файле, то Вы легко сможете определить расположение значений температуры по тегу *<Value>*, который сопровождает каждое значение температуры.

Используйте функцию Flatten to XML для преобразования типов данных LabVIEW в формат XML. Следующий пример генерирует

100 смоделированных значений температуры, строит график массива температур, конвертирует массив чисел в формат XML и записывает данные формата XML в файл `temperature.xml`.



Используйте функцию `Unflatten from XML` для конвертирования данных формата XML в данные LabVIEW. Следующий пример читает 100 значений температуры из файла `temperature.xml` и строит график массива значений температуры.



**Примечание.** Несмотря на то, что Вы можете преобразовать LabVIEW данные типа Variant в XML, попытка обратного преобразования вариантовых данных из XML даст в результате пустые вариантовые данные LabVIEW.

Примеры конвертирования в формат XML и обратно можно найти в библиотеке `examples\file\XMLex.llb`.

### Схема LabVIEW XML

LabVIEW конвертирует данные по установленной XML схеме (шаблону). К настоящему времени Вы не можете создавать собст-

венные схемы, и не можете управлять тем, как LabVIEW размечает тегами каждую порцию данных. Кроме того, Вы не можете конвертировать в XML весь ВП или целую функцию.

LabVIEW XML схема содержится в файле LVXMLSchema.xsd, который находится в директории LabVIEW\vi.lib\utility.

## Группировка данных в массивы и кластеры

---

Используйте элементы управления и функции для массивов и кластеров, чтобы сгруппировать данные. Массив группирует элементы данных одного типа. Кластеры группируют элементы данных смешанных типов.

### Массивы

Массив состоит из элементов и описателей размерности. Элементы – это данные, которые образуют массив. Размерность – это длина, высота или глубина массива. Массив может иметь одну или более размерностей, каждая из которых может содержать до  $2^{31} - 1$  элементов.

Вы можете строить массивы из данных, имеющих тип числовой, булев (Boolean), путь (path), строка (string), осциллограмма (waveform) и кластер (cluster). Рассмотрите использование массивов, когда Вы работаете с набором однотипных данных, и когда требуется выполнять повторяющиеся вычисления. Массивы идеально подходят для хранения данных, которые Вы собираете из осциллограмм или простых данных, генерируемых в циклах, в которых на каждой итерации вырабатывается один элемент массива.

Вы не можете создать массив из массивов. Однако, Вы можете использовать многомерные массивы или можете создать массив кластеров, где каждый кластер содержит один или несколько массивов. Более подробно о типах элементов, которые могут содержать массивы, см. в разделе *Ограничения для массивов* в настоящей Главе. Более подробно о кластерах см. в разделе *Кластеры* в настоящей Главе.

### Индексы

Для указания конкретного элемента в массиве требуется один индекс на каждое измерение. В LabVIEW индексы позволяют Вам пе-

ремещаться по массиву и извлекать из него элементы, строки, столбцы и страницы с помощью кода на блок-диаграмме.

### Примеры массивов

Примером простого массива служит текстовый массив, в котором содержится список девяти планет нашей солнечной системы. LabVIEW представляет это как одномерный массив строк из девяти элементов.

Элементы массива упорядочены. Массив использует индексы так, что Вы можете легко получить доступ к каждому его элементу. Эти индексы имеют нулевую базу отсчета. Это означает, что они изменяются в диапазоне от 0 до  $n-1$ , где  $n$  – количество элементов в массиве. Например, для девяти планет  $n=9$ , таким образом, индекс лежит в диапазоне от 0 до 8. Земля является третьей планетой, следовательно, для нее индекс равен 2.

Другим примером массива может служить осциллограмма, представленная в виде числового массива, каждый последовательный элемент которого является значением напряжения на последовательном временном интервале, как показано на Figure 10-3.

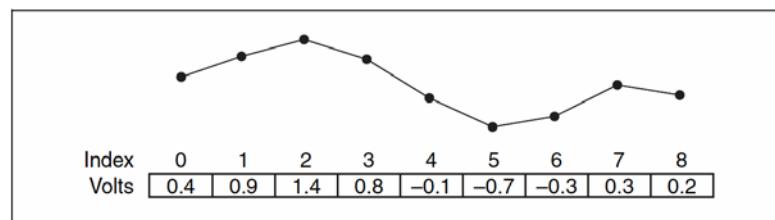
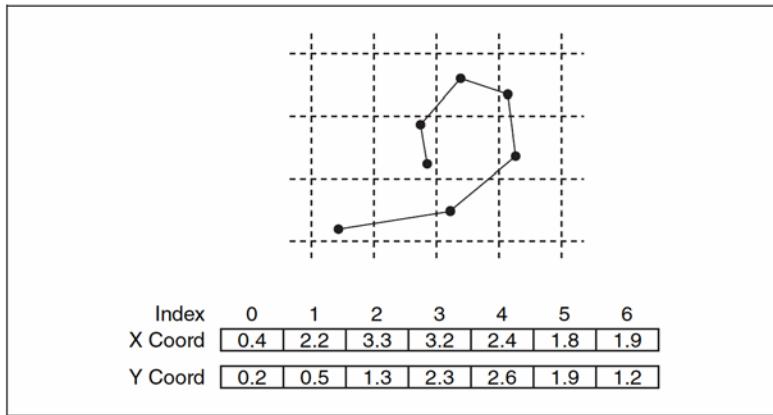


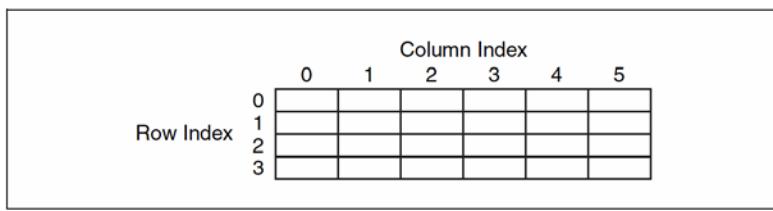
Figure 10-3. Осциллограмма в виде массива чисел

Более сложным примером массива служит графическое представление массива точек, каждая из которых является кластером, содержащим пару чисел, представляющих значения координат X и Y, как показано на Figure 10-4.



**Figure 10-4.** График в виде массива точек

Предыдущие примеры используют одномерные массивы. Двумерные массивы хранят элементы в виде решетки. В этом случае требуется два индекса для определения элемента: индекс столбца и индекс строки, каждый из которых имеет нулевую базу отсчета. На Figure 10-5 показан массив из 6 столбцов и 4 строк, содержащий  $6 \times 4 = 24$  элементов.

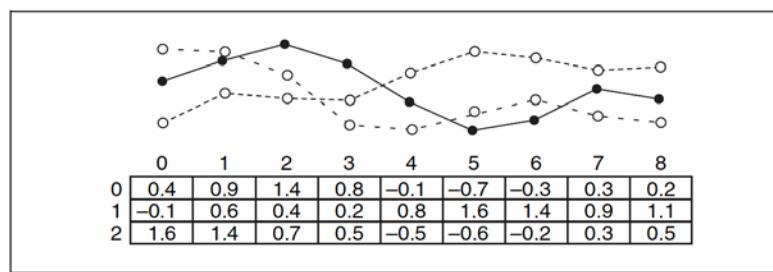


**Figure 10-5.** Двумерный массив из 6 столбцов и 4 строк

К примеру, шахматная доска имеет восемь столбцов и восемь строк для 64 позиций (клеток). Каждая клетка может быть пустой или может содержать одну шахматную фигуру. Вы можете представить шахматную доску в виде двумерного массива строк. Каждая строка является наименованием фигуры, которая занимает соответствующую клетку на доске или является пустой строкой, если эта клетка незанята.

Вы можете обобщить одномерные массивы из примеров на Figure 10-3 и на Figure 10-4 до двух измерений, добавляя строки к этим

массивам. На Figure 10-6 показан набор осциллограмм, представленный в виде двумерного массива чисел. Индекс строки выбирает осциллограмму, а индекс столбца выбирает точку на осциллограмме.



**Figure 10-6.** Несколько осциллограмм в двумерном массиве чисел

Примеры использования массивов можно найти в библиотеке `examples\general\arrays.llb`. Более подробно о построении массивов см. в разделе *Использование циклов для построения массивов* в Главе 8 *Циклы и структуры*.

## Ограничения для массивов

Вы можете создать массив почти для любого типа данных с учетом следующих исключений:

- Нельзя создать массив массивов. Однако Вы можете использовать многомерный массив или использовать функцию Build Cluster Array, чтобы создать массив кластеров, каждый из которых содержит один или несколько массивов.
  - Нельзя создать массив элементов управления «подпанель» (sub-panel control).
  - Нельзя создать массив элементов управления «многостраничное диалоговое окно» (tab control).
  - Нельзя создать массив элементов управления ActiveX.
  - Нельзя создать массив индикаторов Waveform Chart.
  - Нельзя создать массив индикаторов XY Graph, настроенных на несколько графиков.

## Создание массива элементов управления, индикаторов и констант

Чтобы создать на лицевой панели массив элементов управления или индикаторов, поместите на лицевую панель оболочку (array shell) массива, которая показана на Figure 10-7, и перетащите в нее объект данных или элемент, который может иметь один из следующих типов: числовой (numeric), булев (Boolean), строка (string), путь (path), ссылка (refnum) или кластер элементов управления или индикаторов. Оболочка массива находится на подпалитре **Array & Cluster** палитры **Controls** под названием **Array**.

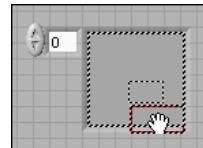


Figure 10-7. Оболочка массива

Оболочка массива автоматически приспособит свой размер к внедренному объекту, будь то маленький булев элемент управления или большой трехмерный график. Чтобы создать на лицевой панели многомерный массив, щелкните правой кнопкой индексный дисплей (index display) в левом верхнем углу оболочки массива и выберите из контекстного меню пункт **Add Dimension**. Вы также можете изменить растягиванием размер индексного дисплея, пока не получите нужное число измерений. Чтобы удалить одно измерение щелкните правой кнопкой индексный дисплей и выберите из контекстного меню пункт **Remove Dimension**. Вы также можете удалить лишние измерения массива, просто перетягивая нижнюю границу индексного дисплея в сторону его уменьшения (вверх).

Чтобы отобразить нужный элемент массива на лицевой панели, либо явно введите нужное значение индекса в индексный дисплей, либо используйте стрелочки инкремента и декремента на индексном дисплее для выбора нужного значения.

Чтобы создать массив констант на блок-диаграмме, выберите из подпалитры **Array** палитры **Functions** оболочку массива констант **Array Constant**, и поместите в нее строковую, числовую или кластерную константу. Вы можете использовать массив констант как основу для сравнения с другими массивами.

## Индексный дисплей массива

Двумерный массив содержит строки и столбцы. Как показано на Figure 10-8, в верхнем окошке индексного дисплея, расположенно-го в левой части оболочки массива, отображается индекс строки, а в нижнем окошке – индекс столбца. В окошке справа от индексного дисплея отображается значение элемента массива, расположенного в заданных строке и столбце. На Figure 10-8 показано, что значени-ем элемента массива в строке 6 и столбце 13 является **66.00**.

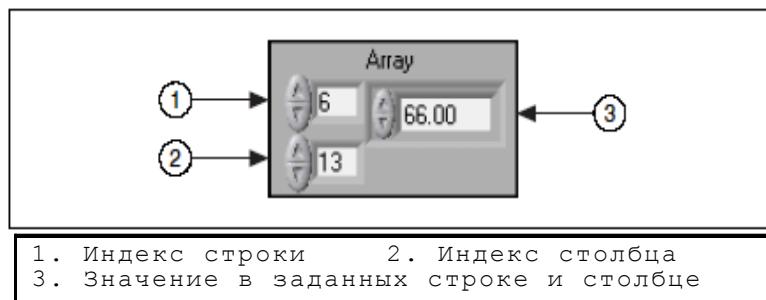


Figure 10-8. Массив элементов управления

Строки и столбцы имеют нулевую базу отсчета. Это означает, что первый столбец имеет индекс 0, второй столбец – индекс 1 и т.д. Установка в индексном дисплее для приведенного ниже массива значений индекса строки 1 и индекса столбца 2 приведет к отобра-жению в правом окошке значения элемента массива 6.

0	1	2	3
4	5	6	7
8	9	10	11

Если Вы попытаетесь отобразить столбец или строку за пределами размерности массива, элемент управления массива станет серым, чтобы показать, что значение в этом элементе не определено и LabVIEW отображает значение по умолчанию для этого типа дан-ных. Значение по умолчанию зависит от типа данных массива.

Используйте инструмент Positioning, чтобы растянуть массив для отображения в правом окошке индикатора нескольких строк или столбцов.

### Функции для работы с массивами

Используйте функции с подпалитры **Array** для создания и манипулирования с массивами при решении следующих задач:

- Извлечение из массива отдельного элемента.
- Вставка, удаление или замещение элементов в массиве.
- Расщепление массивов.

### Автоматическое изменение размеров функций для работы с массивами

Функции Index Array, Replace Array Subset, Insert Into Array, Delete From Array и Array Subset автоматически изменяют свои размеры, чтобы приспособиться к размерности массива, который Вы подключаете к их входу. Например, если Вы подсоедините одномерный массив к одной из этих функций, то функции покажут один индексный вход. Если Вы подсоедините двумерный массив к тем же функциям, то они покажут два индексных входа – один для строки и один для столбца.

Вы можете получить более одного элемента или подмассив (строку, столбец или страницу) с помощью этих функций, используя инструмент Positioning и вручную изменения размер функций. Когда Вы растягиваете одну из этих функций, ее размер будет изменяться в зависимости от того, какой размерности массив подсоединен к входу. Если Вы подсоедините одномерный массив к одной из этих функций, то при растяжении функции будет добавляться по одному индексному входу. Если же Вы подсоедините к этой же функции двумерный массив, то функция будет растягиваться путем добавления по два индексных входа – один для строки и один для столбца.

Индексные входы, которые Вы подсоединяете, определяют конфигурацию подмассива, к которому Вы хотите получить доступ или модифицировать. Например, если на вход функции Index Array подан двумерный массив, и Вы подсоедините только вход **row** (строка), то будет целиком извлечена одномерная строка массива. Если Вы подсоедините только вход **column** (столбец), то будет целиком

извлечен весь одномерный столбец. Если же Вы подключите и вход **row** и вход **column**, то будет извлечен один единственный элемент массива.

Блок-диаграмма, показанная на Figure 10-9, использует функцию Index Array для извлечения строки и одного элемента из двумерного массива.

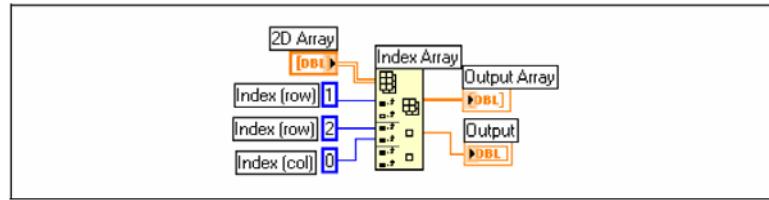


Figure 10-9. Индексация двумерного массива

Чтобы получить доступ к нескольким последовательным значениям в массиве, растяните функцию Index Array, но не присоединяйте значения ко всем добавленным индексным входам. Например, чтобы получить первую, вторую и третью строку из двумерного массива, растяните функцию Index Array на три дополнительные ячейки и присоедините одномерный массив индикаторов к каждому из выходов **sub-array**.

Более подробно о минимизации требований к памяти при использовании функций, работающих с массивами, внутри циклов, см. в разделе *Memory and Speed Optimization* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW Development Guidelines*.

## Кластеры

Кластеры группируют элементы данных разных типов, также как жгут проводов в телефонном кабеле, где каждый из проводов кабеля представляет отдельный элемент кластера. Кластер аналогичен записи или структуре в текстовых языках программирования.

Объединение (bundle) нескольких элементов в кластеры позволяет избавиться от перепутывания проводников на блок-диаграмме и уменьшает необходимое число терминалов на соединительной панели ВПП. Соединительная панель ВП может содержать максимум 28 терминалов. Если ваша лицевая панель содержит больше 28

элементов управления и индикаторов, которые Вы хотите использовать программно, сгруппируйте их в кластер и назначьте этот кластер терминалу соединительной панели.

Хотя элементы и кластера и массива упорядочены, Вы должны сразу разделить (unbundle) все элементы кластера с помощью функции Unbundle, а не индексировать каждый раз по одному элементу, как это можно делать с массивами. Чтобы получить доступ к конкретным элементам кластера используйте функцию Unbundle by Name. Кластеры также отличаются от массивов тем, что они имеют фиксированный размер. Подобно массиву кластер может содержать либо элементы управления, либо индикаторы. Один и тот же кластер не может содержать одновременно и элементы управления и индикаторы.

Большинство кластеров на блок-диаграмме имеют розовый цвет проводника и терминала типа данных. Кластеры из чисел, как самостоятельных элементов, имеют коричневый цвет проводников и терминалов типов данных. Такие числовые кластеры можно подключать непосредственно к числовым функциям вроде Add (сложение) и Square Root (корень квадратный), чтобы одновременно выполнить одну и ту же операцию над всеми элементами кластера.

Элементы кластера имеют логический порядок независимо от их положения в оболочке кластера. Объект, помещенный в кластер первым, будет являться элементом 0, помещенный вторым – элементом 1 и т.д. Если Вы удаляете элемент, такой порядок автоматически корректируется. Упорядоченность кластера определяет порядок, в котором элементы появляются на терминалах функций Bundle и Unbundle на блок-диаграмме. Вы можете просматривать и изменять упорядоченность кластера, щелкая правой кнопкой границу кластера и выбирая из контекстного меню пункт **Reorder Controls In Cluster**.

Для соединения кластеров проводником они оба должны иметь одинаковое число элементов. Соответствующие по порядку элементы этих кластеров должны иметь совместимые типы данных. Например, если элемент числового типа удвоенной точности с плавающей точкой в одном кластере соответствует по порядку внутри другого кластера элементу строкового типа, то проводник, соединяющий на блок-диаграмме два таких кластера, будет неисправным (broken) и ВП не запустится. Если же оба таких элемента – это чис-

ла разного представления, то LabVIEW произведет принудительное преобразование к одному представлению. Более подробно о числовых преобразованиях см. в разделе *Преобразование числовых представлений* в Приложении В *Полиморфные функции*.

Используйте кластерные функции с подпалитры **Cluster** для создания и преобразования кластеров. Например, Вы можете решить следующие задачи:

- Извлечение из кластера отдельных элементов.
- Добавление в кластер отдельных элементов.
- Разбиение кластера на отдельные элементы данных.

## **11. Локальные и глобальные переменные**

---

В LabVIEW Вы можете считывать данные от объектов лицевой панели и записывать данные туда с помощью их терминалов на блок-диаграмме. Однако, каждый объект лицевой панели имеет только один терминал на блок-диаграмме, а ваше приложение может потребовать доступ к данным этих терминалов из нескольких мест блок-диаграммы.

Локальные и глобальные переменные передают информацию между разными местами вашего приложения без непосредственной связи их проводниками. Используйте локальные переменные для доступа к объектам лицевой панели из нескольких мест блок-диаграммы одного и того же ВП. Используйте глобальные переменные для доступа и передачи данных между несколькими разными ВП.

---

**Более подробно...**

Более подробно относительно использования локальных и глобальных переменных см. справочную систему *LabVIEW Help*.

---

### **Локальные переменные**

---

Используйте локальные переменные для доступа к объектам лицевой панели из нескольких мест на блок-диаграмме одного и того же ВП и для обмена данными между структурами блок-диаграммы, которые нельзя соединить проводником.

Посредством локальной переменной Вы можете записать или прочитать данные от элемента управления или индикатора на лицевой панели. Запись в локальную переменную аналогична прохождению данных к некоторому другому терминалу. Однако, с помощью локальной переменной Вы можете записать данные даже в элемент управления и прочитать данные из индикатора. Таким образом, с помощью локальной переменной Вы получаете доступ ко всем объектам лицевой панели, как для записи, так и для чтения.

Например, если пользовательский интерфейс требует от пользователя ввести входное имя (log in), то Вы можете очищать приглашения **Login** и **Password** всякий раз, когда новый пользователь вхо-

входит в систему. Используйте локальную переменную для чтения из строковых элементов управления **Login** и **Password**, когда пользователь входит в систему, и записывайте пустые строки в эти элементы управления, когда пользователь покидает систему.

## Создание локальных переменных

Щелкните правой кнопкой существующий объект лицевой панели или терминал на блок-диаграмме и выберите из контекстного меню пункт **Create»Local Variable**, чтобы создать локальную переменную. Иконка локальной переменной для выбранного объекта появится на блок-диаграмме.

 Вы также можете взять локальную переменную с подпалитры **Structures** палитры **Functions** и поместить ее на блок-диаграмму. Узел локальной переменной, показанный слева, еще не связан ни с каким элементом управления или индикатором. Щелкните правой кнопкой этот узел и с помощью пункта контекстного меню **Select Item** выберите нужный объект лицевой панели. Этот пункт меню содержит список всех объектов лицевой панели, имеющих собственные метки.

LabVIEW использует собственные метки для связи локальных переменных с объектами лицевой панели, поэтому помечайте объекты лицевой панели выразительными метками. Более подробно о собственных и свободных метках см. в разделе *Использование текстовых меток* в Главе 4 *Построение лицевой панели*.

## Глобальные переменные

---

Используйте глобальные переменные для доступа и передачи данных между несколькими ВП, которые запускаются одновременно. Глобальные переменные – это встроенные объекты LabVIEW. Когда Вы создаете глобальную переменную, LabVIEW автоматически создает специальный глобальный ВП, который имеет лицевую панель, но не имеет блок-диаграммы. Добавьте элементы управления и индикаторы на лицевую панель этого глобального ВП, чтобы определить типы данных глобальных переменных, которые в него входят. На самом деле эта лицевая панель есть просто контейнер, к данным которого могут иметь доступ другие ВП.

Например, предположим, что у Вас есть два ВП, работающих одновременно. Каждый из них содержит цикл While Loop и записывает отсчеты данных в построитель графика. Первый ВП содержит булев элемент управления, с помощью которого должны завершаться оба ВП. Вы можете использовать глобальную переменную для завершения обоих циклов с помощью одного булева элемента управления. Если бы оба эти цикла находились на блок-диаграмме одного и того же ВП, то для завершения этих циклов можно было бы воспользоваться локальной переменной.

## Создание глобальных переменных



Возьмите глобальную переменную, узел которой показан слева, с подпалитры **Structures** палитры **Functions** и поместите ее на блок-диаграмму. Сделайте двойной щелчок по узлу глобальной переменной, чтобы отобразить лицевую панель этого глобального ВП. Поместите элементы управления и индикаторы на эту лицевую панель так же, как это делается для обычной лицевой панели.

LabVIEW использует собственные метки для идентификации глобальных переменных, поэтому помечайте объекты лицевой панели выразительными метками. Более подробно о собственных и свободных метках см. в разделе *Использование текстовых меток* в Главе 4 *Построение лицевой панели*.

Вы можете создать несколько отдельных глобальных ВП, каждый с одним объектом на лицевой панели, или Вы можете создать один глобальный ВП с несколькими объектами на лицевой панели. Глобальный ВП с несколькими объектами более эффективен, поскольку Вы можете группировать связанные переменные вместе. Блок-диаграмма ВП может содержать несколько узлов глобальных переменных, которые связаны с элементами управления или индикаторами лицевой панели глобального ВП. Такие узлы глобальных переменных являются либо копиями первого узла глобальной переменной, который Вы помещаете на блок-диаграмму глобального ВП, либо они являются узлами глобальных переменных глобального ВП, который Вы разместили в текущем ВП. Размещение глобальных ВП в других ВП производится тем же образом, как и размещение ВПП в другом ВП. Всякий раз, когда Вы помещаете новый узел глобальной переменной на блок-диаграмму, LabVIEW создает новый ВП, связанный только с этим узлом глобальной переменной и копирует его. Более подробно о ВПП см. в разделе *Виртуальные подприборы (ВПП)* в Главе 7 *Создание ВП и ВПП*.

После того, как Вы завершите размещение объектов на лицевой панели глобального ВП, сохраните его и вернитесь на блок-диаграмму исходного ВП. Затем Вы должны выбрать объект лицевой панели глобального ВП, к которому хотите установить доступ. Щелкните правой кнопкой узел глобальной переменной и выберите из контекстного меню пункт **Select Item**. Будет отображен список всех объектов лицевой панели, имеющих собственные метки.

## Чтение и запись значений переменных

---

После того, как Вы создадите локальную или глобальную переменную, Вы можете читать данные из переменной или записывать данные в нее. По умолчанию новая переменная принимает данные. Такой тип переменной работает как индикатор и является записывающей локальной или глобальной переменной. Когда Вы записываете новые данные в локальную или глобальную переменную, связанный с ней элемент управления или индикатор на лицевой панели обновляет свое значение.

Вы также можете настроить переменную так, чтобы она вела себя как источник данных, то есть сделать читаемую локальную или глобальную переменную. Чтобы настроить переменную так, чтобы она вела себя как элемент управления, щелкните переменную правой кнопкой и выберите из контекстного меню пункт **Select To Read**. Когда такой узел выполняется, ВП считывает данные из связанного с ней элемента управления или индикатора на лицевой панели.

Чтобы опять настроить переменную так, чтобы она вела себя как индикатор, щелкните переменную правой кнопкой и выберите из контекстного меню пункт **Select To Write**.

На блок-диаграмме читаемые локальные или глобальные переменные отличаются от записываемых так же, как различаются терминалы элементов управления от терминалов индикаторов. Читаемая локальная или глобальная переменная имеет толстую границу на значке терминала, а записываемая переменная – тонкую.

Примеры использования локальных и глобальных переменных приведены в библиотеках `examples\general\ locals.llb` и `examples\general\ globals.llb`.

## Предосторожности при использовании локальных и глобальных переменных

---

Локальные и глобальные переменные являются дополнительными понятиями LabVIEW. По своей сути они не входят составной частью в принятую в LabVIEW модель потока данных. Блок-диаграмма может стать трудно читаемой, когда Вы используете локальные и глобальные переменные, поэтому пользоваться ими нужно осторожно. Неправильное использование локальных и глобальных переменных вместо соединительной панели или для доступа к значениям в каждом кадре структур последовательности, может привести к неожиданному поведению ВП. Злоупотребление локальными и глобальными переменными для исключения длинных проводников или для использования их вместо потока данных, снижает эффективность. Более подробно о модели потока данных, принятой в LabVIEW, см. разделе *Поток данных на блок-диаграмме* в Главе 5 *Построение блок-диаграммы*.

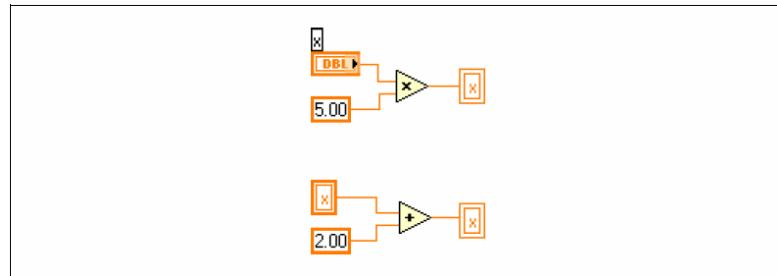
### Инициализация локальных и глобальных переменных

Проверьте, чтобы перед запуском ВП все локальные и глобальные переменные содержали известные значения. В противном случае эти переменные могут содержать значения, которые могут привести к неверной работе ВП.

Если Вы не инициализировали переменные перед тем, как ВП первый раз прочитает их значение, то они будут содержать значения по умолчанию, принятые для связанных с ними объектов лицевой панели.

### Эффект гонок

Эффект гонок имеет место в тех ситуациях, когда две или больше частей кода, выполняемых параллельно, изменяют значение одного и того же совместно используемого ресурса. Такая ситуация типична для локальных или глобальных переменных. На Figure 11-1 показан пример эффекта гонок.



**Figure 11-1.** Эффект гонок

Выход этого ВП зависит от порядка, в котором выполняются операции. Поскольку между этими операциями отсутствует зависимость по данным, то заранее нельзя определить какая из них будет выполняться первой. Чтобы исключить эффект гонок, не записывайте в одну и ту же переменную в нескольких местах блок-диаграммы. Более подробно о зависимости по данным см. в разделе *Зависимость по данным и искусственная зависимость по данным* в Главе 5 *Построение блок-диаграммы*.

### Соображения о памяти при использовании локальных переменных

Локальные переменные создают копии для буферов данных. Когда Вы читаете из локальной переменной, Вы создаете новый буфер для данных из связанного с ней элемента управления.

Если Вы используете локальные переменные для передачи больших объемов данных из одного места на блок-диаграмме к другому, то это обычно приводит к большим затратам памяти и, следовательно, приводит к снижению быстродействия по сравнению с передачей тех же данных посредством проводников. Если Вам нужно хранить данные в процессе выполнения, рассмотрите возможность применения сдвигающих регистров.

### Соображения о памяти при использовании глобальных переменных

Когда Вы читаете из глобальной переменной, LabVIEW создает копию данных, хранящихся в этой глобальной переменной.

Когда Вы манипулируете большими массивами и строками, время и объем памяти, отводимые для манипуляции с глобальными переменными, могут быть значительными. Манипулирование с гло-

бальными переменными особенно неэффективно, когда это касается массивов, поскольку даже если Вы модифицируете один элемент массива, LabVIEW сохраняет и модифицирует весь массив. Если Вы читаете из глобальной переменной в нескольких местах приложения, Вы тем самым создаете несколько буферов памяти, что неэффективно и снижает общую эффективность.

## 12. Графики и диаграммы

---

Используйте индикаторы вывода графиков и диаграмм для отображения данных в графической форме.

Индикаторы-графики (Graph) и индикаторы-диаграммы (Chart) отличаются способом отображения и обновления данных. ВП с индикаторами Graph обычно собирают данные в массивы и затем строят зависимости в виде графиков, что похоже на электронную таблицу, которая вначале накапливает данные, а потом генерирует график зависимости. В противоположность этому, индикатор Chart добавляет новые точки данных к тем, которые уже имеются в дисплее и образуют предысторию накопления данных. На диаграмме можно видеть текущие считанные или измеренные данные совместно с полученными ранее.

---

**Более подробно...**

Более подробно относительно использования графиков и диаграмм см. справочную систему *LabVIEW Help*

---

### Типы графиков и диаграмм

---

Индикаторы графиков и диаграмм включают следующие типы:

- **Waveform Chart and Graph** – Отображают данные, обновляемые в постоянном темпе.
- **XY Graph** – Отображает данные, поступающие неравномерно, как при вводе с синхронизацией (триггером).
- **Digital Waveform Graph** – Отображает данные в виде импульсов или группы цифровых линий. Импульсы используются при передаче данных от одного компьютера к другому.
- **(Windows) 3D Graphs** – Отображают трехмерные данные на трехмерный график объекта лицевой панели ActiveX.

Примеры с использованием индикаторов Graph и Chart можно найти в библиотеке `example\general\graphs`.

## Опции индикаторов Graph и Chart

---

Хотя индикаторы Graph и Chart строят графики данных по-разному, у них есть некоторые общие опции, доступные из контекстного меню. Более подробно об опциях, доступных только для индикаторов Graph или только для индикаторов Chart, см. в разделах *Настройка индикаторов Graph* и *Настройка индикаторов Chart* в настоящей Главе.

Индикаторы для вывода осциллографов в виде графиков и диаграмм (Waveform Graph, Waveform Chart) и индикаторы для вывода двухкоординатных графиков (XY Graph и Express XY Graph) имеют опции, отличные от опций индикаторов для графиков интенсивности (Intensity Graph, Intensity Chart), вывода цифровых данных (Digital Waveform Graph) и вывода трехмерных графиков и диаграмм (3D Surface Graph, 3D Parametric Graph, 3D Curve Graph). Более подробно об опциях индикаторов для вывода графиков интенсивности, цифровых и трехмерных графиков и диаграмм см. в разделах *Индикаторы Intensity Graph и Intensity Chart*, *Трехмерные графики* и *Индикаторы Digital Waveform Graph* в настоящей Главе.

### Несколько шкал X и Y в индикаторах Graph и Chart

Все индикаторы типа Graph допускают использование нескольких шкал X и Y. Все индикаторы типа Chart допускают использование нескольких шкал Y. Используйте несколько шкал на графиках и диаграммах при отображении графиков, которые не могут совместно использовать общую шкалу X или Y. Чтобы добавить новую шкалу на график или на диаграмму, щелкните правой кнопкой нужную шкалу на индикаторе и выберите из контекстного меню **Duplicate Scale**.

### Сглаживание линий графиков на индикаторах Graph и Chart

Можно улучшить внешний вид линий кривых на графиках и диаграммах, используя режим сглаженных линий (anti-aliased lines). Если этот режим включен, линии графиков будут плавными. Режим сглаженных линий не затрагивает и не изменяет параметры толщины линии, стиль линии, стиль точки и др.



**Примечание.** Режим сглаженных линий не доступен для индикатора Digital Waveform Graph.

Для активизации режима сглаженных линий, щелкните правой кнопкой элемент индикатора Plot Legend (перед этим он должен находиться в состоянии видимости) и выбрать из контекстного меню пункт **Anti-aliased**. Если элемент Plot Legend невидим, щелкните правой кнопкой индикатор Graph или Chart и выберите из контекстного меню пункт **Visible Items»Plot Legend**.



**Примечание.** Поскольку прорисовка кривых в режиме сглаженных линий может потребовать интенсивных вычислений, использование этого режима может снизить эффективность работы.

## Настройка внешнего вида индикаторов Graph и Chart

Настройка внешнего вида индикаторов Graph и Chart осуществляется путем включения или отключения видимости отдельных компонентов. Щелкните правой кнопкой индикатор Graph или Chart и выберите из контекстного меню пункт **Visible Item**, с помощью которого можно скрыть или отобразить следующие компоненты:

- **Plot Legend** (панель настройки кривых) – Определяет цвет и стиль линий кривой (кривых). Для отображения нескольких кривых, растяните значок этого элемента.
- **Scale Legend** (панель настройки шкалы) – Служит для задания меток для шкал и настройки их свойств.
- **Graph Palette** (палитра инструментов управления графиком) – Позволяет масштабировать и форматировать изображение в индикаторе в процессе работы ВП.
- **X Scale** (шкала X) и **Y Scale** (шкала Y) – Задают форматы шкал X и Y. Более подробно о шкалах см. в разделе *Форматирование осей* в настоящей Главе.
- **Cursor Legend** (панель настройки курсора, имеется только у индикатора Graph) – Отображает маркер в точке с заданными координатами. Можно отображать несколько курсоров на одном и том же индикаторе Graph.
- **X Scrollbar** (полоса прокрутки по оси X) – Обеспечивает перемещение вдоль оси X в индикаторе Graph или Chart. Используйте полосу прокрутки для просмотра данных, которые в данный момент не отображаются в индикаторе Graph или Chart.

## Настройка индикаторов Graph

Вы можете модифицировать поведение курсора, опции шкал и осей графика. Компоненты индикатора Graph показаны на Figure 12-1.

Вы можете модифицировать графики осциллографм и графики интенсивности, чтобы приспособить их к вашим потребностям в представлении данных или для повышения информативности их отображения. Для индикаторов Chart доступны компоненты: **Plot Legend**, **Scale Legend**, **Graph Palette**, **Digital Display** (цифровой дисплей), **X Scale** и **Y Scale**.

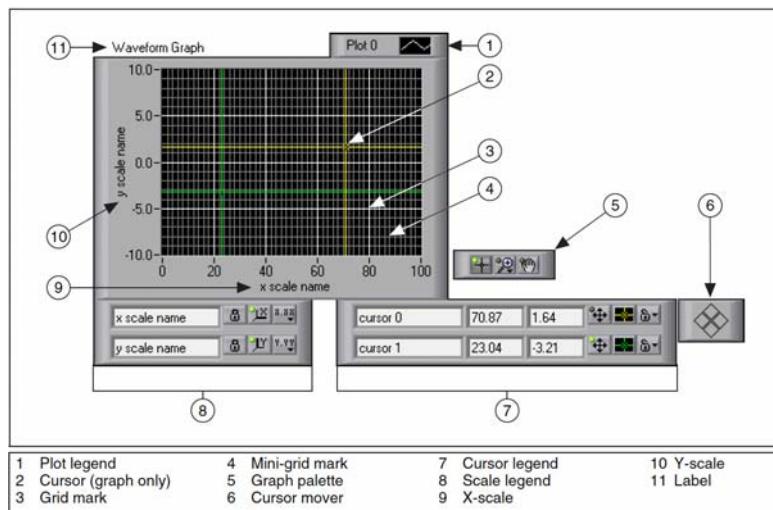


Figure 12-1. Компоненты индикатора Graph

Большинство перечисленных компонентов можно добавить, щелкнув по индикатору правой кнопкой, выбирая из контекстного меню пункт **Visible Items** и затем выбирая необходимый компонент из списка.

