

**ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ «ОБРАЗОВАНИЕ»
РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

П.М. МИХЕЕВ, Д.В. ЧУПРОВ, В.В. АНДРЕЕВ

**СОВРЕМЕННЫЕ ГРАФИЧЕСКИЕ СРЕДЫ
ПРОГРАММИРОВАНИЯ**

Учебное пособие

Москва

2008

*Инновационная образовательная программа
Российского университета дружбы народов*

**«Создание комплекса инновационных образовательных программ
и формирование инновационной образовательной среды,
позволяющих эффективно реализовывать государственные интересы РФ
через систему экспорта образовательных услуг»**

Экспертное заключение –
заведующий кафедрой теоретических основ радиотехники,
доктор технических наук, профессор *В.П. Федосов*
(Таганрогский государственный радиотехнический университет)

П.М. Михеев, Д.В. Чупров, В.В. Андреев

Современные графические среды программирования: Учеб. пособие. –
М.: РУДН, 2008. – 213 с.: ил

Содержание пособия направлено на обеспечение базовой подготовки в области использования среды графического программирования LabVIEW; введение в теорию и методику современного сбора данных; получение практических навыков в области современных методов сбора и обработки экспериментальных данных с использованием новейших цифровых технологий; приобретение студентами базовых знаний в области автоматизации физического эксперимента.

Предназначено для студентов третьего курса бакалавриата, обладающих базовыми знаниями в области программирования и построения вычислительных алгоритмов.

Учебное пособие выполнено в рамках инновационной образовательной программы Российского университета дружбы народов, направление «Комплекс экспортоориентированных инновационных образовательных программ по приоритетным направлениям науки и технологий», и входит в состав учебно-методического комплекса, включающего описание курса, программу и электронный учебник.

© Михеев П.М., Чупров Д.В., Андреев В.В., 2008

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ.....	5
1. Общие сведения о LabVIEW	6
1.1. Введение в LabVIEW	6
1.1.1. Программная среда LabVIEW	6
1.1.2. Виртуальные приборы (ВП).....	6
1.1.3. Последовательность обработки данных	11
1.1.4. Организация программной среды LabVIEW	13
1.1.5. Встроенная помощь среды LabVIEW и руководство пользователя.....	27
1.2. Виртуальные приборы (ВП).....	28
1.2.1. Компоненты ВП	28
1.2.2. Создание ВП	32
1.2.3. Типы и проводники данных	33
1.2.4. Редактирование ВП.....	36
1.3. Создание подпрограмм ВП	45
1.3.1. Подпрограммы ВП	45
1.3.2. Создание иконки ВП и настройка соединительной панели.....	45
1.3.3. Использование подпрограмм ВП.....	51
1.3.4. Преобразование экспресс-ВП в подпрограмму ВП	53
1.3.5. Превращение выделенной секции блок-диаграммы в подпрограмму ВП	54
2. Инструменты для построения алгоритмов.....	56
2.1. Многократные повторения и Циклы	56
2.1.1. Цикл While (по Условию).....	56
2.1.2. Цикл For (с фиксированным числом итераций).....	60
2.1.3. Организация доступа к значениям предыдущих итераций цикла.....	61
2.2. Принятие решений в ВП и структуры.....	65
2.2.1. Функция Select и принятие решений.....	65
2.2.2. Структура Case	66
2.2.3. Узел Формулы	70
2.2.4. Узел Математики (MathScript Node)	72
3. Группирование данных и графическое отображение	74
3.1. Массивы	74
3.1.1. Создание массивов с помощью цикла.....	76
3.1.2. Функции работы с массивами	78
3.1.3. Полиморфизм	80
3.2. Кластеры	81
3.2.1. Что такое кластер	81
3.2.2. Функции работы с кластерами.....	84
3.2.3. Кластеры ошибок	88
3.3. Графическое отображение данных.....	92
3.3.1. Использование графика Диаграмм.....	92
3.3.2. График Осциллограмм и двухкоординатный график Осциллограмм	94
3.3.3. График интенсивности.....	97
4. Работа со строковыми данными и файлами	100
4.1. Строки. Функции работы со строками.....	100
4.1.1. Создание строковых элементов управления и отображения данных	100
4.1.2. Использование некоторых функций обработки строк	103

4.1.3. Преобразование числовых данных в строку	104
4.2. Функции файлового ввода/вывода	108
4.3. Форматирование строк таблицы символов.....	114
4.4. Функций файлового ввода/вывода высокого уровня	116
5. Настройка ВП.....	119
5.1. Настройка внешнего вида лицевой панели	119
5.2. Отображение лицевых панелей подпрограмм ВП во время работы.....	122
5.3. Назначение и использование «горячих» клавиш	123
5.4. «Нередактируемые» ВП	124
6. Сбор данных и управление в LabVIEW	126
6.1. Конфигурация системы сбора данных	126
6.2. Сбор данных в LabVIEW	135
6.2.1. Выполнение операций аналогового ввода.....	136
6.2.2. Запись полученных данных в файл	139
6.2.3. Выполнение операций аналогового вывода	140
6.2.4. Информация о счетчиках.....	143
6.2.5. Ввод и вывод цифровых сигналов.....	144
7. Работа с измерительным оборудованием	146
7.1. Управление измерительными приборами.....	146
7.1.1. Управление в LabVIEW измерительными приборами.....	146
7.1.2. Использование Instrument I/O Assistant.....	146
7.1.3. Архитектура программного обеспечения виртуальных интерфейсов.....	149
7.1.4. Драйверы измерительных приборов	152
7.1.5. Использование ВП драйвера устройства	153
7.2. Работа с GPIB приборами.....	157
7.2.1. Адресация в интерфейсе GPIB.....	158
7.2.2. Остановка передачи данных.....	159
7.2.3. Ограничения	159
7.2.4. Архитектура программных средств	160
7.2.5. Программные средства настройки	161
7.3. Работа с RS-232 приборами.....	163
7.3.1. Последовательная связь.....	163
7.3.2. Настройка последовательного порта.....	164
7.3.3. Обзор аппаратных средств	166
7.3.4. Обзор программных средств	170
7.3.5. Передача сигнальных данных.....	171
Список литературы.....	175
ОПИСАНИЕ КУРСА И ПРОГРАММА.....	176

ПРЕДИСЛОВИЕ

Курс предназначен для студентов третьего курса бакалавриата, обладающих базовыми знаниями в области программирования и построения вычислительных алгоритмов. Курс нацелен на обеспечение базовой подготовки в области использования среды графического программирования LabVIEW; введение в теорию и методику современного сбора данных; получение практических навыков в области современных методов получения и обработки экспериментальных данных с использованием новейших цифровых технологий; приобретение студентами базовых знаний в области автоматизации физического эксперимента.

1. Общие сведения о LabVIEW

1.1. Введение в LabVIEW

1.1.1. Программная среда LabVIEW

Программа, написанная в среде LabVIEW, называется виртуальным прибором (ВП). ВП симулируют реальные физические приборы, например осциллограф или мультиметр. LabVIEW содержит полный набор инструментов для сбора, анализа, представления и хранения данных.

В LabVIEW интерфейс пользователя – лицевая панель создается с помощью элементов управления (кнопки, переключатели и др.) и отображения (графики, светодиоды и др.). После этого на блок-диаграмме ВП осуществляется программирование с использованием графических представлений функций для управления объектами на лицевой панели.

LabVIEW используется для программирования различных **DAQ**-устройств, систем контроля изображения и движения, аппаратных средств, имеющих интерфейсы типа **GPIB**, **VXI**, **PXI**, **RS-232** и **RS-485**. LabVIEW имеет встроенные возможности для работы в компьютерных сетях Интернет, используя LabVIEW Web Server и программные стандарты TCP/IP и Active X. С помощью программной среды LabVIEW можно разрабатывать программно-аппаратные комплексы для тестирования, измерения, ввода данных, анализа и управления внешним оборудованием. LabVIEW – это 32-х разрядный компилятор, который создает как автономные модули (**.EXE**), так и совместно используемые динамические библиотеки (**.DLL**).

1.1.2. Виртуальные приборы (ВП)

ВП состоят из четырех основных компонентов – лицевой панели, блок-диаграммы, иконки и соединительной панели.

Лицевая панель – это интерфейс пользователя ВП. Пример лицевой панели представлен ниже (рис. 1.1).

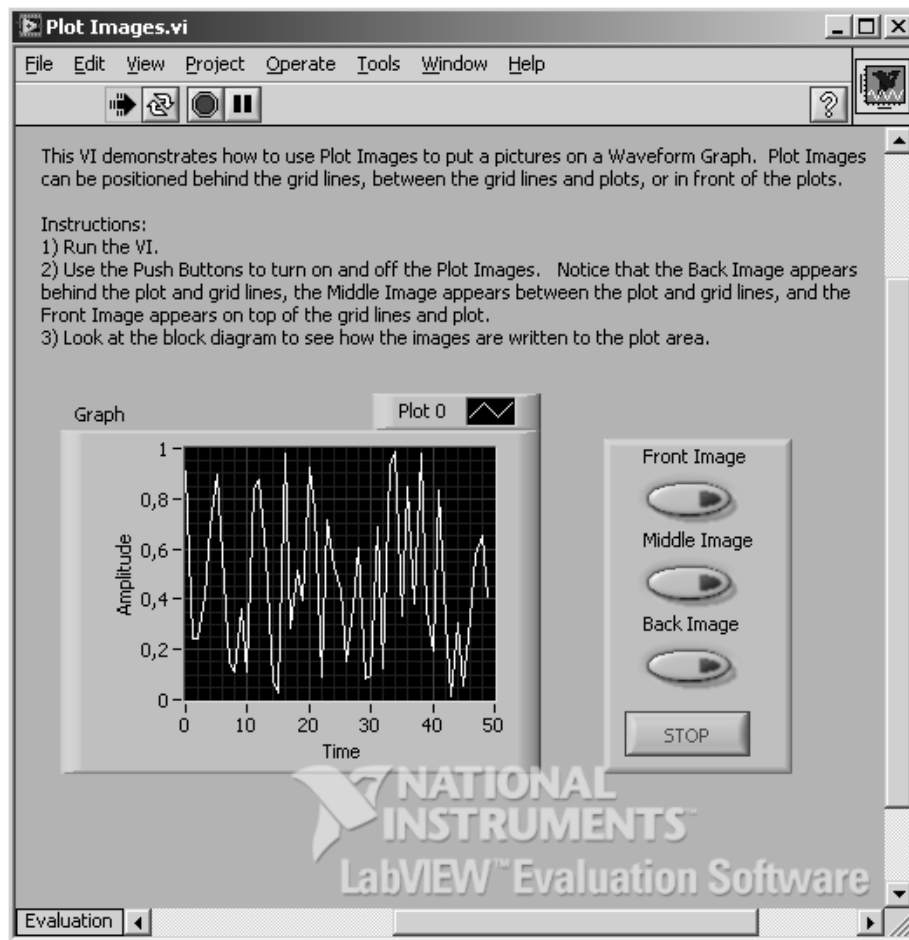


Рис. 1.1. Пример лицевой панели.

Лицевая панель создается с использованием палитры **Элементов (Controls)**. Эти элементы могут быть либо средствами ввода данных – элементы **Управления**, либо средствами отображения данных – элементы **Отображения**. Элементы **Управления** – кнопки, переключатели, ползунки и другие элементы ввода. Элементы **Отображения** – графики, цифровые табло, светодиоды и т.д. Данные, вводимые на лицевой панели ВП, поступают на блок-диаграмму, где ВП производит с ними необходимые операции. Результат вычислений передается на элементы отображения информации на лицевой панели ВП.

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

Следующий пример (рис. 1.2) показывает блок-диаграмму и соответствующую ей лицевую панель:

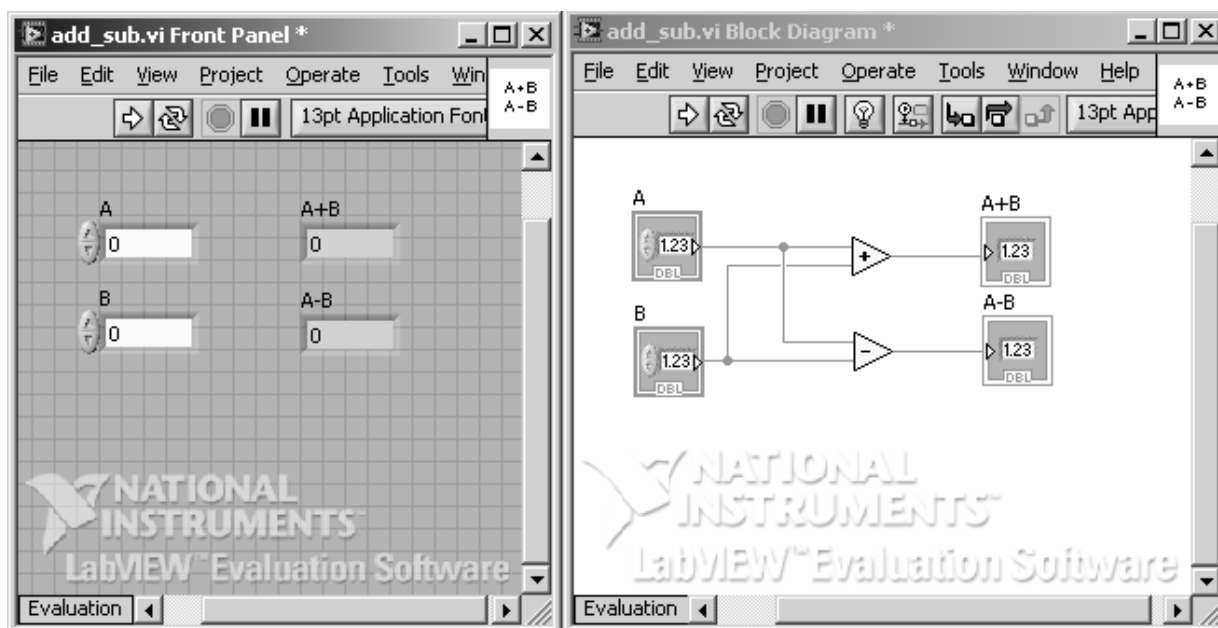


Рис. 1.2. Лицевая панель и соответствующая ей диаграмма.

Для использования созданного виртуального прибора внутри другого ВП в качестве подпрограммы, после создания лицевой панели и блок-диаграммы, необходимо оформить иконку и настроить соединительную панель (область полей ввода/вывода данных). Подпрограмма ВП соответствует подпрограмме в текстовых языках программирования.

Каждый ВП имеет показанную иконку в верхнем правом углу лицевой панели и блок-диаграммы. Иконка – графическое представление ВП. Она может содержать текст и/или рисунок. Если ВП используется в качестве подпрограммы, иконка идентифицирует его на блок-диаграмме другого

ВП. Необходимо также настроить соединительную панель (область полей ввода/вывода данных), чтобы использовать ВП в качестве подпрограммы. Соединительная панель – это набор полей, соответствующий элементам ввода/вывода данных этого ВП. Поля ввода/вывода аналогичны списку параметров вызываемой функции в текстовых языках программирования. Область полей ввода/вывода данных позволяет использовать ВП в качестве подпрограммы. ВП получает данные через поля ввода данных и передает их на блок-диаграмму через элементы **Управления** лицевой панели. Результаты отображаются в его полях вывода данных посредством элементов **Отображения** лицевой панели.

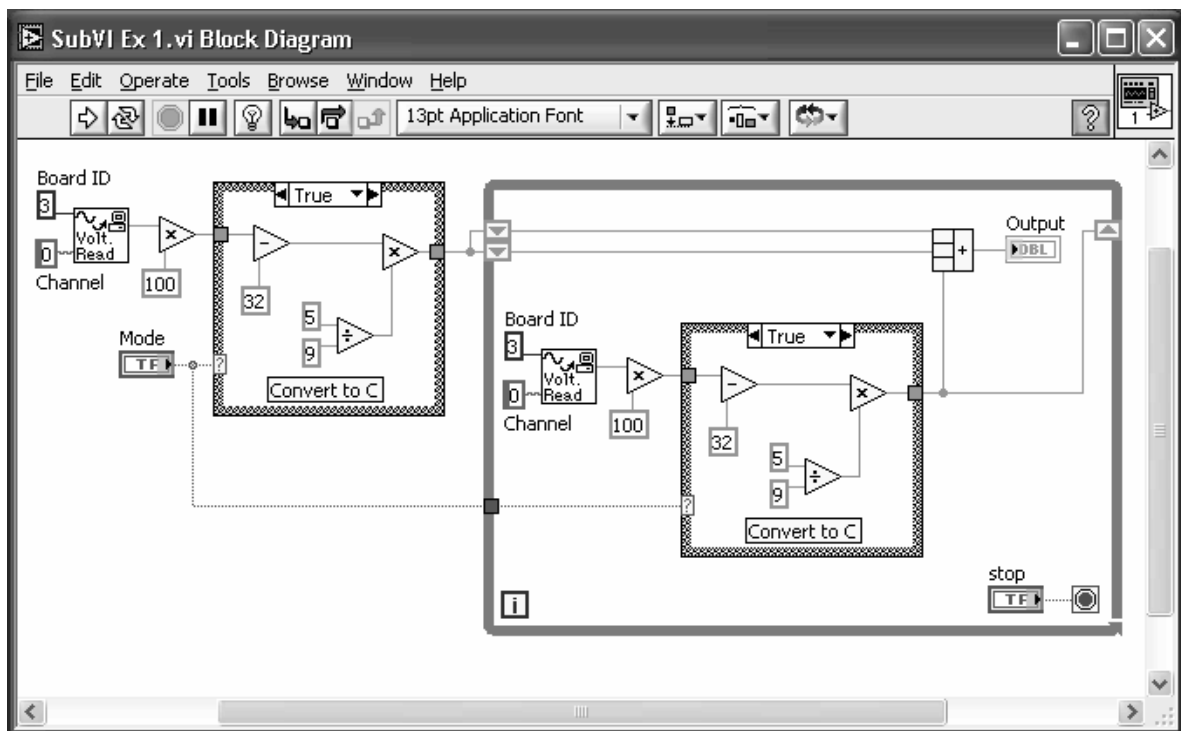


Рис. 1.3. Диаграмма с двумя одинаковыми операциями.

Преимущество LabVIEW заключается в иерархической структуре ВП. Созданный виртуальный прибор можно использовать в качестве подпрограммы на блок-диаграмме ВП более высокого уровня. Количество уровней в иерархии не ограничено. Использование подпрограммы ВП помогает быстро изменять и отлаживать блок-диаграмму. При создании

ВП следует обратить внимание на то, что некоторые операции многократно повторяются. Для выполнения таких операций необходимо использовать подпрограммы ВП или циклы. Раздел 2.1 настоящего пособия содержит дополнительную информацию об использовании циклов. Например, приведенная выше блок-диаграмма (рис. 1.3) содержит две идентичные операции.

Можно создать подпрограмму ВП, которая выполнит эту операцию, и можно вызвать эту подпрограмму дважды. Возможно многократное использование подпрограммы ВП в другом виртуальном приборе. Следующий пример (рис. 1.4) демонстрирует использование **Temperature VI** в качестве подпрограммы на блок-диаграмме.

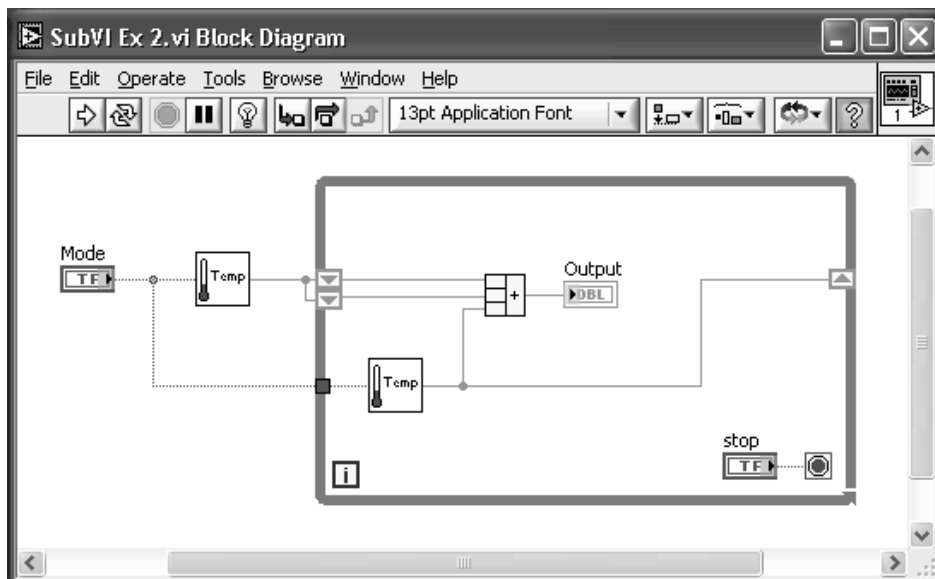


Рис. 1.4. Замена повторяющейся операции иконкой подпрограммы ВП.

Важной функциональной особенностью пакета LabVIEW является автосохранение. При неправильном выключении или при повреждении системы LabVIEW производит сохранение во временный файл открытых файлов с расширениями (.vi), (.vit), (.ctl), (.ctt). LabVIEW *не сохраняет* проекты (.lvproj), библиотеки проектов (.lvlib), XControls (.xctl), или классы (.lvclass).

Если LabVIEW удалось сохранить файлы перед тем, как произошло незапланированное выключение или ошибка системы, то при следующем запуске LabVIEW появится окно **Select Files to Recover**. Выберите файлы, которые вы хотите перезаписать, и нажмите кнопку **Recover**. Если вы не хотите перезаписывать файлы, то, ничего не выбирая, нажмите кнопку **Discard**. При нажатии кнопки **Cancel** выбранные файлы будут помещены в папку *LVAutoSave\archives*. Чтобы настроить функцию автосохранения в LabVIEW, выберите на линейке инструментов в верхней части окна пункт **Tools»Options**, затем из списка **Category**, находящегося в левой части окна выберите пункт **Environment**. После этого вы сможете включить или отключить функцию автосохранения, а также установить интервал времени, через который происходит автосохранение.

1.1.3. Последовательность обработки данных

В Visual Basic, C++, Java и большинстве других текстовых языков программирования порядок выполнения всей программы определяется расположением функций программы. В среде LabVIEW используется потоковая модель обработки данных. Узлы блок-диаграммы выполняют заложенные в них функции, если данные поступили на все необходимые поля ввода/вывода. По окончании выполнения операции одним узлом результаты операции по проводникам данных передаются следующему узлу и т.д. Другими словами, готовность входных данных определяет последовательность выполнения узлов блок-диаграммы.

В качестве примера можно рассмотреть блок-диаграмму (рис. 1.5, а), которая складывает два числа и затем вычитает из получившейся суммы константу «50.0». В этом случае блок-диаграмма выполняется слева направо не потому, что объекты помещены в этом порядке, а потому, что одно из полей ввода функции **Subtract** (Вычитание) не определено, пока

не выполнялась функция **Add** (Сложение) и не передала данные к функции **Subtract** (Вычитание). Не следует забывать, что узел выполняется только тогда, когда определены его поля ввода данных.

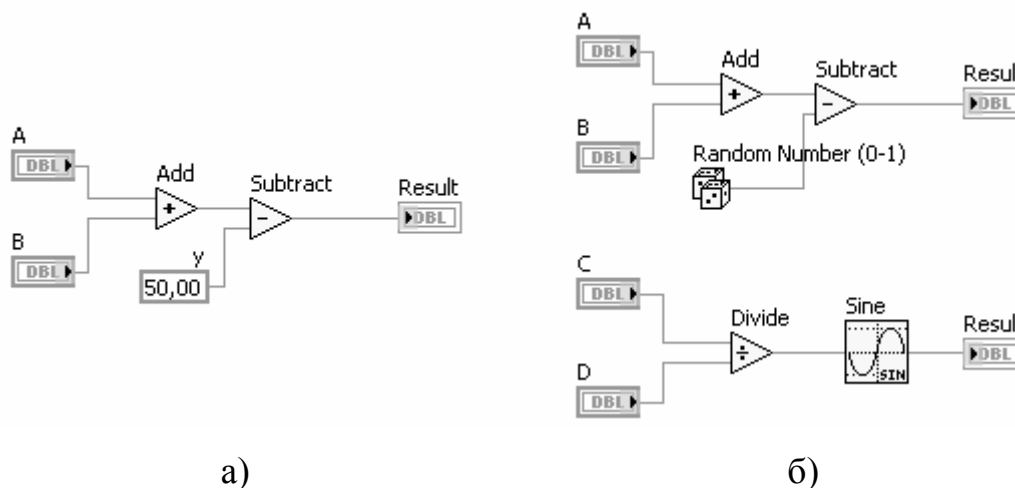


Рис. 1.5. Последовательность выполнения операций на диаграмме:

- а) очередность выполнения узлов задается поступающими на вход данными;
- б) пример, когда последовательность узлов выполнения диаграммы не определена.

В следующем примере (рис. 1.5, б) рассмотрена последовательность выполнения функций **Add** (Сложение), **Random Number** (Генератор случайных чисел) и **Divide** (Деление). Как видно, в данном случае последовательность выполнения функций не определена, так как поля ввода данных функций **Add** (Сложение) и **Divide** (Деление) инициализируются одновременно, а **Random Number** (Генератор случайных чисел) не имеет полей ввода данных. В случае, когда необходимо выполнить одну часть кода блок-диаграммы раньше другой, а зависимости данных между функциями нет, для установки порядка выполнения следует использовать методы программирования.

1.1.4. Организация программной среды LabVIEW

При запуске LabVIEW появляется окно запуска для работы с системой (рис. 1.6), с помощью которого можно создать новый ВП, проект, открыть уже существующий ВП, найти примеры или получить доступ к подсказке *LabVIEW Help*. Окно запуска появляется также при закрытии всех лицевых панелей и блок-диаграмм.

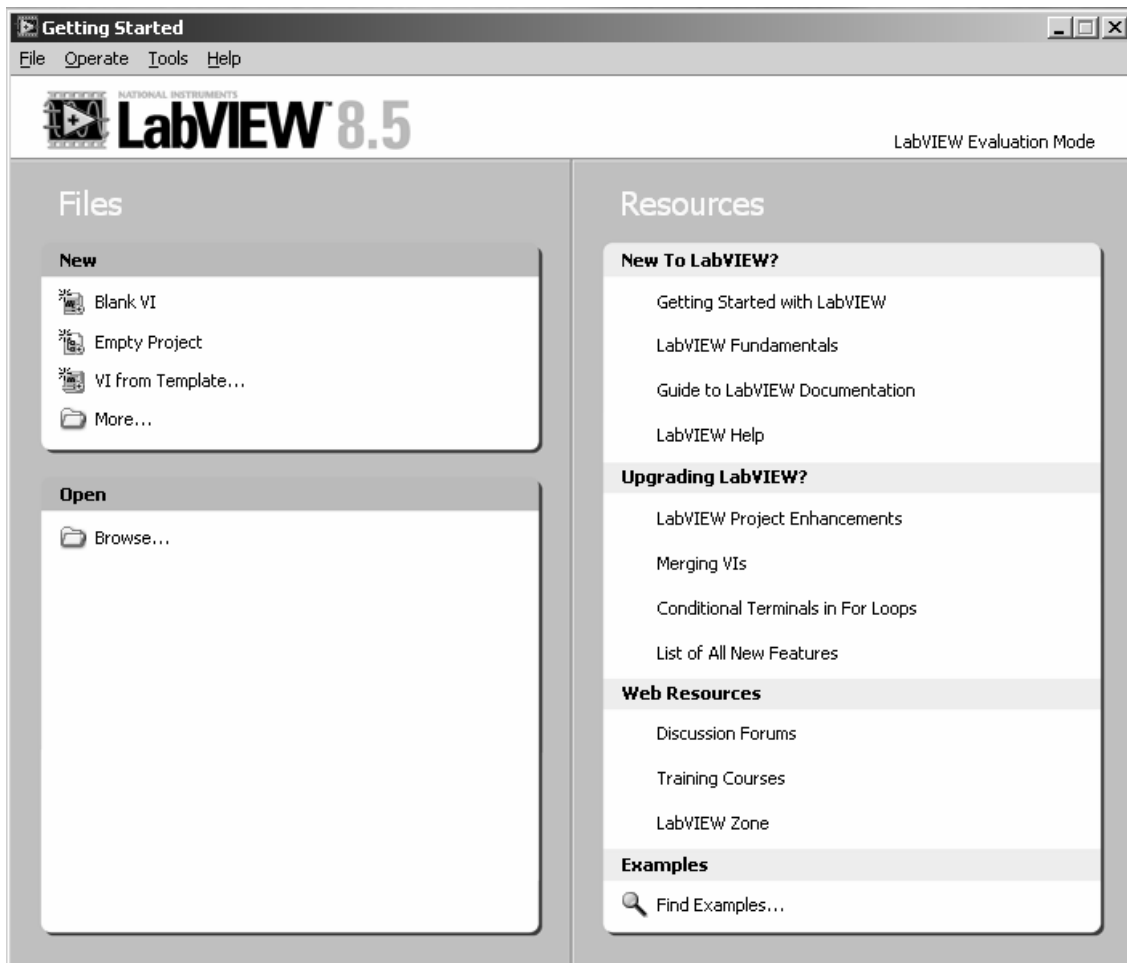


Рис. 1.6. Стартовое окно при запуске LabVIEW.

Окно запуска содержит следующие компоненты:

- Панель меню со стандартными пунктами, например, **File»Exit**
- Окно **Files**, позволяющее открыть или создать ВП. В данном окне находятся вкладки **New** и **Open**. Используя вкладку **New**, вы можете

создать новый ВП, новый проект или загрузить шаблон ВП. Используя вкладку **Open**, можно открыть созданный ранее ВП.

- Окно **Resources** позволяет обратиться за помощью или дополнительной информацией к различным текстовым и Internet-ресурсам, а также посмотреть встроенные примеры.

Создание нового или открытие уже существующего ВП

После нажатия на строку **Blank VI** во вкладке **New** (рис.1.6), на экране появится лицевая панель и блок-диаграмма пустого ВП.

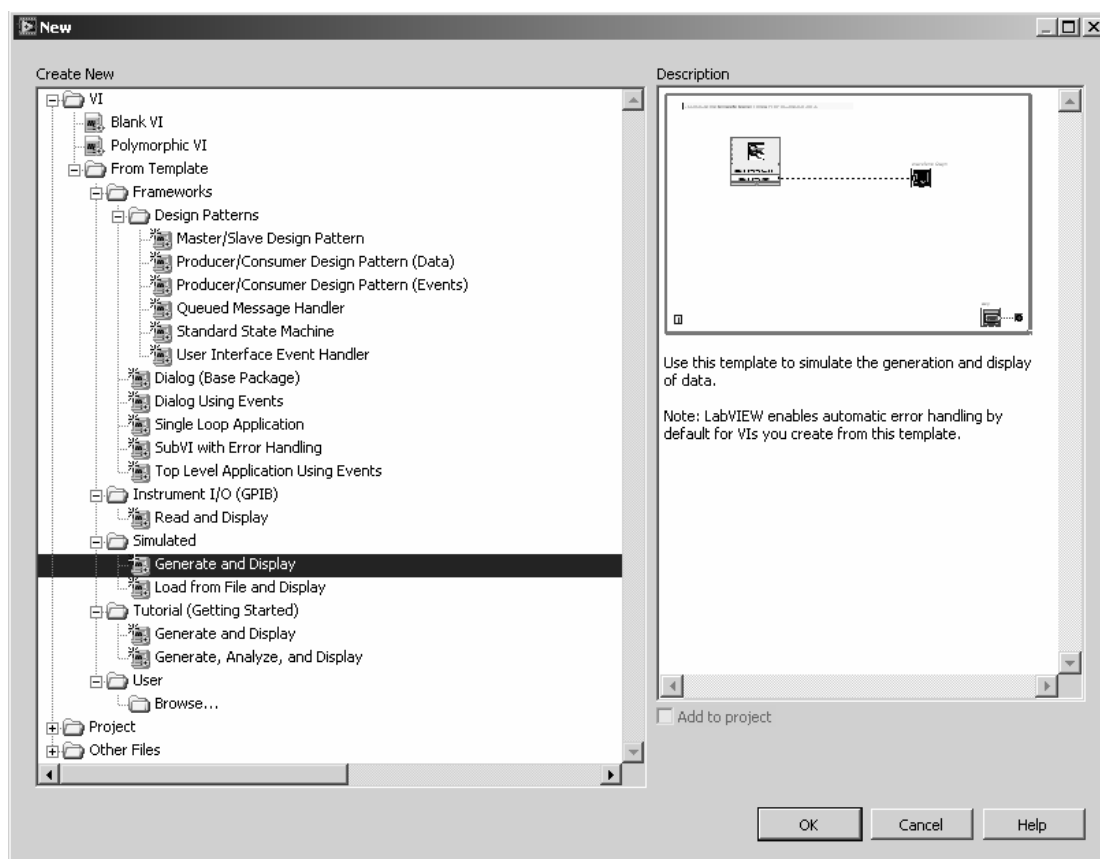


Рис. 1.7. Диалоговое окно выбора шаблона ВП.

Чтобы открыть шаблон ВП, нажмите левой кнопкой мыши во вкладке **New** на строку **VI from Template**. После этого на экране появится диалоговое окно **New** (рис. 1.7). После выбора шаблона из спискового окна **Create New** (создание нового ВП) в секции **Description** отобразится

описание шаблона и блок-диаграмма ВП. Для создания ВП нажмите кнопку **ОК**.

Лицевая панель и окно блок-диаграммы

После нажатия кнопки **Blank VI** появляется окно лицевой панели. Это одно из двух окон LabVIEW, используемых для создания ВП. Другое окно содержит блок-диаграмму.

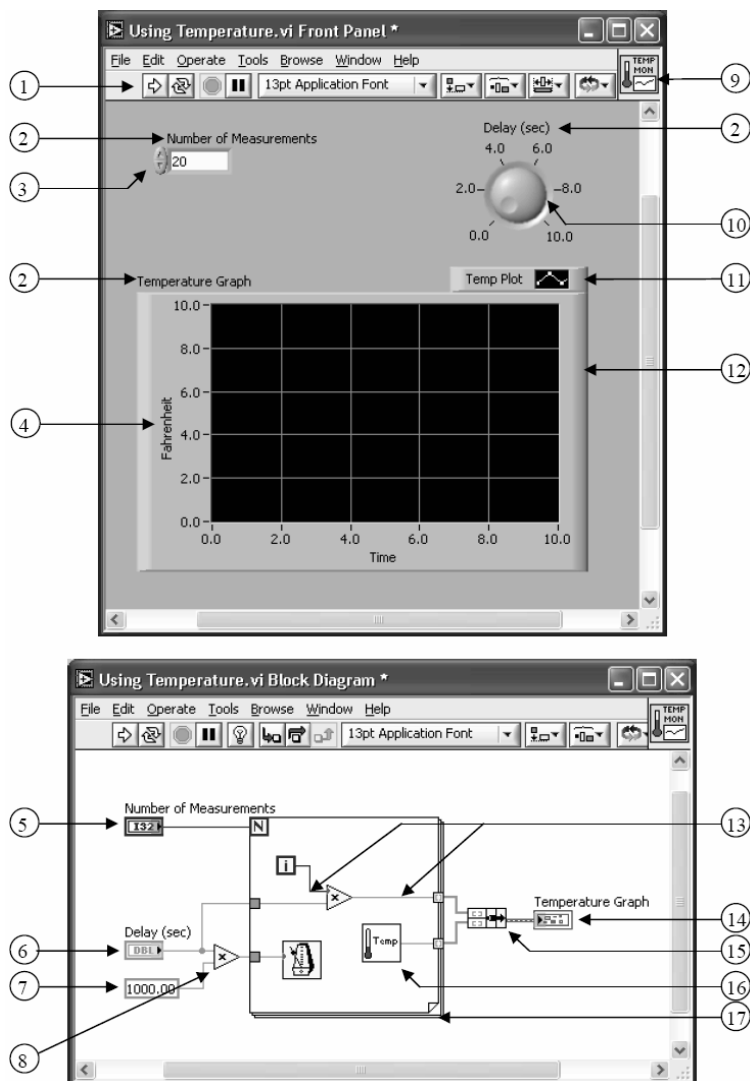


Рис. 1.8. Основные компоненты лицевой панели и блок-диаграммы.

На задний план лицевой панели виртуального прибора вы можете импортировать какое-либо изображение. Для этого нажмите правой кнопкой мыши на полосе прокрутки лицевой панели и из контекстного меню выберите **Properties**. Затем в диалоговом окне **Pane Properties**

выберите изображение из списка **Background**. LabVIEW поддерживает форматы BMP, JPEG и PNG. Рисунок 1.8 демонстрирует лицевую панель и соответствующую ей блок-диаграмму.

На рисунке 1.8. цифрами обозначены:

1. Инструментальная панель.
2. Собственная метка.
3. Цифровой элемент управления.
4. Свободная метка.
5. Терминал данных цифрового элемента управления.
6. Терминал данных кнопки.
7. Числовая константа.
8. Функция Умножение.
9. Иконка ВП.
10. Ручка управления.
11. Панель управления графиком.
12. Двухкоординатный график осциллограмм.
13. Проводники данных.
14. Терминал данных графика осциллограмм.
15. Функция Объединение в структуру (Bundle).
16. Иконка подпрограммы ВП.
17. Цикл с фиксированным числом итераций (For).