### Курсыры в индикаторах Graph

Курсыры на графике позволяют Вам считывать значения в заданных точках графика. Эти значения отображаются на панели управления курсором (**Cursor Legend**). Чтобы добавить курсор на график, щелкните правой кнопкой, выберите из контекстного меню пункт **Visible Items»Cursor Legend** и щелкните где-нибудь в пре-

делах строки на панели управления курсором, чтобы активировать курсор. Используйте инструмент **Positioning**, чтобы растянуть панель управления курсором для добавления еще нескольких курсоров.

Вы можете поместить курсоры и дисплей курсора на все графики и снабдить каждый курсор меткой. Вы можете заблокировать курсор на графике и можете перемещать несколько курсоров одновременно. На одном графике может быть любое число курсоров.

### Автомасштабирование

Индикаторы Graph могут автоматически подстраивать свои горизонтальные и вертикальные шкалы к данным, которые к ним присоединены. Такое поведение шкал называется автомасштабированием (autoscaling). Чтобы включить или выключить автомасштабирование, щелкните правой кнопкой индикатор и выберите из контекстного меню пункт **X Scale»Autoscale X** или **Y Scale»Autoscale Y**. Для индикатора Graph режим автомасштабирования является по умолчанию активным. Однако режим автомасштабирования может снизить эффективность.

Для прямого изменения вертикального или горизонтального масштаба на индикаторе, воспользуйтесь инструментом **Operating** или **Labeling**.

#### Панель настройки шкалы для индикатора **Waveform Graph**

Используйте панель настройки шкалы, чтобы задать метки шкал и настроить их свойства.



Используйте инструмент **Operating**, чтобы щелкнуть кнопку **Scale Format** (формат шкалы), показанную слева, и настроить формат, точность и режим отображения.



Используйте кнопку **Scale Lock** (блокировка шкалы), показанную слева, чтобы включать/выключать режим автомасштабирования для каждой шкалы, видимость шкал, метки шкал и кривые, и чтобы форматировать метки шкал, сетки, линии сетки и цвета этих линий.

## Форматирование осей

Используйте вкладку **Format and Precision** в диалоговом окне **Graph Properties** или **Chart Properties** для настройки внешнего вида и параметров осей X и Y на индикаторах **Graph** или **Chart**.

По умолчанию ось X конфигурируется для использования записи чисел с плавающей точкой (floating point) и имеет метку **Time**, а ось Y конфигурируется для использования автоматического форматирования (automatic formatting) и имеет метку **Amplitude**. Щелкните правой кнопкой индикатор **Graph** или **Chart** и выберите из контекстного меню пункт **Properties**, чтобы отобразить диалоговое окно **Graph Properties** или **Chart Properties** и сконфигурировать оси индикатора **Graph** или **Chart**.

Используйте вкладку **Format and Precision** в диалоговом окне **Graph Properties** или **Chart Properties** для задания числового формата осей на индикаторах **Graph** или **Chart**. Выберите вкладку **Scales**, чтобы изменить наименования осей и настроить внешний вид шкал на осях. По умолчанию числа на осях индикаторов **Graph** или **Chart** представляются шестью цифрами до автоматического перехода к экспоненциальной нотации.

Поместите птичку на пункте **Direct entry mode preferred** (режим прямого ввода), чтобы отобразить текстовые опции, которые позволяют Вам непосредственно вводить форматные строки. Введите форматную строку, которая определяет внешний вид шкал и представление чисел на осях. Более подробно о форматных строках см. в разделе *Using Format String* в справочной системе *LabVIEW Help*.

## Динамическое форматирование индикаторов Graph

Подсоедините проводник с динамическим типом данных (dynamic data type) к индикатору Waveform Graph, чтобы автоматически форматировать надписи на осях и временные метки на оси X. Например, если Вы настроили экспресс ВП **Simulate Signal** на генерацию синусоидального сигнала и использование абсолютного времени и подсоединили его выход к индикатору Waveform Graph, то надпись на графике автоматически изменится на **Sine** и на оси X отобразится время и дата запуска ВП.

Щелкните правой кнопкой индикатор **Graph** и выберите из контекстного меню пункт **Ignore Attribute**, чтобы игнорировать название

графика, которое входит в качестве атрибута в динамический тип данных. Щелкните правой кнопкой индикатор Graph и выберите из контекстного меню пункт **Ignore Time Stamp**, чтобы игнорировать конфигурацию временной метки, которую включает динамический тип данных.



**Примечание.** Если данные, которые Вы отображаете на индикаторе Waveform Graph, содержит абсолютную временную метку, то выберите пункт **Ignore Time Stamp**, чтобы игнорировать временную метку. Более подробно об использовании динамического типа данных с индикаторами Graph и об экспресс ВП см. в руководстве *Getting Started with LabVIEW* (LabVIEW 7 Express Вводный курс). Более подробно об использовании динамического типа данных см. в разделе *Динамический тип данных* в Главе 5 *Построение блок-диаграммы*.

### Использование режима плавного обновления

Когда LabVIEW обновляет объект при выключенном режиме плавного обновления (**Smooth Updates**), происходит удаление старого содержимого и вывод нового значения, что приводит к заметному мерцанию. Использование режима плавного обновления устраняет мерцание, вызванное удалением и повторным выводом значения. Щелкните правой кнопкой индикатор Graph и выберите из контекстного меню пункт **Advanced»Smooth Updates**, чтобы использовать экранный буфер для уменьшения мерцания. Использование режима **Smooth Updates** может снизить эффективность в зависимости от компьютера и видеосистемы, которые Вы используете.

## Настройка индикаторов Chart

В отличие от индикаторов Graph, которые отображают осциллограмму как единое целое, переписывая новые данные поверх уже сохраненных, индикаторы Chart обновляются периодически и сохраняют предысторию ранее поступивших данных.

Вы можете настроить индикаторы Waveform Chart и Intensity Chart, чтобы приспособить их к вашим потребностям в представлении данных или для повышения информативности их отображения. Опции, доступные для индикаторов Chart включают полосу прокрутки (**Scrollbar**), панель настройки кривых (**Plot Legend**), панель настройки шкалы (**Scale Legend**), палитру инструментов управления графиком (**Graph Palette**), цифровой дисплей (**Digital Display**)

и представление шкал во времени. Вы можете модифицировать длину предшествующей диаграммы, режимы обновления и способы отображения кривых.

### Длина предыстории диаграммы

LabVIEW сохраняет точки данных, уже включенных в диаграмму, в буфер, который называется предысторией диаграммы (chart history). По умолчанию размер буфера предыстории диаграммы составляет 1024 точки данных. Вы можете изменить буфер предыстории, щелкнув правой кнопкой по индикатору Chart и выбирая из контекстного меню пункт **Chart History Length**. Для просмотра ранее накопленных данных используйте полосу прокрутки. Чтобы отобразить полосу прокрутки щелкните правой кнопкой индикатор Chart и выберите из контекстного меню пункт **Visible Items»X Scrollbar**.

### Режимы обновления диаграммы

Индикаторы Chart используют три режима отображения данных. Щелкните правой кнопкой индикатор Chart и выберите из контекстного меню пункт **Advanced»Update Mode**. Затем выберите один из режимов **Strip Chart**, **Scope Chart** или **Sweep Chart**. По умолчанию устанавливается режим **Strip Chart**. Эти режимы имеют следующий смысл:

- **Strip Chart** (ленточная диаграмма) – Показывает текущие данные, непрерывно прокручивая слева направо вдоль диаграммы, при этом старые данные остаются слева, о новые добавляются справа. Режим ленточной диаграммы похож на самописец с непрерывной бумажной лентой.
- **Scope Chart** (периодическая развертка диаграммы) – Показывает одну кривую данных, рисуя ее вдоль индикатора слева направо. Каждое новое значение изображается на кривой справа от предыдущего значения. Когда кривая достигнет правой границы индикатора, LabVIEW сотрет кривую и начнет строить ее заново, начиная с левой границы. Такая периодическая смена изображения похожа на то, как это происходит на экране осциллографа.
- **Sweep Chart** (периодическая развертка диаграммы с маркером) – Работает подобно режиму **Scope Chart**, за исключением того, что кривая для старых данных продолжает оставаться справа, а кривая для новых данных, отделенная вертикальной линией (маркером)

ром), добавляется слева. Когда кривая достигает правой границы области просмотра, она не удаляется, пока не «сотрется» проходящим по ней маркером.

### Расположение кривых в одной области просмотра или один над другим

Вы можете отображать несколько кривых на одном индикаторе Chart, используя одну вертикальную шкалу (режим совмещенных кривых – overlaid plots) или несколько вертикальных шкал (режим разделенных кривых – stacked plots). На Figure 12-2 показаны примеры совмещенных и раздельных кривых.

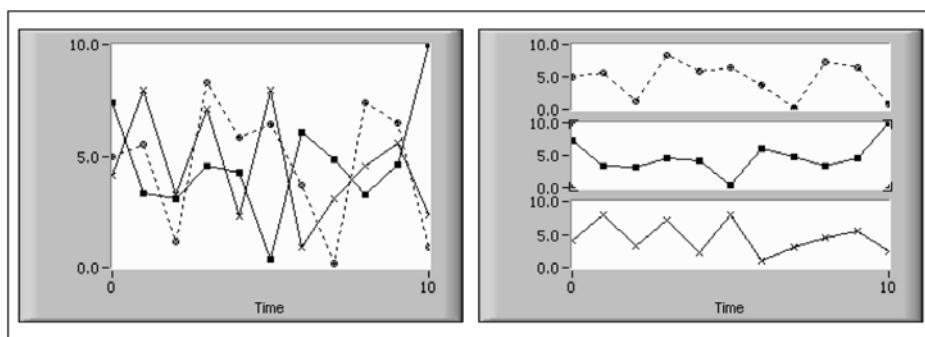


Figure 12-2. Индикаторы Chart с совмещенными и раздельными кривыми

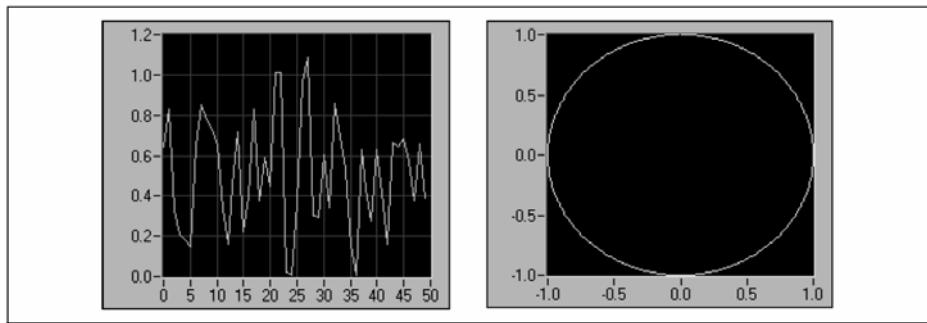
Щелкните правой кнопкой индикатор Chart и выберите из контекстного меню пункт **Stack Plots**, чтобы просматривать диаграмму из нескольких кривых раздельно, каждая со своей вертикальной шкалой. Для просмотра нескольких кривых в совмещенном режиме на одном поле с общей вертикальной шкалой, выберите пункт **Overlay Plots**.

Примеры различных типов индикаторов Chart при подаче на их вход данных различного типа можно найти в виртуальном приборе Charts из библиотеки examples\general\graphs\charts.llb.

## Индикаторы Waveform Graph и XY Graph

Индикаторы Waveform Graph отображают измерения, полученные в результате равномерной дискретизации (с постоянным временным шагом). Индикаторы XY Graph отображают набор точек, неважно,

с равномерной дискретизацией или нет. На Figure 12-3 показаны примеры индикаторов Waveform Graph и XY Graph.



**Figure 12-3. Индикаторы Waveform Graph и XY Graph**

Индикаторы Waveform Graph отображают кривые только однозначных функций вида  $y=f(x)$ , у которых точки равномерно распределены вдоль оси  $x$ . Например, этот могут быть сигналы, зависящие от времени. Индикаторы XY Graph являются более универсальными, так как они могут отображать в декартовых координатах объекты, задаваемые многозначными функциями. Примерами таких объектов могут служить кривая в виде окружности и сигналы с неравномерной дискретизацией. Оба типа индикаторов и Waveform Graph и XY Graph могут отображать кривые, состоящие из произвольного числа точек.

Оба типа индикаторов и Waveform Graph и XY Graph могут принимать на входе данные нескольких типов, что позволяет свести к минимуму манипуляции с данными, которые Вам придется производить до подключения их на вход индикаторов.

### **Типы данных для построения одной кривой на индикаторе Waveform Graph**

Индикатор Waveform Graph позволяет подавать на его вход два типа данных, которые приведут к построению одной кривой.

При подаче на вход одного массива, значения его элементов интерпретируются как точки на графике, причем аргумент  $x$  при переходе от одной точки к другой увеличивается на 1, начиная с  $x=0$ . На вход индикатора Waveform Graph может быть подан также кластер

типа waveform, содержащий в качестве компонентов:  $x_0$  – начальное значение аргумента  $x$ , его приращение  $\Delta x$  и массив  $y$  данных.

Примеры использования типов данных, подключаемых на вход индикатора Waveform Graph и приводящих к построению одной кривой можно найти в виртуальном приборе Waveform Graph из библиотеки examples\general\graphs\gengraph.11b.

## Построение нескольких кривых на индикаторе Waveform Graph

При поступлении на вход индикатора Waveform Graph двумерного массива строится несколько кривых, причем каждой строке массива соответствует одна кривая. Индикатор Waveform Graph интерпретирует эти данные как точки на графике, причем аргумент  $x$  при переходе от одной точки к другой увеличивается на 1, начиная с  $x=0$ . Чтобы каждый столбец массива (а не его строка) интерпретировался как одна кривая графика, подсоедините данные в виде двумерного массива к входу индикатора Waveform Graph, щелкните его правой кнопкой и выберите из контекстного меню пункт **Transpose Array**. Такой прием в частности полезен, когда Вы опрашиваете несколько каналов DAQ устройства, поскольку устройство возвращает данные в виде двумерного массива, и при этом каждый канал сохраняется как отдельный столбец массива. Пример индикатора Waveform Graph, который получает на вход такой тип данных, можно найти в ВП Waveform Graph из библиотеки example\general\graphs\gengraph.11b.

Для построения нескольких кривых на индикатор Waveform Graph можно подать также кластер из значения  $x$ , значения  $\Delta x$  и двумерного массива данных  $y$ . В этом случае индикатор Waveform Graph интерпретирует данные из массива  $y$  как точки на графике, причем аргумент  $x$  при переходе от одной точки к другой увеличивается на  $\Delta x$ , начиная с  $x=0$ . Такой тип данных характерен при отображении графиков нескольких сигналов, равномерно дискретизированных с одной и той же частотой. Примером индикатора Waveform Graph, который получает на вход такой тип данных, является индикатор (Xo, dX, Y) Multi Plot3 в ВП Waveform Graph из библиотеки example\general\graphs\gengraph.11b.

Для построения нескольких кривых на индикатор Waveform Graph можно подать массив кластеров. Каждый кластер содержит массив точек данных  $y$ . Этот внутренний массив задает точки на кривой, а

внешний массив содержит по одному кластеру на каждую кривую. На Figure 12-4 показан такой массив из кластера *y*.

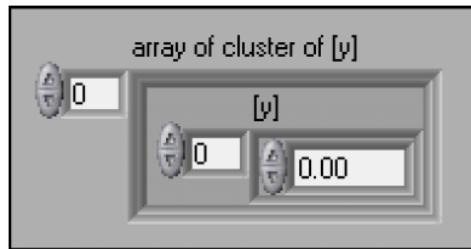


Figure 12-4. Массив из кластера *y*

Используйте такой способ вывода нескольких кривых вместо подключения на вход двумерного массива, если число точек в каждой кривой различно. Например, когда Вы получаете данные от нескольких каналов с разными временными параметрами, используйте такую структуру данных вместо двумерного массива, поскольку все строки двумерного массива должны содержать одинаковое число элементов. Число элементов во внутренних массивах массива кластеров может быть различным. Примером индикатора Waveform Graph, который получает на вход такой тип данных, является индикатор (Y) Multi Plot2 в ВП Waveform Graph из библиотеки `example\general\graphs\gengraph.llb`.

Для построения нескольких кривых на индикатор Waveform Graph можно подать кластер из начального значения *x*, значения  $\Delta x$  и массива, который содержит кластеры. Каждый из этих кластеров содержит массив значений данных *y*. Используйте функцию *Bundle* для объединения массивов в кластеры и функцию *Build Array* для построения из полученных кластеров массива. Можно также использовать функцию *Build Cluster Array*, которая создает массив кластеров, которые содержат то, что Вы подадите на входы. Примером индикатора Waveform Graph, который получает на вход такой тип данных, является индикатор (Xo, dX, Y) Multi Plot2 в ВП Waveform Graph из библиотеки `example\general\graphs\gengraph.llb`.

Для построения нескольких кривых на индикатор Waveform Graph можно подать массив кластеров из значений *x*, значений  $\Delta x$  и массивов данных *y*. Это наиболее общий тип данных, который

может подключаться к входу индикатора Waveform Graph, поскольку в этом случае можно задавать индивидуальные значения стартовых значений и приращений по оси X для каждой кривой. Примером индикатора Waveform Graph, который получает на вход такой тип данных, является индикатор (X<sub>0</sub>, dX, Y) Multi Plot1 в ВП Waveform Graph из библиотеки example\general\graphs\gengraph.llb.

### **Типы данных для построения одной кривой на индикаторе XY Graph**

Для построения одной кривой на индикаторе XY Graph можно подать кластер, содержащий массив *x* и массив *y*. Кроме того, для построения одной кривой на индикаторе XY Graph можно подать массив точек, где каждая точка сеть кластер, содержащий значение *x* и значение *y*. Пример типов данных для построения одной кривой на индикаторе XY Graph можно найти в ВП XY Graph из библиотеки example\general\graphs\gengraph.llb.

### **Типы данных для построения нескольких кривых на индикаторе XY Graph**

Для построения нескольких кривых на индикаторе XY Graph можно подать массив кривых, где каждая кривая есть кластер, содержащий массив *x* и массив *y*. Кроме того, для построения нескольких кривых на индикаторе XY Graph можно подать массив кластеров кривых, где каждая кривая есть массив точек. Каждая точка представляет собой кластер, содержащий значение *x* и значение *y*.

Пример типов данных для построения нескольких кривых на индикаторе XY Graph можно найти в ВП XY Graph из библиотеки example\general\graphs\gengraph.llb.

## **Индикаторы Waveform Chart**

---

Индикаторы Waveform Chart – это специальный тип числовых индикаторов, предназначенных для отображения одной или нескольких кривых. Примеры использования индикаторов Waveform Chart можно найти в библиотеке example\general\graphs\charts.llb

Если Вы подаете на индикатор Chart одно или несколько значений за один раз, то LabVIEW интерпретирует эти данные как точки на

диаграмме, при этом аргумент  $x$  получает каждый раз приращение 1, начиная с  $x=0$ . Индикатор Chart трактует эти данные как новые точки одной кривой. Если Вы подадите на вход индикатора Chart тип данных waveform, то индекс  $x$  будет соответствовать формату представления времени.

Частота поступления новых данных на индикатор Chart определяет, как часто будет перерисовываться кривая на диаграмме.

Для построения нескольких кривых на индикатор Waveform Chart нужно подать данные, объединенные в кластер, образованный числовыми скалярами, каждый из которых представляет одну очередную точку на каждой кривой.

Если Вы хотите добавлять несколько точек для кривых за одно обновление, подсоедините массив кластеров из чисел к входу индикатора Chart. Каждое число представляет одну точку значения  $y$  на каждой кривой.

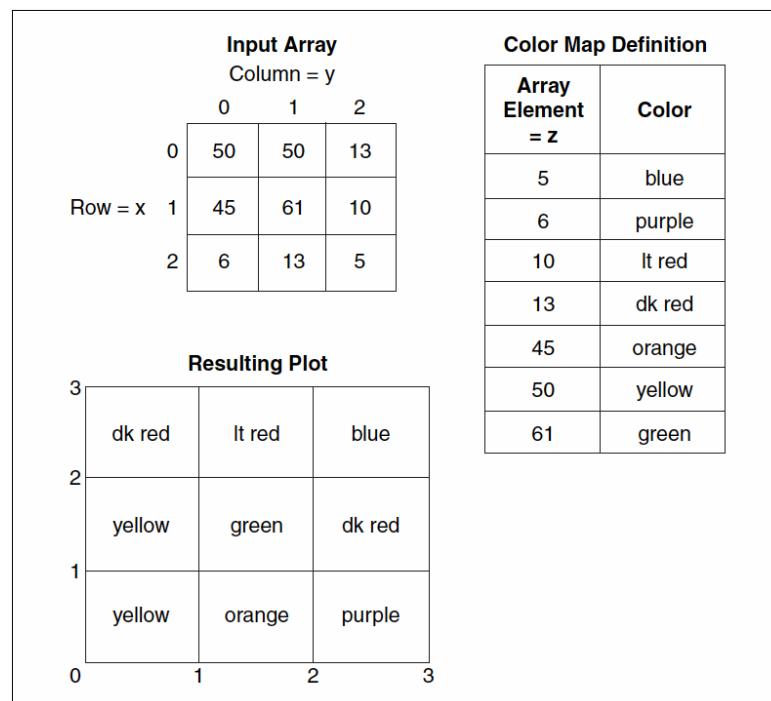
Если Вы не можете определить число отображаемых кривых до запуска вашего ВП, или Вы хотите добавлять несколько точек для нескольких кривых за одно обновление, подсоедините к входу индикатора Chart двумерный массив чисел или кластеров типа waveform. Также как и индикатор Waveform Graph, индикатор Waveform Chart по умолчанию интерпретирует строки массива как новые данные для каждой кривой. Если же нужно рассматривать в качестве новых данных для каждой кривой столбцы массива, то щелкните по индикатору правой кнопкой и выберите из контекстного меню пункт **Transpose Array**.

## Индикаторы Intensity Graph и Intensity Chart

---

Используйте индикаторы Intensity Graph и Intensity Chart (графики и диаграммы интенсивности) для отображения трехмерных данных в виде двумерных изображений посредством раскрашивания областей на декартовой плоскости в различные цвета. Например, Вы можете использовать индикаторы Intensity Graph и Intensity Chart для отображения пространственно распределенных данных, таких как температурные поля или карты местности, на которых числовая величина (температура или высота) отображается цветом. Индикаторы Intensity Graph и Intensity Chart принимают на входе двумерный массив чисел. Каждое число массива представляет определен-

ный цвет. Индексы элементов в этом двумерном массиве определяют места на плоскости, которые будут закрашены в эти цвета. Принцип действия индикатора Intensity Chart показан на Figure 12-5.



**Figure 12-5.** Отображение цветов в индикаторе Intensity Chart

Строки данных поступают в индикатор и отображаются на графике или диаграмме в виде столбцов. Если Вы хотите, чтобы строки данных отображались на дисплее в виде строк, щелкните правой кнопкой индикатор Graph или Chart и выберите из контекстного меню пункт **Transpose Array**.

Индексы элементов массива соответствуют нижнему левому углу цветового блока. Цветовой блок ограничен областью, которая лежит между соседними точками, определяемыми индексами элементов массива. Индикаторы Intensity Graph и Intensity Chart могут отображать до 256 дискретных цветовых оттенков.

После того как блок данных отобразится на индикаторе Intensity Chart, исходная (чистая) плоскость для построения изображения сдвинется вправо относительно последнего блока данных. При получении очередной порции данных, соответствующее им изображение будет появляться справа от уже существующего изображения. Когда область отображения индикатора Intensity Chart полностью заполнится, самые старые данные будут прокручиваться влево и исчезать из поля зрения. Этот процесс происходит точно так же, как и в индикаторе Strip Chart. Более подробно о режимах обновления изображения в индикаторах типа Chart см. в разделе *Режимы обновления диаграммы* в настоящей Главе.

Примеры использования индикаторов Intensity Graph и Intensity Chart можно найти в библиотеке examples\general\graphs\intgraph.llb.

## Схема отображения цветов

Вы можете установить схему отображения цветов (color mapping) для индикаторов Intensity Graph и Intensity Chart интерактивно, точно так же, как это делается для элемента управления Color Ramp (цветовой клин). Более подробно об элементе управления Color Ramp см. в разделе *Цветовой клин (Color Ramp)* в Главе 4 *Построение лицевой панели*.

Вы можете установить схему отображения цветов для индикаторов Intensity Graph и Intensity Chart программно с помощью узла свойств (Property Node), который может быть использован двумя способами. Обычно соответствие значение-цвет указывается в узле свойств. В рамках этого способа нужно специфицировать свойство Z Scale: Marker Values []. Это свойство содержит массив кластеров, в котором каждый кластер содержит числовое граничное значение и соответствующий ему цвет. Когда Вы задаете схему отображения цветов таким способом, Вы можете задать цвет, соответствующий верхней границе, с помощью свойства Z Scale: High Color, и цвет, соответствующий нижней границе, с помощью свойства Z Scale: Low Color. Индикаторы Intensity Graph и Intensity Char допускают до 254 различных цветов, что вместе с верхним и нижним граничными цветами составляет 256. Если Вы все же зададите более 254 цветов, то будет создана таблица соответствия из 254 цветов, полученная интерполяцией заданных цветовых соответствий.

Если Вы отображаете побитовое отображение (bitmap) на индикаторе Intensity Graph, то задавайте цветовую таблицу с помощью свойства Color Table. Согласно этому способу Вы можете задать массив до 256 цветов. Данные, поступающие на индикатор, отображаются в индексы этой таблицы, основываясь на цветовой шкале (color scale) индикатора. Если этот диапазон установлен от 0 до 100, то значение 0 данных отображается в индекс 1, а значение 100 отображается в индекс 254. Все, что меньше 0, отображается в нижний граничный цвет (индекс 0), а все, что больше 100, отображается в верхний граничный цвет (индекс 255).



**Примечание.** Цвета, которые Вы хотите отобразить на индикаторах Intensity Graph или Intensity Chart, ограничены чистыми цветами и возможностями вашей видео карты и монитора.

### Необязательные компоненты индикатора Intensity Chart

Индикатор Intensity Chart имеет много общих с индикатором Waveform Chart необязательных элементов, которые можно сделать видимыми или скрытыми, щелкнув по индикатору правой кнопкой и выбрав из контекстного меню пункт **Visible Item**. Кроме того, поскольку индикатор Intensity Chart имеет цвет в качестве третьей координаты, то имеется шкала похожая на элемент управления Color Ramp (цветовой клин), которая определяет диапазон и схему отображения значений цвета.

Подобно индикатору Waveform Chart, индикатор Intensity Chart поддерживает функцию сохранения данных от предыдущих обновлений входных данных (предыстория данных), для чего используется буфер. Для настройки этого буфера щелкните индикатор и выберите из контекстного меню пункт **Chart History Length**. По умолчанию размер этого буфера составляет 128 точек данных. Индикатор Intensity Chart довольно интенсивно использует память. Например, отображение диаграммы чисел с однократной точностью с предысторией из 512 точек и со 128-ю значениями потребует  $512*128*4$  байт (размер числа с однократной точностью), или 256 КБ.

### Необязательные компоненты индикатора Intensity Graph

Индикатор Intensity Graph работает так же, как и индикатор Waveform Chart, за исключением невозможности сохранения предыду-

ших данных и отсутствия режимов обновления изображения. Каждый раз при поступлении новых данных на вход индикатора Intensity Graph новые данные будут замещать старые.

Индикатор Intensity Graph может иметь курсоры, так же, как и другие индикаторы типа Graph. Каждый курсор отображает значения  $x$ ,  $y$  и  $z$  для выбранной точки графика.

## Индикаторы Digital Waveform Graph

---

Используйте индикатор Digital Waveform Graph для отображения цифровых данных, особенно когда Вы работаете с временными диаграммами или логическими анализаторами. Более подробно о вводе/выводе цифровых данных см. в руководстве *LabVIEW Measurements Manual*.

Индикатор Digital Waveform Graph допускает на входе данные типа digital waveform (цифровая осциллограмма), digital data (цифровые данные) и массивы этих типов данных. По умолчанию индикатор Digital Waveform Graph сжимает цифровые шины так, что индикатор отображает цифровые данные в виде одной кривой. Если Вы подключите массив цифровых данных, то индикатор Digital Waveform Graph отобразит каждый элемент массива в виде отдельной кривой в том порядке, как элементы идут в массиве.

Индикатор Digital Waveform Graph, показанный на Figure 12-6, отображает цифровые данные в виде одной кривой. В этом примере ВП конвертирует числа из массива **Numbers** в тип digital data (цифровые данные) и отображает двоичные представления этих чисел в цифровом индикаторе **Binary Representation**. На индикаторе **Digital Waveform Graph** число 0 отображается без линии сверху; это символизирует, что все биты этого числа равны нулю. Число 255 отображается без линии снизу; это символизирует, что все биты этого числа равны 1.

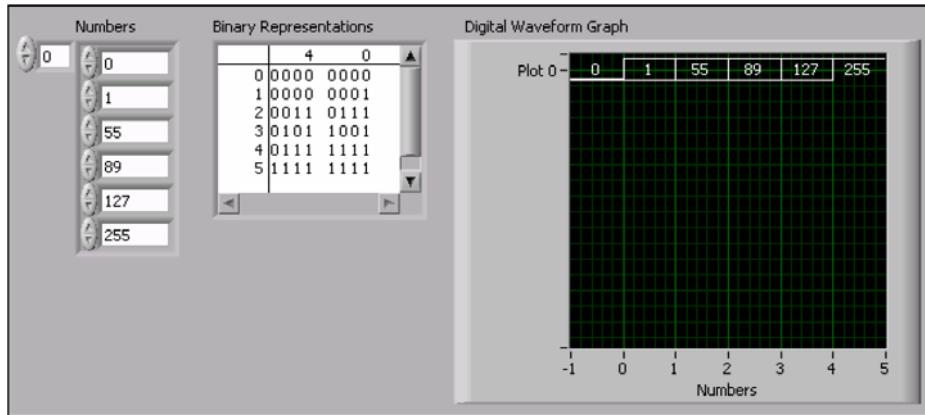


Figure 12-6. Отображение цифровых данных в виде отдельной кривой

Более подробно о конфигурировании кривых на индикаторе Digital Waveform Graph см. в справочной системе *LabVIEW Help*.

Чтобы изобразить в виде кривой каждую выборку (sample) цифровых данных, щелкните правой кнопкой мыши на оси *y* и выберите из контекстного меню пункт **Expand Digital Buses**. После этого каждая кривая будет соответствовать своей выборке из цифровых данных.

Индикатор Digital Waveform Graph, показанный на Figure 12-7, отображает шесть чисел, заданных в элементе управления **Number**.

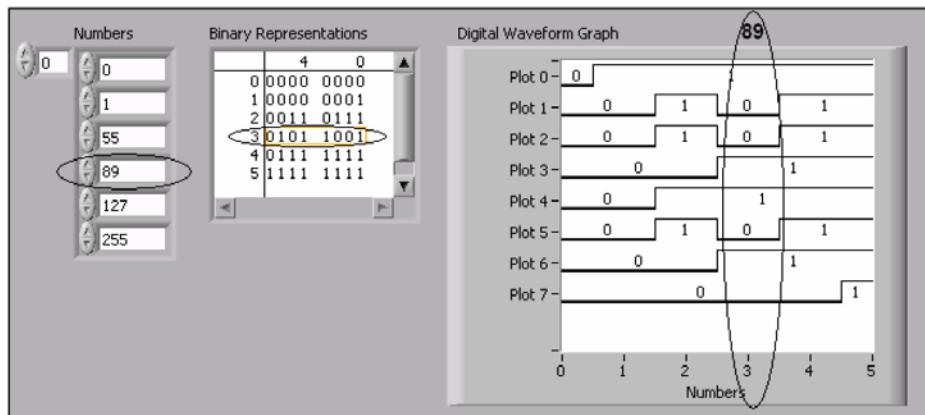


Figure 12-7. Поразрядное отображение целых чисел

Цифровой индикатор **Binary Representations** отображает двоичные представления чисел. Каждый столбец в этой таблице представляет один бит. Например, число 89 представляется в памяти 7-ю битами (0 в столбце 7 означает, что этот бит не используется). Точка 3 на графике в индикаторе **Digital Waveform Graph** отображает 7 бит, необходимых для представления числа 89 и нулевое значение для представления неиспользуемого восьмого бита на кривой Plot 7.

ВП на Figure 12-8 преобразует массив чисел в цифровые данные и использует функцию Build Waveform для объединения начального времени (start time), временного шага (dt) и чисел, введенных в цифровой элемент управления, чтобы потом отобразить цифровые данные.

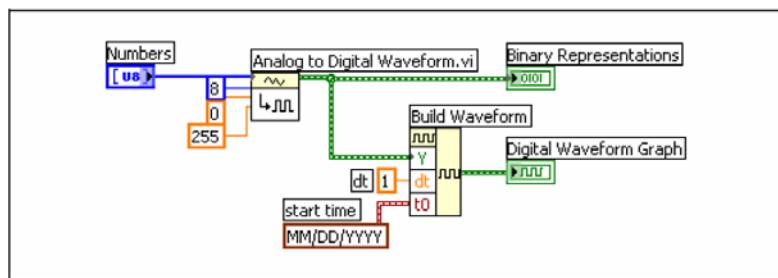


Figure 12-8. Преобразование массива чисел в цифровые данные

Более подробно об элементе управления Digital Data (цифровые данные), о типе данных digital waveform (цифровая осциллограмма) и о конвертировании данных к типу digital data (цифровые данные) см. в разделе Элемент управления цифровая осциллограмма (*Digital Waveform Control*) и Элемент управления цифровыми данными (*Digital Data Control*) в Главе 4 *Построение лицевой панели*.

### Маскирование данных

ВП на Figure 12-7 выводит график, в котором каждая кривая представляет один бит данных. Вы можете также селектировать, переупорядочивать и комбинировать биты в данных перед их отображением на графике. Селектирование, переупорядочивание и комбинирование битового представления называется маскированием (masking) данных.

Используйте маску для комбинирования кривых двух или более различных битов и отображения их затем на одной кривой. Если у Вас есть массив 8-битных целых чисел, то Вы можете отображать до 8 бит на одной кривой. Если у Вас есть массив из 16-битных целых чисел, то Вы можете отображать до 16 бит на одной кривой и т.д. Вы также можете отображать один и тот же бит дважды или большее число раз на одной кривой.

## Трехмерные графики

---



**Примечание.** Трехмерные элементы управления и индикаторы доступны только в пакетах LabVIEW Full Development System и LabVIEW Professional Development System, работающих в среде Windows.

Для многих данных реального мира, таких как распределение температуры по поверхности, связанный частотно-временной анализ, траектория движения самолета и т.п. может потребоваться визуализация в трех измерениях. С помощью индикаторов для трехмерной графики Вы можете визуализировать трехмерные данные и изменять способ их отображения путем модификации свойств индикаторов.

Доступны следующие индикаторы для трехмерной графики:

- **3D Surface Graph** (график трехмерной поверхности) – Рисует поверхность в трехмерном пространстве. Когда Вы перетаскиваете этот индикатор с палитры на лицевую панель, LabVIEW соединяет его с ВПП, который принимает данные, представляющие эту поверхность.
- **3D Parametric Graph** (параметрический трехмерный график) – Рисует сложную поверхность в трехмерном пространстве. Когда Вы перетаскиваете этот индикатор с палитры на лицевую панель, LabVIEW соединяет его с ВПП, который принимает данные, представляющие эту поверхность.
- **3D Curve Graph** (график трехмерной кривой) – Рисует линию в трехмерном пространстве. Когда Вы перетаскиваете этот индикатор с палитры на лицевую панель, LabVIEW соединяет его с ВПП, который принимает данные, представляющие эту поверхность.

Для построения кривых и поверхностей используйте индикаторы трехмерной графики совместно с ВПП для трехмерной графики с подпалитры **3D Graph Properties**. Кривая образуется из отдельных точек на графике, каждая из которых имеет координаты  $x$ ,  $y$  и  $z$ . ВП затем соединяет эти точки линией. Кривая идеально подходит для визуализации траектории движущихся объектов, таких, например, как траектория летящего самолета.

Трехмерные графики используют технологию ActiveX и ВП, которые управляют трехмерным представлением. С помощью ВП с подпалитры **3D Graph Properties** можно управлять поведением трехмерной графики во время исполнения, устанавливая базовые свойства, свойства осей, свойства сетки и свойства проекции.