Инструментальная панель лицевой панели

Инструментальная панель используется для запуска и редактирования ВП. Пример инструментальной панели показан на рисунке 1.9.

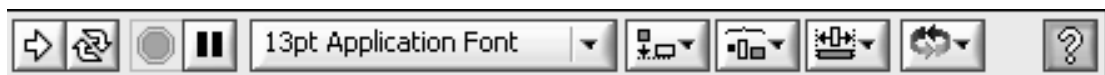















Рис. 1.9. Инструментальная панель лицевой панели ВП.

Таблица 1.1. Назначение кнопок лицевой панели.

	Кнопка запуска Run – запускает ВП.
	Во время работы ВП кнопка Run меняет свой вид, как показано слева, если этот виртуальный прибор высокого уровня.
	Если ВП работает в качестве подпрограммы, то так выглядит кнопка Run .
	Кнопка Run выглядит в виде «сломанной» стрелки, как показано слева, во время создания или редактирования ВП. В таком виде кнопка показывает, что ВП не может быть запущен на выполнение. После нажатия этой кнопки появляется окно Error list , в котором перечислены допущенные ошибки.
	Кнопка непрерывного запуска Run Continuously – ВП выполняется до момента принудительной остановки.
	Во время выполнения ВП появляется кнопка Abort Execution . Эта кнопка используется для немедленной остановки выполнения ВП. (По возможности следует избегать использования кнопки Abort Execution для остановки ВП. Следует позволить ВП закончить передачу данных или выполнить остановку программным способом, гарантируя остановку ВП в определенном состоянии. Например, можно установить на лицевой панели кнопку, по нажатию которой ВП останавливается.)
	Кнопка Pause приостанавливает выполнение ВП. После нажатия кнопки Pause LabVIEW подсвечивает на блок-диаграмме место остановки выполнения. Повторное нажатие – продолжение работы ВП.
	Text Settings – выпадающее меню установок текста, включая размер, стиль и цвет.
	В меню Align Objects производится выравнивание объектов по осям (по вертикали, по осям и т.д.).

	<p>В меню Distribute Objects производится выравнивание объектов в пространстве (промежутки, сжатие и т.д.).</p>
	<p>В меню Resize Objects производится приведение к одному размеру многократно используемых объектов лицевой панели.</p>
	<p>Меню Reorder используется при работе с несколькими объектами, которые накладываются друг на друга. Выделив один из объектов с помощью инструмента ПЕРЕМЕЩЕНИЕ, в меню Reorder следует выбрать его порядок отображения на лицевой панели.</p>
	<p>Кнопка Context Help выводит на экран окно Context Help (контекстной справки)</p>

Инструментальная панель блок-диаграммы







При запуске ВП на блок-диаграмме появляется инструментальная панель, показанная на рисунке 1.10.



Рис. 1.10. Инструментальная панель блок-диаграммы ВП.

Назначение кнопок этой панели объясняется в таблице 1.2. Кнопки, имеющие тот же вид, что и кнопки лицевой панели, выполняют идентичные функции, поэтому в таблице не указаны.

Таблица 1.2. Назначение кнопок инструментальной панели.

	Кнопка Highlight Execution предназначена для просмотра потока данных через блок-диаграмму (режим отладки). Повторное нажатие кнопки отключает этот режим.
	Кнопка Retain Wire Values предназначена для сохранения данных прошедших по проводникам. Включив ее, можно посмотреть значения данных в любом проводнике ВП в любой момент времени.
	Кнопка Step Into используется при пошаговом выполнении цикла от узла к узлу, подпрограммы ВП и т.д. При этом узел мигает, обозначая готовность к выполнению.
	Кнопка Step Over позволяет пропустить в пошаговом режиме цикл, подпрограмму и т.д.
	Кнопка Step Out позволяет выйти из цикла, подпрограммы и т.д. Выход из узла предполагает завершение выполнения этого узла в пошаговом режиме и переход в следующий.
	Кнопка Warning появляется, когда есть потенциальная проблема с блок-диаграммой, но она не запрещает выполнение ВП. Кнопку Warning можно активизировать, войдя в пункт главного меню Инструменты , далее: Опции , Отладка (Tools » Options » Debugging) .

Контекстное меню.

Контекстное меню используется наиболее часто. Все объекты LabVIEW, свободное рабочее пространство лицевой панели и блок-диаграммы имеют свои контекстные меню. Контекстное меню используется для изменения поведения объектов блок-диаграммы и лицевой панели. Контекстное меню вызывается щелчком правой кнопкой мыши на объекте, лицевой панели или блок-диаграмме. Пример контекстного меню показан на рисунке 1.11.

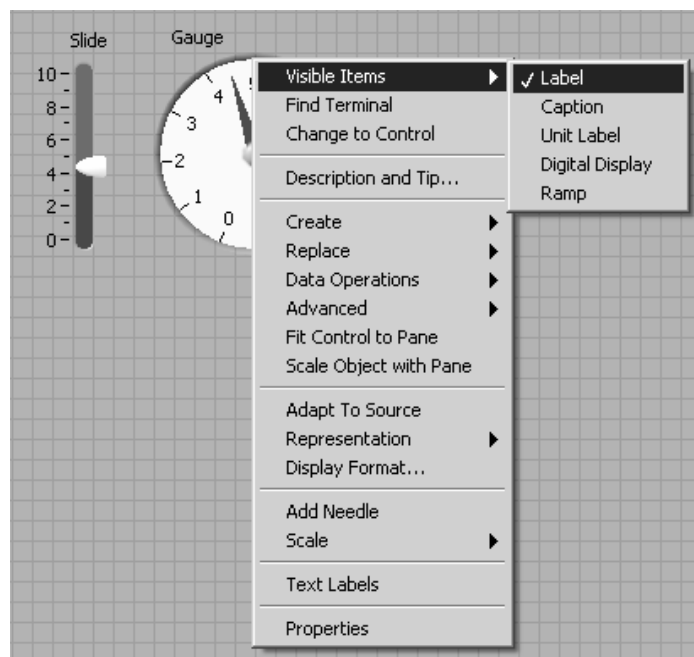


Рис. 1.11. Пример контекстного меню.

Главное меню

Главное меню в верхней части окна ВП содержит пункты, общие с другими приложениями, такие как **Open**, **Save**, **Copy**, **Paste**, а также специфические пункты меню LabVIEW. Некоторые пункты главного меню содержат сведения о «горячих» клавишах вызова этих пунктов. Следует обратить внимание на то, что во время выполнения ВП некоторые пункты главного меню недоступны.

Пункт меню **File** используется для открытия, закрытия, сохранения и печати ВП.

Пункт меню **Edit** используется для поиска и внесения изменений в компоненты ВП.

Пункт меню **View** используется для отображения различных палитр, иерархии ВП и открытия различных окон, позволяющих работать в LabVIEW.

Пункт меню **Project** используется для работы с файловой системой проекта. Пункт меню **Operate** используется для запуска, прерывания выполнения и изменения других опций ВП.

Пункт меню **Tools** используется для связи с приборами и DAQ-устройствами, сравнения ВП, формирования приложений и конфигурации LabVIEW.

Пункт меню **Window** используется для отображения окон LabVIEW и палитр.

Пункт меню **Help** используется для получения информации о палитрах, меню, инструментах, ВП и функциях, для получения пошаговой инструкции использования LabVIEW и информации о компьютерной памяти.

Помимо главного и контекстного меню LabVIEW имеет три вспомогательные палитры, используемые для создания и выполнения ВП: **Tools Palette** (Палитра Инструментов), **Controls Palette** (Палитра Элементов) и **Functions Palette** (Палитра Функций). Эти палитры можно поместить в любом месте экрана.







Палитра инструментов




Создавать, редактировать и отлаживать ВП можно с помощью **Tools Palette** (Палитры Инструментов). Палитра Инструментов доступна как на лицевой панели, так и на блок-диаграмме. Термин «инструмент» подразумевает специальный операционный режим курсора мыши. При выборе определенного инструмента значок курсора изменяется на значок данного инструмента. Палитра Инструментов доступна через пункт главного меню **Window»Show Tools Palette**. Палитру Инструментов можно размещать в любой области рабочего пространства блок-диаграммы и лицевой панели. Удерживая нажатой клавишу **<Shift>** и щелкнув правой клавишей мыши, можно вывести на экран временную версию **Tools Palette** (Палитры Инструментов), которая будет автоматически убираться с экрана после ее однократного использования.



Рис. 1.12. Палитра инструментов.

Таблица 1.3. Назначение кнопок палитры инструментов.

	<p>Если включен автоматический выбор инструмента, то при наведении курсора на объект лицевой панели или блок-диаграммы LabVIEW автоматически выбирает соответствующий инструмент из палитры Tools (Инструментов). Автоматический выбор инструментов включается нажатием на кнопку Automatic Tool Selection палитры Tools (Инструментов) или нажатием клавиш <Shift-Tab>.</p>
 	<p>Инструмент УПРАВЛЕНИЕ используется для изменения значения элементов управления или ввода текста. При наведении курсора на такой элемент, как строковый элемент управления, значок инструмента меняется, как показано слева.</p>
 	<p>Инструмент ПЕРЕМЕЩЕНИЕ используется для выбора, перемещения или изменения размеров объектов. При наведении инструмента на объект изменяемого размера значок инструмента меняется, как показано слева.</p>
 	<p>Инструмент ВВОД ТЕКСТА используется для редактирования текста и создания свободных меток. При создании свободных меток значок инструмента меняется, как показано слева.</p>
	<p>Инструмент СОЕДИНЕНИЕ создает проводники данных, соединяя объекты на блок-диаграмме.</p>
	<p>Инструмент ВЫЗОВ КОНТЕКСТНОГО МЕНЮ вызывает контекстное меню соответствующего объекта по щелчку левой кнопки мыши.</p>

	Инструмент БЫСТРАЯ ПРОКРУТКА ЭКРАНА используется для просмотра окна без использования полосы прокрутки.
	Инструмент ВВОД КОНТРОЛЬНОЙ ТОЧКИ позволяет расставлять контрольные точки на ВП, функциях, узлах, проводниках данных, структурах и приостанавливать в них выполнение программы.
	Инструмент УСТАНОВКА ОТЛАДОЧНЫХ ИНДИКАТОРОВ дает возможность исследовать поток данных в проводниках блок-диаграммы. Используется для просмотра промежуточных значений при наличии сомнительных или неожиданных результатов работы ВП.
	Инструмент КОПИРОВАНИЕ ЦВЕТА предназначен для копирования цвета с последующей вставкой с помощью инструмента РАСКРАШИВАНИЕ .
	Инструмента РАСКРАШИВАНИЕ позволяет изменить цвет объекта. Он также отображает текущий передний план и параметры настройки цвета фона.

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

Палитра элементов и палитра функций

Палитра **Элементов (Controls)** и палитра **Функций (Functions)** содержат разделы, в которых размещены объекты для создания ВП. При нажатии на значок раздела на экран выводится окно, содержащее его объекты. Для использования объекта палитры следует щелкнуть на нем мышью и поместить выбранный объект на лицевую панель или блок-диаграмму. Для перемещения по разделам палитры, выбора элементов, ВП и функций следует использовать кнопки навигации. Для открытия ВП

можно также щелкнуть правой кнопкой мыши иконку ВП на палитре и выбрать **Open VI** из контекстного меню.

Палитра **Элементов** (рис. 1.12) используется для размещения элементов управления и отображения на лицевой панели. Она доступна только на лицевой панели. Чтобы отобразить палитру **Элементов**, следует либо выбрать в пункте главного меню **Window»Show Controls Palette**, либо щелкнуть правой кнопкой мыши в рабочем пространстве лицевой панели. Используя кнопку в верхнем левом углу палитры, можно зафиксировать ее на экране.

Палитра **Функций** (рис. 1.13) используется для создания блок-диаграммы. Она доступна только на блок-диаграмме. Чтобы отобразить палитру **Функций**, следует либо выбрать в пункте главного меню **Window»Show Functions Palette**, либо щелкнуть правой кнопкой мыши в рабочем пространстве блок-диаграммы. Используя кнопку в верхнем левом углу палитры, можно зафиксировать ее на экране.

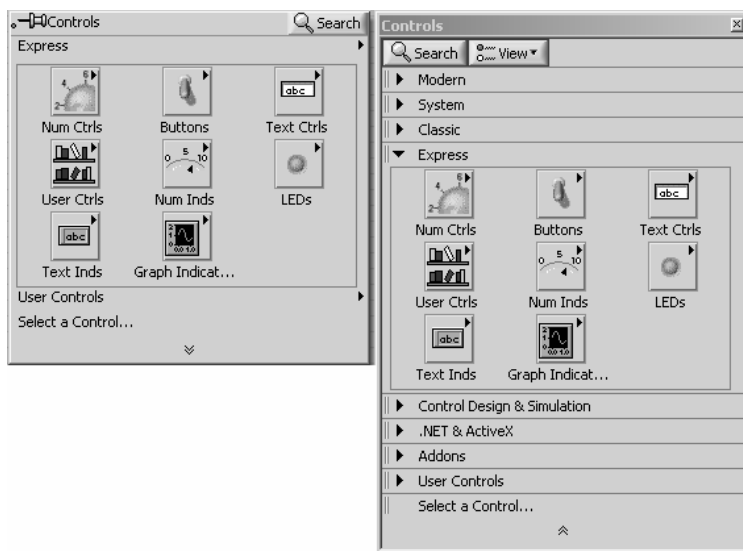


Рис. 1.12. Палитра элементов.

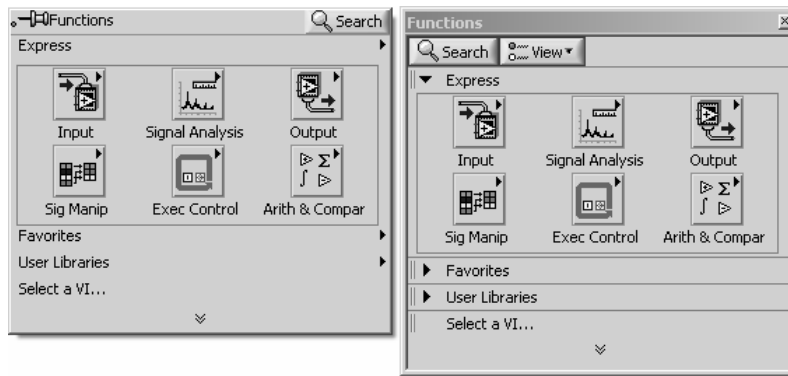


Рис. 1.13. Палитра функций.

Пункты на палитрах организованы в соответствии с категориями. Вы можете выбрать способ отображения различных категорий на палитре (**Category(Standard)**, **Category (Icons and Text)**, **Icons**, **Icons and Text**, **Text**, **Tree**), используя кнопку **View** в верхней части палитры.

При выборе способа отображения пунктов на палитре **Category(Standard)** и **Category (Icons and Text)** вы можете изменить порядок расположения пунктов на палитре, используя в контекстном меню к категории опции **Move this Category Up** и **Move this Category Down**. Кроме того, вы можете перетащить категорию, кликнув на двойную линию, расположенную слева от категории.

При выборе способа отображения пунктов на палитре **Text** и **Tree** вы можете упорядочить их в алфавитном порядке. Для этого выберите кнопку **View» Sort Alphabetically**.

При нажатии на кнопку **Search** вы можете перейти в режим поиска какой-либо функции, узла или ВП по названию. Например, чтобы найти функцию **Random Number** (Генератор случайных чисел), следует нажать кнопку **Search** на палитре **Functions** (Функций) и ввести в поле ввода текста «Random Number». LabVIEW выведет на экран список узлов и функций, в названии которых встречается введенный текст. Выбрав в результатах поиска искомую функцию, можно перенести ее на блок-диаграмму с помощью мыши.

Пункт главного меню **File»Open** открывает диалоговое окно, позволяющее выбрать ВП и загрузить его в память компьютера. На рисунке 1.14 представлен вид диалогового окна, появляющегося во время загрузки ВП.

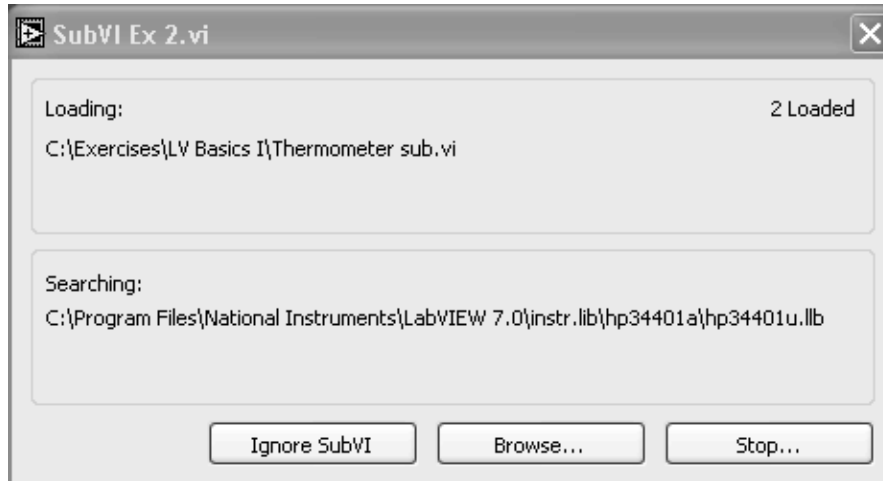


Рис. 1.14. Процесс загрузки ВП.

В этом окне перечисляются все подпрограммы выбранного ВП по порядку их загрузки в память. Остановить загрузку можно в любое время, нажав кнопку **Stop**.

Если LabVIEW не может сразу найти подпрограмму, то поиск продолжается по всем директориям, прописанным в пути поиска файлов **Search Path**. Пути поиска файлов можно редактировать, используя пункты меню Инструменты (**Tools » Options » Paths**). Можно сделать и так, чтобы LabVIEW игнорировал подпрограмму, нажав кнопку **Ignore Sub VI**, или использовать ручной поиск подпрограммы, нажав кнопку **Browse** (Обзор).

1.1.5. Встроенная помощь среды LabVIEW и руководство пользователя

Окно контекстной справки **Context Help** помогает при создании и редактировании ВП. Более подробная информация расположена в **LabVIEW Help** (Встроенной Помощи).

Окно **Context Help** (контекстной справки) выводится на экран из пункта главного меню **Help»Show Context Help** или вводом **<Ctrl-H>** с клавиатуры. При наведении курсора на объект лицевой панели или блок-диаграммы (подпрограммы ВП, функции, константы, элемента управления или отображения данных) в окне контекстной справки появляется иконка этого объекта с указанием всех полей ввода/вывода данных. При наведении курсора на опции диалогового окна в окне контекстной справки появляется описание этих опций. При этом поля, обязательные для соединения, выделены жирным шрифтом, рекомендуемые для соединения поля представлены обычным шрифтом, а дополнительные (необязательные) поля выделены серым или вообще не показаны. На рисунке 1.15 приведен пример окна контекстной справки **Context Help**.