Индикатор 3D Surface Graph для отображения точек на графике использует координатные данные  $x$ ,  $y$  и  $z$ . Индикатор соединяет затем эти точки, формируя образ трехмерной поверхности, соответствующей этим данным. Например, Вы могли бы использовать индикатор 3D Surface Graph для построения объемной карты местности.

Когда Вы выбираете с палитры Controls один из индикаторов для трехмерной графики, LabVIEW помещает на лицевую панель контейнер ActiveX, который содержит трехмерный элемент управления. Кроме того, LabVIEW помещает на блок-диаграмму ссылку на этот элемент управления и соединяет ее с одним из трех ВП для трехмерной графики (**3D Surface.vi**, **3D Parametric.vi** или **3D Curve.vi**).

## **Тип данных waveform (осциллограмма)**

---

Тип данных waveform (осциллограмма) передает сами данные, начальное время и  $\Delta t$ . Вы можете создавать осциллограммы, используя функцию Build Waveform. Многие ВП и функции, предназначенные для ввода и обработки осциллограмм, по умолчанию принимают и возвращают данные типа waveform. Когда Вы подсоединяете данные типа waveform к входу индикаторов Waveform Graph и Waveform Chart, автоматически строятся графики осциллограмм на основе данных, начального времени и  $\Delta t$ . Более подробно об использовании данных типа waveform см. в Главе 5 *Creating a Typical Measurement Applications* руководства *LabVIEW Measurements Manual*.

## Тип данных digital waveform (цифровая осциллограмма)

---

Тип данных digital waveform (цифровая осциллограмма) передает начальное время,  $\Delta x$ , сами данные и атрибуты цифровой осциллограммы. Вы можете использовать функцию Build Waveform для создания цифровых осциллограмм. Когда Вы подсоединяете данные типа digital waveform к входу индикаторов Digital Waveform Graph, автоматически строятся графики осциллограмм на основе данных и временных параметров. Для отдельного просмотра выборок и сигналов цифровой осциллограммы, подсоедините данные типа digital waveform к цифровому индикатору Digital Data. Более подробно об использовании данных типа digital waveform см. в Главе 5 *Creating a Typical Measurement Applications* руководства *LabVIEW Measurements Manual*.

## 13. Графика и звук

---

Используйте ВП с палитры **Graphics & Sound** для отображения или изменения графики или звука в ваших ВП.

---

**Более подробно...**

Более подробно относительно использования графики и звука в ваших ВП см. справочную систему *LabVIEW Help*.

---

### Использование индикатора Picture (изображение)

---

Индикатор Picture – это универсальный индикатор для отображения изображений, которые могут содержать линии, окружности, текст и другие типы графических элементов. Поскольку управление индикатором Picture осуществляется на пиксельном уровне, Вы можете создавать практически любой графический объект. Чтобы создавать, изменять и просматривать изображения в LabVIEW, вместо применения специальных графических приложений используйте индикатор Picture и графические ВП с палитры **Graphics & Sound**.

Индикатор Picture имеет пиксельную координатную систему, в которой начало (0,0) – это пиксель, находящийся в левом верхнем углу индикатора. Горизонтальная координата (x) увеличивается при перемещении слева направо, а вертикальная координата (y) увеличивается при перемещении сверху вниз.

Если изображение слишком велико для отображения на индикаторе, то LabVIEW усекает изображение таким образом, чтобы Вы могли видеть только те пиксели, которые попадают в область отображения индикатора. Используйте инструмент Positioning для изменения размера индикатора и повторного запуска ВП, чтобы просмотреть все изображение целиком. Вы также можете отобразить вертикальную и горизонтальную полосы прокрутки индикатора Picture, чтобы просмотреть пиксели, не попавшие область просмотра индикатора. Щелкните правой кнопкой индикатор и выберите из контекстного меню пункт **Visible Items»Scrollbar**, чтобы отобразить полосы прокрутки индикатора.

Вы можете использовать свойства сервера ВП в классе Picture, чтобы программно модифицировать свойства индикатора Picture, та-

кие, например, как размер индикатора или его области просмотра. Более подробно об использовании свойств сервера ВП см. в Главе 17 *Программное управление ВП*.



Когда Вы помещаете индикатор Picture на лицевую панель, он появляется в виде пустой прямоугольной области, при этом на блок-диаграмме появляется соответствующий терминал, показанный слева.

Чтобы отобразить изображение в индикаторе Picture, Вы должны программно записать изображение в индикатор. Нельзя вставлять в индикатор Picture изображение, скопированное в буфер обмена (clipboard). Вы можете использовать ВП с палитры **Picture Functions** для задания набора инструкций (команд) по рисованию изображений. Каждый такой ВП получает набор входных данных, которые описывают инструкции по рисованию изображений. На основании этих данных ВП создает компактное описание спецификаций, которые затем подаются на индикатор Picture для формирования изображения. Более подробно об использовании ВП с палитры **Picture Functions** для получения изображений на индикаторе Picture см. в разделе *ВП с палитры Picture Functions* настоящей Главы.

## ВП с палитры Picture Plots (изображение кривых)

Используйте ВП с палитры **Picture Plots** (изображение кривых) для создания с помощью индикатора Picture графиков кривых общего вида. Эта палитра включает ВП Polar Plot (кривые в полярных координатах), Plot Waveform (построение осциллограммы), XY Graph (двукоординатный график), Smith Plot (диаграмма Смита – зависимость комплексного импеданса), Radar Plot (отображение кривых в стиле экрана радара) и Draw Scale (отображение шкалы).

ВП с палитры **Picture Plots** используют низкоуровневые графические функции, чтобы создать графический образ ваших данных и управлять графическим кодом для расширения функциональности. Эти графические образы не являются интерактивными, как встроенные в LabVIEW элементы управления, но Вы можете использовать их для визуализации информации способами, которые для встроенных элементов управления пока не доступны. Например, Вы можете использовать ВП Plot Waveform для создания графика с несколько отличными функциональными возможностями по сравнению со встроенным индикатором Waveform Graph.

## Использование ВП Polar Plot в качестве ВПП

Используйте ВП Polar Plot для рисования отдельных смежных квадрантов полярного графика или всего графика целиком. Так же как и для встроенных индикаторов вывода графиков, Вы можете указывать цвета различных компонентов, включая сетку, и указывать диапазон и формат для шкал.

ВП Polar Plot имеет широкие функциональные возможности. Именно поэтому на вход этого ВП подается довольно сложный кластер входных данных. Вы можете использовать и значения по умолчанию и настроечные элементы управления для упрощения этого ВП. Вместо создания на блок-диаграмме входного кластера по умолчанию, скопируйте и поместите на лицевую панель настроечные элементы управления из примера Polar Plot Demo.vi, находящегося в библиотеке examples\picture\demos.llb.

## Использование ВП Plot Waveform и Plot XY в качестве ВПП

Используйте ВП Plot Waveform, эмулирующий поведение встроенного индикатора Waveform Graph, чтобы вычертить кривую осциллографа в различных стилях, включая стиль точки, стиль линий и стиль полосок. Так же как и в случае встроенного индикатора Waveform Graph, Вы можете указывать цвета различных компонентов, включая сетку, и указывать диапазон и формат для шкал.

ВП Plot Waveform имеет широкие функциональные возможности. Именно поэтому на вход этого ВП подается довольно сложный кластер входных данных. Вы можете использовать и значения по умолчанию и настроечные элементы управления для упрощения этого ВП. Вместо создания на блок-диаграмме входного кластера по умолчанию, скопируйте и поместите на лицевую панель настроечные элементы управления из примера Waveform and XY Plots.vi, находящегося в библиотеке examples\picture\demos.llb.

ВП Plot XY и ВП Plot Multi-XY похожи на ВП Plot Waveform. Отличия управления ими касаются косметических аспектов их внешнего вида, поскольку они имеют три дополнительных стиля – два стиля разброса кривых (scatter plot) и стиль кривой, при котором линия рисуется при каждом отдельном значении  $x$ , чтобы отметить минимальное и максимальное значения  $y$  для этого значения  $x$ .

## Использование ВП Smith Plot в качестве ВПП

Используйте ВП Smith Plot (диаграмма Смита) для исследования поведения линий передачи, например в области телекоммуникаций. Линия передачи – это среда, через которую Вы передаете энергию или сигналы. Линия передачи может быть проволокой или атмосферой (эфиром), через которые предаются сигналы. Линии передачи оказывают воздействие на передаваемые сигналы. Это воздействие характеризуется импедансом линии передачи и может приводить к ослаблению сигналов и к фазовым сдвигам.

Импеданс линии передачи является мерой активной и реактивной составляющих сопротивления линии. Импеданс  $z$  обычно записывается в виде комплексного числа  $z = r + jx$ , где  $r$  – составляющая активного сопротивления,  $x$  – составляющая реактивного сопротивления.

Используйте ВП Smith Plot для отображения импеданса линий передачи. Диаграмма состоит из окружностей, соответствующих постоянным значениям активного и реактивного сопротивления.

Вы можете изобразить заданный импеданс  $r + jx$  указанием пересечения окружностей соответствующих значениям  $r$  и  $x$ . После того, как будет отображен импеданс, Вы можете использовать ВП Smith Plot в качестве визуального средства для подбора импеданса и вычисления коэффициента отражения линии передачи.

ВП Smith Plot имеет широкие функциональные возможности. Именно поэтому на вход этого ВП подается довольно сложный кластер входных данных. Вы можете использовать и значения по умолчанию и настроечные элементы управления для упрощения этого ВП. Вместо создания на блок-диаграмме входного кластера по умолчанию, скопируйте и поместите на лицевую панель настроечные элементы управления из примеров, использующих ВП Smith Plot, и находящихся в библиотеке examples\picture\demos.11b.

Для загрузки значений отображаемых импедансов их можно представлять комплексными числами вида  $r + jx$ .

Чтобы исключить потерю подробностей на диаграмме Смита, используйте ВП Normalize Smith Plot для нормировки данных. Дан-

ные, нормированные с помощью ВП Normalize Smith Plot, можно непосредственно подавать на ВП Smith Plot. Обычно требуется масштабирование данных диаграммы Смита относительно характеристического импеданса ( $Z_0$ ) системы.

## ВП с палитры Picture Functions

---

Используйте ВП с палитры Picture Functions для рисования фигур и введения текста в индикатор. Вы можете нарисовать точки, линии, фигуры и пиксельные области ( pixmap). Пиксельные области не выровненных данных являются двумерными массивами элементов, представляющих цвет, где каждое значение соответствует цвету или индексу в массиве цветовых значений RGB, зависящих от насыщенности цвета.

Первая строка палитры **Picture Functions** содержит ВП, которые предназначены для рисования точек и линий. Точка описывается кластером из двух 16-битовых целых чисел со знаком, которые представляют пиксельные координаты  $x$  и  $y$ .

Когда Вы используете ВП с палитры **Picture Functions**, изображение запоминает позицию графического курсора (graphics pen). Для большинства ВП с этой палитры можно указывать абсолютные координаты – то есть, относительно начала координат (0,0). Для ВП Draw Line (чертить линию) и Move Pen (переместить графический курсор) можно указывать или абсолютные или относительные координаты. Относительные координаты указываются относительно текущего положения графического курсора. Вы можете использовать ВП Move Pen для изменения положения графического курсора без рисования линии. Положение графического курсора изменяют только следующие ВП: Draw Point, Move Pen, Draw Line, Draw Multiple Lines.

Вторая строка палитры **Picture Functions** содержит ВП, которые применяются при рисовании фигур. Каждый из этих ВП чертит фигуру в прямоугольной области изображения. Этот прямоугольник задается кластером из четырех чисел, которые представляют левый, верхний, правый и нижний пиксели.

Третья строка палитры **Picture Functions** содержит ВП, которые применяются для рисования текста на изображении. ВП Get Text

Rect не создает текст. Вместо этого он используется для вычисления размера прямоугольника, ограничивающего область строки.

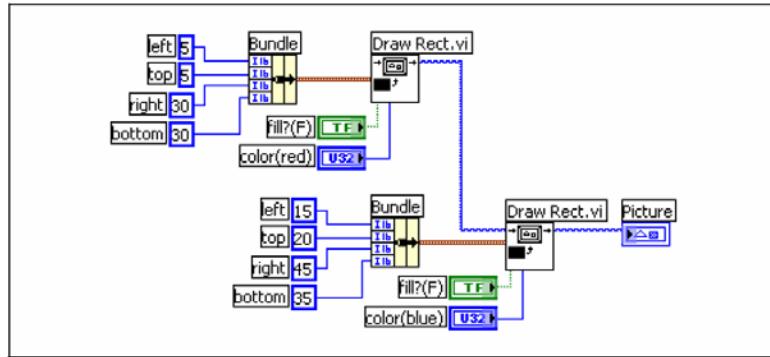
Четвертая строка палитры **Picture Functions** содержит ВП, которые применяются для рисования пиксельных областей (выровненных и не выровненных), для применения маски к изображению, для получения фрагментов исходных изображений и для конвертирования данных типа picture (изображение) в кластер выровненных графических данных (flattened image data cluster).

Последняя строка палитры **Picture Functions** содержит пустую константу типа picture (изображение), которая используется, когда необходимо начать с пустого изображения или сделать в нем изменения. Последняя строка палитры содержит также ВП для преобразования красной, зеленой и синей составляющих цвета в соответствующий RGB цвет, и для обратного преобразования значения цвета в красную, зеленую и синюю составляющие.

Вы можете подсоединять изображения, которые Вы создаете с помощью ВП с палитры **Picture Functions**, только к индикатору Picture или к входу **picture** ВП с этой же палитры. Фактически LabVIEW рисует новое изображение тогда, когда обновляется состояние индикатора Picture на лицевой панели.

Каждый ВП с палитры **Picture Functions** присоединяет свою инструкцию по рисованию изображения к инструкциям, поступающим через вход **picture**, и возвращает объединенные инструкции через выход **new picture**.

Представленная ниже блок-диаграмма использует ВП Draw Rect, для построения изображения двух пересекающихся прямоугольников.



## Задание и модификация цветов с помощью ВП с палитры Picture Functions

Многие ВП с палитры Picture Functions имеют вход **color** для задания цвета фигур и текста. Простейший способ задания цвета состоит в использовании константы color box (цветовой бокс), щелкнув которую можно выбрать нужный цвет.

Чтобы задавать значение цвета путем некоторых вычислений, а не с помощью константы color box, Вы должны понимать, каким образом в константе color box значение цвета связано с его числовым значением.

Цвет представляется 32-битовым целым со знаком, причем три его младших байта представляют красную, зеленую и синюю составляющие. Чтобы задать интервал синих оттенков, создайте массив 32-битных целых, у которых значение синего изменяется и превышает значения красного и зеленого. Чтобы задать интервал серых оттенков, создайте массив 32-битных целых, при этом значения красного, зеленого и синего цветов у каждого элемента массива должны быть одинаковы.

## ВП с палитры Graphics Formats

Используйте ВП с палитры Graphics Formats для чтения и записи данных в файлы некоторых стандартных графических форматов, включая bitmap (.bmp), Portable Network Graphics (.png) и Photographic Experts Group (.jpg). Вы можете использовать эти ВП для решения следующих задач:

- Чтение данных из графического файла и отображение их на индикаторе Picture.
- Чтение данных для последующего изменения изображения.
- Запись изображения с целью просмотра его в другом приложении.

Данные типа `bitmap` представляют собой двумерный массив, в котором значение каждой точки зависит от насыщенности цвета. Например, в черно-белом или 1-битном изображении, каждая точка является булевым значением. В 4-битном и 8-битном изображениях каждая точка является индексом цветовой таблицы. Для 24-битных изображений `true-color` каждая точка является смесью красного, зеленого и синего (RGB) значений.

ВП, которые читают и записывают графические файлы, работают с данными в простом выровненном (flattened) формате. Этот формат близок к записываемым на диск графическим файлам, в которых данные представлены в виде одномерного массива. Эти графические файлы являются пиксельными массивами (`Pixmap`), которые подобны концепции битовых массивов (`bitmap`). Вы можете отображать такие выровненные данные непосредственно с помощью ВП `Draw Flattened Pixmap`.

Для манипуляции с данными как с двумерным массивом, Вы должны конвертировать их к соответствующему формату, используя ВП `Unflatten Pixmap` или ВП `Flatten Pixmap`.

## ВП с палитры Sound

---

Используйте ВП с палитры Sound (звук) для использования звуковых файлов и функций в вашем ВП. Вы можете использовать эти ВП для решения следующих задач:

- Создание ВП, который проигрывает звуковые файлы, такие как записанные сообщения, при выполнении некоторых действий.
- Создание ВП, который проигрывает звуковые файлы, когда ВП начинает или завершает свое выполнение, или когда достигается некоторая точка в ВП.
- Настройка входного звукового устройства для ввода звуковых данных. Используйте ВП `Sound Input` для ввода звуковых данных.

Вы можете также читать любую звуковую информацию, поступающую через устройство.

- Настройка выходного звукового устройства на получение звуковых данных от других звуковых ВП. Вы можете управлять уровнем звука, проходящим через устройство, проигрывать или приостанавливать звук, а также удалять звук из вашей системы.

## 14. Файловый ввод/вывод

---

Операции файлового ввода/вывода передают данные в файлы и из файлов. Используйте ВП и функции с палитры **File I/O** для управления всеми аспектами файлового ввода/вывода, включая следующие:

- Открытие и закрытие файлов с данными.
- Чтение и запись файлов с данными.
- Чтение и запись файлов с данными в формате электронных таблиц.
- Перемещение и переименование файлов и директорий.
- Изменение характеристик файлов.
- Создание, модификация и чтение конфигурационного файла.

Используйте высокоуровневые ВП для выполнения общих операций ввода/вывода. Используйте низкоуровневые ВП и функции для каждой отдельной операцией ввода/вывода.

---

### Более подробно...

Более подробно относительно операций файлового ввода/вывода см. справочную систему *LabVIEW Help*.

---

## Основы файлового ввода/вывода

---

Типичные операции файлового ввода/вывода включают следующий процесс.

1. Создайте или откройте файл. Укажите, где находится существующий файл, или где Вы хотите создать новый файл, задавая путь или взаимодействуя с диалоговым окном, чтобы указать его место расположения. После открытия файл представляется своим ссылочным номером (refnum). Более подробно о ссылочных номерах см. в разделе *Ссылки на объекты или приложения* в Главе 4 *Построение лицевой панели*.
2. Читайте данные из файла или записывайте в него.

### 3. Закройте файл.

Большинство ВП и функций с палитры **File I/O** выполняют только один шаг файловой операции ввода/вывода. Однако, некоторые высокоуровневые ВП, созданные для общих операций файлового ввода/вывода, выполняют все три шага. Хотя эти ВП не всегда так же эффективны, как и низкоуровневые функции, ими легче пользоваться.

## Выбор формата файлового ввода/вывода

---

Использование ВП файлового ввода/вывода зависит от формата файлов. Вы можете читать или записывать данные в файлы в трех форматах – текстовый (text), двоичный (binary) и протокол данных (datalog). Тип нужного вам формата зависит от данных, которые Вы измеряете или создаете, и от приложения, которые будет использовать эти данные.

Воспользуйтесь следующими рекомендациями по выбору формата файла:

- Если Вы хотите сделать ваши данные доступными для других приложений, таких как Microsoft Excel, используйте текстовые файлы, поскольку они являются самыми употребительными и легко переносимыми.
- Если вам нужен произвольный доступ к файлам при чтении или при записи, или если требования к скорости записи и к месту на диске являются критическими, то используйте двоичные файлы, поскольку они эффективнее текстовых файлов и по скорости и по месту на диске.
- Если Вы хотите из LabVIEW манипулировать сложными записями данных или различными типами данных, используйте файлы протоколов данных, поскольку они дают наилучший способ хранения данных, если Вы намерены осуществлять доступ к данным только из LabVIEW и вам приходится хранить сложные структуры данных.

### Когда использовать текстовые файлы

Используйте текстовый формат файлов для ваших данных, чтобы сделать их доступными для других пользователей или приложений, если место на диске и скорость файловых операций не являются

критичными, если Вам не нужен произвольный доступ при чтении или записи и если требования к точности представления чисел не очень важны.

Текстовые файлы являются самым легким форматом для использования и распространения. Почти все компьютеры могут читать и записывать текстовые файлы. Многие типы программ, ориентированные на текстовые данные, могут читать текстовые файлы. Большинство приложений для управления приборами используют текстовые строки.

Храните данные в текстовом формате, когда Вы хотите получить доступ к ним из других приложений, таких как текстовый редактор или электронные таблицы. Чтобы сохранить данные в текстовом формате, используйте строковые функции для преобразования всех данных в текстовые строки. Текстовые файлы могут содержать информацию о данных различного типа.

Текстовые файлы обычно занимают больше места, чем двоичные файлы и файлы протокола данных, если исходные данные не являются текстовыми, как, например, данные графиков или диаграммы, поскольку ASCII представление числовых данных обычно более расточительно по сравнению с исходным числовым представлением. Например, Вы можете сохранить число -123.4567 в четырех байтах как число однократной точности с плавающей точкой. Однако его ASCII представление займет 9 байтов, по одному на каждый символ.

Кроме того, имеются трудности произвольного доступа к числовым данным в текстовых файлах. Хотя каждый символ в строке занимает в точности 1 байт, количество знакомест для представления числа в текстовом виде обычно не является фиксированным. Чтобы найти девятое число в текстовом файле, потребуется вначале прочитать и преобразовать предыдущие восемь чисел.

Вы можете потерять точность представления чисел, если храните числовые данные в текстовых файлах. Компьютеры хранят числовые данные в двоичной форме, а записываются в текстовый файл они обычно в десятичной нотации. Таким образом, при записи данных в текстовый файл может произойти потеря точности. В случае двоичных файлов потеря точности не происходит.

Примеры использования файлового ввода/вывода с текстовыми файлами можно найти в библиотеках examples\file\smplfile.llb и examples\file\sprdsht.llb.

## Когда использовать двоичные файлы

Сохранение двоичных данных, таких как целые числа, предполагает использование фиксированного числа байтов для каждого числа. Например, сохранение любого числа от 0 до 4 миллиардов (к примеру 1, 1000 или 1000000) в двоичном формате займет по 4 байта на одно число.

Используйте двоичные файлы для сохранения числовых данных и для обеспечения произвольного доступа к любому числу из этого файла. Двоичные файлы удобны для чтения машиной, в отличие от текстовых файлов, которые более удобны для чтения человеком. Двоичные файлы являются самым компактным и быстрым форматом для хранения данных. Вы можете использовать несколько типов данных в двоичных файлах, но это не является традиционным.

Двоичные файлы являются самыми эффективными, поскольку они занимают наименьшее дисковое пространство и поскольку не требуется конвертировать данные в текстовый формат и обратно при каждой операции сохранения или извлечения данных. Двоичный файл позволяет представлять в каждом байте дискового пространства 256 значений. Часто двоичные файлы содержат побайтный образ данных в том виде, как они хранятся в памяти, за исключением случаев чисел расширенной точности (extended) и комплексных чисел (complex). Когда файл содержит побайтный образ представления данных в памяти, чтение из него происходит максимально быстро, поскольку не требуются никакие преобразования. Более подробно о хранении данных в LabVIEW см. в заметках к приложению (Application Notes) *LabVIEW Data Storage*.



**Примечание.** Текстовые и двоичные файлы известны также как файлы с потоком байтов (byte stream files). Это означает, что они хранят данные в виде последовательности символов или байтов.

Примеры чтения и записи массивов значений удвоенной точности с плавающей точкой в двоичный файл можно найти в ВП Read

Binary File и Write Binary File из библиотеки  
examples\file\smplfile.llb.

## Когда использовать файлы протоколов данных

Используйте файлы протоколов данных (datalog files) для доступа и манипуляции с данными только из LabVIEW и для быстрого и легкого сохранения сложных структур данных.

Файлы протокола данных хранят данные в виде последовательности одинаково структурированных записей (record), подобно электронным таблицам, где каждая строка представляет собой запись. Все записи в файле протокола данных должны иметь одинаковый тип данных, ассоциированный с этим файлом. LabVIEW записывает каждую запись файла как кластер, содержащий сохраняемые данные. Однако компоненты записи протокола данных могут иметь произвольный тип, определяемый при создании этого файла.

Например, Вы можете создать протокол данных, типом записи которого является кластер из строки и числа. Следовательно, каждая запись протокола данных есть кластер из строки и числа. Однако, первой записью может быть ("abc", 1), в то время как второй записью может быть ("xyz", 7).

Использование файлов протокола данных требует мало манипуляций, что делает запись и чтение намного быстрее. Это также упрощает извлечение данных, поскольку Вы можете читать необходимые блоки данных без просмотра всех предыдущих в этом файле. Произвольный доступ в файлах протокола данных осуществляется быстро и легко, поскольку все, что Вам нужно для доступа к записи, – это ее номер. При создании файла протокола данных LabVIEW последовательно назначает всем записям уникальные номера.

Вы можете получить доступ к файлу протокола данных и с лицевой панели и с блок-диаграммы. Более подробно о доступе к файлам протокола данных с лицевой панели см. в разделе *Регистрация данных лицевой панели* настоящей Главы.

LabVIEW сохраняет запись в файле протокола данных каждый раз, когда связанный с ним ВП запускается. Нельзя перезаписывать запись после того, как LabVIEW запишет ее в файл протокола дан-

ных. Когда Вы читаете файл протокола данных, то можете за один раз считывать одну или несколько записей.

Примеры чтения и записи файлов протокола данных можно найти в библиотеке examples\file\datalog.llb.

## Использование ВП высокого уровня ввода/вывода

---

Используйте ВП высокого уровня ввода/вывода для выполнения общих операций ввода/вывода, таких как запись или чтение следующих типов данных:

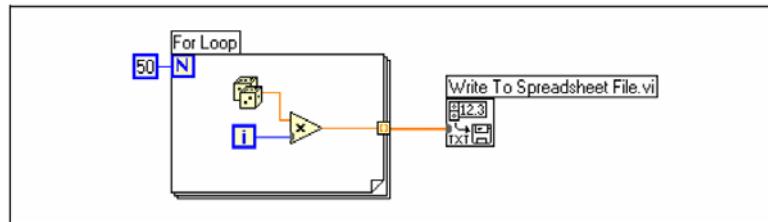
- Символы в текстовых файлах.
- Строки из текстовых файлов.
- Одно- и двумерные массивы чисел одинарной точности в текстовых файлах электронных таблиц.
- Одно- и двумерные массивы чисел одинарной точности или 16-битных целых чисел со знаком в двоичных файлах.

Вы можете сэкономить свое время и усилия на программирование, используя высокие ВП для записи и чтения в файлы. Высокие ВП выполняют операции чтения и записи совместно с операциями открытия и закрытия файлов. Избегайте употребления высоких ВП внутри циклов, поскольку эти ВП при каждом своем запуске выполняют операции открытия и закрытия файлов. Более подробно об удержании файлов в открытом состоянии при выполнении нескольких операций чтения/записи см. в разделе *Дисковый поток* в настоящей Главе.

Высокие ВП ожидают на своем входе путь к файлу. Если Вы не подсоедините путь к файлу, то появится диалоговое окно, чтобы Вы могли указать файл, из которого собираетесь считывать данные или записывать в него. Если будет иметь место ошибка, то появится диалоговое окно с описанием этой ошибки. Вы можете выбрать прекращение или продолжение выполнения.

На Figure 14-1 показано, как использовать высокий ВП Write To Spreadsheet File для записи чисел в файл электронной таблицы приложения Microsoft Excel. Когда запускается этот ВП,

LabVIEW спросит Вас, записывать данные в существующий файл или нужно создать новый.



**Figure 14-1.** Использование высокоуровневого ВП для записи в файл электронной таблицы

Используйте ВП с подпалитры **Binary Files** для записи и чтения файлов в двоичном формате. Данные могут иметь тип целых чисел или чисел однократной точности с плавающей точкой.

## Использование ВП и функций низкого уровня и с подпалитры **Advanced File I/O**

Используйте ВП и функции низкого уровня, а также ВП и функции с подпалитры **Advanced File I/O** для индивидуального управления каждой операцией файлового ввода/вывода.

Используйте основные низкоуровневые функции для создания или открытия файла, записи данных в файл или чтения из него и для закрытия файла. Остальные низкоуровневые функции используйте для выполнения следующих задач:

- Создание директорий.
- Перемещение, копирование или удаление файлов.
- Просмотр содержимого директории.
- Изменение характеристик файла.
- Манипуляция с путем к файлу.



Путь к файлу, показанный слева, является в LabVIEW типом данных, который определяет место размещения файла на диске. Путь включает устройство (тот), на котором находится файл, перечень директорий, между верхним уровнем файловой системы и фай-

лом и имя файла. Ввести или отобразить путь с использованием синтаксиса данной платформы можно с помощью путевого элемента управления File Path Control и индикатора File Path Indicator. Более подробно о них см. в разделе *Элементы управления и индикаторы для путей к размещению файлов* в Главе 4 *Построение лицевой панели*.

На Figure 14-2 показано, как использовать низкоуровневые ВП и функции для записи чисел в файл электронной таблицы приложения Microsoft Excel. Когда Вы запустите этот ВП, ВП Open\Create\Replace откроет файл numbers.xls. Функция Write File записет строку чисел в этот файл. Функция Close File закроет этот файл. Если Вы не закроете файл, данные для него остаются в памяти и файл остается недоступным для других приложений или других пользователей.

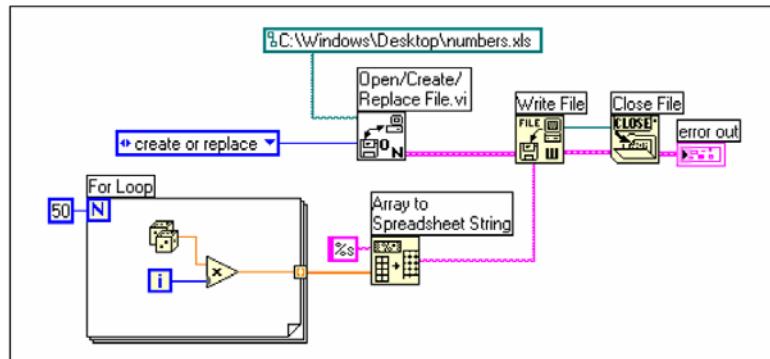


Figure 14-2. Использование низкоуровневого ВП для записи в файл электронной таблицы

Сравните ВП на Figure 14-2 с ВП на Figure 14-1, который решает ту же задачу. На Figure 14-2 для преобразования массива чисел в символьную строку использована функция Array To Spreadsheet String. На Figure 14-1 ВП Write To Spreadsheet File открывает файл, конвертирует массив чисел в символьную строку, записывает данные и закрывает файл.

Пример использования ВП и функций для низкоуровневого файлового ввода/вывода представлен в ВП Write Datalog File Example из библиотеки examples\file\datalog.llb.

## Дисковый поток

Вы также можете использовать ВП и функции низкоуровневого файлового ввода/вывода для создания дискового потока (disk streaming), который позволяет сэкономить ресурсы памяти. Дисковый поток – это технология файлового ввода/вывода, при которой файлы остаются открытыми на протяжении нескольких операций записи, например, внутри цикла. Хотя высокуровневые операции записи проще в использовании, но они открывают и закрывают файл при каждом своем выполнении. Ваши ВП могут стать более эффективными, если Вы избавитесь от частого открытия и закрытия одних и тех же файлов.

Дисковый поток уменьшает количество раз, когда функция взаимодействует с операционной системой для открытия и закрытия файла. Чтобы создать типичную операцию в стиле дискового потока, поместите ВП Open/Create/Replace File перед циклом, а функцию Close File после цикла. Внутри цикла можно производить многократное непрерывное записывание в файл без его открытия и закрытия.

Дисковый поток является идеальным в длинных операциях ввода/вывода данных, когда критичным является быстродействие. Вы можете записывать данные в файл непрерывно, пока не завершится процесс ввода/вывода. Для улучшения результата исключите на это время запуск других ВП и функций, таких как ВП и функции для анализа данных.

## Создание текстовых файлов и файлов электронных таблиц

---

Чтобы записать данные в текстовый файл (text file), Вы должны конвертировать ваши данные в символьную строку. Чтобы записать данные в файл электронной таблицы (spreadsheet file), Вы должны представить символьную строку в формате строки электронной таблицы, которая представляет собой обычную символьную строку с разделителями типа символов табуляции. Более подробно о форматировании символьных строк см. в разделе *Форматирование строк* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.

Запись текста в текстовый файл не требует форматирования, поскольку большинство текстовых редакторов могут читать неформатированную строку.

тированные текстовые файлы. Чтобы записать строку текста в текстовый файл, используйте ВП Write Characters To File, который автоматически открывает и закрывает файл.

Используйте ВП Write To Spreadsheet File или функцию Array To Spreadsheet String для преобразования набора чисел из осцилограммы, из диаграммы или полученных от внешних устройств в одну строку электронной таблицы. Более подробно об использовании этих ВП и функций см. в разделах *Использование ВП высокого уровня файла ввода/вывода* и *Использование ВП и функций низкого уровня и с подпалитры Advanced File I/O* в настоящей Главе.

Чтение текста из приложений типа текстового редактора может привести к ошибке, поскольку текстовые редакторы обычно используют форматирование текста с помощью различных шрифтов, цветов, стилей и размеров, которые не могут быть обработаны с помощью ВП высокого уровня файла ввода/вывода.

Если Вы хотите записать числа и текст в электронную таблицу или в текстовый редактор, используйте функции с палитр **String** и **Array** для форматирования данных и комбинирования строк. Затем запишите данные в файл. Более подробно об использовании этих функций для форматирования и комбинирования данных см. в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.

## **Форматирование и запись данных в файлы**

Используйте функцию Format Into File для форматирования строковых, числовых, путевых и булевых данных как текста и записи форматированных данных в файл. Довольно часто эти функции можно использовать вместодельного форматирования строки с помощью функции Format Into String и последующей записи результирующей строки с помощью ВП Write Characters To File или функции Write File.

Более подробно о форматировании строк см. в разделе *Форматирование строк* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*.

## Сканирование данных из файлов

Используйте функцию Scan From File, чтобы просмотреть текст в файле на предмет распознавания записи строковых, числовых, путевых или булевых значений и затем преобразовать текст в нужный тип данных. Часто можно использовать эту функцию вместо чтения данных из файла с помощью функции Read File или с помощью ВП Read Characters From File и последующего сканирования результирующей строки с помощью функции Scan From String.

## Создание двоичных файлов

---

Чтобы создать двоичный файл, соберите набор чисел и запишите их в файл. Используйте ВП Write To I16 и Write To SGL для сохранения в файле одно- и двумерных массивов целых чисел и чисел однократной точности с плавающей точкой. Используйте ВП Read I16 и Read SGL для чтения созданных таким образом файлов.

Чтобы записать числа других типов, такие как числа удвоенной точности с плавающей точкой или 32-битовые целые числа, используйте низкоуровневые функции или функции с палитры **Advanced File Functions**. При чтении файла используйте функцию Read File и указывайте нужный тип данных для чисел.

Примеры записи и чтения чисел с плавающей точкой в двоичный файл можно найти в ВП Read Binary File и в ВП Write Binary File из библиотеки examples\file\smplfile.llb.

## Создание файлов протоколов данных

---

Вы можете создать и читать файлы протоколов данных (datalog files) путем включения режима протоколирования лицевой панели или используя низкоуровневые функции или функции с палитры **Advanced File Functions** для сбора данных и записи их в файл. Более подробно о создании и использовании файлов протоколов данных для лицевой панели см. в разделе *Регистрация данных лицевой панели* в настоящей Главе.

Вам нет необходимости форматировать данные в файле протокола данных. Однако когда Вы записываете или читаете файл протокола данных, необходимо указывать тип данных. Например, если Вы измеряете и вводите температуру вместе с датой и временем

измерения, то можно записать эти данные в файл протокола данных, указав при этом, что данные являются кластером из одного числа и двух символьных строк. Пример записи данных в файл протокола можно найти в ВП Simple Temp Datalogger из библиотеки examples\file\datalog.11b.

При чтении файла, который включает показания температуры, записанные вместе с временем и датой измерений, следует указать, что Вы хотите читать кластер из одного числа и двух символьных строк. Пример чтения из файла протокола данных можно найти в ВП Simple Temp Datalog Reader из библиотеки examples\file\datalog.11b.

## Запись в файл осциллографом

Используйте ВП Write Waveforms To File и ВП Export Waveforms to Spreadsheet File для сохранения осциллографом в файлах.

Если Вы предполагаете использовать осциллограф, которую Вы создаете, только внутри ВП, то сохраните ее в виде файла протокола (.log). Более подробно о протоколировании данных см. в разделе *Когда использовать файлы протоколов данных* в настоящей Главе.

ВП на Figure 14-3 вводит несколько осциллографом, отображает их на графике и затем записывает в файл протокола данных.

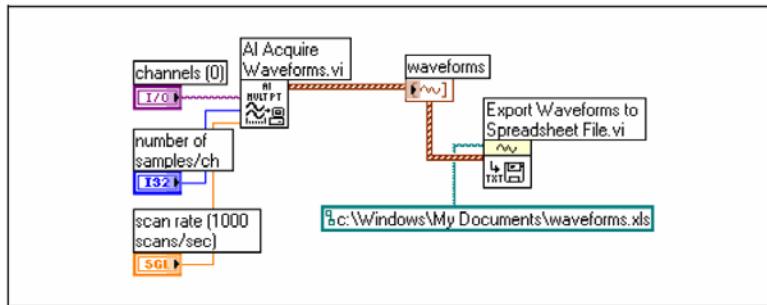


Figure 14-3. Запись нескольких осциллографом в файл протокола данных

## Чтение осциллограмм из файлов

Используйте ВП Read Waveform from File для чтения осциллограмм из файла. После того, как одна осциллограмма будет считана, можно добавлять или изменять компоненты данных типа waveform (осциллограмма) с помощью функции Build Waveform, или извлекать компоненты осциллограммы с помощью функции Get Waveform Attribute.

ВП на Figure 14-4 читает осциллограмму из файла, изменяет компоненту осциллограммы **dt** и отображает измененную осциллограмму на графике.

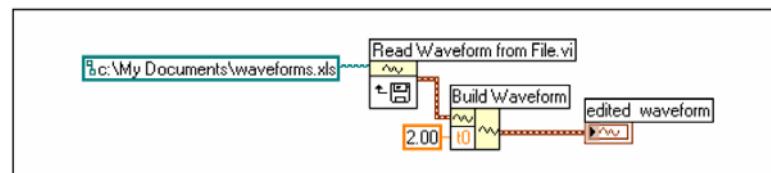


Figure 14-4. Чтение осциллограммы из файла

ВП Read Waveform from File может также читать из файла несколько осциллограмм. В этом случае ВП возвращает массив осциллограмм, который может отображаться в виде нескольких кривых на одном графике. Если Вы хотите получить доступ к одной осциллограмме из такого файла, то следует выбрать элемент из массива по его индексу, как это показано на Figure 14-5. Более подробно об индексации массивов см. в разделе *Массивы* в Главе 10 *Группировка данных с использованием строк, массивов и кластеров*. В данном ВП происходит доступ к файлу, содержащему несколько осциллограмм. Функция Index Array читает первую и третью осциллограмму из файла и строит их кривые на отдельных индикаторах Waveform Graph. Более подробно о графиках см. в Главе 12 *Графики и диаграммы*.

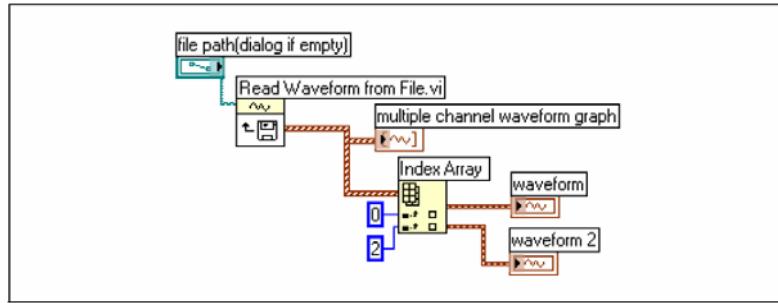


Figure 14-5. Чтение нескольких осциллограмм из файла

## Потоковые параметры

Многие ВП и функции с палитры **File I/O** имеют потоковые параметры (flow-through parameters), обычно это ссылочный номер (ref-num) или путь (path), при этом на выходе возвращаются те же значения этих параметров, которые поступают на вход. Используйте эти параметры для управления порядком выполнения функций. Соединяя потоковый выход одного узла с потоковым входом узла, который должен выполняться следующим, Вы тем самым создаете искусственную зависимость по данным. Без таких потоковых параметров Вам пришлось бы использовать структуры Sequence (последовательность), чтобы гарантировать нужный порядок выполнения операций. Более подробно об искусственной зависимости по данным см. в разделе *Зависимость по данным и искусственная зависимость по данным* в Главе 5 *Построение блок-диаграммы*.

## Создание конфигурационных файлов

Используйте ВП с палитры **Configuration File** для чтения и создания стандартных в Windows конфигурационных файлов (.ini) и для записи зависимых от платформы данных, таких как пути, в не зависимом от платформы формате, чтобы этими файлами могли воспользоваться ВП, созданные под различными платформами. ВП с палитры **Configuration File** не используют стандартный формат конфигурационных файлов. Хотя Вы и можете использовать ВП с палитры **Configuration File** на любой платформе для чтения и записи файлов, созданных с помощью ВП, но Вы не сможете использовать ВП с палитры **Configuration File** для создания или модификации конфигурационных файлов в формате Mac OS или UNIX.

Примеры использования ВП с палитры **Configuration File** можно найти в библиотеке examples\file\config.llb.



**Примечание.** Стандартным расширением для конфигурационного файла в Windows является .ini, однако ВП с палитры **Configuration File** работают с файлами с любым расширением, обеспечивая при этом правильный формат их содержимого. Более подробно о содержимом конфигурационных файлов см. в разделе *Формат конфигурационного файла в Windows*.

## Использование установок конфигурационных файлов

Стандартные конфигурационные файлы в Windows имеют специальный формат для хранения данных в текстовом файле. Вы можете легко получить программный доступ к данным из .ini файла, поскольку они представлены в жестком формате.

Например, рассмотрим конфигурационный файл со следующим содержимым:

```
[Data]  
Value=7.2
```

Вы можете использовать ВП с палитры **Configuration File** для чтения этих данных, как показано на Figure 14-6. Этот ВП использует ВП Red Key для чтения ключа (**key**) с именем **Value** из раздела (**section**) с именем **Data**. Этот ВП работает независимо от того, как этот файл изменялся, лишь бы он был в формате конфигурационного файла Windows.

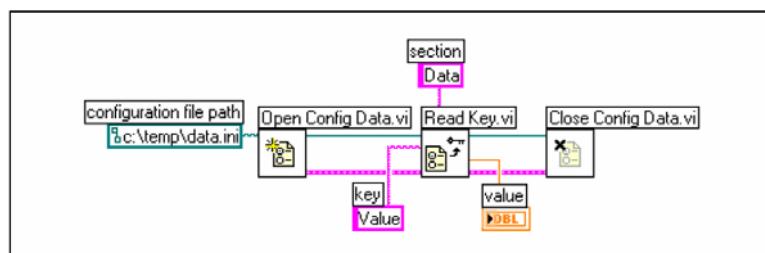


Figure 14-6. Чтение данных из .ini файла

## Формат конфигурационного файла в Windows

Конфигурационные файлы Windows – это текстовые файлы, разбитые на поименованные разделы (section). Имя каждого раздела заключено в квадратные скобки. Имя раздела должно быть уникальным. Разделы содержат пары: ключ (key)/значение (value), разделенные знаком равенства (=). Внутри каждого раздела ключевое имя должно быть уникальным. Ключевое имя представляет некоторый конфигурационный параметр, а наименование значения представляет собой установленное значение для этого параметра. Ниже приведен пример, поясняющий устройство конфигурационного файла:

```
[Section 1]
key1=value
key2=value
[Section 1]
key1=value
key2=value
```

Используйте следующие типы данных в ВП с палитры **Configuration File** для задания поля value каждого ключевого параметра:

- String (символьная строка)
- Path (путь)
- Boolean (булев)
- 64-bit double precision floating point numeric (64-битные числа с военной точности с плавающей точкой)
- 32-bit signed integer (32-битные целые числа со знаком)
- 32-bit unsigned integer (32-битные целые числа без знака)

С помощью ВП с палитры **Configuration File** можно записывать и считывать строку или пропускать строку данных. Эти ВП считывают и записывают строку данных побайтно без конвертирования данных в ASCII. В конвертированных или пропущенных строках конфигурационного файла LabVIEW сохраняет любые неотображаемые текстовые символы с помощью их эквивалента в виде шестнадцатеричного escape-кода, как, например, код \0D для символа возврата каретки. Кроме того, в LabVIEW символ обратного слэша

\ задается в конфигурационном файле с помощью двойного обратного слэша \\. Установите на входе **read raw string?** или на входе **write raw string?** ВП с палитры **Configuration File** значение TRUE, если хотите задать данные в виде символьной строки, или FALSE, если хотите задать данные в виде escape-кода.

Когда ВП с палитры **Configuration File** осуществляют запись в конфигурационный файл, они заключают в кавычки строковые или путевые данные, которые содержат символ пробела. Если же строка содержит символ кавычки, LabVIEW сохраняет его как \". Если Вы читаете или записываете конфигурационный файл с помощью текстового редактора, Вы должны иметь в виду, что LabVIEW заменяет кавычки на \".

Путевые данные LabVIEW сохраняет в формате .ini файла UNIX в не зависимости от текущей платформы. ВП интерпретируют абсолютный путь /c/temp/data.dat, сохраненный в конфигурационном файле, следующим образом:

- (Windows) c:\temp\data.dat
- (Mac OS) c:temp:data.dat
- (UNIX) /c/temp/data.dat

ВП интерпретируют относительный путь temp/data.dat, сохраненный в конфигурационном файле, следующим образом:

- (Windows) temp\data.dat
- (Mac OS) :temp:data.dat
- (UNIX) temp/data.dat

## Регистрация данных лицевой панели

---

Используйте регистрацию данных лицевой панели (logging front panel data), чтобы записывать данные для использования их другими ВП и для составления отчетов. Например, Вы можете зарегистрировать данные от индикатора Graph в одном ВП и использовать эти данные в индикаторе Graph другого ВП.

Всякий раз, когда запускается ВП, регистрация данных лицевой панели сохраняет информацию о лицевой панели в отдельном файле

протокола данных (datalog file), который имеет размеченный текстовый формат. Вы можете извлекать из него данные следующими способами:

- Использовать тот же ВП, из которого Вы записали данные, чтобы извлечь данные интерактивно.
- Использовать ВП в качестве ВПП, чтобы извлечь данные программно.
- Использовать для извлечения данных ВП и функции с палитры **File I/O**.

Каждый ВП устанавливает связь (binding) с файлом протокола, которая хранит место размещения файла протокола данных, в котором LabVIEW регистрирует данные с лицевой панели. Связь с файлом протокола – это ассоциация между некоторым ВП и файлом протокола, в который протоколируются данные этого ВП.

Файл протокола данных содержит записи, которые включают временнную метку и данные при каждом запуске ВП. Когда Вы обращаетесь к файлу протокола данных, Вы выбираете нужную запись путем запуска ВП в режиме восстановления (retrieval mode) и используя элементы управления на лицевой панели для просмотра данных. Когда Вы запускаете ВП в режиме восстановления, числовые элементы управления появляются в верхней части лицевой панели, таким образом, можно перемещаться по записям. Пример такого числового элемента управления см. на Figure 14-7.

## **Автоматическая и интерактивная регистрация данных лицевой панели**

Для активизации автоматической регистрации выберите из меню пункт **Operate»Log at Completion** (регистрация при завершении). При первой регистрации данных лицевой панели ВП LabVIEW спросит у Вас имя файла протокола данных. После этого каждый раз при запуске этого ВП LabVIEW будет добавлять новую запись в файл протокола данных. Вы не можете перезаписывать запись после того, как LabVIEW запишет ее в файл протокола данных.

Чтобы занести Ваши данные интерактивно, выберите из меню пункт **Operate»Data Logging»Log**. LabVIEW тут же добавит новую запись в файл протокола данных. Интерактивное сохранение данных позволяет Вам самим выбирать, когда нужно протоколировать

данные. При автоматической регистрации данные протоколируются при каждом запуске ВП.



**Примечание.** Индикатор Waveform Chart протоколирует по одной точке данных каждый раз, когда происходит регистрация данных лицевой панели. Если Вы подсоедините к индикатору типа Chart массив, то файл протокола будет содержать подмножество массива, которое к данному моменту отображено на диаграмме индикатора Chart.

### Интерактивный просмотр протокола данных лицевой панели

После того, как Вы создадите протокол данных, Вы можете просмотреть его интерактивно, выбирая из меню пункт **Operate»Data Logging»Retrieve**. Появится панель восстановления данных, показанная на Figure 14-7.



Figure 14-7. Панель инструментов восстановления данных

Подсвеченное число указывает номер записи данных, которые Вы просматриваете. Числа в квадратных скобках указывают интервал номеров записей, которые зарегистрированы для данного ВП в файле протокола. При каждом запуске ВП в протокол заносится одна запись. При этом индицируется дата и время выбранной записи. Щелкнув по маленьким стрелкам («вверх» и «вниз») можно просматривать следующую или предыдущую записи. Для этой же цели можно использовать клавиши со стрелкой вверх и стрелкой вниз на клавиатуре.

В дополнение к панели инструментов восстановления данных, на лицевой панели будут происходить изменения в соответствии с выбранной на панели инструментов записи протокола данных. Например, когда Вы щелкните по стрелке «вверх» и перейдете к следующей записи, в элементах управления и индикаторах отобразятся данные, соответствующие моменту времени, при котором осуществлялась эта запись в протокол. Для выхода из режима восстановления и возврата в ВП, чей файл протокола просматривался, щелкните кнопку **OK**.

## **Удаление записи**

Находясь в режиме восстановления (retrieval mode), Вы можете удалить конкретные записи. В режиме восстановления пометьте удаляемые записи, щелкнув по кнопке **Trash** в процессе их просмотра. Если Вы щелкните по кнопке **Trash** повторно, пометка на удаление этой записи снимется.

Для удаления всех помеченных таким образом записей, находясь в режиме восстановления, выберите из меню пункт **Operate»Data Logging»Purge Data**.

Если Вы не удалили ранее помеченные вами записи и щелкнули кнопку **OK**, то LabVIEW запросит у Вас подтверждение на удаление этих записей.

## **Разрыв связи с файлом протокола**

Используйте связь с файлом протокола (log-file binding), чтобы для данного ВП использовать соответствующий файл протокола данных при операциях регистрации (logging) или восстановлении (retrieving) лицевой панели. Могут быть два и более файлов протокола, связанных с одним ВП. Это может помочь в тестировании или проверке данных ВП. Например, Вы можете сравнить данные, внесенные в протокол при первом запуске ВП, с данными, внесенными в протокол при втором запуске того же ВП. Чтобы ассоциировать несколько файлов протокола с одним ВП, Вы должны разорвать (очистить) связь с файлом протокола, выбирая из меню пункт **Operate»Data Logging»Clear Log File Binding**. После этого LabVIEW запросит у Вас спецификацию файла протокола либо при следующем запуске этого ВП, либо при активном режиме автоматической регистрации, либо при интерактивном просмотре запротоколированных данных.

## **Изменение связи с файлом протокола**

Чтобы изменить связь с файлом протокола для регистрации или восстановления данных лицевой панели из различных файлов протокола, выберите из меню пункт **Operate»Data Logging»Change Log File Binding**. LabVIEW предложит вам выбрать другой существующий файл протокола или создать новый. Вы можете изменить связь с файлом протокола, когда Вам потребуется восстановить другие данные в этом ВП или добавить данные из этого ВП в другой файл протокола.

## Программное восстановление данных лицевой панели

Вы можете также восстановить запротоколированные данные, используя ВПП, либо с помощью ВП и функций с палитры **File I/O**.

### Восстановление данных лицевой панели с помощью ВПП

Если щелкнуть правой кнопкой по ВПП и выбрать из контекстного меню пункт **Enable Database Access** (активизировать доступ к базе данных), то вокруг изображения ВПП появится желтый прямоугольник, как показано на Figure 14-8.

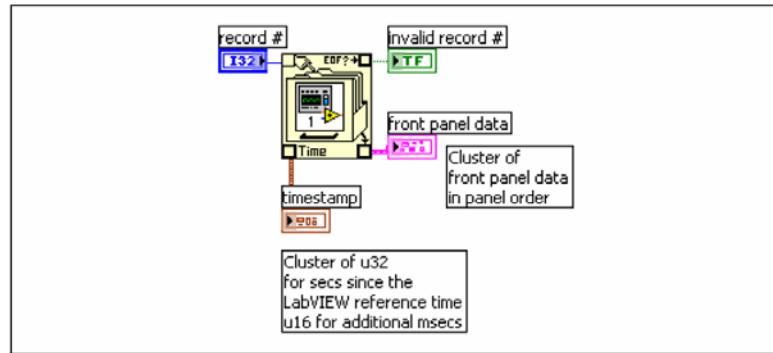


Figure 14-8. Восстановление данных из протокола

Желтый прямоугольник по виду напоминает картотеку и имеет терминалы для получения доступа к данным из файла протокола. Когда Вы активизируете доступ к базе данных для данного ВПП, входы и выходы этого ВПП фактически действуют как выходы, возвращающие ранее запротоколированные данные. Вход **record #** задает номер записи для восстановления, выход **invalid record #** указывает, существует ли запись с заданным номером, вход **timestamp** задает время создания записи, а выход **front panel data** – возвращает кластер объектов лицевой панели. Вы можете использовать данные от объектов лицевой панели, подсоединяя кластер **front panel data** на вход функции **Unbundle**.

Вы можете также восстановить данные на конкретных входах и выходах, подсоединяясь непосредственно к соответствующему терминалу ВПП, как показано на Figure 14-9.

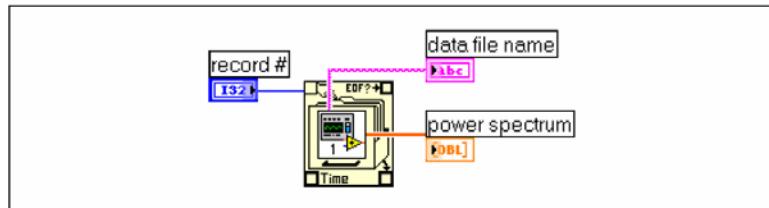


Figure 14-9. Восстановление запротоколированных данных через терминалы ВПП

Если Вы запустите ВП, этот ВПП не будет запускаться. Вместо этого запротоколированные данные с его лицевой панели будут возвращаться в виде кластера на лицевую панель ВП.



**Примечание.** Если ВПП или экспресс ВП отображается в виде расширяемого узла, то активизировать режим доступа к базе данных нельзя.

#### Определение записей

Если ВПП имеет  $n$  протокольных записей, то Вы можете подсоединить к его терминалу **record #** любое число от  $-n$  до  $n-1$ . Вы можете ссылаться на записи в протоколе относительно первой запротоколированной записи, используя неотрицательные номера записей. При этом 0 представляет первую запись, 1 представляет вторую запись и т.д. вплоть до номера  $n-1$ , который предоставляет последнюю запись.

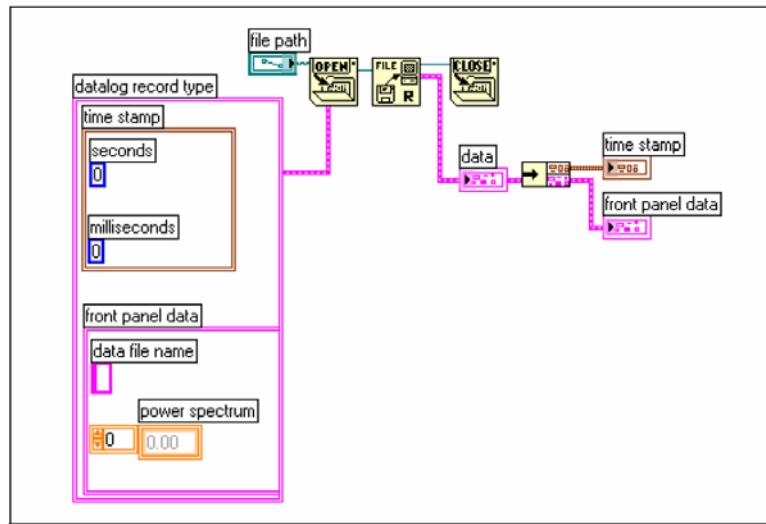
Вы можете ссылаться на записи относительно последней запротоколированной записи, используя отрицательные числа. При этом  $-1$  представляет последнюю запись,  $-2$  представляет предшествующую последней записи и т.д. вплоть до номера  $-n$ , который представляет первую запись. Если Вы подадите на терминал **record #** число за пределами интервала от  $-n$  до  $n-1$ , то на выходе **invalid record #** будет значение TRUE и восстановление данных в этом ВПП не произойдет.

#### Восстановление данных лицевой панели с помощью функций с палитры File I/O

Вы также можете восстановить запротоколированные данные лицевой панели с помощью ВП и функций с палитры **File I/O**. К примеру, это может быть функция **Read File**. Тип данных каждой записи в файле протокола данных лицевой панели содержит два класте-

ра. Один кластер содержит метку времени, а другой кластер содержит данные лицевой панели. Кластер временной метки включает 32-битное целое без знака, которое представляет секунды, и 16-битное целое без знака, которое представляет миллисекунды, истекшего системного времени. Начало системного времени отсчитывается от 00 час.00 мин. 00 сек. 1 января 1904 г. универсального (календарного) времени.

Вы можете получить доступ к файлам протоколов данных лицевой панели с помощью тех же функций с палитры **File I/O**, с помощью которых получали доступ к файлам протоколов данных, созданных программно. Введите **datalog record type** в качестве типа файла, подаваемого на вход функции File Open, как показано на Figure 14-10.



**Figure 14-10.** Восстановление запротоколированных данных с помощью функции File Open

## Директория LabVIEW Data

Используйте текущую директорию для данных LabVIEW Data для хранения файлов с данными, таких как .lvm или .txt-файлы, генерируемых LabVIEW. Чтобы упростить размещение и поиск таких файлов, LabVIEW инсталлирует директорию LabVIEW Data в директорию, принятую операционной системой для хранения ва-



ших данных. По умолчанию экспресс ВП Write LabVIEW Measurement File сохраняет в этой директории файлы .lvm, а экспресс ВП Read LabVIEW Measurement File считывает файлы из этой же директории. Константа Default Data Directory, показанная слева, и свойство Default Data Directory также возвращают значение текущей директории для данных LabVIEW.

Выберите пункт меню **Tools»Options**, а затем категорию **Path**, чтобы изменить значение директории для данных LabVIEW. Текущая директория для данных (Default Data Directory) отличается от общей текущей директории (Default Directory), которая предназначена для новых ВП, пользовательских элементов управления, шаблонов ВП или других документов LabVIEW, которые Вы создаете.

### **Файл измерительных данных**

Файл измерительных данных (LabVIEW measurement file) имеет расширение (.lvm) и включает данные, которые генерируются экспресс ВП Write LabVIEW Measurement File. Файл данных LabVIEW представляет собой текстовый файл, использующий в качестве разделителя символ табуляции, который Вы можете открыть с помощью приложений электронных таблиц или текстовых редакторов. Кроме собственно данных файлы .lvm включают дополнительную информацию о данных, такую как дата и время создания данных.

Более подробно о сохранении и восстановлении данных с помощью экспресс ВП см. в руководстве *Getting Started with LabVIEW*.

Файлы измеренных данных с расширением .lvm могут быть использованы в приложении DIAdem.

## 15. Документирование и печать виртуальных приборов

---

Вы можете использовать LabVIEW для документирования и распечатывания виртуальных приборов.

Раздел *Документирование ВП* в настоящей Главе описывает, как записать информацию о блок-диаграмме и/или лицевой панели в печатную документацию о ВП на любой стадии его разработки.

Раздел *Распечатывание ВП* в настоящей Главе описывает возможности создания распечаток ВП. Некоторые возможности более подходят для печати информации о самих ВП, а другие более подходят для создания отчетов с исходными данными и результатами работы ВП. Некоторые факторы зависят от того, какой метод печати Вы используете, включая то, как Вы хотите распечатывать – вручную или программно, как много возможностей по форматированию отчетов вам потребуются, потребуются ли функциональные возможности для реализации их в стандартном приложении в виде исполняемого файла, и на какой платформе Вы собираетесь запускать ваши ВП.

---

**Более подробно...**

Более подробно относительно документирования и распечатывания ВП см. справочную систему *LabVIEW Help*.

---

### Документирование ВП

---

Вы можете использовать LabVIEW для отслеживания разработки, документирования итогового ВП и создания инструкций для пользователей ВП. Вы можете просматривать документацию, находясь внутри LabVIEW, печатать ее и сохранять в виде HTML, RTF или текстовых файлов.

Чтобы создать эффективную документацию для ВП, добавьте комментарии к истории изменений ВП, включая номера изменений.

## Установка истории изменений ВП

Используйте окно **History** для отображения истории изменений (revision history) ВП, включая номера изменений (revision numbers). Как только Вы произведете изменения в ВП, запишите и отследите их в окне **History**. Чтобы открыть это окно, Выберите из меню пункт **Tools»VI Revision History**. Вы можете также распечатать историю изменений или сохранить ее в виде HTML, RTF или текстового файла. Более подробно о печати истории изменений или сохранения ее в файл см. в разделе *Печать документации* в настоящей Главе.

### Номера изменений

Номера изменений (revision numbers) служат легким способом отслеживания изменений ВП. Номера изменений начинаются с нуля и увеличиваются при каждом сохранении ВП. Чтобы отобразить текущий номер изменений в титульной строке ВП и в титульной строке окна **History**, выберите из меню пункт **Tools»Options**, затем из выпадающего списка категорию **Revision History** и поместите галочку на опции **Show revision number in titlebar**.

Номер, который LabVIEW отображает в окне **History**, является следующим номером изменений, который равен текущему номеру изменений плюс один. Когда Вы добавите комментарий в историю изменений, заголовок этого комментария будет включать следующий номер изменений. Номер изменений не увеличивается, когда Вы сохраняете ВП и при этом изменяете только историю изменений.

Номера изменений не зависят от комментариев в окне **History**. Пропуски номеров изменений между комментариями свидетельствуют о том, что Вы сохраняли ВП без комментариев.

Поскольку история изменений является инструментом исключительно разработки, LabVIEW автоматически удаляет историю изменений, если Вы удалите блок-диаграмму из вашего ВП. Более подробно об удалении блок-диаграммы см. в разделе *Распространение виртуальных приборов* в Главе 7 *Создание ВП и ВПП*. Окно **History** недоступно в версии ВП реального времени (real-time). Страница **General** диалогового окна **VI Properties** отображает номер изменений, даже для ВП без блок-диаграммы. Чтобы очистить

историю изменений и сбросить текущий номер изменений в ноль, щелкните кнопку **Reset** в окне **History**.

## Создание описаний ВП и объектов

Создавайте описания (descriptions) для ВП и их объектов, таких как элементы управления и индикаторы, чтобы описать назначение ВП или объекта и дать пользователю инструкции по их использованию. Вы можете просматривать описания в LabVIEW, распечатывать или сохранять их в HTML, RTF или текстовых файлах.

Чтобы создать, изменить или просмотреть описания ВП, выберите пункт меню **File»VI Properties** и затем из выпадающего меню **Category** выберите пункт **Documentation**. Чтобы создать, изменить или просмотреть описания объекта, щелкните его правой кнопкой и выберите из контекстного меню пункт **Description and Tip**. Подсказки (tip strip) – это краткие пояснения, которые появляются, когда Вы перемещаете курсор над объектом при работающем ВП. Если Вы не введете текст подсказки в окошко **Tip** в диалоговом окне **Description and Tip**, то, естественно, подсказка на лицевой панели появляться не будет. Описание (description) ВП или объекта появляется в окне **Context Help**, когда Вы помещаете курсор над иконкой ВП или над объектом, соответственно.

## Печать документации

Выберите пункт меню **File»Print**, чтобы распечатать документацию ВП или сохранить ее в виде HTML, RTF или текстового файла. Вы можете выбрать для печати документации встроенный формат или создать свой собственный. Создаваемая вами документация может содержать следующие пункты:

- Иконка и соединительная панель.
- Лицевая панель и блок-диаграмма.
- Элементы управления, индикаторы и терминалы типов данных.
- Описания ВП и объектов.
- Иерархия ВП.
- Перечень ВПП.
- История изменений.



**Примечание.** Документация для некоторых типов ВП не может включать все перечисленные выше пункты. Например, полиморфный ВП не имеет лицевой панели или блок-диаграммы, поэтому Вы не можете включить эти пункты в документацию, которую создаете для полиморфного ВП.

### **Сохранение документации в HTML, RTF или текстовые файлы**

Вы можете сохранить документацию ВП в HTML, RTF или текстовые файлы. Вы можете импортировать HTML, RTF файлы в большинство текстовых редакторов, а также можете использовать их для создания компилированных файлов справки. Кроме того, Вы можете использовать сгенерированные LabVIEW HTML файлы для отображения документации ВП на веб-сайтах. Более подробно об использовании HTML и RTF файлов для создания файлов справки см. в *Создание собственных файлов справки* в настоящей Главе.

Когда Вы сохраняете документацию в RTF файл, укажите, хотите ли Вы создать файл, пригодный для создания файла справки или для редактирования в текстовом редакторе. В формате, пригодном для файла справки, LabVIEW сохраняет графику во внешних bitmap файлах. В формате, пригодном для текстового редактора, LabVIEW внедрит графику в документ. В случае HTML файла, LabVIEW сохраняет все графические элементы во внешних файлах в форматах JPEG, PNG или GIF.

#### **Выбор графических форматов для HTML файлов**

Когда Вы сохраняете документацию в HTML файл, Вы можете выбрать формат графических файлов и цветовое разрешение.

Формат JPEG хорошо сжимает графику, но может привести к потере некоторых деталей изображения. Этот формат хорошо работает с фотографиями. Для штриховых рисунков, лицевой панели и блок-диаграммы алгоритм сжатия JPEG может дать размытие изображения и нечеткую передачу цветов. JPEG формат всегда использует 24-битную графику. Если Вы выберите меньшее цветовое разрешение, например черно-белое, то изображение сохранится с указанным вами разрешением, но результат все равно будет представлен с 24-битным разрешением цвета.

Формат PNG также хорошо сжимает изображение, правда иногда уступает в этом формату JPEG. Однако PNG сжатие не приводит к

потере деталей изображения. Кроме того, этот формат поддерживает 1-битную, 4-битную, 8-битную и 24-битную графику. Для малого разрешения изображение сжимается намного лучше, чем в случае формата JPEG. Формат PNG заменяет старый формат Graphics Interchange Format (GIF). Хотя формат PNG обладает достоинствами по сравнению с форматами JPEG и PNG, он не всегда поддерживается интернет браузерами.

Формат GIF также хорошо сжимает изображения, и он поддерживается большинством интернет браузеров. В связи с лицензионными ограничениями LabVIEW не позволяет сохранять изображения в виде сжатых GIF файлов, но в будущем это возможно. Используйте конвертер графических форматов для конвертирования несжатых GIF файлов, которые сохраняет LabVIEW, в сжатые GIF файлы. Для повышения качества сжатых GIF файлов при сохранении документации выберите формат PNG и затем с помощью конвертера графических форматов преобразуйте PNG файлы в GIF файлы. Начиная с PNG формата, Вы получите лучшее качество графики, поскольку PNG формат является точным воспроизведением исходного изображения. Модифицируйте HTML файл, который генерирует LabVIEW, так, чтобы он ссылался на GIF файлы с расширением .gif.

#### **Соглашения для имен графических файлов**

Когда Вы создаете HTML или RTF документацию с внешней графикой, LabVIEW сохраняет терминалы типов данных для элементов управления и индикаторов в графических файлах с соответствующими именами. Если ВП содержит несколько терминалов одного и того же типа, LabVIEW создает только один графический файл для этого типа. Например, если ВП содержит три входных терминала, имеющих тип 32-битного целого со знаком, то LabVIEW создаст один файл с именем 32.x, где x есть расширение, соответствующее графическому формату.

#### **Создание собственных файлов справки**

Вы можете использовать HTML или RTF файлы, которые генерирует LabVIEW, чтобы создать свои собственные компилированные файлы справки. (**Windows**). Вы можете компилировать отдельные HTML файлы, которые генерирует LabVIEW, в HTML файл справки.

Вы можете компилировать RTF файлы, которые генерирует LabVIEW, в (**Windows**) WinHelp, (**Mac OS**) QuickHelp или (**UNIX**) HyperHelp файлы.

Чтобы создать ссылки из ВП на HTML файлы или компилировать файлы справки, выберите пункт меню **File»VI Properties** и затем из спадающего меню **Category** выберите пункт **Documentation**.

## Печать виртуальных приборов

---

Вы можете использовать следующие основные методы печати ВП:

- Выберите пункт меню **File»Print Window**, чтобы распечатать содержимое активного окна.
- Выберите пункт меню **File»Print**, чтобы распечатать более подробную информацию о ВП, включая информацию о лицевой панели, блок-диаграмме, ВПП, элементах управления, истории изменений и т.п. Более подробно об использовании этого метода для печати ВП см. в разделе *Печать документации* настоящей Главы.
- Используйте сервер ВП для программной печати любого окна ВП или документации ВП в произвольный момент времени. Более подробно об использовании данного метода для печати ВП см. в Главе 17 *Программное управление ВП*.

### Печать активного окна

Выберите из меню пункт **File»Print Window**, чтобы распечатать содержимое активной лицевой панели или блок-диаграммы с минимальным количеством подсказок и сообщений. LabVIEW распечатает рабочее пространство активного окна, включая все объекты, не вошедшие в видимую часть этого окна. Не будут выведены на печать титульная строка, строка меню, панель инструментов и полосы прокрутки.

Для настройки того, как LabVIEW будет выводить на печать ВП при выборе пункта меню **File»Print Window** или при программной печати, выберите пункт меню **File»VI Properties** и затем из спадающего меню **Category** – пункт **Print Options**. Более подробно о программной печати см. раздел *Программная печать ВП* в настоящей Главе.

## Программная печать ВП

Используйте один из следующих методов программной печати ВП вместо интерактивной печати через диалоговые окна, которые появляются, когда Вы выбираете пункты меню **File»Print Window** или **File»Print**:

- Установите, чтобы ВП автоматически печатал свою лицевую панель при каждом завершении исполнения.
- Создайте ВПП, который распечатывает ВП.
- Используйте ВП с палитры **Report Generation** для печати отчетов и сохранения отчетов в виде HTML файлов, которые содержат документацию ВП или данные, которые ВП возвращает.
- Используйте сервер ВП для программной распечатки окна ВП или для распечатки документации ВП или сохранения ее в HTML, RTF или текстовые файлы в любой момент времени. Более подробно об использовании данного метода для печати ВП см. в Главе 17 *Программное управление ВП*.



**Примечание.** Если Вы печатаете документацию ВП из скомпилированного приложения, Вы сможете распечатать только лицевые панели.

### Печать лицевой панели ВП после его исполнения

Выберите пункт меню **Operate»Print at Completion** (распечатать после завершения) для печати лицевой панели ВП после того, как он завершит свое исполнение. Вы можете также выбрать **File»VI Properties** и затем **Print Options** из спадающего меню **Category**, чтобы установить птичку на опции **Automatically Print Panel Every Time VI Complete Execution** (автоматическая печать панели после каждого завершения выполнения ВП).

Выбор этой опции подобен выбору команды **File»Print Window**, при условии, что лицевая панель является активным окном.

Если Вы используете ВП в качестве ВПП, то распечатка будет производиться, когда этот ВПП завершит свое исполнение, но до того, как управление будет передано вызывающему его ВП.

## Использование ВПП для печати данных из ВП верхнего уровня

В некоторых случаях Вы можете не захотеть, чтобы ВП распечатывался при каждом завершении своей работы. Может потребоваться, чтобы печать производилась, только если пользователь щелкнет кнопку или выполнится некоторое условие, скажем ошибка тестирования. Вы также можете захотеть большего контроля над форматом печати, или печатать только подмножество элементов управления. В таких случаях можно использовать ВПП, который настроен на печать при завершении.

Создайте ВПП и сконфигурируйте его лицевую панель в том виде, который Вы хотите вывести на печать. Вместо того чтобы выбирать пункт меню **Operate»Print at Completion** в ВП верхнего уровня, выберите этот пункт меню для ВПП. Когда Вы захотите сделать распечатку, вызовите этот ВПП и подсоедините к нему данные, которые хотите вывести на печать.

## Генерация и печать отчетов

Используйте ВП с палитры **Report Generation** для печати отчетов или сохранения HTML отчетов, которые содержат документацию ВП или данные, которые возвращает ВП. Используйте ВП Easy Print или ВП Documentation для генерации базового отчета, который содержит документацию ВП. Используйте ВП Easy Text Report для генерации базового отчета, который содержит данные, которые возвращает ВП. Для генерации более сложных отчетов используйте другие ВП с палитры **Report Generation**.



**Примечание.** Вы можете генерировать отчеты только в пакетах LabVIEW Full Development Systems и Professional Development Systems.