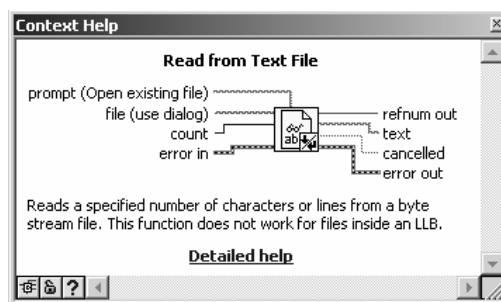


Рис. 1.15. Окно контекстной справки.

Для отображения **LabVIEW Help** (Встроенной Помощи) можно выбрать в пункте главного меню помощь – **Help»VI, Function, & How-To Help** или в окне **Context Help** (контекстной справки) щелкнуть на **Detailed help**.

Встроенная Помощь LabVIEW содержит детальные описания большинства палитр, меню, инструментов, ВП и функций, включает в себя пошаговую инструкцию использования особенностей LabVIEW и связана с *LabVIEW Tutorial* (руководством пользователя), PDF-версией учебника LabVIEW и технической поддержкой на Web-сайте National Instruments.

1.2. Виртуальные приборы (ВП)

1.2.1. Компоненты ВП

ВП состоит из четырех основных компонентов – лицевой панели, блок- диаграммы, иконки и соединительной панели. Подробная информация о создании иконки и соединительной панели – в разделе 1.3 «Подпрограммы ВП».

Лицевая панель

На лицевой панели создаются элементы управления и отображения, которые являются интерактивными средствами ввода и вывода данных этого ВП. Элементы Управления – кнопки, переключатели и другие устройства ввода данных. Элементы Отображения – графики, светодиоды и другие индикаторы. Элементы Управления моделируют устройства ввода данных и передают данные на блок-диаграмму ВП. Элементы отображения моделируют устройства вывода и отображения данных, которые получает или генерирует блок-диаграмма.

Для размещения элементов Управления и Отображения данных на лицевой панели используется палитра Controls (Элементов). Палитра Controls (Элементов) доступна только с лицевой панели. Для вывода на экран палитры Controls (Элементов) следует выбрать пункты главного меню Window»Show Controls Palette или щелкнуть правой кнопкой мыши в рабочем пространстве лицевой панели.

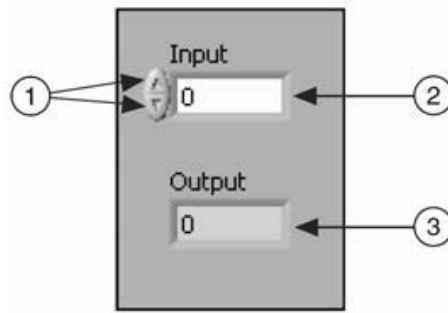


Рис. 1.16. Числовые элементы управления и отображения:

- 1 – кнопки изменения значения;
- 2 – числовой элемент управления;
- 3 – числовой элемент отображения.

Ввод или изменение значения элемента управления осуществляется либо с помощью кнопок приращения значений, либо нужное значение просто вводится в элемент с помощью инструмента ВВОД ТЕКСТА, после чего следует нажать кнопку **<Enter>**.

Логические элементы управления и отображения используются для ввода и отображения значения логической переменной (Boolean), принимающей одно из двух возможных значений (TRUE или FALSE; ИСТИНА или ЛОЖЬ). Логические объекты моделируют выключатели, кнопки и светодиоды. Вертикальный переключатель и круглый светодиод показаны на рисунке 1.17.

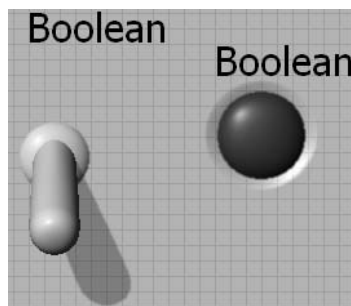


Рис. 1.17. Элементы переключатель и светодиод.

Почти все элементы управления и отображения данных можно редактировать, используя их контекстное меню. Для вызова контекстного

меню следует щелкнуть правой кнопкой мыши на объекте. Например, для редактирования метки – щелкнуть правой кнопкой мыши на метке.

Блок-диаграмма

Блок-диаграмма состоит из узлов, терминалов и проводников данных (рис. 1.18).

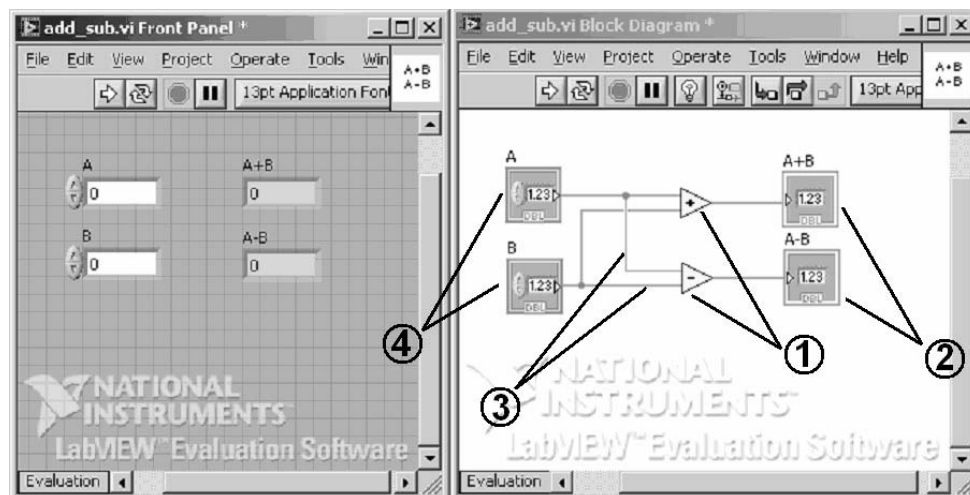




Рис. 1.18. Элементы на блок-диаграмме:

- 1 – узлы;
- 2 – терминалы данных элементов отображения;
- 3 – проводники данных;
- 4 – терминалы данных элементов управления.

Объекты лицевой панели на блок-диаграмме отображаются в виде терминалов данных:  Терминал данных – это графическое изображение прямоугольной формы с буквенно-численными обозначениями. Буквенно-численное обозначение на терминале данных определяет тип данных, который может использоваться в элементах управления или отображения. Например, приведенный DBL-терминал определяет, что данный элемент управления использует числа двойной точности с плавающей запятой.

Терминал данных может отображаться в виде иконки: . Для этого достаточно щелкнуть правой кнопкой мыши в поле терминала

данных и выбрать **View as Icon** (отображать в виде иконки) из контекстного меню. Снять метку для отображения в стандартном виде. Отображение терминала данных в стандартном виде позволяет сохранить место на блок-диаграмме.

Терминалы данных обеспечивают обмен данными между лицевой панелью и блок-диаграммой; они подобны переменным и константам текстовых языков программирования. Различают терминалы данных следующих типов – терминалы элементов управления и отображения данных, терминалы узлов. Терминалы элементов управления и отображения относятся к средствам управления и отображения данных на лицевой панели. Данные, введенные в элементы управления на лицевой панели (А и В на рисунке 1.18), поступают на блок-диаграмму через эти терминалы. Когда функции **Add** (Сложение) и **Subtract** (Вычитание) завершают свои вычисления, то на выходе выдают новое значение данных. Эти значения поступают на терминалы элементов отображения данных и передаются на лицевую панель.

Узлы

Узлы – это объекты на блок-диаграмме, которые имеют одно или более полей ввода/вывода данных и выполняют алгоритмические операции ВП. Они аналогичны операторам, функциям и подпрограммам текстовых языков программирования. Узлы включают в себя функции, подпрограммы ВП и структуры. Подпрограмма ВП – виртуальный прибор, который можно использовать на блок-диаграмме другого ВП в качестве подпрограммы. Структуры – это элементы управления процессом, такие как структура **Case** (Варианта), цикл **While** (цикл по условию) и т.д. Узлы **Add** (Сложение) и **Subtract** (Вычитание), показанные на предыдущей блок-диаграмме, - узлы функций.

1.2.2. Создание ВП

Для создания ВП откройте новый ВП или шаблон и сохраните его. После этого можно конструировать лицевую панель и блок-диаграмму. Диалоговое окно **New** используется для создания различных компонент в среде LabVIEW при построении приложений. Можно начинать с пустого ВП или с шаблона для упрощения программирования. Диалоговое окно **New** содержит следующие компоненты:

Create New – отображает шаблоны, с помощью которых можно создавать ВП или другие документы LabVIEW. Для этого достаточно выбрать шаблон и нажать кнопку ОК.

VI – содержит различные ВП.

Blank VI – открывает пустые лицевую панель и блок-диаграмму.

VI from Template – открывает лицевую панель и блок-диаграмму, содержащие компоненты для построения различных видов ВП.

Frameworks – открывает лицевую панель и блок-диаграмму, содержащие компоненты для построения ВП, включающих специальные виды выполняемых функций.

Instrument I/O – открывает лицевую панель и блок-диаграмму, содержащие компоненты, необходимые для связи с внешними устройствами, подсоединенными к компьютеру.

Simulated – открывает лицевую панель и блок-диаграмму, содержащие компоненты, необходимые для моделирования получения данных с устройства.

Tutorial (Getting Started) – открывает лицевую панель и блок-диаграмму, содержащие компоненты, необходимые для построения ВП, предназначенных для выполнения упражнений руководства Getting Started.

User – открывает лицевую панель и блок-диаграмму ВП, созданного ранее.

Project – открывает окно проекта в LabVIEW.

Other Files – позволяют создать классы, глобальные переменные, библиотеки и т.д.

Description – отображает блок-диаграмму и описание выбранного из списка **Create New** шаблона ВП в случае, если шаблон имеет описание.

Сохранение ВП

Выбрав из пункта главного меню **File** подпункт **Save, Save All** или **Save As**, можно сохранить ВП либо как отдельный файл, либо как группу из нескольких ВП в файл библиотеки ВП LabVIEW. Файл библиотеки ВП имеет расширение *.llb. National Instruments рекомендует сохранять ВП в виде отдельных файлов, организованных в каталоги, особенно если над одним и тем же проектом работают несколько разработчиков. LabVIEW использует диалоги загрузки и сохранения файлов, заданные по умолчанию. Эту функцию можно отключить с помощью пунктов главного меню **Tools»Options**, выбрав из выпадающего меню пункт **Miscellaneous**.

1.2.3. Типы и проводники данных

В среде LabVIEW проводники данных используются для соединения многочисленных терминалов данных. Поля ввода/вывода должны быть совместимыми с типами данных, передаваемыми им по проводникам. Например, нельзя соединять поле вывода массива с полем ввода данных численного типа. Кроме того, характер соединения должен быть корректным. Проводники должны быть подсоединены лишь к одному источнику данных и, по крайней мере, к одному полю ввода данных. Например, нельзя соединять 2 элемента отображения. Компонентами,

определяющими совместимость соединения, являются тип данных элемента управления и/или отображения и тип данных поля ввода/вывода.

В данном курсе используются следующие типы данных:

Numeric (численный тип)

Floating point — число с плавающей запятой, отображается в виде оранжевых терминалов. Может быть представлено в виде **single** (32 bit), **double** (64-bit) или **extended** (128-bit) **precision** (с одиночной, двойной или расширенной точностью). Число с плавающей запятой может быть комплексным.

Integer — целочисленный тип, отображается в виде голубых терминалов. Возможны три представления целых чисел: 8, 16 и 32 бита. Один бит может использоваться для знака числа, если это число является знаковым целым.

Boolean — логический тип, отображается в виде зеленых терминалов. Логический тип может принимать только два значения: 0 (FALSE) или 1 (TRUE).

String — строковый тип, отображается в виде розовых терминалов. Строковый тип данных содержит текст в ASCII формате.

Path — путь к файлу, отображается в виде терминалов. Путь к файлу близок строковому типу, однако, LabVIEW форматирует его, используя стандартный синтаксис для используемой платформы.

Array — массивы включают типы данных составляющих элементов и принимают соответствующий им цвет.










Cluster — кластеры включают различные типы данных. Кластерный тип данных отображается коричневым цветом, если все его элементы численные, если же элементы кластера являются данными различных типов, он отображается розовым.

Waveform — сигнальный тип данных является кластером элементов, содержащим данные, начальное значение времени и интервал времени между измерениями.

Dynamic — динамический тип, отображается в виде темно-синих терминалов. Кроме данных сигнала, динамический тип содержит дополнительную информацию, например, название сигнала или дату и время его получения. Большинство экспресс-ВП принимают и/или возвращают данные динамического типа. Данные динамического типа можно направлять к любому элементу отображения или полю ввода, принимающему данные численного, логического или сигнального типа.

Данные между объектами блок-диаграммы передаются по соединительным линиям – проводникам данных. Проводник данных аналогичен переменным в текстовых языках программирования. Каждый проводник данных имеет единственный источник данных, но может передавать их ко многим ВП и функциям. Проводники данных различаются цветом, стилем и толщиной линии, в зависимости от типа передаваемых данных.

Таблица 1.4. Примеры основных типов проводников данных.

Тип данных	Одно значение	Одномерный (1D) массив	Двумерный (2D) массив	Цвет проводника
Численный				Оранжевый – с плавающей точкой, Голубой – целочисленный.
Логический				Зеленый
Строковый				Розовый

В среде LabVIEW объекты соединяются проводниками данных после их помещения на блок-диаграмму. В автоматическом режиме среда

LabVIEW подключает те поля ввода/вывода данных, которые наиболее совместимы, несовместимые поля остаются несоединенными. Если выбранный объект помещается на блок-диаграмме недалеко от другого объекта, среда LabVIEW показывает пунктирные временные проводники данных, намечающие области возможного соединения. Следует обратить внимание, что при отпускании кнопки мыши LabVIEW автоматически подключает проводник данных к полю ввода/вывода данных, выбранного объекта.

Корректировка параметров автоматического подключения проводников осуществляется через пункты главного меню **Tools>>Options>>Block Diagram**.

Соединение объектов проводниками данных вручную производится с помощью инструмента СОЕДИНЕНИЕ. После наведения инструмента СОЕДИНЕНИЕ на поле ввода или вывода данных на экране появляется подсказка, которую можно использовать для уточнения места подключения проводника.

1.2.4. Редактирование ВП

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

Создание объектов

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

Constant – создание констант, отображающихся только на блок-диаграмме.

Control – создание элемента управления на лицевой панели ВП.

Indicator – создание элемента отображения данных на лицевой панели.

Выделение объектов

Выделение объектов на лицевой панели и блок-диаграмме производится с помощью инструмента ПЕРЕМЕЩЕНИЕ. Когда объект выделен, его окружает пунктирная линия. Для выбора нескольких объектов следует во время их выделения нажать и удерживать клавишу **<Shift>**. Можно также выделить несколько объектов, щелкнув мышью в свободном пространстве и обведя их курсором.

Перемещение объектов

Перемещение объектов осуществляется при помощи инструмента ПЕРЕМЕЩЕНИЕ. Перемещать объекты можно также при помощи стрелок на клавиатуре. Для перемещения объекта с шагом в несколько пикселей в момент перемещения следует нажать и удерживать клавишу **<Shift>**. Можно ограничить направление движения выбранного объекта только по горизонтали или только по вертикали, если в момент его перемещения удерживать клавишу **<Shift>**. Первоначально выбранное направление движения (горизонтальное или вертикальное) определяет направление перемещения объекта.

Удаление объектов

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

Отмена и восстановление действий

Если в процессе редактирования ВП была допущена ошибка, можно отменить или восстановить действия, выбрав **Undo** (Отменить) или **Redo** (Восстановить) в пункте главного меню **Edit** (Редактирование). Установка количества действий, подлежащих отмене или восстановлению,

производится в пункте главного меню **Tools»Options**. Для этого из выпадающего меню следует выбрать раздел **Block Diagram**. Установка небольшого числа повторений сохраняет ресурсы памяти компьютера.

Копирование объектов

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

После переноса выбранного объекта на новое место, отпускается сначала кнопка мыши, а затем клавиша **<Ctrl>**. В этом месте появляется копия объекта, а первоначальный объект остается на старом месте. Этот процесс называется копированием либо клонированием. Можно копировать объекты и стандартным способом, выбирая пункты главного меню **Edit»»Copy** и затем **Edit»»Paste**.

Метки объектов

Метки используются для идентификации объектов. Среда LabVIEW имеет два вида меток: свободные и собственные. Собственные метки принадлежат объекту, описывают только его и двигаются вместе с ним. Собственную метку можно перемещать независимо от объекта, но при перемещении объекта метка перемещается вместе с ним. Свободные метки не принадлежат объектам, их можно создавать, перемещать, вращать или удалять независимо. Они используются для описания объектов, ввода комментариев на лицевой панели и блок-диаграмме. Для создания свободной метки используется инструмент ВВОД ТЕКСТА. Выбрав этот инструмент, необходимо щелкнуть в свободном пространстве одной из панелей и ввести текст. После ввода текста метки поместить курсор в пространство вне метки или нажать кнопку **<Enter>** на инструментальной панели.

По умолчанию нажатие на клавиатуре клавиши **<Enter>** добавляет новую строку. Чтобы закончить ввод текста с клавиатуры, следует нажать **<Shift-Enter>**. Можно закончить ввод текста с клавиатуры нажатием

клавиши **<Enter>**, для этого в пункте главного меню следует выбрать **Tools»Options**, далее, в выпадающем меню найти **Front Panel** и отметить пункт **End text entry with Return key**.

Специальный вид свободной метки используется для ввода комментариев на блок-диаграмму. Эта свободная метка находится на палитре **Functions»Programming»Structures»Decorations**.

Выделение и удаление проводников данных

Сегмент проводника данных – это отдельная горизонтальная или вертикальная его часть. Место соединения двух сегментов – излом проводника данных. Точка, в которой встречаются два, три или четыре проводника данных, называется точкой соединения. Проводник данных содержит все сегменты между точками соединения, между терминалом данных и точкой соединения, между терминалами данных, если нет точек соединений. Для выделения сегмента используется инструмент ПЕРЕМЕЩЕНИЕ. Двойной щелчок мыши выделяет проводник данных, тройной щелчок – множество проводников данных.

LabVIEW поддерживает функцию автоматического масштабирования проводников данных, поэтому перемещение объектов не приводит к нарушению проводника данных.

Разорванные проводники данных

Разорванный проводник данных выглядит, как черная штриховая линия с красным крестом посередине, как показано ниже. Разрыв проводников данных происходит по причинам разного рода. Например, при попытке соединения объектов с несовместимыми типами данных. Описание причины разрыва проводника данных появляется в окне всплывающей подсказки после наведения на проводник инструмента СОЕДИНЕНИЕ. Тройной щелчок инструментом ПЕРЕМЕЩЕНИЕ на проводнике и последующее нажатие клавиши **<Delete>** удаляет

выделенный проводник. Удаление всех разорванных проводников производится через пункт главного меню **Edit»Remove Broken Wires**.