Используйте ВП с палитры **Report Generation** для решения следующих задач:

- Добавление в отчет текста, графики, таблиц или документации ВП.
- Установка для текста шрифта, размера, стиля и цвета.
- Установка ориентации отчета – горизонтально или вертикально.
- Установка верхних и нижних колонтитулов.

- Установка полей и табуляции.

## Дополнительные способы печати

Если стандартные методы печати в LabVIEW вас не устраивают, Вы можете использовать следующие дополнительные способы:

- Построчная печать данных. Если у вас построчный принтер, подсоединеный к последовательному порту, то используйте ВП с подпалитры **Serial** для отправки на него текста. Реализация этого способа требует некоторых знаний командного языка принтера.
- Экспорт данных в другие приложения, такие как Microsoft Excel, сохранение данных в файл и печать через другие приложения.
- **(Windows, Mac OS, UNIX)** Использование ВП System Exec.
- **(Mac OS)** Использование ВП AESend Print Document.
- **(Windows)** Использование ActiveX для печати данных через другие приложения. Более подробно об ActiveX см. в Главе 19 *Связь в среде Windows*.

## 16. Конфигурирование ВП

---

Вы можете конфигурировать ВП и ВПП, чтобы они работали в соответствии с требованиями вашего приложения. Например, если Вы планируете использовать ВП в качестве ВПП, который требует взаимодействия с пользователем, настройте его так, чтобы его лицевая панель появлялась всякий раз, когда происходит его вызов.

Вы можете конфигурировать ВП различными способами, либо находясь внутри него, либо программно с помощью сервера ВП. Более подробно об использовании сервера ВП для настройки поведения ВП см. в Главе 17 *Программное управление ВП*.

---

**Более подробно...**

Более подробно относительно конфигурирования ВП см. справочную систему *LabVIEW Help*.

---

### Конфигурирование внешнего вида и поведения ВП

---

Выберите пункт меню **File»VI Properties**, чтобы сконфигурировать внешний вид и поведение ВП. Используйте спадающее меню **Category** в верхней части диалогового окна, чтобы выбрать несколько различных категорий настроек, включая следующие:

- **General** – Отображает текущий путь к директории, в которой сохранен ВП, его номер изменений, историю изменений и любые изменения, сделанные после последнего сохранения ВП. Вы также можете использовать эту страницу для редактирования иконки или размера выравнивающей сетки для этого ВП.
- **Documentation** – Используйте эту страницу, чтобы добавить описание ВП и ссылку на раздел файла справки. Более подробно об опциях документирования см. в разделе *Документирование ВП* в Главе 15 *Документирование и печать виртуальных приборов*.
- **Security** – Используйте эту страницу для блокировки ВП или защиты его с помощью пароля.
- **Window Appearance** – Используйте эту страницу для настройки внешнего вида окна.

- **Window Size** – Используйте эту страницу, чтобы установить размер окна.
- **Execution** – Используйте эту страницу для настройки процесса выполнения ВП. Например, Вы можете настроить ВП так, чтобы он запускался сразу после того, как он будет открыт, или чтобы приостанавливался, когда он будет вызван в качестве ВПП. Вы также можете настроить так, чтобы ВП запускался с различными приоритетами. Например, если важно, чтобы ВП запускался без ожидания завершения других операций, настройте его так, чтобы он запускался с критичным по времени (наивысшим) приоритетом. Более подробно о создании многопоточных ВП см. в руководстве (Application Note) *Using LabVIEW to Create Multi-threaded VIs for Maximum Performance and Reliability*.
- **Editor Options** – Используйте эту страницу для установки разметки выравнивающей сетки для текущего ВП и для изменения стиля элементов управления или индикаторов, которые LabVIEW создает, когда Вы щелкаете правой кнопкой терминал и выбираете из контекстного меню пункт **Create»Control** или **Create»Indicator**. Более подробно о выравнивающей сетке см. в разделе *Выравнивание и распределение объектов* в Главе 4 *Построение лицевой панели*.

## Изменение меню

---

Вы можете создать специальное меню для каждого ВП, который Вы строите, и можете настроить ВП так, что строка меню будет видна, либо будет скрыта. Чтобы показать или скрыть строку меню выберите пункт меню **File»VI Properties**, затем из спадающего меню **Category** выберите пункт **Windows Appearance**, затем щелкните кнопку **Customize** и поставьте или удалите птичку на опции **Show Menu Bar**.

Конфигурирование меню включает создание меню и предоставление кода блок-диаграммы, который выполняется, когда пользователь выберет различные пункты меню.



**Примечание.** Специальное меню появляется только во время запуска ВП.

## Создание меню

Вы можете создать свое пользовательское меню (custom menu) или модифицировать первоначальное меню (default menu) LabVIEW статически, когда редактируете ВП, или программно, когда запускаете ВП. Когда Вы выбираете пункт меню **Edit»Run-Time Menu** и создаете меню в диалоговом окне **Menu Editor**, LabVIEW создает файл меню времени исполнения (run time menu) с расширением **.rтm**, таким образом, вы можете иметь специальную строку меню в вашем ВП вместо исходной строки меню. После того, как Вы создадите и сохраните **.rтm** файл, Вы должны установить соответствующий относительный путь между вашим ВП и **.rтm** файлом.

Используйте диалоговое окно **Menu Editor**, чтобы связать **.rтm** файл специального меню с ВП. Когда ВП запустится, он загрузит меню из **.rтm** файла. Вы также можете использовать диалоговое окно **Menu Editor**, чтобы построить специальное меню либо с пунктами приложения (application items), которые являются пунктами исходного меню LabVIEW, либо с пунктами пользователя (user items), которые Вы добавляете сами. Реализацию пунктов приложения LabVIEW задает автоматически, а реализацию пользовательских пунктов меню Вы должны задать сами с помощью блок-диаграммы. Более подробно об обработке пользовательских пунктов меню см. в разделе *Обработка пунктов меню* в настоящей Главе.

## Обработка пунктов меню

Когда Вы создаете пользовательское меню, Вы назначаете каждому пункту меню уникальный идентификатор в виде символьной строки, нечувствительной к регистру символов, который называется тегом (tag). Когда пользователь выбирает пункт меню, Вы получаете программный доступ к тегу этого пункта с помощью функции **Get Menu Selection**. LabVIEW предоставляет для каждого пункта меню обработчик на блок-диаграмме, основываясь на значении тега каждого пункта меню. Обработчик представляет собой комбинацию структур **While Loop** и **Case**, которые позволяют вам определить, какой из имеющихся в наличии пунктов меню выбран, и выполнить соответствующий программный код.

После того, как Вы построите пользовательское меню, постройте на блок-диаграмме структуру **Case**, которая выполнит, или обрабатывает, каждый пункт пользовательского меню. Этот процесс называется

ется обработкой пунктов меню (menu selection handling). Все пункты меню приложения (application menu) LabVIEW обрабатывает неявно с помощью встроенных обработчиков.

На Figure 16-1 функция Get Menu Selection читает пункт меню, который выбирает пользователь, и передает этот пункт меню в структуру Case, где происходит реализация этого пункта меню.

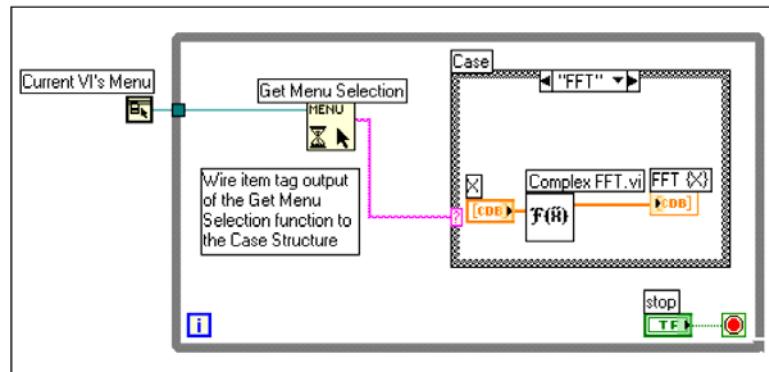


Figure 16-1. Блок-диаграмма обработки пунктов меню

Если вам известно, что некоторые пункты меню требуют большое время на их обработку, подсоедините булев элемент управления к входу **block menu** функции Get Menu Selection и установите этот элемент управления в состояние TRUE. При этом строка меню станет неактивной и пользователь не сможет выбрать какой-нибудь другой пункт меню, пока не будет обработан уже выбранный пункт. Чтобы активизировать строку меню после того, как LabVIEW обработает выбранный пункт меню, подсоедините значение TRUE к функции Enable Menu Tracking.

Вы можете также использовать для обработки меню структуру Event. Более подробно о структуре Event см. в Главе 9 *Событийно управляемое программирование*.

## **17. Программное управление ВП**

---

Вы можете получить доступ к серверу виртуальных приборов (серверу ВП) через блок-диаграмму, с помощью технологии ActiveX и посредством протокола TCP, который обеспечивает обмен между виртуальными приборами и другими копиями (instance) LabVIEW. Благодаря этому Вы можете осуществлять программное управление ВП и LabVIEW. Вы можете выполнить операции сервера ВП на локальном компьютере или дистанционно через сеть.

---

**Более подробно...**

Более подробно относительно программного управления виртуальными приборами см. справочную систему *LabVIEW Help*.

---

### **Возможности сервера виртуальных приборов**

---

Используйте сервер виртуальных приборов (сервер ВП) для выполнения следующих программных операций:

- Удаленный вызов ВП.
- Настройка копии LabVIEW так, чтобы она стала сервером, который экспортирует виртуальные приборы, которые Вы можете вызывать из других копий LabVIEW через интернет. Например, если у вас есть приложение для сбора данных, которое вводит и накапливает данные в удаленном месте, Вы можете по мере необходимости получать эти данные на вашем локальном компьютере. Изменяя предустановки LabVIEW, Вы можете сделать некоторые ВП доступными через интернет так, что получение новых данных будет таким же легким, как вызов ВПП. Сервер ВП сам отрабатывает детали сетевого обмена. Кроме того, сервер ВП работает через платформы, таким образом, клиент и сервер могут работать на различных платформах.
- Редактирование свойств ВП и LabVIEW. Например, вы можете динамически определить размещение окна ВП или прокрутить лицевую панель таким образом, чтобы какая-то его часть стала видимой. Вы также можете программно сохранять на диск любые изменения.

- Обновление свойств сразу нескольких ВП вместо того, чтобы вручную изменять свойства каждого ВП с помощью диалогового окна **File»VI Properties**.
- Получение информации о каждой копии LabVIEW, такой как номер версии и редакция. Вы также можете получать информацию об окружении, такую как вид платформы, на которой работает LabVIEW.
- Динамическая загрузка виртуальных приборов в память, когда другой ВП требует их вызова вместо того, чтобы загружать все ВПП при открытии ВП.
- Создание развивающейся (plug-in) архитектуры приложения, позволяющей развивать функциональность приложения уже после распространения его пользователям. Например, Вы можете иметь целый набор ВП для фильтрации данных, все из которых имеют одни и те же параметры. В результате проектирования этого приложения с возможностью динамической загрузки этих ВП из директории plug-in, Вы можете поставлять это приложение с установленным комплектом таких ВП, а для некоторых пользователей обеспечить возможность расширения функциональности путем помещения новых ВП для фильтрации в директорию plug-in.

## Построение приложений сервера ВП

---

Программная модель приложений сервера ВП основана на ссылочных номерах (refnum). Ссылочные номера используются также в файловом вводе/выводе, в сетевых соединениях и в других объектах LabVIEW. Более подробно о ссылочных номерах см. в разделе *Ссылки на объекты или приложения* в Главе 4 *Построение лицевой панели*.

Обычно Вы открываете ссылочный номер на копию LabVIEW или на ВП. Затем Вы используете ссылочный номер в качестве параметра для других ВП. Эти ВП получают (читают) или устанавливают (записывают) свойства, выполняют методы или динамически загружают указанный ВП. В завершение Вы закрываете ссылочный номер, в результате чего указанный ВП выгружается из памяти.

Используйте следующие функции и узлы с палитры **Application Control** для построения приложений сервера ВП:

- **Open Application Reference** (открыть ссылку на приложение) – Открывает ссылку на локальное или удаленное приложение, к которому Вы получаете доступ через сервер, или для доступа к удаленной копии LabVIEW.
- **Open VI Reference** (открыть ссылку на ВП) – Открывает ссылку на ВП на локальном или на удаленном компьютере или на динамически загруженный с диска ВП.
- **Property Node (узел свойств)** – Получает и устанавливает свойства ВП, объекта или приложения. Более подробно о свойствах см. в разделе *Узлы свойств* в настоящей Главе.
- **Invoke Node (узел вызовов)** – Вызывает методы из ВП, объекта или приложения. Более подробно о методах см. в разделе *Узлы вызовов* в настоящей Главе.
- **Call By Reference Node (вызов через ссылочный узел)** – Вызывает динамически загруженный ВП.
- **Close Reference** (закрыть ссылку) – закрывает все открытые ссылки на ВП, объект или приложение, к которым был доступ через сервер ВП.

## Ссылки на приложение и на ВП

Вы имеете доступ к функциональности сервера ВП через ссылки на два основных класса объектов – на объект приложение и на объект ВП. После того, как Вы создадите ссылку на один из этих объектов, Вы можете подать эту ссылку на ВП или функцию, которые выполняют операцию над этим объектом.

Ссылочный номер приложения (application refnum) ссылается на локальную или удаленную копию LabVIEW. Вы можете использовать свойства и методы приложения, чтобы изменить предустановки и вернуть информацию о системе. Ссылочный номер ВП (VI refnum) ссылается на ВП в некоторой копии LabVIEW.

С помощью ссылочного номера на копию LabVIEW (refnum to an instance of LabVIEW) Вы можете получать информацию об окружении LabVIEW, к примеру, платформу, на которой запущена LabVIEW, номер версии или список всех ВП, загруженных в данный момент в память. Вы можете также установить такие данные, как текущее имя пользователя или список ВП, экспортруемых в другие копии LabVIEW.

Когда Вы создаете ссылочный номер на ВП, LabVIEW загружает ВП в память. Этот ВП остается в памяти до тех пор, пока Вы не закроете этот ссылочный номер. Если имеются несколько ссылочных номеров на один и тот же ВП, открытых одновременно, то этот ВП останется в памяти до тех пор, пока Вы не закроете все ссылочные номера на этот ВП. С помощью ссылочного номера на ВП Вы можете обновлять все свойства ВП, доступные через диалоговое окно **File»VI Properties**, такие, например, как положение окна лицевой панели. Вы также можете программно распечатать документацию ВП, сохранить ВП в другом месте и произвести экспорт и импорт его символьных строк для перевода на другой язык.

## Манипуляция установками приложения и ВП

---

Используйте сервер ВП, чтобы получать и переключать установки приложения и ВП посредством узла свойств (Property Node) и узла вызовов (Invoke Node). Многие установки приложений и ВП Вы можете получать и переключать только через узлы свойств и вызовов.

Примеры использования свойств и методов классов приложения (Application Class) и ВП (VI Class) можно найти в директории `examples\viserver`.

### Узлы свойств

Используйте узел свойств (Property Node) для получения и изменения свойств приложения или ВП. Чтобы выбрать нужное свойство для данного узла, воспользуйтесь инструментом Operating и щелкните им терминал свойств, либо щелкните правой кнопкой белую область узла и выберите из контекстного меню пункт **Properties**.

Вы можете читать или записывать многие свойства, используя один узел, однако некоторые свойства не являются записываемыми, их можно только читать. Используйте инструмент Positioning, чтобы растянуть узел свойств и добавить новые терминалы. Маленькая стрелка справа от названия свойства указывает, что это свойство читается (выходной терминал). Аналогичная стрелка слева от названия свойства указывает, что это свойство записывается (входной терминал). Щелкнув правой кнопкой терминал и выбирая из контекстного меню **Change to Read** или **Change to Write**, можно изменить состояние этого свойства.

Узел свойств выполняется сверху вниз. Узел свойств не выполняется, если перед его исполнением возникает ошибка, таким образом, всегда осуществляется проверка на возможные ошибки. Если ошибка возникает в некотором свойстве, LabVIEW игнорирует оставшиеся свойства и возвращает ошибку. При этом кластер **error out** будет содержать информацию о том свойстве, которое вызвало ошибку.

Если узел свойств открывает и возвращает ссылку на приложение или ВП, то используйте функцию Close Reference, чтобы закрыть эту ссылку на приложение или на ВП. LabVIEW закрывает ссылки на элементы управления, когда они становятся ненужными. Вам нет необходимости явно закрывать ссылки на элементы управления.

### **Неявно связанные узлы свойств**

Когда Вы создаете узел свойств объекта лицевой панели, щелкнув правой кнопкой по объекту и выбирая из контекстного меню пункт **Create»Property Node**, LabVIEW создает на блок-диаграмме узел свойств, неявно связанный с этим объектом лицевой панели. Поскольку такие узлы свойств неявно связаны с объектом, для которого они были созданы, они не имеют входного терминала для ссылочного номера, и нет необходимости соединять узел свойств с терминалом ссылки на объект лицевой панели или на элемент управления. Более подробно о ссылках на элементы управления см. в разделе *Управление объектами лицевой панели* в настоящей Главе.

## **Узлы вызовов**

Используйте узлы вызовов (Invoke Nodes) для выполнения действий, или методов, приложения или ВП. В отличие от узла свойств, каждый отдельный узел вызовов выполняет только один метод приложения или ВП. Выберите метод с помощью инструмента Operating, щелкнув им терминал метода или щелкнув правой кнопкой по белой области узла и выбирая из контекстного меню пункт **Methods**.

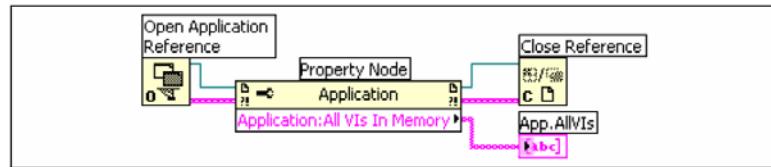
Имя метода всегда является первым в списке параметров узла вызовов. Если метод возвращает значение, то терминал метода отображает возвращаемое значение (в правой части терминала имеется

стрелка). В противном случае терминал метода не имеет возвращаемого значения.

Узел вызовов отображает список параметров сверху вниз, причем самым верхним является имя метода, а имена необязательных параметров помещаются внизу и пишутся серым шрифтом.

## Манипуляция свойствами и методами класса Приложение

Вы можете получить или установить значения свойств локальной или удаленной копии LabVIEW, выполнить методы LabVIEW или и то и другое. На Figure 17-1 показано, как отобразить на лицевой панели список всех ВП, находящихся в памяти локального компьютера, в виде массива символьных строк.



**Figure 17-1.** Отображение списка всех ВП, находящихся в памяти локального компьютера

Если Вы не подсоедините ссылочный номер к входу **reference**, узел свойств или узел вызовов будет использовать ссылку на текущую копию LabVIEW. Если Вы хотите манипулировать свойствами или методами другой копии LabVIEW, нужно подсоединить ссылку на приложение к входу **reference**.

Чтобы найти ВП в памяти удаленного компьютера, подсоедините строковый элемент управления к входу **machine name** функции Open Application Reference, как это показано на Figure 17-2, и введите IP адрес или имя домена. Вы также должны выбрать свойство **Exported VIs in Memory**, поскольку свойство **All VIs in Memory**, используемое на Figure 17-1, применимо только к локальным копиям LabVIEW.

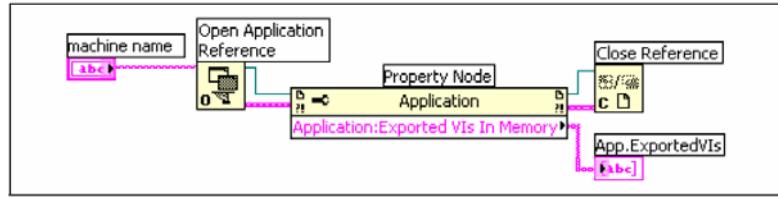


Figure 17-2. Отображение списка всех ВП, находящихся в памяти удаленного компьютера

### Манипуляция свойствами и методами класса Виртуальный прибор

Вы можете получать или устанавливать значения свойств ВП, выполнять методы ВП, или и то и другое. На Figure 17-3 LabVIEW повторно инициализирует значения объектов лицевой панели ВП к их значениям по умолчанию с помощью узла свойств.

Если Вы не подсоедините ссылочный номер к входу **reference**, то узел свойств (Property Node) или узел вызовов (Invoke Node) будет использовать ссылку на ВП, содержащему этот узел свойств или узел вызовов. Если Вы хотите манипулировать свойствами или методами другого ВП, то нужно подсоединить ссылочный номер этого ВП к входу **reference**.

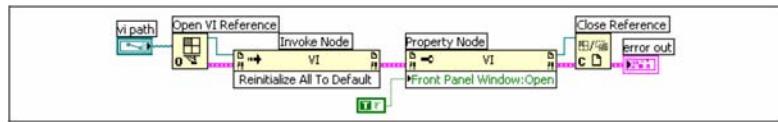


Figure 17-3. Использование свойств класса ВП и узла вызовов

Узел свойств работает подобно узлу вызовов. После того, как Вы подсоедините ссылочный номер ВП к узлу свойств, Вы можете получить доступ ко всем свойствам класса ВП.

### Манипуляция свойствами и методами класса Приложение и класса ВП

В некоторых ВП Вам может потребоваться доступ к свойствам и методам и класса Приложение и класса ВП. Вы должны открывать и закрывать ссылочные номера на класс Приложение и на класс ВП раздельно, как показано на Figure 17-4.

Figure 17-4 иллюстрирует, как определить список ВП в памяти локального компьютера и отобразить на лицевой панели путь к каждому ВП. Чтобы найти все ВП, находящиеся в памяти, Вы должны получить доступ к свойству класса Приложение. Чтобы определить путь к каждому из этих ВП, Вы должны получить доступ к свойству класса ВП. Количество ВП, находящихся в памяти задает число итераций цикла For Loop. Поместите функции Open VI Reference и Close Reference внутрь цикла For Loop, поскольку Вам нужен ссылочный номер на каждый ВП, находящийся в памяти. Не закрывайте ссылочный номер на приложение, пока цикл For Loop не завершил получение путей ко всем ВП.

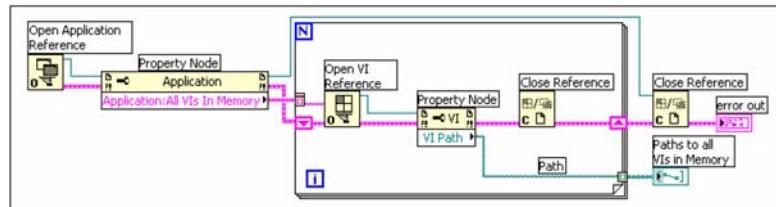


Figure 17-4. Использование свойств и методов класса Приложение и класса ВП

## Динамическая загрузка и вызов ВП

Вы можете динамически загружать ВП вместо того, чтобы использовать вызовы статически связанных ВПП. ВПП являются статически связанными, когда Вы помещаете их непосредственно на блок-диаграмму вызывающего их ВП. Они загружаются одновременно с загрузкой вызывающего ВП.

В отличие от статически связанных ВПП, динамически загружаемые ВПП не загружаются до тех пор, пока вызывающий ВП не осуществит вызов ВПП. Если Вы имеете большой вызывающий ВП, то Вы можете сократить время загрузки и память, используя динамически загружаемые ВПП, поскольку ВПП не будут загружаться, пока они действительно не потребуются, и после завершения операции ненужные ВПП можно удалить из памяти.

## Узлы вызова по ссылке и строго типизированные ссылочные номера ВП

Используйте узел вызова по ссылке (Call By Reference Node) для динамического вызова ВП.

Узел вызова по ссылке требует строго типизированный ссылочный номер ВП (strictly typed VI refnum). Строго типизированный ссылочный номер ВП идентифицирует соединительную панель ВП, который Вы вызываете. Он не создает постоянной ассоциации с ВП и не содержит другой информации о ВП, такой как его имя или расположение. Вы можете подсоединять входы и выходы узла вызова по ссылке точно так же, как делаете это для любого другого ВП.

На Figure 17-5 показано, как используется узел вызова по ссылке для динамического вызова ВП Frequency Response. Узел вызова по ссылке требует использования функций Open VI Reference и Close Reference, подобно тому, как эти функции используются для узла свойств и для узла вызовов.

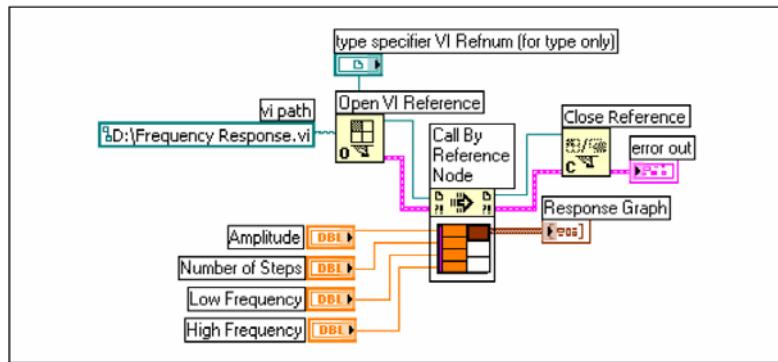


Figure 17-5. Использование узла вызова по ссылке

ВП, который Вы специфицируете с помощью строго типизированного ссылочного номера, передает только информацию о соединительной панели. Это значит, что постоянная ассоциация (связь) между этой ссылкой и ВП не создается. Особенно старайтесь избегать неверного подсоединения к соединительной панели, получаемой с помощью ссылочного номера на выбранный ВП. Выбор конкретного ВП осуществляется с помощью входа **vi path** функции Open VI Reference.

## Редактирование и запуск ВП на удаленных компьютерах

Важным аспектом ссылочных номеров на Приложение и на ВП является их сетевая прозрачность. Это означает, что Вы можете от-

крыть ссылочные номера на объекты удаленных компьютеров точно так же, как Вы открывали бы ссылочные номера на такие же объекты вашего компьютера.

После того, как Вы откроете ссылочный номер на удаленный объект, Вы можете трактовать его как локальный объект, но с некоторыми ограничениями. Для операции на удаленном объекте сервер ВП посыпает информацию об этой операции через сеть и посыпает результат обратно. Приложение смотрится почти одинаково, независимо от того, работает оно удаленно или локально.

## Управление объектами лицевой панели

---

Ссылки на элементы управления соответствуют ссылкам на объекты пользовательского интерфейса в текстовых языках программирования. Однако, ссылки на элементы управления не соответствуют указателям (pointers) в текстовых языках программирования.

Используйте элементы управления Control Refnum (ссылочный номер элемента управления) с палитры **Refnum** или с палитры **Classic Refnum**, чтобы передать ссылку на объект лицевой панели в другие ВП. Вы можете также щелкнуть правой кнопкой объект лицевой панели и выбрать из контекстного меню пункт **Create>Reference**. После того, как ссылка на элемент управления будет передана в ВПП, используйте узлы свойств (Property Nodes) и узлы вызовов (Invoke Nodes) для чтения и записи свойств и вызова методов ссылаемого объекта лицевой панели.

Более подробно об использовании событий для программного управления поведением блок-диаграммы посредством объектов лицевой панели см. в разделе *Структуры выбора и последовательности* в Главе 8 *Циклы и структуры*.

### Строго и слабо типизированные ссылочные номера элементов управления

Строго типизированные ссылочные номера элементов управления (strictly typed control refnums) принимают только ссылочные номера элементов управления точно такого же типа данных. Например, если типом строго типизированного ссылочного номера на элемент управления является 32-битный целочисленный движок (slider), то к терминалу ссылочного номера элемента управления Вы можете

подсоединить только 32-битный целочисленный движок. Вы не сможете подключить к этому терминалу 8-битный целочисленный движок, скалярный движок с удвоенной точностью или кластер 32-битных целочисленных движков.

Ссылки на элемент управления, которые Вы создаете из элемента управления, по умолчанию являются строго типизированными. На лицевой панели красная звездочка в нижнем левом углу ссылки на элемент управления указывает, что данная ссылка является строго типизированной. На блок-диаграмме на узле свойств или на узле вызовов, подсоединенному к терминалу ссылки на элемент управления, появляется надпись (*strict*), что указывает на то, что данная ссылка на элемент управления является строго типизированной.



**Примечание.** Поскольку механические действия с блокировкой (latch mechanical actions) несовместимы со строго типизированными ссылками на элемент управления, то булевы элементы управления, обладающие механическим действием с блокировкой, вырабатывают слабо типизированные ссылки на элемент управления.

Слабо типизированные ссылки на элементы управления (weakly typed control refnums) являются более гибкими в возможности принимать тип данных. Например, если типом слабо типизированной ссылки на элемент управления является *slide* (движок), то Вы можете подсоединить к терминалу ссылки на элемент управления 32-битный целочисленный движок, движок с однократной точностью или кластер 32-битных целочисленных движков. Если типом слабо типизированной ссылки на элемент управления является *control* (элемент управления), то Вы можете подсоединить к терминалу ссылки на элемент управления ссылку на элемент управления любого типа.



**Примечание.** Когда Вы подсоединяете узел свойств к терминалу слабо типизированной ссылки на элемент управления, свойство **Value** вырабатывает варианты данные, которые могут требовать преобразования типа перед их использованием. Свойство **History Data** для диаграмм (Chart) доступно, только если ссылка на этот индикатор Chart является строго типизированной. Более подробно о варианты данных см. в раз-

разделе *Обработка вариантовых данных* в Главе 5 *Построение блок-диаграммы*.

## 18. Сетевые коммуникации в LabVIEW

---

Виртуальные приборы могут связываться через сеть с другими процессами, включая те, которые работают в других приложениях или на удаленных компьютерах. Используйте сетевые коммуникационные возможности LabVIEW для решения следующих задач:

- Совместное использование данных различными ВП, работающими в сети с использованием технологии DataSocket.
- Публикация изображения лицевой панели и документации ВП в интернете.
- Отправка данных из ВП по электронной почте (Email).
- Построение ВП, которые сообщаются с другими приложениями и другими ВП через такие низкоуровневые протоколы, как TCP, UDP, Apple events и PPC Toolbox.

---

**Более подробно...**

Более подробно относительно коммуникационных возможностей LabVIEW см. справочную систему LabVIEW Help

---

### Выбор между файловым вводом/выводом, сервером виртуального прибора, технологией ActiveX и работой в сети

---

Работа в сети не всегда может быть наилучшим решением для вашего приложения. Если Вы хотите создать файл, который содержит данные, доступные для чтения другими ВП или приложениями, воспользуйтесь ВП и функциями с палитры **File I/O** (файловый ввод/вывод). Более подробно об использовании ВП и функций с палитры **File I/O** см. в Главе 14 *Файловый ввод/вывод*.

Если Вы хотите управлять другими ВП, используйте сервер виртуальных приборов. Более подробно об управлении ВП и другими приложениями LabVIEW на локальном и удаленном компьютерах см. в Главе 17 *Программное управление ВП*.

**(Windows)** Если Вы хотите получить доступ к многим возможностям приложений Microsoft таким, как внедрение графиков осцил-

лограмм в электронную таблицу Excel, воспользуйтесь ВП и функциями с палитры **Active X**. Более подробно о взаимодействии ActiveX совместимых приложений и LabVIEW см. в Главе 19 *Связность в среде Windows*.

## LabVIEW в качестве сетевого клиента и сервера

---

Вы можете использовать LabVIEW в качестве клиента, чтобы регистрировать данные и использовать возможности других приложений, или в качестве сервера, чтобы сделать возможности LabVIEW доступными для других приложений. Более подробно об использовании сервера виртуальных приборов для управления ВП на локальном или удаленном компьютере см. в Главе 17 *Программное управление ВП*. Управление виртуальным прибором осуществляется путем доступа к его свойствам и методам с помощью узла свойств и узла вызовов, соответственно.

Перед тем, как Вы сможете получить доступ к свойствам и сможете вызывать методы другого приложения, Вы должны установить сетевой протокол, через который будет происходить доступ к свойствам и методам. Протоколы, которые Вы можете использовать, включают HTTP и TCP/IP. Протокол, который Вы выберите, зависит от приложения. Например, протокол HTTP идеален для Web публикации, но его нельзя использовать для построения ВП, который получает данные, создаваемые другими ВП. Чтобы сделать это, используйте протокол TCP/IP.

Более подробно о коммуникационных протоколах, которые поддерживает LabVIEW, см. в разделе *Низкоуровневые коммуникационные приложения* в настоящей Главе.

**(Windows)** Более подробно об использовании ActiveX технологии с LabVIEW в качестве ActiveX сервера или клиента см. в Главе 19 *Связность в среде Windows*.

## Использование технологии DataSocket

---

Используйте технологию DataSocket корпорации National Instrument для совместного обмена данными с другими ВП на Web или на вашем локальном компьютере. DataSocket работает совместно с установленными коммуникационными протоколами для измерения

и автоматизации подобно тому, как Web броузер работает совместно с различными интернет технологиями.

Технология DataSocket обеспечивает доступ к нескольким механизмам входа и выхода для лицевой панели через диалоговое окно **DataSocket Connection**. Щелкните правой кнопкой объект лицевой панели и выберите из контекстного меню пункт **Data Operation»DataSocket Connection**, чтобы отобразить диалоговое окно **DataSocket Connection**. Вы публикуете (записываете) или подписываетесь (читаете) данные путем указания URL (Uniform Resource Locator – единый указатель ресурса). Во многом это похоже на то, как Вы указываете URL для Web броузера.

Например, если Вы хотите данные в индикаторе термометра на лицевой панели совместно использовать с другими компьютерами на Web, опубликуйте данные термометра, указав URL в диалоговом окне **DataSocket Connection**. Пользователи других компьютеров подписываются на эти данные, помещая термометр на свою лицевую панель и выбирая URL в диалоговом окне **DataSocket Connection**. Более подробно об использовании технологии DataSocket на лицевой панели см. в разделе *Использование DataSocket на лицевой панели* в настоящей Главе.

Более подробно о технологии DataSocket можно почитать в статье *Integrating the Internet into Your Measurement System*. Эта статья доступна в PDF формате. Соответствующий файл *datasock.pdf* размещен в директории */manuals*. Его также можно найти на Web сайте [ni.com](http://ni.com).

## Задание URL

URL используют такие коммуникационные протоколы для передачи данных, как *dstp*, *ftp* и *file*.

Протокол, который Вы используете в URL, зависит от типа данных, которые Вы хотите опубликовать, и от того, как Вы конфигурируете вашу сеть.

Чтобы опубликовать данные или подписаться на данные с помощью DataSocket, Вы можете использовать следующие протоколы:

- **DataSocket Transport Protocol (dstp)** – Исконный протокол для соединений DataSocket. Когда Вы используете этот протокол, ВП связывается с сервером DataSocket. Вы должны обеспечить данные именным тегом, который присоединяется к URL. Соединение DataSocket использует именной тег для адресации конкретного пункта данных на сервере DataSocket. Чтобы использовать этот протокол Вы должны запустить сервер DataSocket.
- **(Windows) OLE for Process Control (opc)** – Разработан специально для совместного получения данных в реальном времени, как например данные, вырабатываемые в системах промышленной автоматики. Чтобы использовать этот протокол, Вы должны запустить OPC сервер.
- **(Windows) logos** – внутренняя технология National Instruments для передачи данных между сетью и вашим локальным компьютером.
- **File Transfer Protocol (ftp)** – Вы можете использовать этот протокол, чтобы специфицировать файл, из которого читаются данные.

**Примечание.** Чтобы прочитать текстовый файл с FTP сайта с использованием технологии DataSocket, добавьте [text] в конец URL.

- **file** – Вы можете использовать этот протокол, чтобы обеспечить связь с локальным или сетевым файлом, который содержит данные.

В Table 18-1 показаны примеры URL для каждого из протоколов.

Используйте URL для dstp, opc и logos для совместного использования данных, поскольку эти протоколы могут обновлять удаленные или локальные элементы управления и индикаторы. Используйте URL для ftp и file для чтения данных из файлов, поскольку эти протоколы не могут обновлять удаленные и локальные элементы управления и индикаторы.

**Table 18-1.** Пример URL для DataSocket

URL	Пример
dstp	dstp://servername.com/numeric, где numeric – это именной тег данных
opc	opc:\National Instruments.OPCTest\item1 opc:\\machine\National Instruments.OPCModbus\Modbus Demo Box.4:0 opc:\\machine\National Instruments.OPCModbus\Modbus Demo Box.4:0?updaterate=100&deadband=0.7
logos	logos://computer_name/process/data_item_name
ftp	ftp://ftp.ni.com/datasocket/ping.wav
file	file:ping.wav file:c:\mydata\ping.wav file:\\machine\mydata\ping.wav

Примеры использования соединений DataSocket можно найти в библиотеке examples\comm\datasktx.llb.