Следует обратить внимание на то, что использование пункта главного меню **Remove Broken Wires** требует определенной осторожности. Иногда проводник является разорванным, потому что еще не закончено создание блок-диаграммы.

Редактирование текста (изменение шрифта, стиля и размера)

Выбрав пункт меню **Text Settings** на инструментальной панели, можно изменить шрифт, стиль, размер и провести выравнивание любого текста внутри меток или на дисплеях элементов управления и отображения. На некоторых элементах управления и отображения данных текст может быть помещен более чем в одном месте, например оси графиков. В этом случае текст в каждом поле можно изменять независимо. Текст выделяется инструментом **ВВОД ТЕКСТА**, как показано на рисунке 1.19, и на инструментальной панели выбирается пункт меню **Text Settings**.

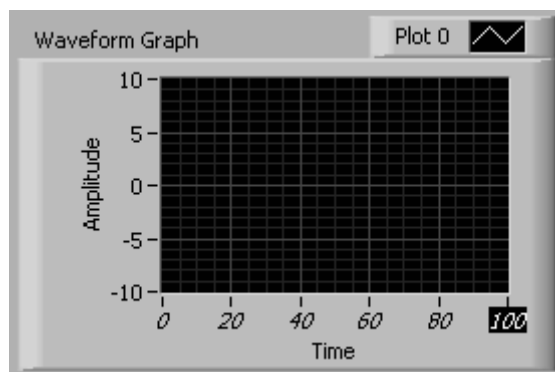


Рис. 1.19. Редактирование текста на графике.

Изменение размеров объектов

Большинство объектов лицевой панели допускают изменение размеров. Чтобы подготовить объект к изменению размера, необходимо навести на него инструмент **ПЕРЕМЕЩЕНИЕ**. По углам объекта появляются маркеры, показанные слева. Затем курсор следует установить на один из маркеров и, удерживая нажатой левую кнопку мыши,

переместить маркер, размер шрифта при этом не меняется. Промежуточные границы изменяемого размера обозначаются штриховой линией. Когда нужный размер элемента достигнут, кнопку мыши следует отпустить. Удержание клавиши <Shift> во время перемещения маркеров сохраняет пропорции объекта. Можно изменять размеры и объектов блок-диаграммы, таких как структуры и константы.

Выравнивание и распределение объектов в пространстве

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

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

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

Приведение нескольких объектов к одному размеру

Приведение нескольких объектов к одному виду производится с помощью выпадающего меню **Resize Objects** (Изменение размеров объектов). Предусмотрена возможность изменения размера всех выбранных объектов по ширине или высоте до ширины/высоты

наименьшего или наибольшего объекта, также имеется возможность задать размер всех выбранных объектов в пикселях.

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

Копирование объектов между ВП или между другими приложениями

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

Окрашивание объектов

Можно изменять цвет большинства объектов ВП, но не всех. Например, терминалы данных и проводники данных блок-диаграммы используют только определенные цвета, соответствующие типу представленных данных. Изменение цвета объекта или фона рабочего пространства производится с помощью инструмента РАСКРАШИВАНИЕ. Для этого следует щелкнуть правой кнопкой мыши на выбранном элементе или рабочем пространстве любой из панелей. Можно изменить заданные по умолчанию цвета большинства объектов, выбирая пункты меню **Tools»»Options** и затем **Colors**. Можно также сделать объект прозрачным, выбрав **T (transparent)** в меню **Colors**.

Отладка ВП

Если ВП не запускается, это означает, что он не готов к работе. В процессе создания или редактирования ВП кнопка **Run** принимает вид разорванной стрелки, как было показано выше. Если после завершения редактирования блок-диаграммы стрелка все еще имеет разорванный вид, то ВП работать не будет.

Для поиска ошибок нажмите кнопку **Run** или выберите пункт главного меню **Windows»Show Error List**, чтобы вывести на экран окно **Список ошибок**, в котором перечислены все допущенные ошибки. После двойного щелчка левой кнопкой мыши на описании ошибки выделится объект, содержащий эту ошибку.

Режим анимации выполнения ВП

Режим анимации выполнения блок-диаграммы активируется щелчком правой кнопки мыши по кнопке **Highlight Execution**. После нажатия кнопки «лампочка» загорится – режим активирован. Выполнение ВП в этом режиме сопровождается подсветкой движения данных по блок-диаграмме от одного узла к другому. При этом числовые значения передаваемых данных будут отображаться на выходных терминалах узлов диаграммы в виде всплывающих окон. Этот режим используется для пошаговой отладки ВП и наблюдения за выполнением блок-диаграммы. Не следует забывать, что режим анимации замедляет скорость выполнения ВП.

Режим пошаговой отладки ВП

Режим пошаговой отладки ВП используется для просмотра выполнения ВП на блок-диаграмме. Активация пошагового режима осуществляется нажатием кнопок **Step Over** или **Step Into** на инструментальной панели. Чтобы увидеть подсказку, следует поместить курсор поверх кнопок **Step Over**, **Step Into** или **Step Out**. Подсказка описывает событие, которое последует после нажатия этой кнопки.

Пошаговый режим можно использовать и для просмотра выполнения подпрограммы ВП. При использовании пошагового режима отладки ВП в режиме анимации на иконке подпрограммы ВП появится зеленая стрелка. Зеленая стрелка показывает, что данная подпрограмма ВП в данный момент времени выполняется.

Отладочные индикаторы

Инструмент **УСТАНОВКА ОТЛАДОЧНЫХ ИНДИКАТОРОВ** предназначен для проверки промежуточного значения данных в проводнике данных в процессе выполнения ВП. В режиме пошагового выполнения или при остановке в контрольной точке с помощью отладочных индикаторов можно визуализировать значения данных в проводнике или поле ввода/вывода узла данных, если узел уже получил свое значение. Можно установить несколько локальных отладочных индикаторов для одновременного наблюдения за данными в разных точках блок-диаграммы.

Чтобы создать локальный отладочный индикатор, следует щелкнуть правой кнопкой мыши на выбранном проводнике данных и выбрать в контекстном меню пункт **Custom Probe**. При отладке ВП вы можете сохранять в памяти значения данных, прошедших по проводникам. То есть, когда вы помещаете **ОТЛАДОЧНЫЙ ИНДИКАТОР** на блок-диаграмму, в нем сразу же будет отображаться значение данных, прошедших по проводу в момент последнего выполнения. Чтобы сделать функцию доступной, нажмите на панели инструментов в верхней части окна блок-диаграммы кнопку **Retain Wire Values**. После этого LabVIEW будет сохранять значения данных в каждой точке потока.

Контрольные точки

Инструмент **ВВОД КОНТРОЛЬНОЙ ТОЧКИ** предназначен для размещения контрольных точек в узлах или проводниках данных блок-диаграммы. В месте установки контрольной точки в момент прохождения

через нее данных возникает пауза в выполнении программы. Когда ВП приостановил свое выполнение в контрольной точке, LabVIEW подсвечивает узел или проводник данных в месте установки контрольной точки. LabVIEW обводит красной границей узел и блок-диаграмму и отмечает красным маркером проводник данных. После наведения курсора на контрольную точку черное поле инструмента ВВОД КОНТРОЛЬНОЙ ТОЧКИ становится белым. Для удаления существующей контрольной точки по ней следует щелкнуть инструментом ВВОД КОНТРОЛЬНОЙ ТОЧКИ.

1.3. Создание подпрограмм ВП

1.3.1. Подпрограммы ВП

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

1.3.2. Создание иконки ВП и настройка соединительной панели

Каждый виртуальный прибор в правом верхнем углу лицевой панели и в окне блок-диаграммы отображает иконку. Иконка – графическое

представление прибора. Она может содержать текст, рисунок или и то и другое одновременно. Если ВП используется в качестве подпрограммы, то иконка идентифицирует его на блок-диаграмме другого ВП.

Установленная по умолчанию иконка ВП содержит номер, который указывает, сколько новых приборов открылось после запуска LabVIEW. Создать собственную иконку, отличную от заданной по умолчанию, можно, щелкнув правой кнопкой мыши по иконке в правом верхнем углу лицевой панели или блок-диаграммы. Затем выбрать пункт **Edit Icon** (Редактирование иконки) из контекстного меню. **Icon Editor** (Редактор иконки) можно также вызвать двойным щелчком левой кнопки мыши в верхнем правом углу одной из панелей. Редактирование иконки доступно также из пункта главного меню **File**, далее **VI Properties** (Свойства ВП), где в диалоговом окне **Category** (Категория) следует выбрать пункт **General** (Общие) и нажать кнопку **Edit Icon** (Редактирование иконки).

Проектирование иконки выполняется в области редактирования, расположенной в центре окна **Icon Editor** (Редактора иконки), при помощи инструментов, расположенных слева от области редактирования. Вид иконки и доступный на блок-диаграмме и в правом верхнем углу обеих панелей размер иконки появляется справа от области редактирования, в соответствующем поле, как показано на рисунке 1.20.

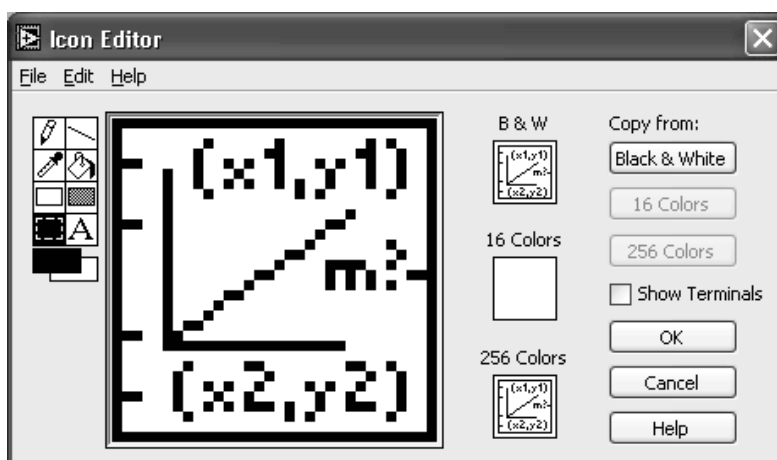











Рис. 1.20. Редактирование иконки ВП.

В зависимости от типа монитора, иконка может быть создана для черно-белого, 16-цветного или 256-цветного режима. Для печати, в случае отсутствия цветного принтера, LabVIEW использует черно-белую иконку. По умолчанию установлен 256-цветный режим. Меню **Edit** (редактирование) используется для вырезания, копирования и вставки картинок из иконки или в нее. При выборе фрагмента иконки для вставки картинки LabVIEW изменяет размер картинки для соответствия размеру выбранной области. Предусмотрена возможность перемещения графических символов из файловой системы в верхний правый угол лицевой панели или блок-диаграммы. LabVIEW автоматически преобразует изображение в иконку размером 32×32 точки.

Для копирования цветной иконки в черно-белую (или наоборот) достаточно выбрать опцию **Copy from**, находящуюся в правой части диалогового окна **Icon Editor**. Нажать кнопку **OK** для окончательной замены. В случае если сплошная граница вокруг иконки не нарисована, фон иконки будет прозрачным. При выборе иконки на блок-диаграмме маркеры выбора появляются вокруг каждого графического элемента иконки. Набор инструментов для редактирования иконки расположен в левой части окна **Icon Editor**.

Таблица 1.5. Описание инструментов в окне **Icon Editor**.

	Инструмент КАРАНДАШ позволяет рисовать или стирать по одной точке.
	Инструмент ЛИНИЯ позволяет рисовать прямые линии. Для рисования вертикальных, горизонтальных и диагональных линий необходимо во время рисования нажать и удерживать клавишу <Shift> .
	Инструмент КОПИРОВАНИЕ ЦВЕТА предназначен для копирования цвета символа в поле редактирования иконки.
	Инструмент ЗАПОЛНЕНИЕ ЦВЕТОМ предназначен для заполнения ограниченной области заданным цветом переднего плана.

	Инструмент ПРЯМОУГОЛЬНИК выводит в область редактирования прямоугольную границу заданным цветом переднего плана. Двойной щелчок левой кнопкой мыши на ПРЯМОУГОЛЬНИК обводит иконку рамкой заданным цветом переднего плана.
	Инструмент ЗАПОЛНЕННЫЙ ЦВЕТОМ ФОНА ПРЯМОУГОЛЬНИК выводит в область редактирования прямоугольную границу заданным цветом переднего плана, заполненную цветом фона. Двойной щелчок левой кнопкой мыши на ЗАПОЛНЕННОМ ЦВЕТОМ ФОНА ПРЯМОУГОЛЬНИКЕ обводит иконку рамкой цвета символа и заполняет цветом фона.
	Инструмент ВЫБОР предназначен для выделения фрагмента иконки, что позволяет вырезать, копировать, перемещать или вносить другие изменения в выделенный фрагмент. Чтобы очистить область редактирования иконки достаточно дважды щелкнуть левой кнопкой мыши на инструменте ВЫБОР и нажать кнопку <Delete> .
	Инструмент ВВОД ТЕКСТА позволяет вводить текст в область редактирования иконки. Выбор шрифта производится двойным щелчком левой кнопкой мыши на инструменте ВВОД ТЕКСТА. (Windows) Доступна опция « Small Fonts ».
	Инструмент ПЕРЕДНИЙ ПЛАН/ФОН отображает цвета фона и переднего плана (символа). При нажатии на каждый прямоугольник появляется палитра выбора цвета.

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

Show Terminals – выводит в область редактирования поля ввода/вывода данных.

OK – сохраняет внесенные в иконку изменения.

Cancel – закрывает **Icon Editor** без сохранения.

Строка меню в окне **Icon Editor** содержит опции редактирования, такие как **Undo** (Отмена), **Redo** (Повтор), **Cut** (Вырезать), **Copy** (Копировать), **Paste** (Вставить) и **Clear** (Очистить).

Настройка соединительной панели

Для использования ВП в качестве подпрограммы ВП необходимо настроить соединительную панель, показанную слева. Соединительная панель является совокупностью полей ввода/вывода данных, соответствующих элементам управления и отображения этого ВП, подобно набору параметров вызова функции в текстовых языках программирования. Соединительная панель определяет поля входных и выходных данных ВП. Таким образом, ВП можно использовать в качестве подпрограммы. Каждому полю ввода или вывода данных назначается свой элемент лицевой панели. Для редактирования соединительной панели необходимо щелкнуть правой кнопкой мыши на иконке ВП и выбрать из контекстного меню пункт **Show Connector** (Показать поля ввода/вывода данных). Вместо иконки появится соединительная панель, в которой каждый прямоугольник соответствует полю ввода или вывода данных. Количество отображаемых LabVIEW полей ввода/вывода данных соответствует количеству элементов на лицевой панели. На рисунке 1.21 показана лицевая панель, содержащая четыре элемента управления и один элемент отображения. Таким образом, в соединительной панели LabVIEW отображает четыре поля ввода и одно поле вывода данных.

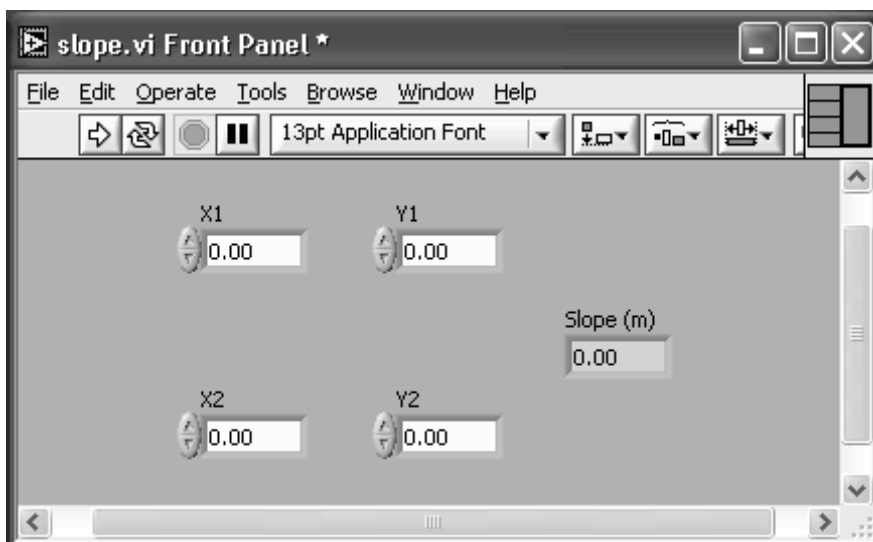


Рис. 1.21. Настройка соединительной панели ВП.

Выбор и редактирование шаблона соединительной панели

Выбор шаблона осуществляется щелчком правой кнопки мыши на соединительной панели и выбором пункта **Patterns** (Шаблон) из контекстного меню. В шаблоне некоторые из полей ввода/вывода данных можно оставить без соединения и задействовать позднее при необходимости. Такая гибкость дает возможность вносить изменения с минимальным отражением на иерархии ВП. Причем не все элементы лицевой панели должны быть обязательно задействованы в соединительной панели. Задействованные поля выделены цветом, соответствующим типу данных элемента. Максимально возможное количество полей ввода/вывода данных ограничено 28.

Наиболее часто используемый шаблон содержит четыре поля ввода и четыре поля вывода. Данный шаблон является стандартным для упрощения соединения. Верхние поля ввода/вывода обычно используются для ссылок, нижние – для обработки ошибок. Подробная информация об обработке ошибок находится в Разделе 3.2, *Кластеры*.

Следует избегать необходимости использования более 16 полей ввода/вывода данных. Наличие более 16 полей снижает удобочитаемость.

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

Привязка полей ввода/вывода данных к элементам лицевой панели

После выбора шаблона соединительной панели необходимо каждому полю назначить свой элемент лицевой панели. Для упрощения использования подпрограммы ВП следует поля ввода данных размещать слева, а поля, связанные с элементами отображения, – справа на соединительной панели. Чтобы назначить поля ввода или вывода данных, следует щелкнуть по выбранному полю левой кнопкой мыши, затем щелкнуть мышью на элементе, который необходимо связать с этим полем, после этого вывести курсор в свободное пространство лицевой панели и снова щелкнуть мышью. Задействованные поля примут цвет, определенный типом данных соответствующего элемента. Можно также сначала щелкнуть левой кнопкой мыши по элементу, а потом по полю ввода/вывода данных. Во время назначения полей ввода/вывода данных используется инструмент СОЕДИНЕНИЕ, однако между элементом лицевой панели и соответствующим ему полем проводник не появляется.