### Форматы данных, поддерживаемые DataSocket

Используйте DataSocket для публикации и подписки на следующие данные:

- **Raw text** (неформатированный текст) – Используйте неформатированный текст для передачи строки в индикатор.
- **Tabbed text** (табулированный текст) – Используйте табулированный текст в качестве электронной таблицы для опубликования данных в виде массива. LabVIEW интерпретирует табулированный текст как массив данных.
- **.wav data** (звуковые данные) – Используйте звуковые данные для подачи звука на вход ВП или функции.
- **Variant data** (вариантные данные) – Используйте варианты для подписки на данные от других приложений, таких как элементы управления ActiveX в системе программирования National Instruments Measurement Studio.

## Использование DataSocket на лицевой панели

Используйте DataSocket соединения лицевой панели для опубликования или подписки на общие данные объектов лицевой панели. Когда Вы совместно используете данные объектов лицевой панели с другими пользователями, Вы публикуете данные. Когда пользователи получают опубликованные данные и просматривают их на своей лицевой панели, пользователи подписываются на данные.

Соединения DataSocket отличаются от соединений Web сервера и от соединений ActiveX, поскольку Вы можете использовать соединения DataSocket непосредственно с лицевой панели без всякого программирования на блок-диаграмме. Каждый элемент управления и индикатор лицевой панели может публиковать или подписываться на данные через собственное соединение DataSocket. Соединения DataSocket для лицевой панели публикуют только данные, а не изображение элемента управления на лицевой панели, таким образом, ВП, которые подписываются через соединение DataSocket, могут выполнять свои собственные операции над данными.

Вы можете установить значение элемента управления непосредственно на лицевой панели и затем опубликовать эти данные, или Вы можете построить блок-диаграмму, подсоединить выход ВП или функции к индикатору и опубликовать данные из этого индикатора. Можно указать следующие типичные сценарии использования соединения DataSocket с элементами управления и индикаторами:

- Публикация значения из элемента управления лицевой панели, чтобы манипулировать элементом управления и публикация данных для других пользователей, которые подписались на них через элемент управления или индикатор. Например, если Вы поместите кнопку на вашу лицевую панель кнопку, которая увеличивает или уменьшает температуру, пользователь другого компьютера может подписаться на эти данные и использовать их в элементе управления, подсоединенном к ВПП или функции, или просматривать их в индикаторе.
- Публикация значения, которое появляется в индикаторе лицевой панели, так, что другой пользователь может подписаться на эти данные и просматривать их в элементе управления или индикаторе на своей лицевой панели или использовать полученные в элементе управления данные для подачи их на вход ВПП или функции. Например, ВП, который вычисляет среднюю температуру и

отображает значение температуры на индикаторе-термометре на лицевой панели, может публиковать данные о температуре.

- Подписка на значение, которое появляется в элементе управления или в индикаторе на лицевой панели другого ВП, чтобы просматривать эти данные на индикаторе лицевой панели вашего ВП. Если Вы подписались на данные из элемента управления, Вы можете использовать эти данные в вашем ВП, подсоединяя этот элемент управления к входу ВПП или функции.
- Публикация из и подписка на элемент управления лицевой панели, в результате чего другие пользователи могут манипулировать элементом управления на лицевой панели вашего ВП с лицевых панелей своих ВП. Когда Вы запускаете ВП, элементы управления на лицевой панели вашего ВП получают текущие значения, которые другие ВП или приложения опубликовали через соединение DataSocket. Когда пользователь изменит значение в элементе управления на лицевой панели, соединение DataSocket опубликует это новое значение для элемента управления на лицевой панели вашего ВП. Если Вы затем поменяете значение в элементе управления, ваш ВП опубликует это значение для лицевых панелей других пользователей.

Объекты лицевой панели, которые подписаны на данные, не обязаны совпадать по типу с объектами, которые опубликовали эти данные. Однако, объекты лицевой панели должны иметь тот же тип данных или, если это числовые данные, они могут подвергаться принудительному преобразованию. Например, Вы можете использовать цифровой индикатор в вашем ВП для просмотра показаний термометра, генерируемых другим ВП. Термометр может выдавать числа с плавающей точкой, а цифровой индикатор может быть целочисленным.

Соединения DataSocket лицевой панели изначально предназначены для совместного использования данных. Для чтения данных из локальных файлов, с FTP серверов или с Web серверов используйте функцию DataSocket Read, функции и ВПП с палитры **File I/O** или ВПП и функции с палитры **Application Control**.

### Чтение и запись общих данных через блок-диаграмму

С блок-диаграммы Вы можете программно читать или записывать данные, используя функции с палитры **DataSocket**.

Используйте функцию DataSocket Write для программной записи общих данных через соединение DataSocket. На Figure 18-1 показано, как записать числовое значение.

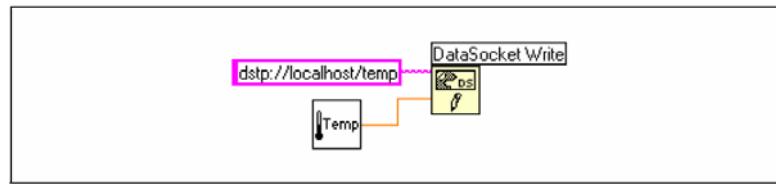


Figure 18-1. Публикация данных с помощью функции DataSocket Write

Функция DataSocket Write является полиморфной, поэтому Вы можете подсоединять многие типы данных к входу **data**. Более подробно о полиморфных ВП и функциях см. в разделе *Полиморфные ВП и функции* в Главе 5 *Построение блок-диаграммы*.

Используйте функцию DataSocket Read для программного считывания данных через соединение DataSocket. На Figure 18-2 показано, как прочитать данные и преобразовать их к числу удвоенной точности с плавающей запятой.

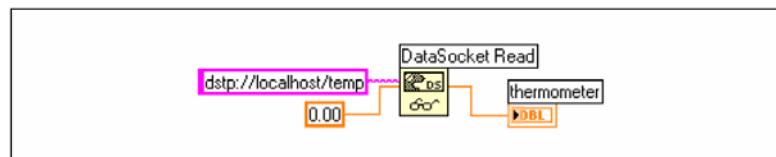


Figure 18-2. Чтение одного значения с помощью функции DataSocket Read

Конвертируйте общие данные к нужному типу, подсоединяя к входу **type** функции DataSocket Read элемент управления или константу. Если Вы не укажете тип, выход **data** функции DataSocket Read будет возвращать данные вариантиного типа, которыми Вы должны манипулировать с помощью функции Variant to Data.

### Программное открытие и закрытие соединений DataSocket

Используйте функции DataSocket Open и DataSocket Close для управления открытием и закрытием соединения DataSocket. Когда Вы открываете соединение DataSocket с помощью функции DataSocket Open, соединение остается открытым пока не выполнится одно из следующих условий: Вы явно закроете это соединение

ние с помощью функции DataSocket Close, Вы закроете весь ВП или ВП завершит свою работу. Вход **URL** функции DataSocket Open принимает только URL для DataSocket. Функция DataSocket Open возвращает ссылочный номер на соединение DataSocket, который Вы можете использовать в качестве входа **URL** для функций DataSocket Read и DataSocket Write.

### Буферизация данных DataSocket

Если Вы используете протокол DataSocket Transport Protocol (dstp), то сервер DataSocket по умолчанию публикует для всех подписчиков только самое последнее значение. Когда один из клиентов опубликует значения для сервера до того, как другой клиент прочтет их, новые значения заменят старые значения, еще не прочитанные другим клиентом. Такие потери необработанных данных могут иметь место на сервере или у клиента. Это может не вызывать проблем, если Вы подписаны на данные DataSocket и хотите получать только самые последние данные, опубликованные сервером. Однако, если Вы хотите получать все данные, публикуемые сервером, то Вы должны буферизировать данные на клиенте.



**Примечание.** Буферизация со стороны клиента применяется также и в других протоколах, таких как **opc**, **logos** и **file**. Чтобы использовать буферизацию в **dstp** Вы, кроме того, должны использовать DataSocket Server Manager для настройки буферизации со стороны сервера. Более подробно о буферизации со стороны сервера см. в справочной системе *DataSocket Help*.

Буферизация в протоколе **dstp** не гарантирует доставку данных. Если данные в буфере сервера или клиента превысят размер буфера, то старые значения из буфера удаляются и заменяются новыми. Чтобы обнаружить потерю в потоке данных, подсоедините публикуемые данные к функции Set Variant Attribute. Это позволит индивидуально идентифицировать каждое публикуемое значение, а у подписчика можно проверить последовательность идентификаторов на наличие пропусков.

Установите вход **mode** функции DataSocket Open в состояние **BufferedRead** или **BufferedReadWrite** и используйте узел свойств, чтобы установить свойства DataSocket, определяющие размер FIFO буфера. Сделайте это так, чтобы гарантировать, что LabVIEW бу-

дет сохранять принимаемые клиентом значения в буфер вместо того, чтобы заменять ими еще непрочитанные значения.



**Примечание.** Если Вы используете свойства DataSocket для установки размера FIFO буфера, Вы должны установить на входе **mode** функции DataSocket Open значение **BufferedRead** или **BufferedReadWrite**. В противном случае этот пункт сервера не будет буферизироваться для этого соединения.

Figure 18-3 демонстрирует буферизацию DataSocket.

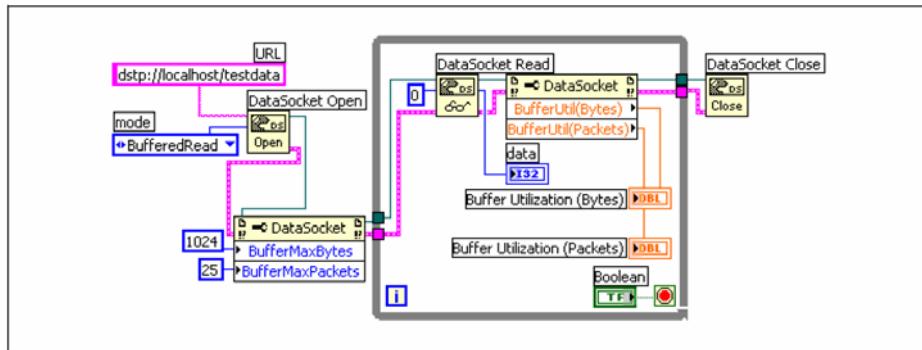


Figure 18-3. Буферизация DataSocket



**Примечание.** Буферизация применяется только тогда, когда Вы используете функцию DataSocket Read, чтобы подписаться на данные, публикуемые сервером. Буферизация не доступна, когда для подписки на данные Вы используете DataSocket соединения лицевой панели.

#### Диагностические отчеты

Используйте свойство Buffer Utilization (Bytes) или свойство Buffer Utilization (Packets), чтобы запросить диагностическую информацию об указанных вами буферах. Используйте эти свойства для проверки, какая часть буфера используется на клиенте, чтобы узнать, достаточен ли текущий размер буфера. Если используемая часть буфера приближается к его максимальному значению, увеличьте размер буфера, чтобы обеспечить хранение всех значений, которые публикует сервер.

Более подробно о задании размера буфера для клиента DataSocket см. в справочной системе *LabVIEW Help*.

### DataSocket и вариантовые данные

В некоторых случаях ВП или другое приложение, которое программно читает данные, не может конвертировать данные обратно к их исходному типу, как например, когда Вы подписываетесь на данные от другого приложения. Кроме того, Вы можете захотеть добавить атрибуты к данным, такие как временная метка или сообщение, которое не входит в тип данных.

В таких случаях используйте функцию To Variant для программного конвертирования данных, которые Вы пишете в соединение DataSocket, к вариантовому типу. На Figure 18-4 показана блок-диаграмма, которая непрерывно вводит данные о показаниях температуры, преобразует их к вариантовому типу и добавляет временную метку в качестве атрибута вариантовых данных.

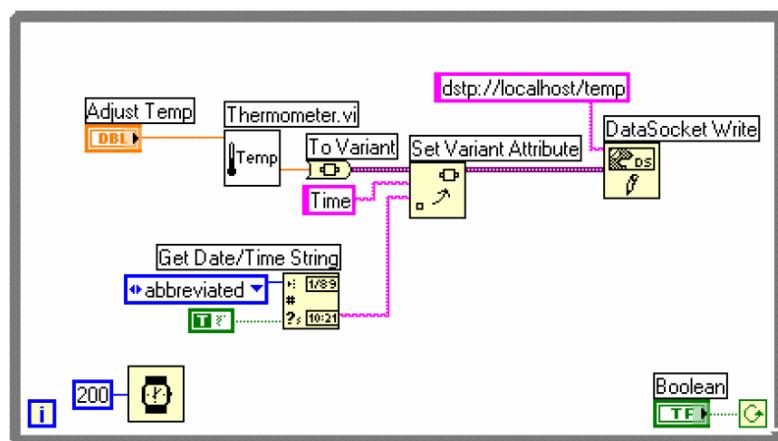
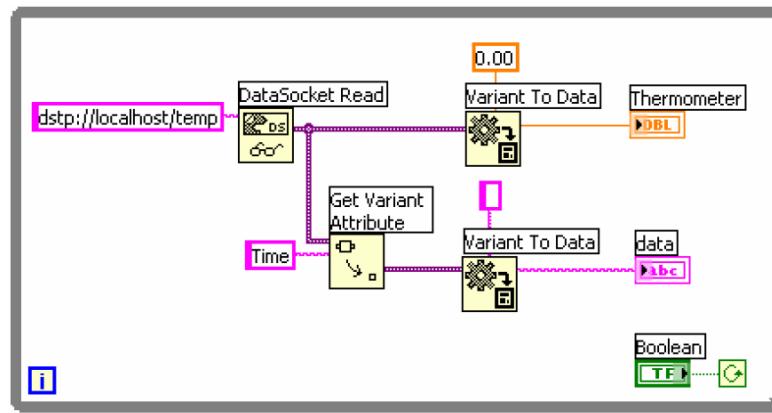


Figure 18-4. Конвертирование общедоступных данных о температуре к вариантовому типу

Когда другой ВП читает общедоступные данные, он должен конвертировать вариантовые данные к тому типу данных, с которым он работает. На Figure 18-5 показана блок-диаграмма, которая непрерывно читает данные о температуре через соединение DataSocket, конвертирует вариантовые данные к показаниям температуры, по-

лучает временную метку, ассоциированную с каждым показанием, и отображает температуру и временную метку на лицевой панели.



**Figure 18-5.** Конвертирование общедоступных вариантов данных к типу данных о температуре

## Опубликование виртуальных приборов на Web

Используйте Web сервер LabVIEW для создания HTML документов, для публикации образов лицевой панели на Web и для внедрения виртуальных приборов на Web страницу.



**Примечание.** Используйте программный пакет LabVIEW Enterprise Connectivity Toolset (набор инструментов для связи в рамках предприятия) для управления ВП на Web и для добавления некоторых функций безопасности для ВП, которые Вы публикуете на Web. Более подробно об этом пакете можно узнать на Web сайте [ni.com](http://ni.com).

### Опции Web сервера

Выберите пункт меню **Tools»Options** и затем пункт **Web Server** из спадающего меню, чтобы установить следующие опции:

- Установка корневой директории и файла протокола.
- Активизация Web сервера.
- Управление доступом броузера к лицевой панели ВП.

- Настройка того, какие лицевые панели ВП являются видимыми на Web.

Перед тем, как Вы сможете опубликовать ВП на Web, Вы должны активизировать (enable) Web сервер на вкладке **Web Server:Configuration** диалогового окна **Options**. Вы также можете активизировать Web сервер с помощью Web Publishing Tool (инструмент публикации на Web), который описан в разделе *Создание HTML документов* в настоящей Главе.

## **Создание HTML документов**

Выберите из меню пункт **Tools»Web Publishing Tool**, чтобы использовать Web Publishing Tool (инструмент публикации на Web) для выполнения следующих задач:

- Создание HTML документа.
- Вставка статических (неподвижных) или движущихся (анимационных) образов лицевой панели в HTML документ. К настоящему времени только Netscape броузер поддерживает движущиеся изображения.
- Внедрение ВП, которым клиенты могут удаленно управлять и просматривать его лицевую панель.
- Вставка текста до и после внедренного изображения лицевой панели ВП.
- Размещение каймы вокруг изображения или внедренной лицевой панели.
- Предварительный просмотр документа.
- Сохранение документа на диск.
- Активизация Web сервера для публикации HTML документов и образа лицевой панели на Web.

## **Опубликование образов лицевой панели**

Используйте .snap URL в Web броузере или HTML документе, чтобы вернуть статический образ лицевой панели виртуального прибора, находящегося в памяти. Параметры запроса в URL специфицируют имя ВП и атрибуты изображения. Например, используйте

<http://web.server.address/.snap?VIName.vi>, где *VIName.vi* – имя ВП, который Вы хотите просмотреть.

Используйте `.monitor` URL, чтобы вернуть движущийся образ лицевой панели виртуального прибора, находящегося в памяти. Параметры запроса в URL специфицируют имя ВП, атрибуты анимации и атрибуты изображения. Например, используйте <http://web.server.address/.monitor?VIName.vi>, где *VIName.vi* – имя ВП, который Вы хотите просмотреть.

### Форматы образов лицевой панели

Web сервер может создавать образы лицевых панелей в графических форматах JPEG и PNG.

Формат JPEG хорошо сжимает графику, но приводит к потере некоторых деталей изображения. Этот формат лучше работает на фотографических изображениях. Для штриховых рисунков, лицевых панелей и блок-диаграмм JPEG сжатие может привести к размытию изображений и нечеткой передаче цветов. Формат PNG также хорошо сжимает графику, хотя не всегда так хорошо, как формат JPEG. Однако, PNG сжатие не приводит к потере деталей изображения.

## Удаленный просмотр и управление лицевыми панелями

---

Вы можете просматривать и управлять лицевыми панелями ВП дистанционно, либо изнутри LabVIEW, либо из Web браузера путем соединения с встроенным в LabVIEW Web сервером. Когда Вы открываете лицевую панель дистанционно на клиенте, Web сервер посыпает лицевую панель клиенту, но блок-диаграмма и все ВП остаются на компьютере сервера. Вы можете взаимодействовать с этой лицевой панелью точно так же, как если бы этот ВП был запущен на клиенте, за исключением того, что блок-диаграмма исполняется на сервере. Используйте эту возможность для опубликования лицевой панели целиком или для управления вашим удаленным приложением безопасно, легко и быстро.



**Примечание.** Используйте Web сервер LabVIEW, если Вы хотите управлять виртуальными приборами как единым целым. Для чтения и записи данных в отдельные элементы управления лицевой панели ис-

пользуйте сервер DataSocket. Более подробно об использовании сервера DataSocket см. в разделе *Использование технологии DataSocket* в настоящей Главе.

## Настройка сервера для клиентов

Перед тем, как клиент сможет дистанционно просматривать лицевую панель и управлять ею с помощью LabVIEW или Web броузера, пользователь компьютера-сервера должен вначале настроить сервер. Чтобы настроить Web сервер выберите из меню пункт **Tools»Options** и затем из выпадающего меню – страницы **Web Server**. Используйте эти страницы для управления доступом к Web серверу и для назначения тех лицевых панелей, которые должны быть видимы дистанционно. Вы также можете использовать эти страницы для установки временных ограничений на то, как долго отдельный клиент может управлять ВП в тех случаях, когда несколько удаленных клиентов ожидают своей очереди на управление этим ВП.

Web сервер позволяет нескольким клиентам одновременно подсоединяться к одной и той же лицевой панели, но в каждый момент времени только один клиент может управлять этой лицевой панелью. Пользователь компьютера-сервера может восстановить управление над любым ВП в любой момент времени. Когда любой, кто имеет доступ на управление лицевой панелью, изменяет значение на лицевой панели, все лицевые панели клиентов также отразят это изменение. Однако, лицевые панели клиентов не отражают все изменения. Вообще говоря, лицевые панели клиентов не отражают изменения, сделанные над внешним видом объектов лицевой панели, но отражают содержащиеся в них фактические значения. Например, если тот, кто имеет доступ к управлению лицевой панелью, изменит режим прокрутки изображения или риски на шкале индикатора Chart или включит или отключит отображение полосы прокрутки, то эти изменения отразятся только на его лицевой панели.

### Лицензия удаленной панели

Вы должны настроить лицензию на поддержку нескольких клиентов, которые могут потенциально соединяться с вашим сервером.



**Примечание (Windows 9.X).** LabVIEW Full Development System и Application Builder включают лицензию удаленной панели, которая позволяет

одному клиенту удаленно просматривать и управлять лицевой панелью. LabVIEW Professional Development System включает лицензию удаленной панели, которая позволяет пяти клиентам удаленно просматривать и управлять лицевой панелью. Вы можете обновить лицензию удаленной панели на поддержку большего числа клиентов в Windows 9.X. (Windows 2000/NT/XP, Mac OS, Unix) LabVIEW Full Development System и Application Builder включают лицензию удаленной панели, которая позволяет одному клиенту удаленно просматривать и управлять лицевой панелью. LabVIEW Professional Development System включает лицензию удаленной панели, которая позволяет пяти клиентам удаленно просматривать и управлять лицевой панелью. Вы можете обновить лицензию удаленной панели на поддержку большего числа клиентов.

## Просмотр и управление лицевыми панелями в LabVIEW или из Web броузера

Клиент может удаленно просматривать и управлять лицевой панелью из LabVIEW или из Web броузера.



**Примечание.** Перед тем, как Вы сможете удаленно просматривать и управлять лицевой панелью, нужно из LabVIEW активизировать Web сервер на компьютере-сервере, на котором размещен ВП, который Вы хотите просматривать и которым хотите управлять.

### Просмотр и управление лицевыми панелями в LabVIEW

Чтобы просмотреть удаленную лицевую панель, используя LabVIEW в качестве клиента, откройте новый ВП и выберите пункт меню **Operate»Connect to Remote Panel**, чтобы открыть диалоговое окно **Connect to Remote Panel**. Используйте это диалоговое окно, чтобы задать интернет адрес сервера и имя ВП, который Вы хотите просматривать. По умолчанию, лицевая панель удаленного ВП изначально находится в режиме наблюдателя (observer mode). Вы можете запросить управление, помещая птичку на опции **Request Control** в диалоговом окне **Connect to Remote Panel**, когда Вы делаете запрос на ВП. Когда ВП появляется на вашем компьютере, Вы также можете щелкнуть в любом месте его лицевой панели и выбрать из контекстного меню пункт **Request Control**. Получить доступ к этому меню можно также, щелкнув по статусной строке внизу окна лицевой панели. Если в этот момент отсутствуют другие клиенты в режиме управления, то появится сообщение о том, что Вы получили управление над этой лицевой панелью. Если

же в этот момент другие клиенты управляют этим ВП, то сервер поставит ваш запрос в очередь, где он будет находиться до тех пор, пока другие клиенты не освободят управление или пока не истечет лимит времени на управление. Только пользователь компьютера-сервера может управлять очередью клиентов, выбирая пункт меню **Tools»Remote Panel Connection Manager**.

Если нужно сохранить данные, вырабатываемые виртуальным прибором, работающим на удаленном компьютере, то используйте технологию DataSocket или протокол TCP вместо механизма удаленных лицевых панелей. Более подробно об использовании DataSocket см. в разделе *Использование технологии DataSocket* в настоящей Главе. Более подробно об использовании TCP см. в разделе *Протоколы TCP и UDP* в настоящей Главе.

Все ВП, которые клиенты хотят просматривать или управлять ими, должны находиться в памяти. Если запрашиваемый ВП находится в памяти, то сервер отправит лицевую панель запрашиваемого ВП клиенту. Если же его в памяти не окажется, то в диалоговом окне **Open Remote Panel** в разделе **Connection Status** появится сообщение об ошибке.

### **Просмотр и управление лицевыми панелями из Web броузера**

Если Вы хотите, чтобы клиенты, у которых нет установленной системы LabVIEW, все же имели возможность удаленно просматривать лицевую панель и управлять ею с помощью Web броузера, они должны установить у себя LabVIEW Run-Time Engine (модуль реального времени). Этот модуль содержит пакет встраивания LabVIEW броузера (LabVIEW browser plug-in package), который инсталлируется в plug-in директорию броузера. Установочный диск LabVIEW содержит инсталлятор модуля LabVIEW Run-Time Engine.

Клиенты устанавливают модуль LabVIEW Run-Time Engine, а пользователь компьютера-сервера создает HTML файл, содержащий тег <OBJECT> и <EMBED>, ссылающийся на ВП, который предназначен для удаленного просмотра и управления. Этот тег содержит URL ссылку на ВП и информацию, которая заставляет Web броузер передать этот ВП на внедренный LabVIEW броузер. Клиенты получают доступ к Web серверу путем введения его Web адреса в строку адреса или в поле URL в верхней части окна Web броузера. Внедренный LabVIEW броузер отобразит лицевую па-

нель и свяжется с Web сервером, в результате чего клиент сможет взаимодействовать с удаленной лицевой панелью. Клиенты запрашивают управление путем выбора пункта **Request Control of VI** в нижней части окна удаленной лицевой панели, либо путем выбора пункта **Request Control of VI** из контекстного меню.

Если в данный момент нет других клиентов, получивших доступ на управление, и если нет других окон броузера в том же соединении, находящихся в режиме управления, то появляется сообщение, свидетельствующее о том, что вы получили управление над данной лицевой панелью. Если имеются другие клиенты, управляющие этим же ВП, то сервер будет держать ваш запрос в очереди до тех пор, пока другие клиенты не освободят управление или пока не истечет лимит времени на управление. Только пользователь компьютера-сервера может управлять очередью клиентов, выбирая пункт меню **Tools»Remote Panel Connection Manager**.



**Примечание.** При просмотре и управлении лицевыми панелями в Web броузере National Instruments рекомендует использовать Netscape 4.7 или старше, или Internet Explorer 5.5 Service Pack 2 или старше.

### **Функциональность, не поддерживаемая при удаленном просмотре лицевых панелей и управлении ими**

В связи с ограничениями Web броузера приложения пользователя ского интерфейса, которые пытаются манипулировать размерами и размещением лицевой панели, могут работать неверно, когда лицевая панель отображается как часть Web страницы. Хотя Web сервер и внедренный LabVIEW броузер позволяют сохранить точность воспроизведения приложений сложного пользовательского интерфейса – в частности, те, которые представляют диалоговые окна и окна ВПП, – некоторые приложения в контексте Web броузера могут работать неправильно. National Instruments рекомендует не экспортirовать такие типы приложений для использования их в Web броузере.

Избегайте использования ВП, которые содержат циклы While Loop, но не содержат функцию ожидания (wait function). Такие ВП препятствуют выполнению фоновых задач в разумных временных рамках, что резко замедляет отклик лицевых панелей при удаленном взаимодействии с ними.

Кроме того, некоторые ВП могут работать на удаленном компьютере не совсем так, как на локальном компьютере. Элементы управления ActiveX, внедренные на лицевую панель, не отображаются на удаленном клиенте, поскольку они строят изображение и работают практически независимо от LabVIEW. Если ВП представлен стандартным диалоговым окном, пользователь с доступом к управлению получит ошибку, поскольку нельзя удаленно просматривать файловую систему. Кроме того, кнопка просмотра (browse button) путевого элемента управления (path control) на удаленной лицевой панели будет неактивной.

Клиенты, удаленно просматривающие лицевую панель, могут видеть различное ее поведение, в зависимости от того является ли присоединенная лицевая панель объектом построенного приложения. В частности, если лицевая панель является объектом построенного приложения, то любые программные изменения лицевой панели, сделанные до того, как клиент соединится с этой лицевой панелью, не будут отражены на компьютере клиента. Например, если узел свойств изменяет заголовок (caption) элемента управления до того, как клиент соединится с этой лицевой панелью, то клиент будет видеть исходный заголовок элемента управления, а не измененный перед соединением.

Только клиенты с доступом к управлению могут удаленно просматривать лицевую панель ВП динамически открытого и запущенного с помощью сервера ВП или лицевую панель ВПП, настроенного на отображение лицевой панели при его вызове. Клиенты без доступа к управлению таким ВП не могут просматривать его лицевую панель.

Блок-диаграммы, которые создают некоторые эффекты пользовательского интерфейса путем опроса свойств элементов управления лицевой панели, могут снизить эффективность своей работы при управлении таким ВП от удаленного компьютера. Вы можете улучшить работу таких ВП с помощью функции Wait for Front Panel Activity (ожидание активности лицевой панели).

## **Отправка данных из ВП по электронной почте**

---

Используйте ВП с палитры **SMTP E-mail** для отправки сообщений с присоединенными данными и файлами по электронной почте, с использованием протокола Simple Mail Transfer Protocol (SMTP).

LabVIEW не поддерживает аутентификацию для SMTP. ВП с палитры **SMTP E-mail** используют для кодирования сообщений формат Multipurpose Internet Mail Extension (MIME). В этом формате в одном сообщении по электронной почте можно отправлять несколько документов, включая двоичные файлы данных. Вы также можете описать свойства каждого вложения, такие как используемые для текстовых файлов наборы символов. Более подробно о пересылке данных от ВП по электронной почте см. в справочной системе *LabVIEW Help*.



**Примечание.** ВП с палитры **SMTP E-mail** доступны только в пакетах LabVIEW Full Development System и LabVIEW Professional Development System.

Кроме адреса получателя электронной почты Вы должны знать Web адрес SMTP сервера. Для указания сервера электронной почты все ВП с палитры **SMTP E-mail** имеют вход **mail server**. Символьная строка, подсоединяемая к входу **mail server** должна быть именем хоста или IP адресом внешнего компьютера-сервера, который может принимать запросы от компьютера, на котором запускаются ВП с палитры **SMTP E-mail**. Если вам неизвестен почтовый сервер, обратитесь к администратору сети, чтобы узнать его правильное имя. После того, как Вы укажете правильный почтовый сервер, ВП с палитры **SMTP E-mail** откроет соединение с сервером и отправит на него команды, которые описывают получателя и содержат сообщение электронной почты. Сервер отправляет сообщение к конкретным получателям или пересыпает его другим SMTP серверам. Примеры использования ВП с палитры **SMTP E-mail** можно найти в библиотеке `examples\comm\smtpex.llb`.



**Примечание.** ВП с палитры **SMTP E-mail** не поддерживают много-байтную кодировку символов, таких, например, как японские

## Выбор набора символов

Входной параметр **character set** ВП с палитры **SMTP E-mail** специфицирует набор символов, который используется в тексте сообщения или во вложении. Набор символов определяет соотношение между символами и их кодами.

Символ – это базовая единица письменных языков: буква, цифра, знак препинания или, в некоторых языках, целые слова. Модификации букв, такие как регистр или знаки ударения, превращают их в отдельные символы. Например,  $\circ$ ,  $\circ$ ,  $\ddot{\text{O}}$ , и  $\text{O}$  – разные символы.

Символьный код – это число, которое представляет данный символ. Поскольку компьютеры понимают только числа, то чтобы работать с символами, они должны ассоциировать каждый символ с числом.

Набор символов (character set) – это соотношение между символами и числами, которые представляют их в компьютере. Например, в наборе символов ASCII кодами символов А, В и С являются, соответственно, 65, 66 и 67.

### **Стандартный для США набор символов ASCII**

Общеупотребительным для электронной почты является стандартный для США набор символов US-ASCII, или просто ASCII. Многие почтовые приложения используют по умолчанию этот набор символов и не работают с другими наборами. Набор символов ASCII представляет буквы и большинство знаков пунктуации, используемых в английском языке. Их количество равно 128. Многие другие символы принадлежат расширению набора ASCII.

Набора символов ASCII может не хватить, поскольку многие другие языки требуют символы, не установленные в ASCII. Например, Вы не сможете написать немецкое слово Müller с помощью набора ASCII, поскольку символ  $\ddot{\text{u}}$  отсутствует в наборе символов ASCII.

### **Набор символов ISO Latin-1**

Поскольку многие языки требуют символы, не установленные в ASCII, страны, которые используют такие языки, создали новые наборы символов. Большинство этих символьных наборов содержат первые 128 символьных кодов в точности повторяющих набор ASCII, а оставшиеся 128 символьных кодов используются для кодирования символов национальных языков. Некоторые из этих символьных наборов используют различные символьные коды для представления одних и тех же символов. Это может вызвать проблемы, когда один символьный набор используется для отображения текста, написанного с помощью другого набора символов. Чтобы устранить эту проблему, стандартный набор символов. Одним из широко распространенных символьных наборов является ISO

Latin-1, известный также как ISO-8859-1. Этот символьный набор представляет символы, используемые в большинстве западноевропейских языков, и большинство приложений электронной почты, которые обрабатывают эти языки, включают этот набор символов.

### **Символьный набор Mac OS**

Корпорация Apple Computer разработала свой собственный расширенный символьный набор еще до того, как был определен набор ISO Latin-1. Символьный набор Mac OS основывается на ASCII, но использует другой, по сравнению с ISO Latin-1, набор 128 расширенных символьных кодов. В связи с этим, электронная почта, которая содержит акцентированные символы, записанные в символьном наборе Mac OS, будут отображаться неверно в почтовых приложениях, которые ожидают текст в кодировке ISO Latin-1. Чтобы устранить эту проблему, почтовые приложения Mac OS конвертируют текст в кодировку ISO Latin-1 перед отправкой его по электронной почте. Когда другое почтовое приложение Mac OS получит текст, помеченный как использующий кодировку ISO Latin-1, оно конвертирует этот текст к кодировке Mac OS.

### **Транслитерация**

Транслитерация – это отображение символов в другом символьном наборе. Используйте транслитерацию, если Вы посыаете электронную почту и вам нужно представить текст почтового сообщения в символах другого символьного набора. Вы можете использовать ВП с палитры **SMTP E-mail** для указания символьного набора, который отображает текст в другой символьный набор перед его отправкой. Например, если Вы создаете сообщение, используя стандартные ASCII символы и указываете, что символьным набором является MacRoman, то ВП с палитры **SMTP E-mail** произведет транслитерацию текста и отправит его как текст в кодировке ISO-8859-1 (ISO Latin-1). используйте вход параметра **translit** ВП с палитры **SMTP E-mail**, чтобы указать, какую транслитерацию может использовать данный ВП. Транслитерация определяется виртуальным символьным набором, конечным символьным набором и транслитерационным, или отображающим, файлом. Транслитерационный файл указывает, как входной символ отображается в выходной символ.

Транслитерационный файл – это двоичный файл (таблица) из 256 байт, имеющий 256 входов. Каждый вход в файл соответствует символу виртуального символьного набора (входного) и содержит

новый символьный код из конечного символьного набора. Например, если Вы хотите отобразить символ а, с символьным кодом 61 в символ А с символьным кодом 41, вход транслитерационного файла с индексом 61 должен содержать числовое значение 41. Если вход содержит числовое значение, равное его индексу, то транслитерационный файл не изменит данный символ виртуального символьного набора. Например, если вход транслитерационного файла с индексом 61 содержит числовое значение 61, то этот символ при транслитерации не изменится.

Когда Вы задаете транслитерационный файл в качестве итогового символьного набора на входе **translit**, то применяется отображение в указанном порядке. Например, если на вход **translit** поступает [MacRoman iso-8859-1 macroman.trl, MacRomanUp MacRoman asciiup.trl], то символьный набор MacRoman изменится на iso-8859-1 с использованием файла macroman.trl и затем MacRomanUp изменится на MacRoman с использованием файла asciiup.trl. Примеры trl-файлов можно найти в директории vi.lib\Utility\SMTP.

## **Низкоуровневые коммуникационные приложения**

---

LabVIEW поддерживает несколько низкоуровневых протоколов, которые Вы можете использовать для коммуникаций между компьютерами.

Каждый протокол отличается друг от друга, особенно в части способа указания места расположения в сети удаленного приложения. Вообще говоря, каждый из этих протоколов не сравним с другими протоколами. Например, если Вы хотите установить сообщение между Mac OS и Windows, то необходимо использовать протокол, который работает на обеих платформах, такой, скажем, как TCP.

### **Протоколы TCP и UDP**

Протокол TCP (Transmission Control Protocol) и протокол UDP (User Datagram Protocol) доступны на всех платформах, которые поддерживает LabVIEW. TCP является надежным протоколом, ориентированным на соединения. Он обеспечивает обнаружение ошибок и гарантирует, что данные поступают в нужном порядке и без дублирования фрагментов. По этим причинам TCP обычно является лучшим выбором для сетевых приложений.

Хотя UDP может дать большую эффективность по сравнению с TCP и не требует соединения, он не дает полной гарантии доставки. Обычно UDP используется в приложениях, для которых гарантия доставки не является критичной. Например, приложение может передавать данные к потребителю так часто, что потеря некоторых фрагментов данных не создает проблем. Более подробно об использовании протоколов TCP и UDP в приложениях LabVIEW см. в руководстве (Application Note) *Using LabVIEW with TCP/IP and UDP*.