1.3.3. Использование подпрограмм ВП

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

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

Редактирование подпрограммы ВП

Вызов лицевой панели подпрограммы ВП из блок-диаграммы другого ВП производится двойным щелчком на нем инструментом УПРАВЛЕНИЕ или ПЕРЕМЕЩЕНИЕ. Это же можно сделать с помощью главного меню, выбрав в пункте **Browse** (Обзор) подпункт **This VI's SubVIs** (Подпрограммы этого ВП). Для вызова блок-диаграммы подпрограммы ВП следует, удерживая клавишу **<Ctrl>**, дважды щелкнуть на нем левой кнопкой мыши. Изменения, внесенные в подпрограмму ВП, доступны вызывающим его программам только после предварительного их сохранения.

Установка значимости полей ввода

В окне контекстной справки **Context Help**, которое доступно из пункта главного меню **Help»Show Context Help**, обязательные для соединения поля обозначены жирным шрифтом, рекомендуемые – нормальным, а дополнительные (не обязательные) – светло-серым шрифтом при условии, что используется режим подробного просмотра **Detailed**. В **Simple** (Кратком) просмотре окна контекстной справки **Context Help** эта информация недоступна. При создании подпрограммы ВП необходимо указать обязательные для соединения поля (также рекомендуемые и дополнительные) с целью предупреждения пользователя от ошибки.

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

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

По умолчанию LabVIEW устанавливает значимость созданного поля в позицию **Recommended** (Рекомендуется). Установка **Required** (Обязательно) необходима для указания соединений, без которых ВП работать не будет. В качестве примера можно рассмотреть **File I/O** (подпрограммы работы с файлами), расположенные на палитре **Functions** (Функций).

1.3.4. Преобразование экспресс-ВП в подпрограмму ВП

Экспресс-ВП называются настраиваемые с помощью диалогового окна узлы функций. Они используются для выполнения стандартных измерений, уменьшая количество соединений проводников данных. Подробнее об экспресс-ВП можно прочитать в руководстве *Getting Started with LabVIEW*. Предусмотрена возможность создания подпрограммы ВП из сконфигурированного экспресс-ВП. Для этого достаточно щелкнуть правой кнопкой мыши по экспресс-ВП и выбрать пункт **Open Front Panel** (открыть лицевую панель) в контекстном меню. Для создания подпрограммы ВП из сконфигурированного экспресс-ВП необходимо выполнить следующую последовательность действий:

1. Сконфигурировать экспресс-ВП.
2. Щелкнуть правой кнопкой мыши по экспресс-ВП и выбрать пункт **Open Front Panel** (открыть лицевую панель) в контекстном меню.

3. Нажать на кнопку **Convert** (преобразовать) в появившемся диалоговом окне с предупреждением, после этого появится лицевая панель ВП.
4. Отредактировать ВП.
5. Выбрать пункт **Operate»Make Current Values Default** или выделять мышью каждый элемент управления и выбирать пункт контекстного меню **Make Current Values Default** для сохранения значений каждого элемента управления.
6. Сохранить ВП, новая подпрограмма ВП, отображенная в виде раскрывающегося узла, заменит экспресс-ВП на блок-диаграмме.

После создания ВП из экспресс-ВП подпрограмма ВП не преобразовывается обратно в экспресс-ВП.

1.3.5. Превращение выделенной секции блок-диаграммы в подпрограмму ВП

Можно упростить блок-диаграмму ВП, создав из часто выполняемых операций подпрограмму ВП. Для этого с помощью инструмента ПЕРЕМЕЩЕНИЕ необходимо выделить интересующую секцию блок-диаграммы и выбрать из пункта главного меню **Edit** (Редактирование) пункт **Create SubVI** (Создать подпрограмму ВП). Выделенная секция сменится иконкой новой подпрограммы ВП. LabVIEW создаст элементы управления и отображения данных для новой подпрограммы ВП и соединит поля ввода/вывода данных с существующими проводниками, как показано на рисунке 1.22.

По умолчанию новая подпрограмма ВП использует шаблон для создания соединительной панели и иконку. Дважды щелкните правой кнопкой мыши по иконке подпрограммы ВП для редактирования соединительной панели и иконки и для сохранения ВП.

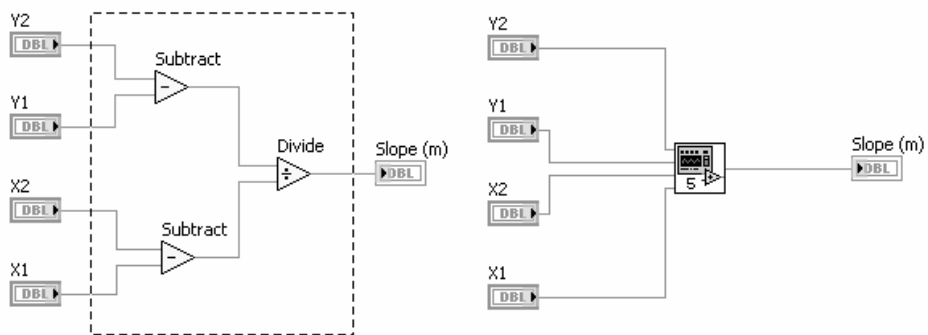


Рис. 1.22. Выделение секции блок-диаграммы.

Однако нельзя создать подпрограмму ВП из секции с количеством входов и выходов более 28, так как 28 – максимальное количество возможных полей ввода/вывода данных подпрограммы ВП.

2. Инструменты для построения алгоритмов

2.1. Многократные повторения и Циклы

Структуры являются графическим представлением операторов цикла и операторов **Case** (Варианта), используемых в текстовых языках программирования. Структуры на блок-диаграмме используются для выполнения повторяющихся операций над потоком данных, операций в определенном порядке и наложения условий на выполнение операций. Среда LabVIEW содержит следующие структуры: цикл **While** (по условию), цикл **For** (с фиксированным числом итераций), структура **Case** (Вариант), структура **Sequence** (Последовательность), структура **Event** (Событие), а также **Formula Node** (узел Формулы), **MathScript Node** (узел Математики) и др.

В этом разделе рассмотрены структуры – Цикл **While** (по условию), Цикл **For** (с фиксированным числом итераций), а также функции, часто используемые с этими структурами, такие как **Shift Register** (Сдвиговый регистр) и **Feedback Node** (узел Обратной связи).

2.1.1. Цикл **While** (по Условию)

Цикл **While** (по условию) работает до тех пор, пока не выполнится логическое условие выхода из цикла. Цикл **While** аналогичен циклам **Do** и **Repeat Until**, используемым в текстовых языках программирования. Рисунок 2.1. демонстрирует (1) цикл **While** в среде LabVIEW, (2) эквивалентную блок-схему работы цикла **While**, (3) пример текстового аналога кода работы цикла **While**.

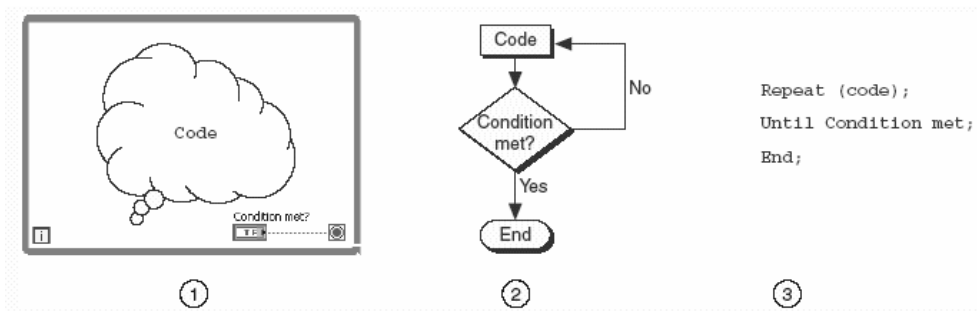


Рис. 2.1. Цикл с остановкой по условию.

Цикл **While** находится в палитре **Functions»Programming»Structures**. После того как цикл выбран в палитре **Functions** (Функций), следует с помощью курсора выделить часть блок-диаграммы, которую необходимо поместить в цикл. После отпускания кнопки мыши, выделенная область блок-диаграммы помещается в тело цикла. Добавление объектов блок-диаграммы в тело цикла осуществляется помещением или перетаскиванием объектов.

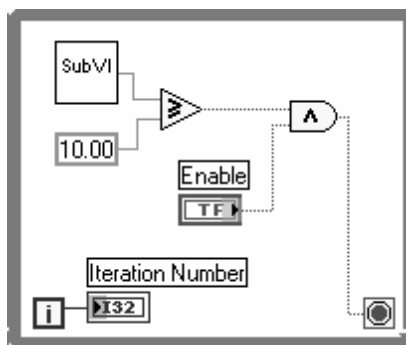




Рис. 2.2. Цикл с остановкой по значению ИСТИНА.

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

Терминал счетчика итераций  содержит значение количества выполненных итераций. Начальное значение терминала всегда равно нулю.

На блок-диаграмме, показанной на рисунке 2.2, условие выхода из цикла **While** определяется значением выходного параметра подпрограммы ВП большего или равного 10,00 и состоянием терминала элемента управления **Enable**. Функция **And** (Логическое «И») на выходе выдает значение TRUE, если на оба поля ввода данных функции поступают значения TRUE. В противном случае функция на выходе выдает значение FALSE, и работа цикла не завершается.

В предыдущем примере велика вероятность получения бесконечно выполняемого цикла. Обычно стремятся получить единственное условие выхода из цикла, нежели одновременное выполнение двух условий.

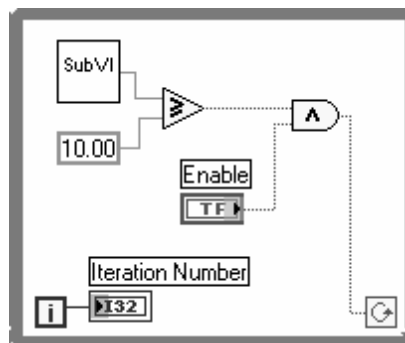



Рис. 2.3. Цикл с остановкой по значению ЛОЖЬ.

Предусмотрена возможность изменения условия выхода и соответствующего ему изображения терминала условия выхода. Щелчком правой кнопки мыши по терминалу условия выхода или по границе цикла необходимо вызвать контекстное меню и выбрать пункт **Continue If True** (Продолжение, если Истина). Также можно воспользоваться инструментом УПРАВЛЕНИЕ, щелкнув им по терминалу условия выхода. Изображение терминала условия выхода поменяется на  **Continue If True** (Продолжение, если Истина). В результате условием выхода из цикла

становится поступающее на терминал условия выхода значение FALSE, как показано на следующей блок-диаграмме. Цикл **While** на рисунке 2.3 выполняется до тех пор, пока выходные данные подпрограммы ВП остаются меньше «10».

Терминалы входных/выходных данных цикла

Данные могут поступать в цикл **While** (или выходить из него) через терминалы входных/выходных данных цикла. Терминалы входных/выходных данных цикла передают данные из структур и в структуры. Терминалы входных/выходных данных цикла отображаются в виде сплошных прямоугольников на границе области цикла **While**.

Прямоугольник принимает цвет типа данных, передаваемых по терминалу. Данные выходят из цикла по его завершении. В случае если данные поступают в цикл **While** через терминал входных/выходных данных цикла, выполнение цикла начинается при поступлении данных в терминал.

На следующей блок-диаграмме терминал счетчика итераций присоединен к терминалу выхода цикла. Значения из терминала выхода цикла не поступают к элементу отображения номера итерации до завершения цикла **While**.

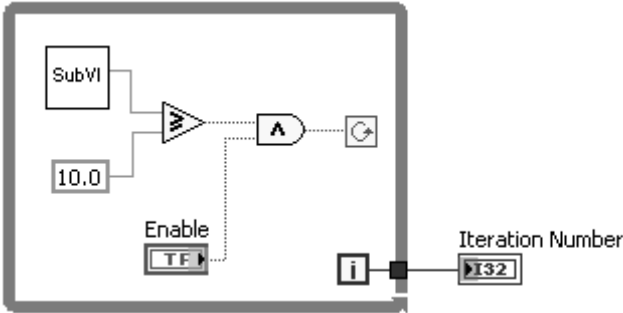


Рис. 2.4. Вывод данных из цикла через выходной терминал.

Лишь последнее значение итерации отображается элементом отображения номера итерации.

2.1.2. Цикл For (с фиксированным числом итераций)

Цикл **For** (с фиксированным числом итераций) выполняет повторяющиеся операции над потоком данных определенное количество раз. Иллюстрация (рис. 2.5) демонстрирует (1) цикл **For** в среде LabVIEW, (2) эквивалентную блок-схему работы цикла **For**, (3) пример текстового аналога кода работы цикла **For**.

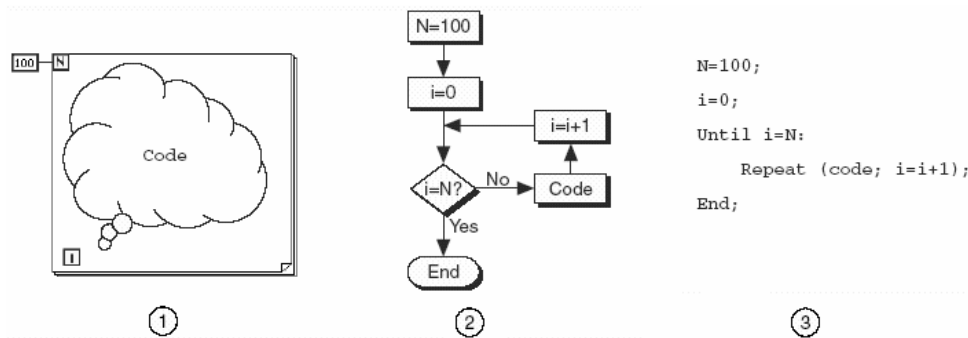


Рис. 2.5. Цикл с фиксированным количеством итераций.

Цикл **For**, расположен в палитре **Функций** в разделе **Functions»Programming»Structures**. Значение, присвоенное терминалу **максимального числа итераций N** цикла **N**, определяет максимальное количество повторений операций над потоком данных. **Терминал счетчика итераций i** содержит значение количества выполненных итераций. Начальное значение счетчика итераций всегда равно 0.

Цикл **For** отличается от цикла **While** тем, что завершает работу, выполнив заданное максимальное число итераций **N**. Цикл **While** завершает работу при выполнении заданного условия выхода из цикла.

Преобразование типов данных

LabVIEW может оперировать с такими типами данных, как целочисленный тип (integer): **byte, word, long**, число с плавающей запятой: **single, double, extended precision**, комплексное число: **single, double, extended precision**.

Когда в поле ввода данных функции поступают операнды разных типов, то значение на выходе функции принимает формат данных более широкого диапазона. При этом LabVIEW автоматически осуществляет преобразование типов и в месте соединения проводника с терминалом появляется изображение серой точки.

Например, терминал максимального числа итераций N цикла For имеет целочисленный тип двойной точности (long integer). На него поступают данные в формате числа двойной точности с плавающей запятой. На терминале числа итераций появляется красный треугольник.

Если в поля ввода данных функции, работающей с данными одного типа, поступают данные двух разных типов, LabVIEW приводит тип данных одного из терминалов к типу данных другого терминала. LabVIEW выбирает тип данных, занимающий большее количество бит. Если типы эквивалентны по количеству занимаемых бит, LabVIEW предпочитает беззнаковый тип данных.

Для изменения типа представления данных на объектах блок-диаграммы необходимо щелкнуть по ним правой кнопкой мыши и из контекстного меню выбрать пункт **Representation**.


Когда LabVIEW проводит преобразование данных из формата числа двойной точности с плавающей запятой в целочисленный формат, то значение $x,5$ округляется до ближайшего целого четного. Например, LabVIEW округляет 2,5 до 2, а 3,5 до 4.

2.1.3. Организация доступа к значениям предыдущих итераций цикла

При работе с циклами зачастую необходим доступ к значениям предыдущих итераций цикла. Например, в случае ВП, измеряющего температуру и отображающего ее на графике, для отображения текущего среднего значения температуры, необходимо использовать значения,

полученные в предыдущих итерациях. Есть два пути доступа к этим данным: **Shift Register** (сдвиговый регистр) и **Feedback Node** (узел обратной связи).

Сдвиговые регистры

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

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

Чтобы инициализировать сдвиговый регистр, необходимо передать на его левый терминал любое значение извне цикла. Если не инициализировать сдвиговый регистр, он использует значение, записанное в регистр во время последнего выполнения цикла или значение, используемое по умолчанию для данного типа данных, если цикл никогда не выполнялся.

Цикл с неинициализированным сдвиговым регистром используется при неоднократном запуске ВП для присвоения выходному значению сдвигового регистра значения, взятого с последнего выполнения ВП. Чтобы сохранить информацию о состоянии между последующими

запусками ВП, следует оставить вход левого терминала сдвигового регистра не определенным.

После завершения выполнения цикла последнее значение, записанное в регистр, останется на правом терминале. При последующей передаче данных из цикла через правый терминал будет передано последнее значение, записанное в регистр.

Предусмотрена возможность создания нескольких сдвиговых регистров в одной структуре цикла. Если в одном цикле выполняется несколько операций, следует использовать сдвиговый регистр с несколькими терминалами для хранения данных, полученных в результате выполнения различных операций цикла. На рисунке 2.6 показано использование двух инициализированных сдвиговых регистров.

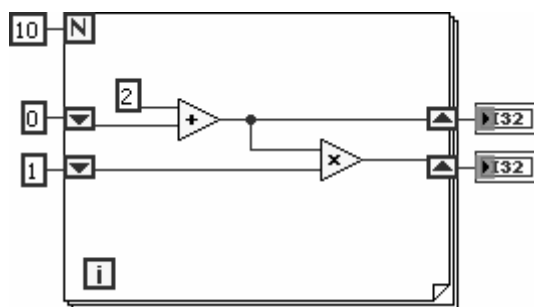


Рис. 2.6. Цикл с двумя сдвиговыми регистрами.

Стек сдвиговых регистров

Для создания стека сдвиговых регистров достаточно щелкнуть правой кнопкой мыши по левому терминалу и выбрать пункт контекстного меню **Add Element**. Стек сдвиговых регистров осуществляет доступ к значениям предыдущих итераций цикла. Стек сдвиговых регистров сохраняет данные предыдущей итерации и передает эти значения к следующей итерации.

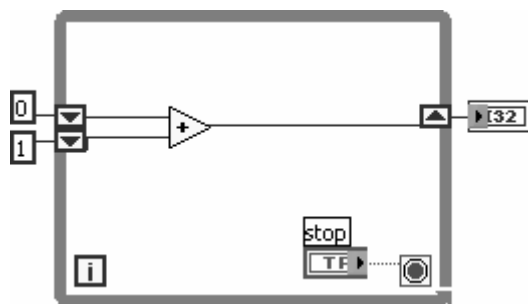



Рис. 2.7. Стек сдвиговых регистров.

Стек сдвиговых регистров может находиться только в левой части цикла, так как правый терминал лишь передает данные из текущей итерации в следующую. Значение последней итерации сохраняется в самом верхнем сдвиговом регистре. Второй терминал сохраняют данные, переданные ему с предыдущей итерации.

Узлы обратной связи

Узел обратной связи  автоматически появляется в циклах **While** или **For** при соединении поля вывода данных подпрограммы ВП, функции или группы подпрограмм ВП и функций с полем ввода данных тех же самых подпрограмм ВП, функций или их групп. Как и сдвиговый регистр, узел обратной связи сохраняет данные любого типа по завершению текущей итерации и передает эти значения в следующую итерацию. Использование узлов обратной связи позволяет избежать большого количества проводников данных и соединений.

Можно поместить узел обратной связи внутри цикла **While** или **For**, выбрав **Feedback Node** в палитре **Structures**. При помещении узла обратной связи на проводник данных до ответвления, передающего данные на выходной терминал цикла, узел обратной связи передает все значения на выходной терминал цикла. При помещении узла обратной связи на проводник после ответвления, передающего данные на выходной терминал цикла, узел обратной связи передаст все значения обратно на поле ввода

данных ВП или функции, а затем передаст последнее значение на выходной терминал цикла.

2.2. Принятие решений в ВП и структуры

2.2.1. Функция Select и принятие решений

Каждый ВП до этого места курса выполнялся в порядке, определяемом потоком данных. Однако, бывают случаи, когда по ходу программы должно быть принято решение. Например, если происходит событие А, то необходимо сделать В, а если происходит С – то D. В программах, написанных на текстовых языках программирования, эта задача решается операторами if – else, операторами case, switch и т.д. В LabVIEW реализовано много различных способов принятия решений. Самый простой из них – функция **Select**.

Функция Select

Функция **Select**, расположенная в палитре **Functions»Programming»Comparison**, в зависимости от значения на логическом входе выбирает одно из двух значений. Если на логическом входе будет значение TRUE, то на выходе функция выдаст значение, поданное на вход «**t**», если же на логическом входе FALSE, то возвращается значение с поля «**f**». Блок-диаграмма ВП, в котором применена функция **Select**, приведена на рисунке 2.8. При необходимости принимать более сложные решения может понадобиться структура **Case**.

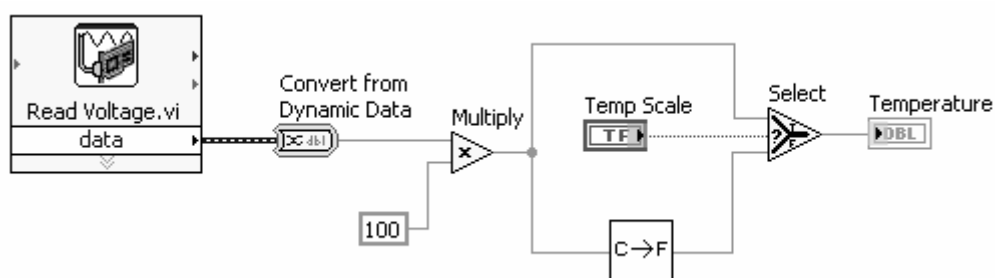


Рис. 2.8. Применение функции Select.

2.2.2. Структура Case

Структура Case (рис 2.9) имеет две или более поддиаграммы вариантов. Только одна поддиаграмма варианта видима в данный момент времени и только одна поддиаграмма варианта работает при выполнении данной структуры. Входное значение терминала селектора структуры определяет, какая поддиаграмма будет выполняться в данный момент времени. Структура **Case** аналогична операторам case или логическим операторам (if...then...else) в текстовых языках программирования.

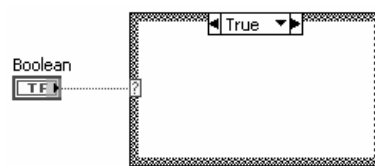
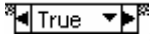



Рис. 2.9. Структура Case.

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

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

Для использования структуры **Case** необходимо отметить вариант по умолчанию. Вариант по умолчанию или поддиаграмма по умолчанию

выполняется, если значение терминала варианта выходит за пределы диапазона или не существует вариантов для возможных значений терминала варианта.

Щелчок правой кнопки мыши на границе структуры **Case** позволяет добавлять, дублировать, перемещать и удалять варианты (поддиаграммы), а также отмечать вариант по умолчанию.

Выбор варианта

В качестве примера использования структуры **Case** вместо функции **Select** приведена блок-диаграмма (рис. 2.10). На переднем плане структуры **Case** показан логический вариант TRUE.

Определение варианта осуществляется либо выбором значения на селекторе структуры **Case**, либо вводом значения с помощью инструмента **ВВОД ТЕКСТА**. При выборе какого-либо варианта, он появляется на переднем плане, как показано на рисунке.

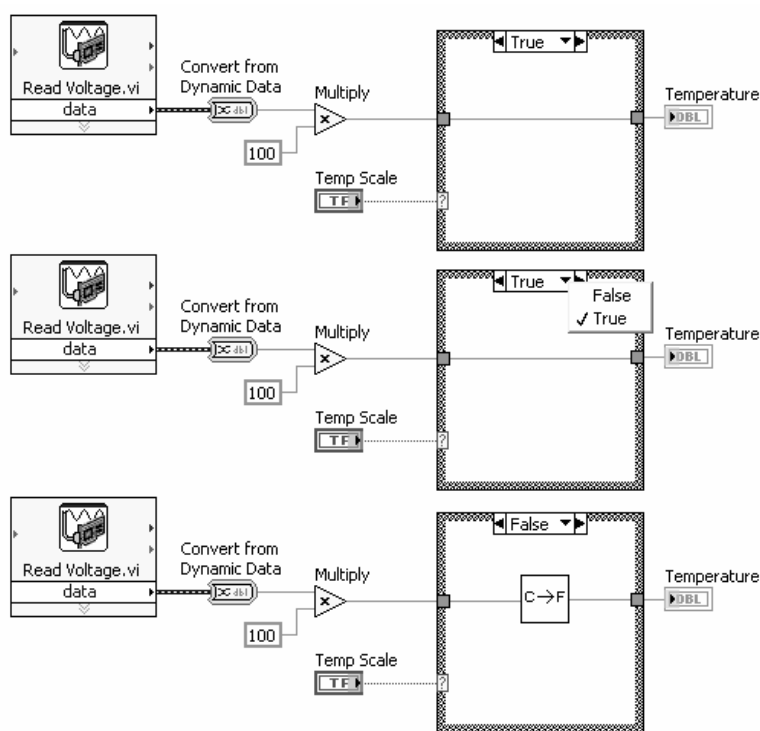


Рис. 2.10. Переключение между вариантами структуры Case.

Значения селектора варианта должны быть того же типа, что и тип данных, подаваемых на терминал селектора варианта. Значение селектора варианта, окрашенное красным цветом, показывает, что его необходимо удалить или отредактировать, иначе ВП не будет выполняться. Нельзя подавать числа с плавающей точкой на терминал селектора варианта, так как возможны ошибки округления и возникновение ситуации неопределенности. Если подать число с плавающей точкой на терминал селектора варианта, LabVIEW округлит это значение до ближайшего четного целого. Если число с плавающей точкой введено непосредственно в селектор варианта, то оно окрашивается в красный цвет и должно быть удалено или отредактировано.

Терминалы входа и выхода

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

Для определения значения выходного терминала следует правым щелчком мыши по терминалу вызвать контекстное меню и выбрать пункты: **Create»Constant** или **Create»Control**.

Следующие примеры (рис. 2.11) показывают, как значения входных терминалов структуры **Case** складываются или вычитаются в зависимости от значения терминала варианта.

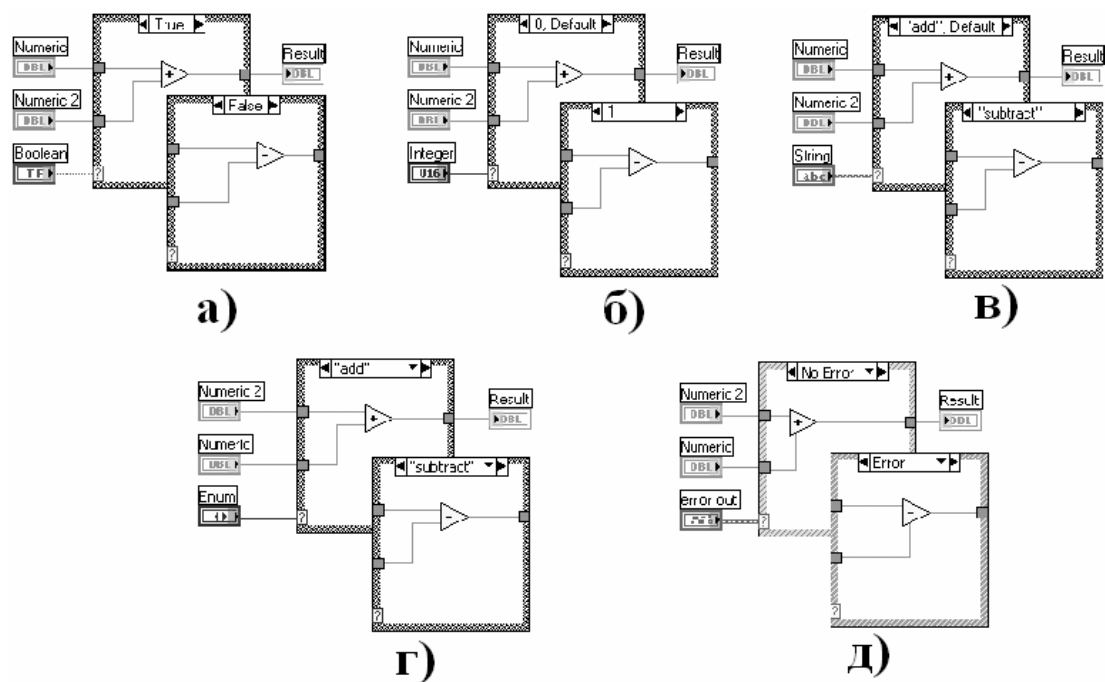


Рис. 2.11. Использование различных типов данных для управления структурой Case.

Логическая структура Case

На рисунке 2.11, а) приведен пример логической структуры **Case**. Варианты структуры наложены друг на друга для упрощения иллюстрации.

Если в терминал логического элемента управления, соединенный проводником данных с терминалом селектора варианта, введено значение **TRUE**, то выполняется сложение; если введено значение **FALSE**, то выполняется вычитание значений числовых элементов управления.

Целочисленная структура Case

На рисунке 2.11, б) показан пример целочисленной структуры **Case**. Терминал **Integer** соответствует элементу управления **ring control** (списка с циклическим перебором значений), расположенному в палитре **Controls»Modern»Ring & Enum**. Если значение элемента управления **ring control** равно «**0**» (сложить), то ВП складывает числа; если равно «**1**» (вычесть), то ВП производит вычитание чисел. Если значение элемента

управления отлично от «0» (сложить) и «1» (вычесть), то ВП складывает числа, т.к. этот вариант выполняется по умолчанию.

Строковая структура Case

На рисунке 2.11, в) показан пример строковой структуры **Case**. Если в поле элемента управления введена строка «**add**», то ВП производит сложение чисел и вычитает их, если введено значение «**subtract**».

Структура Case по перечислениям

На рисунке 2.11, г) показан пример структуры **Case** по перечислениям.

Структура Case для кластера ошибок

На рисунке 2.11, д) показан пример структуры **Case** для кластера ошибок. В этом примере на терминал селектора структуры **Case** подается кластер ошибок **error out**. В этом случае есть только два варианта структуры: «Ошибка» и «Нет ошибки», для которых граница структуры имеет красный и зеленый цвет соответственно. Структура **Case** выполняет вариант, основываясь на информации о наличии ошибки. Структура **Case** реагирует только на логическую переменную **status** кластера ошибок (более подробно о кластерах см. Раздел 3.2).

2.2.3. Узел Формулы

Узел Формулы (**Formula Node**) используется для выполнения математических операций в текстовом виде на блок-диаграмме. Использовать узел Формулы удобно, когда выражения имеют много переменных, или они достаточно сложные. Для ускорения процесса создания алгоритма можно копировать и вставлять имеющиеся текстовые математические коды в узел Формулы вместо их воссоздания на блок-диаграмме.

Создание терминалов входных и выходных данных узла Формулы осуществляется щелчком правой кнопки мыши по границе узла. В

контекстном меню необходимо выбрать пункты **Add Input** или **Add Output**, а затем ввести переменные для входа и выхода. Далее вводится уравнение в рабочую область структуры. Каждое выражение должно заканчиваться разделителем (;). Узел Формулы может также использоваться для принятия решений. На блок-диаграмме (рис 2.12) показаны два эквивалентных способа применения операторов if – then в узле Формулы.

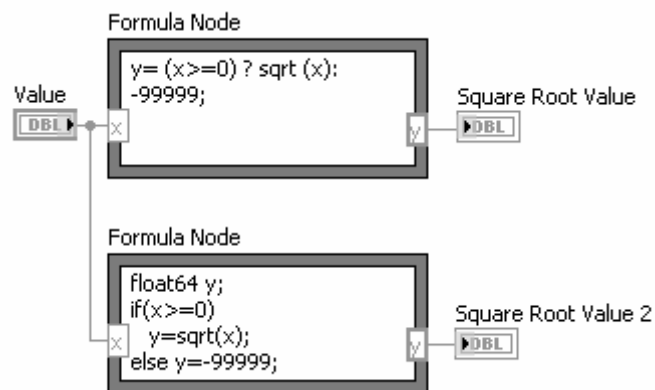


Рис. 2.12. Узел Формулы.

Узел Формулы позволяет производить разнообразные математические операции. Для получения более подробной информации о функциях, операциях и синтаксисе узла Формулы используйте справку **LabVIEW Help**.

ВП **Formula Express**, расположенный в палитре **Functions»Express»Arith/Compare**, является, по сути, встроенным в LabVIEW научным калькулятором. Он может выполнять большинство операций, выполняемых узлом Формулы, однако только по одной операции за раз. Для получения более подробной информации о ВП **Formula Express** используйте справку **LabVIEW Help**.

2.2.4. Узел Математики (MathScript Node)

Подобно узлу Формул узел Математики используется для выполнения математических операций в текстовом виде на блок-диаграмме. Как и при использовании узла Формул, вы можете передавать и получать данные из узла кода. Однако в отличие от узла Формул, узел Математики обладает более расширенными возможностями. Используя узел Математики, вы сможете импортировать в LabVIEW код, написанный в **LabVIEW MathScript**, **MATLAB** или **Xmath**. На рисунке 2.13 приведен пример окна Математики в LabVIEW:

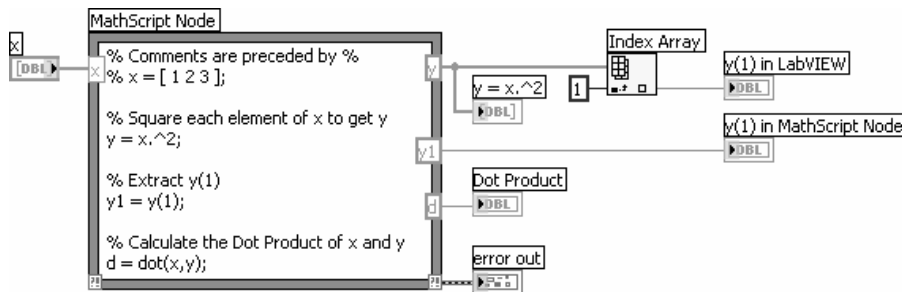


Рис. 2.13. Использование узла Математики.

Вы можете использовать узел **MathScript Node** для создания, загрузки и редактирования кодов, написанных на **MATLAB**, даже если **MATLAB** не установлен на вашем компьютере. Однако надо иметь в виду, что **MathScript** поддерживает не все функции, поддерживаемые **MATLAB**.

Если же у вас на компьютере установлена **Xmath** или **MATLAB** версии 6.5 и выше, то вы можете использовать любые части кода, написанные на **Xmath** или **MATLAB**.

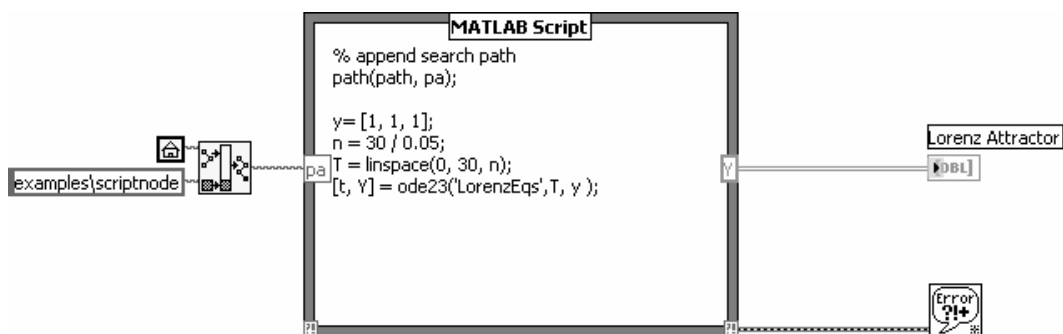


Рис. 2.14. Использование узла MathScript Node.

MATLAB и Xmath инсталлируют свои программы обработки кода, LabVIEW общается с данными программами с помощью соответствующего узла кода. Вы можете заменить используемую программу обработки кода, например конвертировать Xmath Script Node в MATLAB Script Node. Для этого нажмите правой кнопкой мыши на границу узла кода и выберите из контекстного меню **Choose Script Server»Xmath Script Node** или **Choose Script Server»MATLAB Script Node**. Для **MathScript Node** изменить программу обработки кода нельзя.

Узлы **MATLAB Script Node** и **Xmath Script Node** находятся на палитре **Functions»Mathematics»Scripts & Formulas»Script Nodes**. Узел **MathScript Node** находится на палитре **Functions»Mathematics»Scripts & Formulas**, либо на палитре **Functions»Programming»Structures**.

Использование окна LabVIEW MathScript

Для создания математического кода в LabVIEW вы также можете использовать **LabVIEW MathScript Window**. Чтобы открыть окно **LabVIEW MathScript Window**, выберите на линейке инструментов пункт **Tools»MathScript Window**. **LabVIEW MathScript Window** генерирует значения на выходе, содержит историю вызываемых команд, список переменных, которые вы определяете, и отображает выбранные переменные.

Вы можете сохранить скрипты, созданные в **LabVIEW MathScript Window**, и загрузить их в **MathScript Node**. И наоборот, скрипты, созданные и сохраненные в **MathScript Node**, можно загрузить в **LabVIEW MathScript Window**.

Для большей информации о **LabVIEW MathScript** обратитесь к *LabVIEW Help*. В содержании найдите пункт **Fundamentals»Formulas and Equations**.