Используйте ВП UDP Multicast Open вместо функции UDP Open, чтобы открыть соединения, способные только читать, только писать или писать и читать UDP данные, используя групповой IP адрес. Групповой (multicast) IP адрес определяет группу адресатов. Групповые IP адреса лежат в диапазоне от 224.0.0.0 до 239.255.255.255. Когда клиент хочет присоединиться к группе адресатов, он подписывается на групповой IP адрес этой группы. После того, как клиент подпишется к группе адресатов, он будет получать данные, посылаемые на этот групповой IP адрес. Более подробно об использовании групповой адресации в UDP см. в руководстве (Application Note) *Using LabVIEW with TCP/IP and UDP*.

## События Apple и PPC Toolbox (Mac OS)

Более употребительная только в Mac OS форма коммуникации – это события Apple. Используйте события Apple для отправки сообщений на запрос действий или возврат информации от других Mac OS приложений.

Инструментарий межпрограммного обмена PPC (Program-to-Program) Toolbox представляет собой низкоуровневой формой отправки и получения данных между Mac OS приложениями. PPC Toolbox обеспечивает большую эффективность по сравнению с событиями Apple, поскольку требуется меньшая избыточность передаваемой информации. Однако, поскольку PPC Toolbox не определяет типы информации, которую Вы можете передать, многие приложения не поддерживают его. PPC Toolbox является наилучшим методом для пересылки большого объема информации между приложениями, которые поддерживают PPC Toolbox. Более подробно об использовании событий Apple и PPC Toolbox в приложениях LabVIEW см. в руководстве (Application Note) *Using Apple Events and PPC Toolbox to Communicate with LabVIEW Applications on the Macintosh*.

## ВП с палитры Pipe (Unix)

Используйте ВП с палитры **Pipe**, чтобы открыть, закрыть, прочитать и записать в трубопроводы (Pipes), которым в Unix присвоены имена. Используйте поименованные трубопроводы для коммуникации между LabVIEW и остальными процессами.

## Выполнение команд системного уровня (Windows и Unix)

Используйте ВП с палитры **System Exec** для выполнения или запуска других Windows приложений или Unix приложений командной строки изнутри ВП. С помощью этих ВП можно выполнять командные строки системного уровня, которые могут включать любые параметры, поддерживаемые приложением, которое Вы хотите запустить.

## 19. Связность в среде Windows

---

LabVIEW обеспечивает доступ к другим приложениям Windows, использующим технологии .NET или ActiveX. Вы можете использовать LabVIEW в качестве .NET клиента, чтобы получить доступ к объектам, свойствам и методам, связанным с .NET серверами. LabVIEW не является .NET сервером. Другие приложения не могут непосредственно связываться с LabVIEW через .NET. Посредством ВП, поддерживающих .NET, Вы можете соединиться со службами Windows и с программными интерфейсами приложений (API). Каркас .NET включает обслуживание COM+ компонентов, каркас разработки ASP Web и поддержку обслуживающих Web протоколов, таких как SOAP, WSDL и UDDI. Каркас .NET представляет собой программную основу для .NET окружения, которое Вы используете для построения, развертывания и запуска основанных на Web приложений, развитых клиентских приложений и XML Web служб.

С помощью технологии ActiveX Windows приложения, такие как LabVIEW, осуществляют публикацию множества объектов, команд и функций, которые становятся доступными для других Windows приложений. Вы можете использовать LabVIEW в качестве клиента ActiveX для получения доступа к объектам, свойствам, методам и событиям, связанным с другими приложениями, поддерживающими ActiveX. LabVIEW может также действовать в качестве ActiveX сервера, при этом другие приложения могут получать доступ к объектам, свойствам и методам LabVIEW.

.NET обозначает .NET технологию фирмы Microsoft. Вы должны инсталлировать каркас .NET. Более подробно о .NET и об инсталляции каркаса см. на Web сайте MSDN. ActiveX обозначает ActiveX технологию фирмы Microsoft и OLE технологию. Более подробно об ActiveX см. в руководствах: Microsoft Developer's Network documentation, Inside OLE (автор Kraig Brockschmidt), second edition, и Essential COM (автор Don Box).

---

**Более**

**подробно...**

Более подробно относительно использования технологии .NET и ActiveX см. в справочной системе LabVIEW Help и на Web сайте National Instruments [ni.com](http://ni.com)

---

## Окружение .NET

---

Ниже представлен перечень основных элементов, которые образуют окружение (среду) .NET. Это информация дана с целью помочь вам понять суть технологии .NET, однако ее недостаточно для грамотного использования компонентов .NET в LabVIEW.

- **Common Language Runtime (CLR)** (общий язык времени исполнения) – набор библиотек, отвечающих за службы времени исполнения, такие как языковая интеграция, безопасное принуждение, память, сбор мусора, управление процессами и потоками. CLR формирует основу для .NET и использует промежуточный язык IL (Intermediate Language), который генерируют все языки программирования, чтобы облегчить взаимодействие между .NET и другими языками.

Чтобы помочь взаимодействию .NET с другими языками CLR обеспечивает систему типов данных, которая наводит мосты между различными языками программирования и ограничениями операционных систем. Разработчики используют CLR, чтобы рассматривать систему как совокупность типов данных и объектов, а не как совокупность памяти и потоков. Чтобы генерировать информацию в формате метаданных CLR IL, нужны соответствующие компиляторы и компоновщики. В системе Win32 компиляторы всех языков программирования генерируют CLR IL код вместо ассемблерного кода.

- **Class Libraries** (библиотеки классов) – набор классов, которые обеспечивают стандартную функциональность, такую как ввод и вывод, манипуляция над символьными строками, управление безопасностью, сетевые коммуникации, управление потоками, управление текстами, создание пользовательского интерфейса и др. Эти классы содержат те же функции, которые использует система Win32/COM. В каркасе .NET Вы можете классы, созданные в одном из .NET языков, использовать в другом .NET языке.
- **Assemblies** (сборки) – категория развертывания, подобная DLL, OCX или исполняемый модуль для компонента в COM. Сборки являются динамическими библиотеками (DLL) и исполняемыми модулями, которые строятся с использованием .NET CLR. Сборки могут состоять из одного или нескольких файлов. Сборка включает манифест, который содержит информацию об имени сборки, ее версии, размещении, данные о безопасности публикации, спи-

сок файлов, которые образуют сборку, перечень зависимых сборок, ресурсы и экспортируемые типы данных. Однофайловые сборки содержат все данные в единственном файле, включая манифест и все необходимые ресурсы. Многофайловые сборки должны иметь внешние ресурсы, такие как изображения, иконки, звуковые файлы и т.п. или должны иметь один файл для кода ядра, а другие для вспомогательных библиотек.

Сборки могут быть публичными (public) или частными (private). Технология .NET требует, чтобы частные сборки находились в той же директории, что и приложение, а публичные сборки должны размещаться в общедоступном глобальном КЭШе, который называется глобальный кэш для сборок (GAC – Global Assemble Cache). Разработчик приложения обычно пишет частные сборки для использования в этом приложении. Разработчик сборки занимается также управлением версий. Именем сборки является имя файла (без расширения), в котором находится манифест.

- **Global Assemble Cache (GAC)** (глобальный кэш для сборок) – список публичных сборок, доступных в системе. GAC подобен использованию COM регистрации.

## Функции и узлы с палитры .NET

---

Используйте следующие функции и узлы LabVIEW, которые обеспечивают доступ к объектам, свойствам и методам, связанным с .NET сервером. (Они размещены на палитре .NET).

- Используйте Constructor Node (узел конструктора), чтобы выбрать из сборки конструктор .NET класса и создать экземпляр (instance) этого класса для исполнения. Когда Вы помещаете этот узел на блок-диаграмму, LabVIEW отображает диалоговое окно **Select .NET Constructor**.
- Используйте Property Node (узел свойств), чтобы получить (прочитать) и установить (записать) свойства, связанные .NET классом.
- Используйте Invoke Node (узел вызовов), чтобы вызвать методы, связанные с .NET классом.
- Используйте функцию Close Reference (закрыть ссылку), чтобы закрыть все ссылки на .NET объекты, когда необходимость на соединение с ними отпадает.

- Используйте функцию To More Generic Class (к самому общему классу), чтобы преобразовать (upcast) .NET ссылку к ее базовому классу.
- Используйте функцию To More Specific Class (к самому частному классу), чтобы преобразовать (downcast) .NET ссылку к ее производному классу.

## LabVIEW в качестве .NET клиента

---

Когда система программирования LabVIEW запрашивает объекты, ассоциированные с .NET сборками, она действует как .NET клиент. Использование LabVIEW в качестве .NET клиента предусматривает выполнение следующих трех основных шагов.

1. Создайте .NET объект с помощью конструктора и установите ссылку на него.
2. Подсоедините ссылку на .NET объект к узлу свойств или к узлу вызовов и выберите нужное свойство или метод.
3. Закройте ссылку на .NET, чтобы закрыть соединение с объектом.

Для получения доступа к .NET объекту, используйте на блок-диаграмме узел конструктора, чтобы создать нужный .NET объект. С помощью этого узла выберите класс объектов из сборки. Когда Вы поместите узел конструктора на блок-диаграмму, появится диалоговое окно **Select .NET Constructor**, в котором отобразится список всех публичных сборок в GAC. Если Вы хотите выбрать частную сборку, щелкните в диалоговом окне кнопку **Browse** и, перемещаясь по файловой системе, найдите нужную частную сборку. .NET сборки хранятся в файлах с расширениями .dll или .exe. После того, как Вы выберите частную сборку, она появится в спадающем меню **Assembly** диалогового окна **Select .NET Constructor** после его повторного запуска.

Когда Вы выбираете сборку и класс, в диалоговом окне **Select .NET Constructor** в секции **Constructor** появляется конструктор для этого класса. Выберите конструктор и щелкните кнопку **OK**, чтобы закрыть это диалоговое окно. Имя выбранного вами класса появится в узле конструктора.

Узел конструктора похож на функцию Automation Open для ActiveX, за исключением того, что узел конструктора может задавать инициализационные параметры для создания объектов. Вы можете подсоединить ссылку на .NET сервер от узла конструктора к узлу свойств или к узлу вызовов и выбрать нужное свойство или метод из контекстного меню. Используйте функцию Close Reference, чтобы закрыть ссылку на .NET объект.

Более подробно о создании .NET объектов см. в справочной системе *LabVIEW Help*.

## Отображение типов данных

LabVIEW конвертирует типы данных параметров свойств, методов и конструкторов в типы данных LabVIEW таким образом, что LabVIEW может читать и правильно интерпретировать эти данные. LabVIEW отображает типы данных, которые не могут быть конвертированы как ссылочные номера .NET. В Table 19-1 приведены типы данных .NET и типы данных LabVIEW, в которые они конвертируются.

Table 19-1. Типы данных LabVIEW и .NET

.NET типы	LabVIEW типы
System.Int32, System.UInt32, System.Int16, System.UInt16	   
System.String	
System.Boolean	
System.Byte, System.Char, System.UByte	  
System.Single, System.Double, System.Decimal	  
System.Array	Отображается как массив соответствующего типа
Enumeration	 (редко)
DateTime	
Любой другой .NET объект	

## **Развертывание .NET приложений**

---

После того, как Вы создадите ВП, содержащий .NET компоненты, на его основе можно построить либо исполняемый модуль, либо библиотеки LLB или DLL.

### **Развертывание исполняемого модуля**

При построении приложений, использующих .NET объекты, необходимо скопировать частные сборки, которые используются в виртуальных приборах, в директорию стандартного приложения и убедиться, что на целевом компьютере установлен скелет .NET.

### **Развертывание виртуальных приборов**

При развертывании виртуальных приборов необходимо скопировать частные сборки, которые используют данные ВП, в директорию ВП верхнего уровня. Убедитесь, что все сборки, используемые в этих ВП, находятся в той же директории ВП верхнего уровня и что на целевом компьютере установлен скелет .NET.

### **Развертывание библиотек DLL**

При развертывании построенных с помощью LabVIEW библиотек DLL необходимо скопировать частные сборки, используемые виртуальными приборами, в директорию приложения, использующего такие библиотеки DLL, и убедиться, что на целевом компьютере установлен скелет .NET.

## **Конфигурирование приложения .NET клиента**

---

.NET позволяет администрировать приложения с помощью конфигурационных файлов. Конфигурационный файл содержит XML текст и обычно имеет расширение `.config`. Чтобы настроить приложение .NET клиента в LabVIEW, можно создать конфигурационный файл для ВП верхнего уровня или для стандартного приложения. Дайте этому файлу имя, совпадающее с именем приложения, и добавьте расширение `.config`. Например, `My App.vi.config` или `My App.exe.config`.

## Объекты, свойства, методы и события ActiveX

---

Приложения, использующие ActiveX, содержат объекты, имеющие незащищенные свойства и методы, доступные для других приложений. Объекты могут быть видимыми для пользователей, такие как кнопки, окна, картинки, документы и диалоговые окна, или невидимыми для пользователей, такие как объекты приложений. Вы можете получить доступ к приложению путем доступа к объектам, ассоциированным с этим приложением, и установкой свойств или вызовом методов таких объектов.

Более подробно об объектах, свойствах и методах в системе программирования LabVIEW см. в разделе *Манипуляция установками приложения и ВП* в Главе 17 *Программное управление ВП*.

События – это действия, совершаемые объектами, такие как щелчок мыши, нажатие клавиши или нехватка памяти. Всякий раз, когда такие действия совершаются, объект посыпает событие на ожидающий его ActiveX контейнер вместе с сопровождающей это событие информацией.

Более подробно об использовании узла Register Event Callback для обработки событий ActiveX см. в разделе *События ActiveX* в настоящей Главе.

### Виртуальные приборы, функции, элементы управления и индикаторы, работающие с ActiveX

Используйте следующие ВП, функции, элементы управления и индикаторы для доступа к объектам, свойствам, методам и событиям, ассоциированным с другими приложениями, имеющими доступ к ActiveX:

- Используйте элемент управления Automation Refnum, чтобы создать ссылку на об объект ActiveX. Щелкните правой кнопкой этот элемент управления на лицевой панели, чтобы выбрать из списка библиотеку типов и объект, к которому Вы хотите получить доступ.
- Используйте функцию Automation Open, чтобы открыть объект ActiveX.

- Используйте ActiveX контейнер для доступа и отображения объекта ActiveX на лицевой панели. Щелкните правой кнопкой этот контейнер, выберите из контекстного меню пункт **Insert ActiveX Object** и затем выберите нужный вам объект.
- Используйте узел свойств для получения (чтения) или установки (записи) свойств, связанных с объектом ActiveX.
- Используйте узел вызовов для вызова методов, связанных с объектом ActiveX.
- Используйте узел Register Event Callback для обработки событий, которые происходят в объекте ActiveX.
- Используйте элемент управления или индикатор Variant для передачи данных в элементы управления ActiveX и из них. Более подробно об использовании вариантовых данных см. в разделе *Обработка вариантовых данных* в Главе 5 *Построение блок-диаграммы*.

## LabVIEW в качестве клиента ActiveX

---

Когда ВП обращается к объектам, связанным с другим ActiveX приложением, LabVIEW действует как клиент ActiveX. Вы можете использовать LabVIEW в качестве ActiveX клиента следующими способами:

- Откройте приложение, такое как Microsoft Excel, создайте документ и добавьте данные в этот документ.
- Внедрите в контейнер на лицевой панели документ, такой как документ Microsoft Word или электронная таблица Microsoft Excel.
- Поместите кнопку или другой объект из другого приложения, такой как кнопка **Help**, которая вызывает справочный файл другого приложения, на лицевую панель.
- Свяжите элемент управления ActiveX, созданный вами, с другим приложением.

LabVIEW обеспечивает доступ к объекту ActiveX с помощью элемента управления Automation Refnum или посредством контейнера ActiveX. Каждый из них является объектом лицевой панели. Используйте элемент управления Automation Refnum, чтобы выбрать нужный объект ActiveX. Используйте контейнер ActiveX, чтобы выбрать и отобразить на лицевой панели такой объект ActiveX, как

кнопка или документ. Оба объекта появятся на блок-диаграмме как элементы управления Automation Refnum.

## Доступ к ActiveX приложениям

Чтобы получить доступ к ActiveX приложениям, используйте элемент управления Automation Refnum на блок-диаграмме для получения ссылки на приложение. Подсоедините этот элемент управления к входу функции Automation Open, которая откроет вызываемое приложение. Используйте узел свойств для выбора нужных свойств, связанных с объектом. Используйте узел вызовов для вызова нужных методов, связанных с объектом. Затем закройте ссылку на объект с помощью функции Close Reference. Закрытие ссылки удаляет объект из памяти.

К примеру, Вы можете построить ВП, который открывает Microsoft Excel, в результате чего его окно появляется на пользовательском экране. Затем создает рабочую книгу, создает электронную таблицу, создает таблицу в LabVIEW и записывает эту таблицу в электронную таблицу Excel.

Примером использования LabVIEW в качестве клиента Excel является ВП Write Table To XL из библиотеки examples\comm\ExcelExamples.llb.



**Примечание.** Приложения, которые включают настраиваемые ActiveX интерфейсы, появляются с иконкой . Щелкните эту иконку, чтобы выбрать объект такого интерфейса. Более подробно о настраиваемых интерфейсах см. в разделе *Поддержка настраиваемых автоматических интерфейсов ActiveX* в настоящей Главе.

## Вставка объекта ActiveX на лицевую панель

Чтобы вставить объект ActiveX на лицевую панель, щелкните правой кнопкой элемент управления ActiveX Container, выберите из контекстного меню пункт **Insert ActiveX Object** и затем выберите нужный ActiveX элемент управления или документ, который Вы хотите вставить. Вы можете установить свойства для ActiveX объекта либо интерактивно с помощью броузера свойств ActiveX (ActiveX Property Browser) или вкладок свойств, либо программно с помощью узла свойств. Более подробно об установке свойств

ActiveX объекта см. в разделе *Установка свойств ActiveX* в настоящей Главе.

Используйте узел вызовов (Invoke Node) для вызова методов, ассоциированных с объектом.

Например, Вы можете отобразить Web страницу на лицевой панели с помощью элемента управления ActiveX Container, чтобы получить доступ к управлению Web броузером (например, Microsoft Web Browser). Для этого нужно выбрать класс методов Navigate, затем выбрать метод URL и указать нужный URL.

Если Вы используете элемент управления ActiveX Container, то нет необходимости подсоединять на блок-диаграмме элемент управления Automation Refnum к функции Automation Open или закрывать ссылку на объект с помощью функции Close Reference. Вы можете подсоединяться непосредственно к узлу вызовов или к узлу свойств, поскольку ActiveX Container внедряет вызываемое приложение в LabVIEW. Однако, если ActiveX Container включает свойства или методы, которые возвращают ссылки на другие объекты ActiveX, то эти дополнительные ссылки Вы должны закрывать сами.

### **Режим конструирования (design mode) для ActiveX объектов**

Щелкните правой кнопкой элемент управления ActiveX Container и выберите из контекстного меню пункт **Advanced»Design Mode**, чтобы отображать этот элемент управления в режиме конструирования во время редактирования ВП. В режиме конструирования события (events) не генерируются и соответствующие процедуры для их обработки не вызываются. Исходным режимом по умолчанию является режим запуска (run mode), при котором Вы можете взаимодействовать с объектом так же, как потом это будет делать пользователь.

### **Установка свойств ActiveX**

После того, как Вы либо откроете сервер ActiveX, либо вставите элемент управления ActiveX или документ ActiveX, Вы можете установить свойства, связанные с этим элементом управления или документом с помощью броузера свойств ActiveX, страниц свойств или узла свойств.

## **Броузер свойств ActiveX**

Используйте броузер свойств ActiveX (ActiveX Property Browser) для просмотра и установки всех свойств, связанных с элементом управления ActiveX или с документом ActiveX в контейнере ActiveX. Для вызова броузера свойств ActiveX щелкните правой кнопкой элемент управления или документ в контейнере на лицевой панели и выберите из контекстного меню пункт **Property Browser**. Вы также можете выбрать пункт главного меню **Tools»Advanced»ActiveX Property Browser**. Броузер свойств ActiveX дает легкий способ интерактивной установки свойств объекта ActiveX, но его нельзя использовать для программной установки свойств и можно использовать только с объектами ActiveX, находящимися в контейнере на лицевой панели. Броузер свойств ActiveX не доступен в режиме запуска (run mode) и при запущенном ВП.

## **Страницы свойств ActiveX**

Многие ActiveX объекты обладают страницами свойств (property pages), которые группируют свойства, связанные с объектом на отдельных вкладках. Для активизации страниц свойств ActiveX, щелкните правой кнопкой контейнер на лицевой панели и выберите из контекстного меню имя объекта.

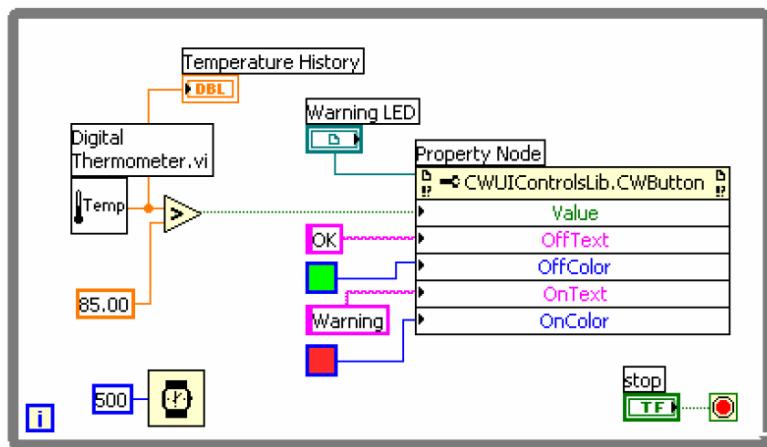
Также как и броузер свойств ActiveX, страницы свойств ActiveX дают легкий способ интерактивной установки свойств ActiveX, но их нельзя использовать для программной установки свойств и они имеются только у ActiveX объектов, помещенных в контейнер. Кроме того, страницы свойств ActiveX имеются не у всех ActiveX объектов. Страницы свойств также не доступны в режиме запуска (run mode) и при запущенном ВП.

## **Узлы свойств**

Используйте узел свойств (Property Node) для программной установки свойств ActiveX объектов. Например, если Вы используете ActiveX объект для выдачи пользователю сообщения о том, что температура вышла за заданный предел, то используйте узел свойств, чтобы установить свойство объекта Value для указания этого предела.

Ниже приведен пример, в котором при превышении температуры порога 85 градусов по Фаренгейту изменяется свойство Value для

ActiveX элемента управления CWButton, который входит в состав библиотеки пользовательского интерфейса ActiveX системы программирования National Instruments Measurement Studio.



В этом примере элемент управления CWButton действует как лампочка, изменяющая свой цвет и отображающая предупреждение *Warning*, когда температура превысит порог. Это предупреждение появляется в качестве текста включенного состояния (свойство *OnText*) элемента управления CWButton.

 **Примечание.** В этом примере для установки свойств OffText, OffColor, OnText и OnColor Вы могли бы использовать броузер свойств ActiveX или страницы свойств ActiveX, поскольку нет необходимости устанавливать эти свойства программно. Более подробно об использовании броузера свойств ActiveX и страниц свойств ActiveX см. в разделах настоящей Главы *Броузер свойств ActiveX* и *Страницы свойств ActiveX*, соответственно.

## LabVIEW в качестве сервера ActiveX

Свойства и методы приложений LabVIEW, виртуальных приборов и элементов управления доступны из других приложений с помощью ActiveX вызовов. Другие ActiveX приложения, такие как Microsoft Excel, могут запрашивать свойства, методы и отдельные ВП из LabVIEW. При этом LabVIEW действует как ActiveX сервер.

Например, Вы можете внедрить ВП для построения графика в электронную таблицу Excel и, находясь в электронной таблице, вводить данные в этот ВП и запускать его. В результате запуска этого ВП на его графике отобразятся введенные вами данные.

Примером использования свойств и методов LabVIEW в электронной таблице Excel является файл examples\comm\fresp.xls.

## **Поддержка настраиваемых автоматических интерфейсов ActiveX**

Если Вы пишите клиент ActiveX, который запрашивает свойства и методы из сервера ActiveX с помощью LabVIEW, то Вы можете получить доступ к настраиваемым интерфейсам, которыми располагает сервер. Чтобы сделать это вам нет необходимости использовать IDispatch. Однако разработчик сервера ActiveX должен сделать так, чтобы параметры свойств и методов в этих настраиваемых интерфейсах имели типы данных Automation (IDispatch). Разработчик сервера должен это сделать для того, чтобы сделать доступными несколько интерфейсов для одного объекта, а не для нескольких объектов. Вы сможете использовать эти интерфейсы в среде LabVIEW. Более подробно относительно настраиваемых интерфейсов см. в документации на среду программирования, в которой разработан сервер.

## **Использование констант для установки параметров в виртуальных приборах с возможностями ActiveX**

Некоторые параметры узлов ActiveX требуется выбирать из списка правильных значений. Устанавливайте значения таких параметров путем выбора их описательных имен из кольцевых констант (ring constant). Чтобы установить кольцевую константу при построении ВП, щелкните правой кнопкой параметр узла, который принимает значения данных, и выберите из контекстного меню пункт **Create Constant**. Значения, доступные в кольцевой константе, будут зависеть от ссылочного номера, поступающего на данный узел. На Figure 19-1 и Figure 19-2 показаны примеры использования кольцевой и числовой констант для установки значения параметра.

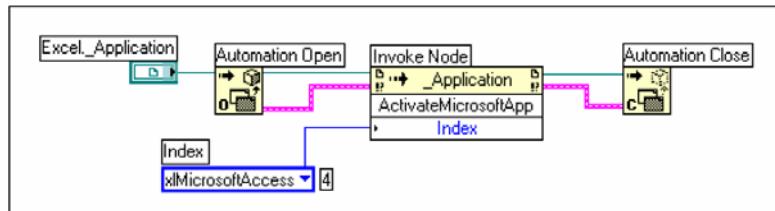


Figure 19-1. Установка значения с помощью кольцевой константы

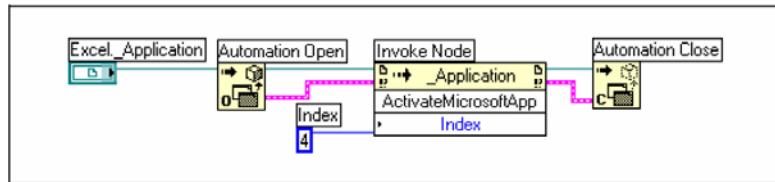


Figure 19-2. Установка значения с помощью числовой константы

Параметры, которые получают значения данных, имеют небольшие стрелочки слева от имени параметра. Чтобы просмотреть числовые значения, соответствующие символическим именам, щелкните правой кнопкой кольцевую константу и выберите из контекстного меню пункт **Visible Items»Digital Display**.

Виртуальные приборы на Figure 19-1 и на Figure 19-2 оба обращаются к приложению Microsoft Excel и выполняют метод. Параметр **Index** метода **ActivateMicrosoftApp** имеет следующие опции: **MicrosoftWord**, **MicrosoftPowerPoint**, **MicrosoftMail**, **MicrosoftAccess**, **MicrosoftFoxPro** и **MicrosoftSchedulePlus**.

Чтобы указать, какое числовое значение параметра **Index** соответствует опции **MicrosoftAccess** на Figure 19-1, выберите опцию **MicrosoftAccess** из спадающего меню кольцевой константы. Числовое значение, соответствующее выбранной опции, появится в окошке рядом с кольцевой константой. Вместо использования кольцевой константы Вы можете ввести числовое значение опции прямо в числовую константу, как это показано на Figure 19-2.

## События ActiveX

Чтобы использовать события ActiveX в приложении, Вы должны зарегистрировать событие и затем обработать его, когда оно насту-

пит. Регистрация события ActiveX подобна динамической регистрации событий, которая объяснялась в разделе *Динамическая регистрация событий* в Главе 9 *Событийно управляемое программирование*. Однако, архитектура ВП с событиями ActiveX отличается от архитектуры ВП с обработкой событий, которая описана в Главе 9 *Событийно управляемое программирование*. Следующие компоненты образуют типичный ВП с событиями ActiveX:

- ActiveX объект, для которого Вы хотите сгенерировать событие.
- Узел Register Event Callback (обработчик зарегистрированного события) для описания и регистрации типа события, которое Вы хотите сгенерировать.
- ВП-обработчик (callback VI), который содержит код, созданный вами и предназначенный для обработки описанного вами события.

Вы можете генерировать и обрабатывать события объектов ActiveX в контейнере или события объектов ActiveX, которые указаны с помощью ссылки Automation Refnum. Например, Вы можете вызвать элемент управления Windows tree (дерево директорий) через контейнер ActiveX и указать, что Вы хотите сгенерировать событие Double Click для пунктов, отображенных в этом элементе управления.

Узел Register Event Callback является растягиваемым узлом; он позволяет обрабатывать несколько событий, подобно узлу Register For Events.

Когда Вы присоединяете ссылку на объект ActiveX к узлу Register Event Callback и указываете событие, которое Вы хотите сгенерировать для этого объекта, происходит регистрация этого события для указанного объекта ActiveX. После такой регистрации, Вы создаете ВП-обработчик (callback VI), который содержит код, предназначенный для обработки этого события.

## Обработка событий ActiveX

Вы должны создать ВП-обработчик (callback VI), который обрабатывает события от элементов управления ActiveX, когда этот элемент управления вырабатывает зарегистрированные события. ВП-обработчик запускается при возникновении таких событий. Чтобы создать ВП-обработчик, щелкните правой кнопкой вход **VI Ref** уз-

ла Register Event Callback и выберите из контекстного меню пункт **Create Callback VI**. LabVIEW создаст ВП, содержащий следующие компоненты:

- **Event common data** (общие данные о событии) включает следующие элементы:
  - **Source** - числовой элемент управления, который определяет в качестве источника события LabVIEW или ActiveX. Значение 1 означает событие ActiveX.
  - **Type** - указывает событие, которое наступает. Это перечислительный тип для события пользовательского интерфейса и 32-битный целый беззнаковый тип для событий ActiveX и событий от других источников. Для событий ActiveX тип события представляет код метода или ID (идентификатор) под которым это событие зарегистрировано.
  - **Time** - это временная метка в миллисекундах, которая отмечает время, когда событие генерируется.
- **CtrlRef** – это ссылка на объект ActiveX (или Automation Refnum), на котором происходит событие.
- **Event Data** (данные о событии) – кластер параметров, специфичный для события, которое обрабатывает ВП-обработчик. LabVIEW определяет подходящий кластер **Event Data**, когда Вы выбираете событие из узла Register Event Callback. Более подробно об уведомляющих и фильтрующих событиях см. в разделе *Уведомляющие и фильтруемые события* в Главе 9 *Событийно управляемое программирование*.
- **Event Data Out** (выход данных о событии) – кластер модифицируемых параметров, специфичных для событий, обрабатываемых ВП-обработчиком. Этот элемент доступен только для фильтруемых событий.
- (Необязательный) **User Parameter** (параметр пользователя) – это данные, которые LabVIEW выдает пользователю через ВП-обработчик, когда ActiveX объект генерирует событие.

 **Примечание.** Вы можете использовать существующий ВП в качестве ВП-обработчика пока его соединительная панель ВП, который Вы предполагаете использовать, совпадает с соединительной панелью данных о событии. National Instruments рекомендует, чтобы ВП-обработчик

был повторно вызываемым (reentrant), поскольку если это не так, то LabVIEW не сможет вызывать его синхронно, если события ActiveX произойдут несколько раз подряд.

## 20. Вызов кода из текстовых языков программирования

---

Большинство стандартных совместных библиотек Вы можете вызывать в LabVIEW с помощью узла Call Library Function (вызов библиотечной функции). Кроме того, Вы можете из LabVIEW вызвать Си код с помощью узла кодового интерфейса – Code Interface Node (CIN).

Более подробно о соображениях, зависящих от платформы, при вызове внешнего кода см. в разделе *Call Library Function Nodes and Interface Nodes* в Главе 6 *LabVIEW Style Guide* руководства *LabVIEW development Guidelines*. Более подробно о вызове кода из текстовых языков программирования см. в руководстве *Using External Code in LabVIEW*.

---

**Более подробно...**

---

Более подробно относительно вызова кода из текстовых языков программирования см. справочную систему LabVIEW Help

---

### Узел вызова библиотечной функции

---

Используйте узел вызова библиотечной функции (Call Library Function Node) для вызова самых общих совместных библиотек или DLL. С помощью этой функции Вы можете создавать в LabVIEW интерфейс для обращения к уже существующим библиотекам или к новым библиотекам, специально написанным для использования в LabVIEW. National Instruments рекомендует использовать узел вызова библиотечной функции для создания интерфейса к внешнему коду.

### Узел кодового интерфейса

---

Используйте узел кодового интерфейса (CIN) в качестве альтернативного метода обращения к исходному коду, написанному на языке Си. Узел вызова библиотечной функции обычно легче в использовании по сравнению с CIN.

## 21. Формулы и уравнения

---

Если Вам нужно использовать в LabVIEW сложные аналитические выражения, то вовсе не обязательно соединять на блок-диаграмме различные арифметические функции. Вы можете создать выражение в обычном математическом стиле и затем внедрить это выражение в ваше приложение.

Для выполнения математических операций в среде LabVIEW используйте формульный узел (Formula Node) или узел выражение (Expression Node). Для повышения функциональности Вы можете сделать связь с математическим приложением MATLAB.

---

**Более подробно...**

Более подробно об использовании выражений и принятого в них синтаксиса, доступных функциях и операторах, а также описание возможных ошибок см. в справочной системе LabVIEW Help

---

### Методы использования выражений в LabVIEW

---

Для выполнения математических операций на блок-диаграмме Вы можете использовать формульный узел (Formula Node), узел выражения (Expression Node) и узел скриптов приложения MATLAB (MATLAB script node).



**Примечание.** Чтобы использовать узел скриптов на вашем компьютере, на нем должно быть установлено приложение MATLAB, поскольку данный узел для исполнения скриптов осуществляет вызов сервера скриптов приложения MATLAB. Ввиду того, что LabVIEW использует технологию ActiveX для реализации узла скриптов, это возможно только в среде Windows. Узел скриптов похож на формульный узел, но, в отличие от него, он позволяет вам импортировать существующий скрипт MATLAB в формате ASCII и запускать этот скрипт в LabVIEW. Так же как и для формульного узла, Вы можете вводить данные в узел скриптов и выводить из него.

## Формульные узлы

---

Формульный узел (Formula Node) – это обычный текстовый узел, который Вы можете использовать для выполнения математических операций на блок-диаграмме. При этом для реализации арифметического выражения вам не потребуется ни доступ к внешнему коду или приложению, ни соединение низкоуровневых арифметических функций. В дополнение к текстовым арифметическим выражениям формульный узел дает возможность использовать текстовые версии операторов: `if`, `while loop`, `for loop` и `do loop`, которые знакомы программистам на Си. Эти программные элементы похожи на те, которые используются при программировании на Си, но не идентичны им.

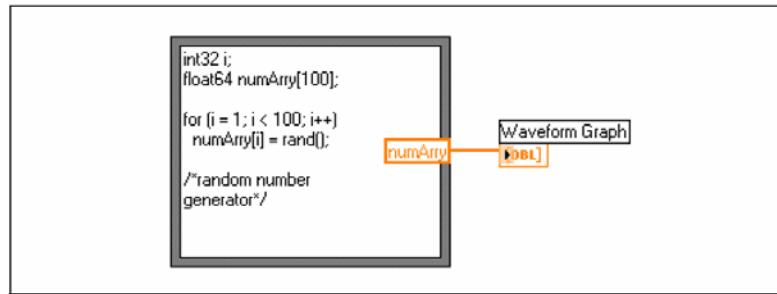
Формульный узел удобен для выражений, которые используют много переменных, и при использовании уже существующего текстового кода. Вы можете копировать и вставлять фрагменты имеющегося текстового кода вместо того, чтобы заново создавать это часть программы в графическом виде.

Формульные узлы используют проверку типов, чтобы гарантировать, что индексы массивов имеют числовой тип, а операнды битовых операций имеют целочисленный тип. Кроме того, формульные узлы проверяют индексы массивов на соответствие диапазону. В случае выхода индекса за диапазон значениям массива присваиваются нулевые значения, а признаку выхода за диапазон присваивается значение `por`, что указывает на отсутствие операции.

Формульные узлы осуществляют также автоматическое преобразование типов.

### Использование формульного узла

Формульный узел имеет растягиваемое поле, подобно циклам: `For Loop`, `While Loop`; структурам: `Case`, `Stacked Sequence` и `Flat Sequence`. Однако, вместо поддиаграммы формульный узел содержит один или несколько операторов в Си-подобном формате, разделенных точкой с запятой, что иллюстрируется приведенным ниже примером. Также как и в СИ, можно добавлять комментарии, ограничивая их парой слэш-звездочки (`/*comment*/`).



Пример использования формульного узла можно найти в ВП Equations из библиотеки examples\general\struct.llb.

## Переменные в формульном узле

Когда Вы работаете с переменными, помните о следующем.

- Количество переменных или выражений в формульном узле не ограничено.
- Не должно быть входов или выходов с совпадающими именами, но вход и выход могут иметь одно и то же имя.
- Для объявления входной переменной щелкните правой кнопкой границу формульного узла и выберите пункт **Add Input** из контекстного меню. Нельзя объявлять входные переменные внутри формульного узла.
- Для объявления выходной переменной щелкните правой кнопкой границу формульного узла и выберите пункт **Add Output** из контекстного меню. Имя выходной переменной должно совпадать либо с именем входной переменной, либо с именем переменной, объявленной внутри формульного узла.
- Вы можете изменить входную переменную на выходную и наоборот, щелкнув ее правой кнопкой и выбирая из контекстного меню **Change to Input** или **Change to Output**.
- Вы можете объявить и использовать переменную внутри формульного узла, не связывая ее ни с входным, ни с выходным соединением узла.
- Необходимо подключать все входные терминалы.

- Переменные могут числовыми скалярами с плавающей точкой; точность их представления может зависеть от конфигурации вашего компьютера. Кроме того, переменные могут быть целыми числами и массивами.
- Переменные не могут иметь единиц.

## Узлы выражения

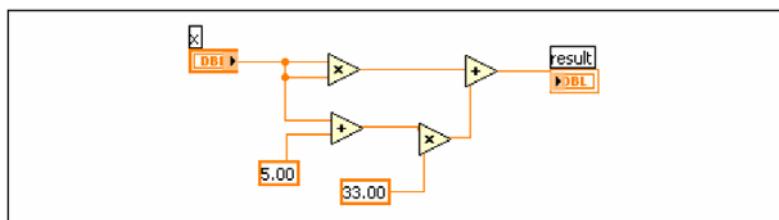
Используйте узел выражения (Expression Node) для вычисления арифметических выражений, или равенств, которые содержат одну единственную переменную. Узел выражения удобен, когда равенство хоть и содержит только одну переменную, но является достаточно сложным.

Узел выражения использует значение, которое поступает на входной терминал в качестве значения переменной. Выходной терминал возвращает результат вычисления выражения.

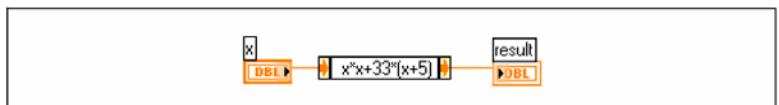
Например, рассмотрим простое выражение:

$$x \times x + 33 \times (x + 5)$$

Ниже показана блок-диаграмма, на которой для представления этого выражения используются числовые функции (с палитры **Numeric**).



С помощью узла выражения эту блок-диаграмму можно значительно упростить, как показано ниже.



## Полиморфизм в узлах выражения

Входной терминал узла выражения (Expression Node) имеет тот же тип данных, что и элемент управления или константа, подсоединенные к нему. Выходной терминал имеет тот же тип данных, что и входной терминал. Данные на входе могут иметь в качестве типа: любой числовой скаляр (кроме комплексного), массив не комплексных числовых скаляров, или кластер не комплексных числовых скаляров. В случае массива или кластера узел выражения применяет записанное выражение к каждому элементу входного массива или кластера.

## Узел скриптов приложения MATLAB

---

Используйте узел скриптов приложения MATLAB (MATLAB script node) для создания, загрузки и редактирования скриптов приложения MATLAB на блок-диаграмме.

Чтобы использовать узел скриптов приложения MATLAB на вашем компьютере должно быть установлено приложение MATLAB. Если у вас уже есть скрипты, написанные в приложении MATLAB, их можно импортировать в узел скриптов.

Чтобы обмениваться значениями между MATLAB и LabVIEW, назначьте входные и выходные терминалы узла скриптов для соответствующих переменных в скрипте приложения MATLAB. Вы можете определить функцию терминала из записи выражения. Например, если скрипт содержит оператор присваивания  $X = i + 3$ , Вы можете назначить  $i$  в качестве входного терминала, чтобы управлять тем, как узел скрипта будет вычислять значение  $X$ , и  $X$  назначить в качестве выходного терминала, чтобы получить окончательный результат вычисления скрипта.

Если у вас еще нет написанного скрипта, то Вы можете поместить узел скрипта на блок-диаграмму и создать скрипт, используя синтаксис приложения MATLAB. LabVIEW будет обмениваться с сервером скриптов (script server engine), который есть ни что иное, как программа, реализующая этот скрипт. Обмен с сервером скриптов и управление им происходит с помощью стандартного протокола. Сервер скриптов инсталлируется вместе с приложением MATLAB.

Из-за особенностей языка скриптов приложения MATLAB узел скриптов не может определить тип данных терминала, который Вы создаете. Вы должны сами назначить каждому терминалу узла скриптов тип данных LabVIEW. В Table 21-1 показано соответствие типов данных LabVIEW и MATLAB.

**Table 21-1.** Типы данных LabVIEW и MATLAB

Тип данных LabVIEW	Тип данных MATLAB
	Real
	String
	String
	Real Vector
	Real Matrix
	Complex
	Complex Vector
	Complex Matrix

Для преобразования типов данных LabVIEW в типы данных MATLAB используйте функции с подпалитры **Conversion** палитры **Numeric** или с подпалитры **String/Array/Path Conversion** палитры **String**.

### Рекомендации по программированию скриптов приложения MATLAB

Приведенные ниже рекомендации по программированию скриптов приложения MATLAB могут облегчить их отладку.

- Написав скрипт, перед внедрением в LabVIEW запустите его в среде MATLAB, чтобы протестировать и отладить.
- Проверяйте типы данных. Когда Вы создаете новый вход или выход, убедитесь, что тип данных терминала назначен правильно. Для отслеживания этого Вы можете использовать функции Error In и Error Out.

- Создайте элементы управления и индикаторы для входов и выходов, чтобы проверить значения, которые узел скриптов передает между LabVIEW и MATLAB. Это позволит вам при необходимости найти место, в котором узел скриптов вычисляет неверное значение.
- Воспользуйтесь при отладке параметрами проверки ошибок. Создайте индикатор для терминала **error out** у узла скриптов, в результате чего Вы сможете видеть информацию о возможных ошибках в процессе выполнения. Формульные узлы могут показывать ошибки и на этапе компиляции.

# Организация LabVIEW

В этом приложении описана структура размещения файлов LabVIEW и рекомендуемые места для сохранения файлов.

## Структура директорий LabVIEW

---

В этом разделе описана структура файловой системы LabVIEW для платформ Windows, Mac OS и UNIX. При установке LabVIEW инсталлируется программный драйвер для аппаратуры GPIB, DAQ, VISA, IVI, Monitor Control и IMAQ, рассчитанный на ту платформу, которая имеется в наличии. Более подробно о конфигурировании вашего оборудования см. в Главе 3 *Configuring Measurement Hardware* руководства *LabVIEW Measurement Manual*.

После завершения инсталляции директория LabVIEW имеет следующую структуру.

### Библиотеки

- `user.lib` – Директория, в которой Вы можете сохранять созданные вами элементы управления и виртуальные приборы. LabVIEW отображает элементы управления на палитрах **User Control**, а виртуальные приборы – на палитрах **User Libraries**. Эта директория не изменяется, даже если Вы обновите или переустановите LabVIEW. Более подробно о сохранении файлов в директории `user.lib` см. в разделе *Добавление ВП и элементов управления в пользовательскую подпалитру (User Subpalette) и в подпалитру приборных драйверов (Instrument Drivers Subpalette)* в Главе 3 *Среда LabVIEW*.
- `vi.lib` – Содержит библиотеки встроенных ВП, которые LabVIEW отображает в соответствующих группах на палитре **Functions**. Не сохраняйте файлы в директории `vi.lib`, поскольку при обновлении или повторной инсталляции LabVIEW эти директории перезаписываются.

- **instr.lib** – Содержит приборные драйвера, используемые для управления приборами PXI, VXI, GPIB, а также приборами последовательного порта и другими приборами компьютера. Когда Вы инсталлируете драйверы приборов National Instruments и помещаете их в эту директорию, LabVIEW добавляет их на палитру **Instrument Drivers**.

## Структура и поддержка

- **menus** – Содержит файлы, которые LabVIEW использует для конфигурирования структуры палитр **Controls** и **Functions**.
- **resource** – Содержит дополнительные файлы поддержки приложений LabVIEW. Не сохраняйте файлы в этой директории, поскольку LabVIEW переписывает эти файлы при обновлении или повторной инсталляции.
- **project** – Содержит файлы, которые становятся пунктами в меню **Tools** среды LabVIEW.
- **templates** – Содержит шаблоны типовых ВП.
- **www** – Место размещения HTML файлов, которые можно просматривать с помощью Web сервера.

## Изучение и инструкции

- **examples** – Содержит примеры ВП. Для обзора и изучения примеров выберите из главного меню пункт **Help»Find Examples**.

## Документация

- **manuals** – Содержит документацию в формате PDF. Эта директория не содержит файлы справки. Для просмотра PDF файлов выберите из главного меню пункт **Help»Search the LabVIEW Bookshelf**.
- **help** – Содержит файлы справки. Для доступа к справочной системе LabVIEW Help выберите из главного меню пункт **Help»VI, Function, & How-To Help**.

## Mac OS

В дополнение к перечисленным выше директориям пользователи компьютеров Mac имеют директорию совместных библиотек, в которой находятся файлы поддержки приложений LabVIEW.

## Предлагаемые места для сохранения файлов

---

При инсталляции LabVIEW создаются директории `vi.lib` и `resource`, предназначенные только для системных целей. Не сохраняйте свои файлы в этих директориях.

Вы можете сохранять свои файлы в следующих директориях:

- `user.lib` – Все элементы управления или ВП, которые Вы хотите отобразить на палитрах **User Controls** или **User Libraries**. Более подробно о сохранении файлов в директории `user.lib` см. в разделе *Добавление ВП и элементов управления в пользовательскую подпалитру (User Subpalette) и в подпалитру приборных драйверов (Instrument Drivers Subpalette)* Главы 3 *Среда LabVIEW*.



**Примечание.** Сохраняйте ВПП в директории `user.lib` только в тех случаях, когда они могут быть использованы в любой части проекта без каких бы-то ни было модификаций. Пути к ВП размещенным в директории `user.lib` задаются относительно директории `labview`. Пути к ВПП, сохраненным где-нибудь еще, задаются относительно родительского ВП. Следовательно, копируя ВП из директории `user.lib` с целью модификации его под конкретное применение, не изменяйте путь к его ВПП, размещенным в директории `user.lib`.

- `instr.lib` – Любой приборный драйвер ВП, который Вы хотите отобразить на палитре **Instrument Drivers**.
- `project` – ВП, которые Вы используете для расширения возможностей LabVIEW. ВП, которые Вы сохраняете в этой директории, появляются в меню **Tools**.
- `www` – Место размещения HTML файлов, к которым Вы можете получить доступ через Web сервер.
- `help` – Любые ВП, файлы PDF и `.hlp` файлы, которые Вы хотите сделать доступными через пункт меню **Help**.
- `LabVIEW Data` – Любые файлы с данными, которые генерирует LabVIEW. Например, `.lvm` или `.txt` файлы.

Вы также можете создать директорию в произвольном месте вашего жесткого диска и сохранять в ней файлы LabVIEW, которые Вы

создадите. Более подробно о создании директорий см. в разделе *Сохранение ВП* в Главе 7 *Создание ВП и ВПП*.

## Полиморфные функции

Функции являются полиморфными в различной степени – ни один, несколько или все их входы могут быть полиморфными. Некоторые входы функций принимают числовые или булевые значения. Некоторые принимают числовые или строковые. Некоторые принимают не только числовые скаляры, но и массивы чисел, кластеры чисел, массивы кластеров чисел и т.п. Некоторые принимают только одномерные массивы, хотя элементы массива могут иметь любой тип. Некоторые функции принимают все типы данных, включая комплексные числа. Более подробно о создании и использовании полиморфных единиц см. в замечаниях к приложению (Application Notes) *Polymorphic Units in LabVIEW*.

---

### Более подробно...

Более подробно относительно полиморфных функций см. справочную систему *LabVIEW Help*.

---

### Преобразование числовых представлений

---

Вы можете преобразовать произвольное числовое представление к любому другому числовому представлению. Когда Вы подсоединяете два или более числовых входов различного типа к функции, эта функция обычно возвращает результат в том из этих форматов, который является длиннее или шире. Принудительное преобразование меньших представлений к более широким осуществляется перед выполнением функций. В тех случаях, когда такое преобразование имеет место, соответствующие терминалы помечаются точкой (coercion dot).

Некоторые функции, такие как Divide (деление), Sine (синус) и Cosine (косинус) всегда выдают результат в виде числа с плавающей точкой. Если Вы подсоедините к их входам целые числа, то они перед выполнением вычислений будут преобразованы к числам с плавающей точкой удвоенной точности.

Для вещественных скалярных величин обычно лучше использовать числовое представление удвоенной точности с плавающей точкой. Использование чисел одинарной точности с плавающей точкой могут дать (а иногда и нет) некоторую экономию в быстродействии, однако значительно более подвержены эффектам переполнения и потери точности. К примеру, функции из библиотеки анализа данных используют числа с плавающей точкой удвоенной точности. При необходимости следует использовать числа с плавающей точкой повышенной точности (extended-precision). Эффективность и точность представление формата повышенной точности варьируется в зависимости от используемой платформы. Более подробно об эффекте переполнения в представлении чисел с плавающей точкой см. в разделе *Неопределенные или неожиданные данные* в Главе 6 *Запуск и отладка виртуальных приборов*.

Для целых чисел обычно наилучшим является представление в виде 32-битных целых со знаком.

Если Вы подсоединяете источник данных к приемнику, и они имеют различное числовое представление, то LabVIEW конвертирует данные в соответствии со следующими правилами:

- **Знаковое или беззнаковое целое к приемнику числа с плавающей точкой** – Конвертация является точной, за исключением случая преобразования 32-битного целого в число с плавающей точкой однократной точности, когда LabVIEW уменьшает точность представления с 32 бит до 24 бит.
- **Число с плавающей точкой к приемнику целого числа со знаком или без знака** – LabVIEW заменяет значения, выходящие за диапазон представления целых чисел, на максимальное или минимальное целое значение. Большинство объектов целого типа, такие как терминал итераций цикла For Loop, округляют числа с плавающей точкой. При этом производится округление дробной части 0.5 к ближайшему четному целому значению. Например, число 6.5 будет округлено к числу 6, а не к числу 7.
- **Целое число к приемнику целого числа** – Значения, выходящие за диапазон представления целых чисел приемника, не преобразуются к минимальному или максимальному значению. Если значение числа-источника меньше в его представлении числа-приемника, то LabVIEW расширяет знак на старшие разряды числа-приемника, если число-источник со знаком. Если число-

источник имеет беззнаковый тип, то старшие разряды числа-приемника заполняются нулевым битом. Если представления числа-источника больше представления числа-приемника, то копируются только младшие значащие биты числа-источника.

## Полиморфизм числовых функций

---

Арифметические функции получают числовые входные данные. За некоторыми исключениями, отмеченными в описании функций, выходные данные имеют то же числовое представление, что и входные данные либо, если входы имеют различное представление, выход имеет представление входа с наиболее широким представлением.

Арифметические функции работают с числами, с массивами чисел, с кластерами чисел, с массивами кластеров чисел, с комплексными числами и т.д. Формальное рекурсивное определение допустимых входных типов данных имеет следующий вид:

*Numeric type* = numeric scalar OR array [*numeric type*] OR cluster [*numeric types*]

Числовые скаляры могут быть числами с плавающей точкой, целыми числами или комплексными числами с плавающей точкой. В LabVIEW не предусмотрено использование массивов, элементы которых являются массивами.

Массивы могут иметь любое количество измерений (dimension) произвольной размера (длины). Кластеры могут содержать произвольное количество элементов. Тип данных на выходе функций имеет такое же числовое представление, как тип данных на их входе. Функции с одним входом осуществляют поэлементную обработку массива или кластера.

Для функций с двумя входами можно использовать следующие комбинации входов:

- **Подобие (Similar)** – Оба входа имеют одинаковую структуру. В этом случае выход имеет ту же структуру, что и входы.

- **Один скаляр** (One scalar) – Один из входов является числовым скаляром, а другой – массивом или кластером. В этом случае выходом будет массив или кластер.
- **Массив** (Array of) – Один из входов является числовым массивом, а другой – имеет числовой тип элементов этого массива. В этом случае выходом будет массив.

Для подобных входов LabVIEW выполняет функцию над соответствующими элементами структур. Например, LabVIEW может сложить два массива элементов поэлементно. Оба массива должны иметь одну и ту же размерность. Вы можете складывать массивы с разным числом элементов; результат такого сложения (массив) будет иметь столько элементов, сколько их у входного массива с наименьшим числом элементов. Кластеры должны иметь одинаковое число элементов, а соответствующие их элементы должны быть одинакового типа.

Нельзя использовать функцию Multiply (умножение) для выполнения матричного умножения. Если Вы примените функцию Multiply к двум матрицам, то LabVIEW возьмет первое число из первой строки первой матрицы, умножит его на первое число первой строки второй матрицы и т.д.

Для операций, принимающих на одном входе скаляр, а на другом входе – структуру в виде массива или кластера, LabVIEW выполнит функцию над скаляром и соответствующими элементами этой структуры. К примеру, LabVIEW может вычесть число из всех элементов массива, независимо от его размерности.

Для операций, принимающих на одном входе данные числового типа, а на другом входе массив элементов того же типа, LabVIEW выполнит функцию над каждым элементом массива. Например, график представляет собой массив точек, каждая из которых задается кластером из двух чисел  $x$  и  $y$ . Чтобы сместить график на 5 единиц в направлении  $x$  и на 8 единиц в направлении  $y$ , Вы можете прибавить к графику (как к массиву точек) точку (5,8).

На Figure B - 1 показаны возможные комбинации полиморфизма для функции Add (сложить).

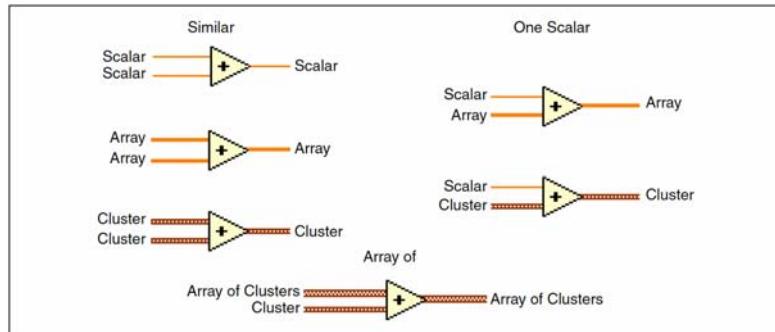


Figure B - 1. Комбинации полиморфизма функции Add (сложить)

## Полиморфизм булевых функций

Логические функции получают либо булевые, либо числовые входные данные. Если входные данные имеют числовой тип, то LabVIEW выполняет операцию поразрядно (в двоичном представлении). Если входные данные являются числами с плавающей точкой, то LabVIEW округляет их к длинному целому (long integer) и такой же тип будут иметь выходные данные.

Логические функции работают с массивами чисел или булевых величин, с кластерами чисел или булевых величин, с массивами кластеров из чисел или булевых величин и т.д.

Формальное рекурсивное определение допустимых входных типов данных имеет следующий вид:

*Logical type* = Boolean scalar OR numeric scalar OR array [*logical type*] OR cluster [*logical types*]

с тем исключением, что недопустимы комплексные числа и массивы массивов.

Логические функции с двумя входами могут иметь такие же комбинации типов входных данных, что и арифметические функции. Однако, логические функции имеют дополнительные ограничения, состоящие в том, что базовые операции могут выполняться только либо над двумя булевыми величинами, либо над двумя числами. Например, Нельзя выполнить операцию AND между булевой вели-

чиной и числом. На Figure B - 2 показаны некоторые допустимые комбинации из булевых величин для функции AND.

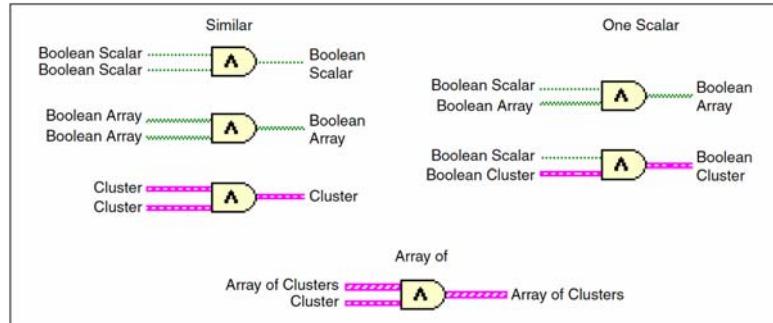


Figure B - 2. Комбинации полиморфизма с булевыми данными для функции AND

## Полиморфизм для функций обработки массивов

Большинство функций обработки массивов принимают  $n$ -мерные массивы данных любого типа. Однако, на схеме подключения в описаниях функций указаны числовые массивы в качестве типа данных по умолчанию.

## Полиморфизм строковых функций

Функции String Length (длина строки), To Upper Case (преобразовать к верхнему регистру), To Lower Case (преобразовать к нижнему регистру), Reverse String (развернуть строку) и Rotate String (циклически сдвинуть строку) принимают строки, кластеры и массивы строк, массивы кластеров. Функции To Upper Case и To Lower Case могут также принимать числа, кластеры чисел и массивы чисел, интерпретируя их как символы в ASCII кодах. Входы параметров ширины и точности представления результирующей записи числа в виде строки должны быть скалярами.

## Полиморфизм функций для конвертирования строк

Функции Path To String (конвертировать путь в строку) и String To Path (конвертировать строку в путь) являются полиморфными. Это значит, что они работают со скалярными значениями, с массивами скаляров, с кластерами скаляров, с массивами кластеров скаляров и

т.д. Выход представляет собой ту же композицию данных, что и на входе, но с элементами нового типа.

## **Полиморфизм дополнительных функций преобразования строк в числа**

Функция Number To Decimal String (преобразовать число к строке десятичных цифр) и аналогичные ей функции: ...To Hex String (к строке шестнадцатеричных цифр), ...To Octal String (к строке восьмеричных цифр), ...To Engineering String (к строке записи числа в инженерном формате), ...To Fractional String (к строке записи числа с дробной частью заданного формата) и ...To Exponential String (к строке записи числа в экспоненциальном формате) принимают данные в виде кластеров и массивов чисел и вырабатывают на выходе кластеры и массивы строк. Функция Decimal String To Number (преобразовать строку десятичных цифр в число) и аналогичные ей функции: Hexadecimal ... (преобразовать строку шестнадцатеричных цифр в число), Octal ... (преобразовать строку восьмеричных цифр в число) и Fract/Exp ... (преобразовать строку цифр с дробной частью заданного формата / экспоненциального формата / инженерного формата в число) принимают кластеры и массивы строк и вырабатывают на выходе кластеры и массивы чисел. Входы параметров ширины и точности представления записи числа в виде строки должны быть скалярами.

## **Полиморфизм функций обработки кластеров**

Функции Bundle (собрать кластер) и Unbundle (разобрать кластер) не показывают типы данных на своих входных и выходных терминалах, пока Вы не подсоедините к ним объекты. Как только Вы это сделаете, терминалы примут вид, похожий на терминалы элементов управления и индикаторов, соответствующего типа данных.

## **Полиморфизм функций сравнения**

Функции Equal? (равно?), Not Equal? (не равно?) и Select (выбор) принимают на входе данные любого типа до тех пор, пока на оба входа поданы данные одного и того же типа.

Функции Greater or Equal? (больше или равно), Less or Equal? (меньше или равно), Less? (меньше), Greater? (больше), Max & Min (максимум и минимум), и In Range? (нахождение в диапазоне) при-

нимают на входе данные любого типа, за исключением комплексных (complex), путей (path) и ссылочных номеров (refnum) и при условии, что данные на обоих входах имеют один и тот же тип. Вы можете сравнивать числа, строки, булевые значения, массивы строк, кластеры чисел, кластеры строк и т.д. Однако нельзя сравнивать число со строкой или строку с булевым значением и т.п.

Функции, которые сравнивают значения с нулем, могут принимать на входе числовые скаляры, кластеры чисел и массивы чисел. Эти функции выдают на выходе булевые значения в виде структур того же вида, что и входные данные.

Функция Not A Number/Path/Refnum (не число/путь/ссылочный номер) принимает на входе те же типы данных, что и функции сравнения с нулем. Эта функция может также принимать на входе путь (path) и ссылочный номер (refnum). Функция Not A Number/Path/Refnum дает на выходе булевые значения в виде структур того же вида, что и входные данные.

Функции Decimal Digit? (десятичная цифра), Hex Digit? (шестнадцатеричная цифра), Octal Digit? (восьмеричная цифра), Printable? (печатный символ), и White Space? (пробел) принимают на входе числовой или строковый скаляр, кластер строк или кластер не комплексных чисел, массив строк или массив не комплексных чисел и т.д. Результат на выходе состоит из булевых значений в виде структур того же вида, что и входные данные.

Функция Empty String/Path? (пустая строка/путь) принимает на входе строковый скаляр, кластер строк, массив строк и т.д. Результат на выходе состоит из булевых значений в виде структур того же вида, что и входные данные.

Вы можете использовать функции Equal? (равно), Not Equal? (не равно), Not A Number/Path/Refnum? (не число/путь/ссылочный номер), Empty String/Path? (пустая строка/путь), и Select (выбор) с входными данными типа путь (path) и ссылочный номер (refnum). Никакие другие функции сравнения не принимают данные такого типа.

Функции сравнения, которые принимают на входе массивы и кластеры обычно возвращают булевые массивы или кластеры того же вида, что и входные. Если Вы хотите, чтобы функция возвращала

одно единственное булево значение, щелкните эту функцию правой кнопкой и выберите из контекстного меню пункт **Comparison Mode»Compare Aggregates**. Более подробно о том, как эти функции сравнивают агрегаты см. в разделе *Сравнение массивов и кластеров* в Приложении С *Функции сравнения*.

## Полиморфизм логарифмических функций

---

Логарифмические функции (функции с палитры **Logarithmic**) принимают числовые входные данные. Если вход является целым числом, то выходом будет число с плавающей точкой удвоенной точности. В противном случае, выход имеет такое же числовое представление, как и вход.

Эти функции работают с числами, с массивами чисел, с кластерами чисел, с массивами кластеров чисел, с комплексными числами т.д. Формальное рекурсивное определение допустимых входных типов данных имеет следующий вид:

*Numeric type* = numeric scalar OR array [*numeric type*]  
OR cluster [*numeric type*]

с тем исключением, что недопустимы массивы массивов.

Массивы могут иметь произвольную величину и любое число измерений. Кластеры могут содержать произвольное количество элементов. Выходной тип данных имеет такое же числовое представление, что и тип данных на входе и функции обрабатывают каждый элемент кластера или массива. Более подробно о двухходовых полиморфных функциях см. в разделе *Полиморфизм числовых функций* в данном приложении. Допустимы следующие комбинации входных типов данных для двухходовых логарифмических функций:

- **Подобие** (Similar) – Оба входа имеют одинаковую структуру. В этом случае выход имеет ту же структуру, что и входы.
- **Один скаляр** (One scalar) – Один из входов является числовым скаляром, а другой – массивом чисел или кластером чисел. В этом случае выходом будет массив или кластер.

## Функции сравнения

Используйте функции сравнения (с палитры **Comparison**) для сравнения булевых значений, символьных строк, чисел, массивов и кластеров. Большинство функций сравнения анализируют один вход или сравнивают два входа и возвращают на выходе результат в виде булева значения.

---

### Более подробно...

Более подробно относительно функций сравнения см. справочную систему *LabVIEW Help*.

---

### Сравнение булевых значений

---

Функции сравнения считают булево значение TRUE больше, чем булево значение FALSE.

### Сравнение символьных строк

---

LabVIEW сравнивает символьные строки, основываясь на числовом эквиваленте ASCII кодов символов. Например, а (десятичный эквивалент 97) больше, чем А (десятичный эквивалент 65), что больше цифрового символа 0 (48), который, в свою очередь, больше символа пробела (32). LabVIEW сравнивает символы один за другим, начиная с начала строки и до тех пор, пока не встретятся неравные символы. На этом сравнение завершается. Например, LabVIEW анализирует символьные строки abcd и abef, пока не дойдет до символа с, который меньше, чем символ е. Наличие символа считается большим, чем пустое место (отсутствие символа). Таким образом, символьная строка abcd больше символьной строки abc, поскольку первая строка длиннее.

## Сравнение чисел

---

Функции сравнения конвертируют входные числовые значения к одинаковому представлению перед их сравнением. Если на один или на оба входа функции сравнения поступает значение NaN (не число), то возвращается значение, означающее неравенство. Более подробно о значении NaN см. в разделе *Неопределенные или неожиданные данные* в Главе 6 *Запуск и отладка виртуальных приборов*.

## Сравнение массивов и кластеров

---

Некоторые функции сравнения имеют два режима для сравнения массивов или кластеров данных. В режиме сравнения агрегатов (Compare Aggregates mode) при сравнении двух массивов или кластеров возвращается одно единственное булево значение. В режиме сравнения элементов (Compare Elements mode) функция сравнивает элементы отдельно и возвращает массив или кластер булевых значений.

В режиме сравнения агрегатов операция сравнения символьных строк и операция сравнения массивов выполняются абсолютно одинаково, при этом функция сравнения рассматривает строку как массив ASCII символов.

Чтобы изменить режим сравнения, щелкните правой кнопкой функцию сравнения и выберите из контекстного меню пункт **Comparison Mode**»**Compare Elements** или пункт **Comparison Mode**»**Compare Aggregates**. Некоторые функции сравнения работают только в режиме сравнения агрегатов, потому в их контекстном меню эти пункты отсутствуют.

## Массивы

При сравнении многомерных массивов присоединенные к функции сравнения массивы должны иметь одинаковое число измерений. Функции сравнения, работающие только в режиме сравнения агрегатов, сравнивают массивы тем же способом, которым они сравнивают символьные строки – по одному элементу, начиная с первого элемента, до тех пор, пока не встретятся неравные элементы.

### **Режим сравнения элементов**

В режиме сравнения элементов (Compare Elements mode) функции сравнения возвращают массив булевых значений той же размерности, что и входные массивы. Каждое измерение выходного массива имеет размер, равный наименьшему размеру того же измерения из двух входных массивов. Проходя все измерения, такие как строки, столбцы или страницы, функции сравнивают значения соответствующих элементов входных массивов, чтобы получить булево значение соответствующего элемента выходного массива.

### **Режим сравнения агрегатов**

В режиме сравнения агрегатов (Compare Aggregates mode) функции сравнения возвращают одно единственное булево значение-результат после сравнения элементов массива. Функции сравнения рассматривают последующие элементы массива только после того, как рассмотрят все предыдущие элементы. Таким образом, для получения результата сравнения выполняются следующие шаги:

- Сравниваются соответствующие элементы входных массивов, начиная с первых элементов.
- Если соответствующие элементы *не* равны, то процесс останавливается и функция возвращает результат этого сравнения.
- Если соответствующие элементы равны, то обрабатывается следующая пара значений, пока не будет найдено неравенство или пока не будет достигнут конец одного из входных массивов.
- Если все значения просмотренных элементов входных массивов равны, но в одном из массивов в конце остаются нерассмотренные элементы, то более длинный массив считается больше короткого. Например, функция сравнения считает массив [1, 2, 3, 2] большим, чем массив [1, 2, 3].

### **Кластеры**

Кластеры, которые Вы сравниваете, должны состоять из одного и того же числа элементов, все элементы кластеров должны быть сравнимых типов и должны располагаться в кластерах в одном и том же порядке. Например, Вы можете сравнить кластер, состоящий из числа типа DBL (удвоенная точность, плавающая точка) и строки, с кластером, состоящим из числа типа I32 (32-битное целое) и строки.

### **Режим сравнения элементов**

В режиме сравнения элементов (Compare Elements mode) функции сравнения возвращают кластер из булевых элементов, по одному на каждый соответствующий элемент во входных кластерах.

### **Режим сравнения агрегатов**

В режиме сравнения агрегатов (Compare Aggregates mode) функции сравнения возвращают одно единственное булево значение-результат. В этом режиме функция сравнивает соответствующие элементы до тех пор, пока не обнаружит их неравенство, что и определяет конечный результат. Функция считает два кластера равными, только если все их элементы равны между собой.

Используйте для кластеров режим сравнения агрегатов, если Вы сравниваете две записи, содержащие данные, отсортированные таким образом, что элементы кластера, стоящие в кластере дальше, рассматриваются после элементов, стоящих в кластере раньше. Например, если Вы сравниваете кластер, содержащий две символьные строки, первая из которых содержит поле фамилии, а вторая – поле имени, то функция должна будет сравнивать поля имени в исходных кластерах только в том случае, если поля фамилии в них совпадают.

## **Техническая поддержка и профессиональные услуги**

При необходимости технической поддержки и профессиональных услуг посетите следующие разделы Web сайта [ni.com](http://ni.com) компании National Instruments:

- **Support** (поддержка) – Интерактивные ресурсы технической поддержки включают:
  - **Self-Help Resources** (ресурсы самостоятельной помощи) – Для получения немедленных ответов и решений посетите нашу обширную библиотеку ресурсов технической поддержки, на английском, японском и испанском языках на сайте [ni.com/support](http://ni.com/support). Эти ресурсы имеются для большинства продуктов и доступны бесплатно для зарегистрированных пользователей. Они включают базу знаний, руководства, мастера пошаговой диагностики, документацию соответствия, коды примеров, руководства по изучению, замечания к приложениям, приборные драйверы, дискуссионные форумы, гlosсарий по измерениям и т.п.
- **Training** (обучение) – Посетите сайт [ni.com/custed](http://ni.com/custed), чтобы получить доступ к самоучителям, видеоматериалам и интерактивным средствам на компакт дисках. National Instruments предлагает различные учебные курсы и сертификационные экзамены для проверки ваших знаний и квалификации специалиста.

Программы учебных курсов разработаны на основе 15-летнего опыта обучения пользователей при участии инженеров-разработчиков NI и имеют практическую направленность. Программы всех курсов имеют модульную структуру, так что вы можете выбрать наиболее подходящие курсы для вашей области деятельности.

Обучение проводится на оборудовании National Instruments в учебном классе компании или на территории заказчика. Курсы имеют проходят под руководством опытных преподавателей, всегда готовых ответить на специфические для вашего приложения вопросы. В ходе проведения курсов слушателям предоставляются методические пособия и учебники на русском языке, по окончании выдаются сертификаты международного образца.

Подробнее о курсах на [ni.com/russia](http://ni.com/russia), раздел Курсы

Курсы на русском языке:

Название курса	Длительность
LabVIEW: Вводный курс	3 часа
LabVIEW: Основы I	3 дня
LabVIEW: Основы II	2 дня
Системы сбора данных	3 дня
LabWindows/CVI: Основы I	3 дня
LabWindows/CVI: Основы II	2 дня
TestStand: Основы I	3 дня
LabVIEW FPGA	3 дня
LabVIEW Real-Time	3 дня

- **System Integration** (системная интеграция) – Если у вас мало времени, имеются ограничения на ваши технические ресурсы или имеются какие-либо проблемы в реализации вашего проекта, вам могут помочь другие члены альянса NI Alliance Program.

На территории России, СНГ и Балтии работают системные интеграторы National Instruments (NI Alliance members) – инженерные фирмы ведущие разработку и сдачу проектов под ключ. За годы работы в России были разработаны и сданы в эксплуатацию системы стендовых испытаний двигателей, стенды структурных испытаний корпусов, крупные системы вибродиагностики и акустических тестов, системы тестирования авионики и многие другие системы.

Информацию о системных интеграторах National Instruments вы можете найти, посетив раздел О National Instruments/ Системные интеграторы на сайте [ni.com/russia](http://ni.com/russia)

Если вы искали помощи на **ni.com** и не нашли ответа, обратитесь **за бесплатной технической поддержкой** в офис National Instruments:

**National Instruments Россия, СНГ, Балтия**

119361 г. Москва, ул. Озерная, д.42 офис 1101

**Телефон в Москве:** + 7(495) 783-68-51

**Телефон в Санкт-Петербурге:** + 7 (812) 951-44-18

**Телефон в Киеве:** + 38 (068) 394-21-22

**Телефон в Риге:** + 371 (22) 38-87-86

Электронная почта: [support.russia@ni.com](mailto:support.russia@ni.com)

**ni.com/russia**      **www.labview.ru**