3. Группирование данных и графическое отображение

3.1. Массивы

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

Данные, составляющие массив, могут быть любого типа: целочисленного, логического или строкового. Массив также может содержать элементы графического представления данных и кластеры. Использовать массивы удобно при работе с группами данных одного типа и при накоплении данных после повторяющихся вычислений. Массивы идеально подходят для хранения данных, полученных с графиков, или накопленных во время работы циклов, причем одна итерация цикла создает один элемент массива.

Нельзя создать массив, состоящий из массивов. Однако можно создать массив кластеров, где каждый кластер будет состоять из одного или более массивов. Более подробную информацию по этому вопросу можно найти в Разделе 3.2, *Кластеры*.

Все элементы массива упорядочены. Чтобы к ним было легко обращаться, каждому элементу присвоен индекс. Нумерация элементов массива всегда начинается с 0. Таким образом, индексы массива находятся в диапазоне от 0 до $(n-1)$, где n – число элементов в массиве.

Например, в массиве из девяти планет солнечной системы $n=9$, следовательно, значение индекса находится в пределах от 0 до 8. Земля является третьей планетой от Солнца, поэтому ее индекс равен 2.

Создание массива элементов управления и отображения

Для создания массива элементов управления или отображения данных необходимо выбрать шаблон массива из палитры **Controls»Modern»Array, Matrix & Cluster** и поместить его на лицевую панель. Затем поместить внутрь шаблона массива элемент управления либо отображения данных. Поместить в шаблон массива запрещенный элемент управления или отображения, например двухкоординатный график осциллограмм (**XY graph**), не удастся.

Поместить объект в шаблон массива следует до того, как он будет использоваться на блок-диаграмме. Если этого не сделать, то шаблон массива не будет инициализирован, и использовать массив будет нельзя. При этом на диаграмме терминал массива будет окрашен в черный цвет, что означает неопределенный тип данных.

В двумерном (2D) массиве элементы хранятся в виде матрицы. Таким образом, для размещения элемента требуется указание индекса столбца и строки. На рисунке 3.1 показан двумерный массив, состоящий из 6 столбцов (длина) и 4 строк (высота). Количество элементов в массиве равно 24 ($6 \times 4 = 24$).

		Индекс колонки					
		0	1	2	3	4	5
Индекс строки	0						
	1						
	2						
	3						

Рис. 3.1. Расположение индексов двумерного массива.

Для увеличения размерности массива необходимо щелкнуть правой кнопкой мыши по элементу индекса и выбрать из контекстного меню пункт **Add Dimension**. С этой целью также можно использовать инструмент ПЕРЕМЕЩЕНИЕ. Для этого надо просто изменить размер элемента индекса.

Создание массива констант

Создать массив констант на блок-диаграмме можно, выбрав в палитре **Functions»Programming»Array** шаблон **Array Constant** и поместив в него числовую константу. Массив констант удобно использовать для передачи данных в подпрограммы ВП.

3.1.1. Создание массивов с помощью цикла

Цикл **For** и цикл **While** могут автоматически накапливать массивы и проводить их индексацию на своих границах. Это свойство называется автоиндексацией. После соединения терминала данных массива с терминалом выхода из цикла каждая итерация цикла создает новый элемент массива. На рисунке 3.2 видно, что проводник данных, соединяющий терминал данных массива с терминалом выхода из цикла стал толще, а сам терминал выхода из цикла окрашен в цвет терминала данных массива.

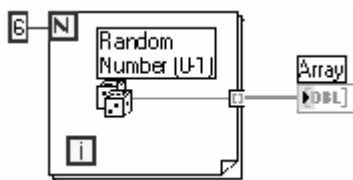


Рис. 3.2. Индексация на выходном терминале цикла FOR.

Автоиндексация отключается щелчком правой кнопки мыши по терминалу входа/выхода из цикла и выбором пункта контекстного меню **Disable Indexing**. Автоиндексацию следует отключать, например, в случае, когда нужно знать только последнее значение.

Ввиду того, что цикл **For** часто используется при работе с циклами, для него в LabVIEW автоиндексация включена по умолчанию. Для цикла **While** автоиндексация по умолчанию отключена. Для того чтобы включить автоиндексацию, необходимо щелкнуть правой кнопкой мыши по

терминалу входа/выхода из цикла и выбрать в контекстном меню пункт **Enable Indexing**.

Создание двумерных (2D) массивов

Для создания двумерных массивов необходимо использовать два цикла **For**, один внутри другого. Как показано на иллюстрации, внешний цикл создает элементы массива в строке, а внутренний цикл создает элементы массива в столбце.

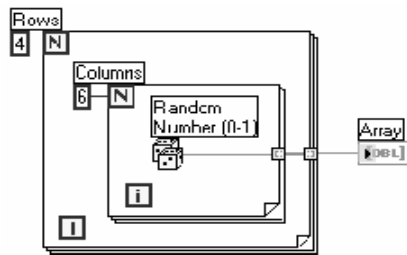


Рис. 3.3. Создание двумерного массива с помощью циклов FOR.

Использование автоиндексации для установки значения терминала количества итераций цикла

При включенной автоиндексации массива, подключенного к терминалу входа в цикл **For**, LabVIEW, автоматически устанавливает значение терминала количества итераций цикла **N**, равного размерности массива. Таким образом, отпадает необходимость задания значения терминалу **N**.

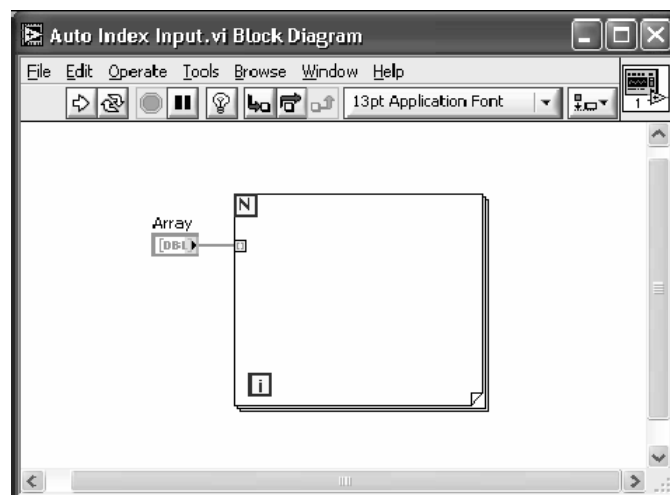


Рис. 3.4. Автоматическая индексация на входе цикла.


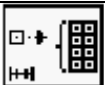
В примере на рисунке 3.4 цикл **For** будет выполнен ровно столько раз, сколько элементов в массиве. Как правило, стрелка на кнопке **Run** сломана, если терминал количества итераций цикла не подключен, однако в этом примере стрелка цела, что говорит о возможности запуска ВП.



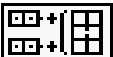
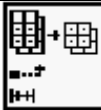

Если автоиндексация установлена более чем для одного терминала входа в цикл или явно задано значение терминала количества итераций цикла **N**, то значением терминала **N** станет меньшая из величин. Например, если соединить массив из 10 элементов с терминалом входа в цикл, а значение терминала количества итераций установить равным 15, то цикл выполнит 10 итераций.

3.1.2. Функции работы с массивами

Для создания и управления массивами используются функции, расположенные в палитре **Functions»Programming»Array**.

Таблица 3.1. Наиболее часто используемые функции работы с массивами.

	<p>Array Size – показывает количество элементов массива каждой размерности. Если массив n-мерный, на выходе функции Array Size будет массив из n элементов. Например, для приведенного ниже массива функция Array Size выдаст значение 3.</p> <table border="1" data-bbox="371 1420 671 1476"> <tr> <td>2</td> <td>4</td> <td>3</td> </tr> </table>	2	4	3
2	4	3		
	<p>Initialize Array – создает n-мерный массив, в котором каждый элемент инициализирован значением поля ввода данных element. Для увеличения размерности массива достаточно добавить поля ввода данных, растянув узел функции. Например, если для функции Initialize Array заданы следующие значения параметров: на поле element подается значение 5, а на поле dimension size (если оно одно) – значение 3, то на выходе получится массив, показанный ниже.</p> <table border="1" data-bbox="371 1883 671 1935"> <tr> <td>5</td> <td>5</td> <td>5</td> </tr> </table>	5	5	5
5	5	5		

  	<p>Build Array – объединяет несколько массивов или добавляет элемент в n-мерный массив. Изменение размера функции увеличивает количество полей ввода данных, что позволяет увеличить количество добавляемых элементов. Например, если объединить два предыдущих массива, то функция Build Array выдаст на выходе следующий массив:</p> <table border="1" data-bbox="371 555 671 667"> <tr> <td>2</td> <td>4</td> <td>3</td> </tr> <tr> <td>5</td> <td>5</td> <td>5</td> </tr> </table> <p>Для объединения входных данных в более длинный массив той же размерности, как показано ниже, достаточно щелкнуть правой кнопкой мыши на функции и выбрать из контекстного меню пункт Concatenate Inputs.</p> <table border="1" data-bbox="371 887 962 943"> <tr> <td>2</td> <td>4</td> <td>3</td> <td>5</td> <td>5</td> <td>5</td> </tr> </table>	2	4	3	5	5	5	2	4	3	5	5	5
2	4	3											
5	5	5											
2	4	3	5	5	5								
	<p>Array Subset – выдает часть массива, начиная с индекса, поступившего на поле index. Длина возвращаемой части массива соответствует значению, указанному в поле length. Например, если подать предыдущий массив на поле ввода функции Array Subset, значение 2 – на поле index и 3 – на поле length, то на поле вывода данных будет следующее:</p> <table border="1" data-bbox="371 1283 671 1339"> <tr> <td>3</td> <td>5</td> <td>5</td> </tr> </table>	3	5	5									
3	5	5											
	<p>Index Array – выдает элемент, соответствующий индексу, значение которого подается на поле ввода index. Например, при использовании предыдущего массива, функция Index Array выдаст значение 2, если на поле ввода данных index подать значение 0.</p> <p>Функцию Index Array можно использовать для выделения строки или столбца из двумерного массива и дальнейшего отображения в виде подмассива. Для этого двумерный массив надо подать в поле ввода данных функции. Функция Index Array должна иметь два поля index. Верхнее поле index указывает строку, а нижнее – столбец. Можно задействовать оба поля index для выбора отдельного элемента или только одно для выбора строки или столбца. Например, в поле ввода данных функции подается массив, показанный ниже.</p>												

	2	4	3
	5	5	5
Функция Index Array в поле вывода данных выдаст следующий массив в случае, если на поле index (строка) подается значение 0.			
	2	4	3

3.1.3. Полиморфизм

Арифметические функции, расположенные в палитре **Functions»Programming»Numeric**, являются полиморфными. Это означает, что на поля ввода этих функций могут поступать данные различных типов (скалярные величины, массивы). Например, можно использовать функцию **Add** для прибавления скалярной величины к массиву или сложения двух массивов. Если на одно поле ввода данных функции **Add** подать скалярную величину 2, а другое соединить с массивом, показанным ниже,

2	4	3
---	---	---

то функция прибавит 2 к каждому элементу массива, и массив будет иметь вид:

4	6	5
---	---	---

Если на вход функции **Add** подать два предыдущих массива, функция сложит каждый элемент первого массива с соответствующим элементом второго и выдаст результат в виде массива, показанного ниже.

6	10	8
---	----	---

Если с помощью функции **Add** сложить два массива разной размерности, например, таких, как предыдущий и показанный ниже,

1	3	2	4
---	---	---	---

то функция сложит каждый элемент первого массива с соответствующим элементом второго и выдаст результат в виде массива размерностью меньшей из двух исходных.

7	13	10
---	----	----

С кластерами арифметические функции работают таким же образом. Подробнее о работе с кластерами можно узнать из Раздела 3.2. *Кластеры*.

3.2. Кластеры

3.2.1. Что такое кластер

Кластеры объединяют элементы разных типов данных, подобно пучку проводов телефонного кабеля, где каждый провод представляет собой отдельный элемент кластера. Кластеры играют ту же роль, что и структуры в языках программирования.

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

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

Создание кластеров из элементов управления и отображения данных

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



Рис. 3.5. Пример отображения кластера на лицевой панели.

Порядок элементов в кластере

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

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

Посмотреть и изменить порядковый номер объекта, помещенного в кластер, можно, щелкнув правой кнопкой мыши по краю кластера и выбрав из контекстного меню пункт **Reorder Controls In Cluster**. Панель инструментов и кластер примут вид, показанный на рисунке 3.6.

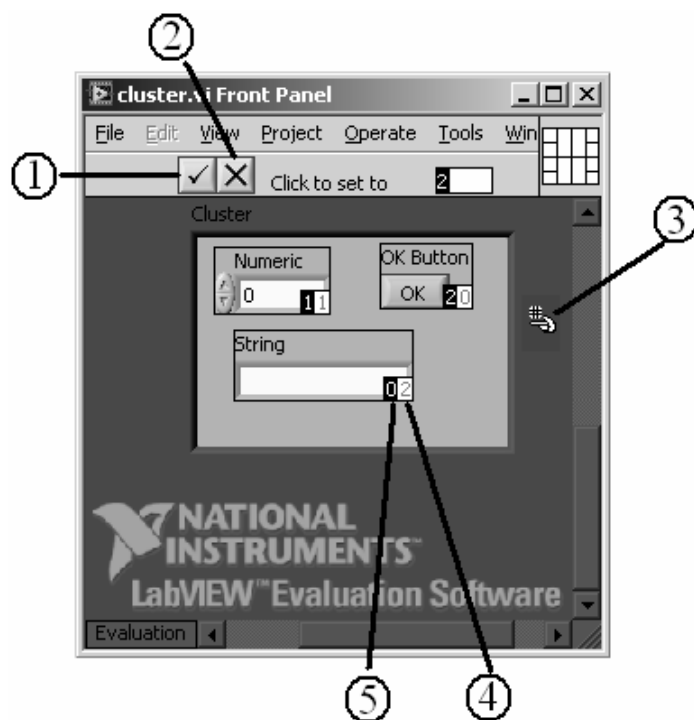


Рис. 3.6. Изменение порядкового номера элемента в кластере.

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

В белом поле (4) указан текущий порядковый номер элемента, в черном

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

Соответствующие элементы, определенные в кластерах одинаковыми порядковыми номерами, должны иметь совместимые типы данных. Например, в одном кластере элемент 0 является числовым элементом

управления, а элемент 1 – строковым элементом управления. Во втором кластере элемент 0 – числовой элемент отображения данных и элемент 1 – строковый элемент отображения данных. Кластер элементов управления корректно соединится с кластером элементов отображения данных.

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

Создание кластера констант

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

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

3.2.2. Функции работы с кластерами

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

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

Name и **Unbundle by Name** в полях ввода/вывода данных содержат имя первого элемента кластера.

Сборка кластеров

Для сборки отдельных элементов в кластер используется функция **Bundle**. Эта же функция используется для изменения данных в элементе уже существующего кластера. Инструмент ПЕРЕМЕЩЕНИЕ используется для добавления полей ввода данных, для этого также можно щелкнуть правой кнопкой по полю ввода данных и выбрать из контекстного меню пункт **Add Input**. При соединении кластера с полем ввода данных **cluster** количество полей ввода данных функции должно соответствовать количеству элементов во входящем кластере. На поле ввода данных **cluster** можно подать только одну требующую замены компоненту. Например, на рисунке 3.7 показан кластер, имеющий три элемента управления.

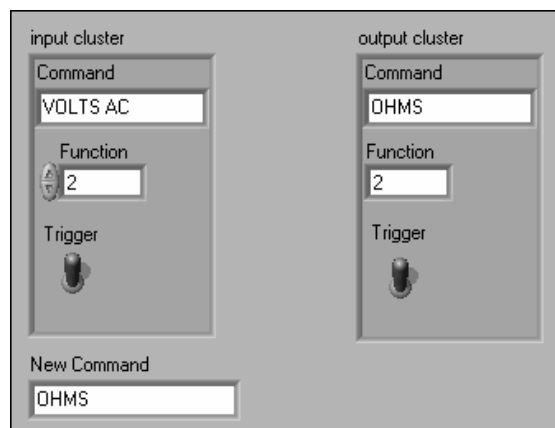


Рис. 3.7. Кластеры.

Если известен логический порядок элементов, можно использовать функцию **Bundle** для изменения значения элемента **Command**, соединив элементы, как показано на рисунке 3.8.

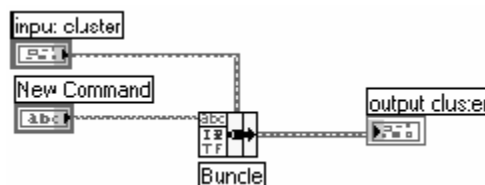


Рис. 3.8. Сборка кластера.

Замена или доступ к элементам кластера

Для замены элемента в уже существующем кластере используется функция **Bundle by Name**. Функция **Bundle by Name** работает так же, как функция **Bundle**, но вместо обращения к элементу кластера по его порядковому номеру обращается к нему по его собственной метке (имени). При этом можно получить доступ только к элементам, имеющим собственную метку. Количество полей ввода данных не требует соответствия с количеством элементов в кластере. С помощью элемента УПРАВЛЕНИЕ можно щелкнуть по полю ввода данных терминала и выбрать желаемый элемент из выпадающего меню. Можно также щелкнуть правой кнопкой мыши по полю ввода данных и выбрать элемент в разделе контекстного меню **Select Item**. На рисунке 3.9 показано, как можно использовать функцию **Bundle by Name** для изменения значений элементов **Command** и **Function**.

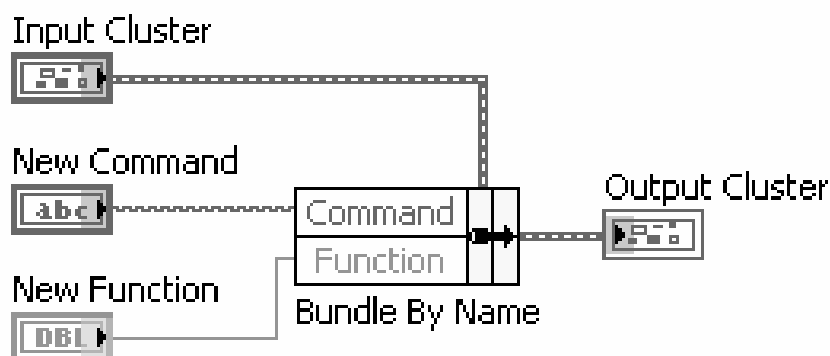


Рис. 3.9. Замена элемента кластера.

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

Разделение кластера

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

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

Например, функция **Unbundle**, при использовании кластера, показанного ниже, имеет четыре поля вывода данных, которые соотносятся с четырьмя элементами кластера. Необходимо знать порядок элементов в кластере для корректного сопоставления логического элемента соответствующему вертикальному переключателю в кластере. В этом примере элементы упорядочены сверху вниз, начиная с 0. Если использовать функцию **Unbundle by Name**, то полей вывода данных может быть произвольное количество и обращаться к отдельным элементам можно в произвольном порядке.

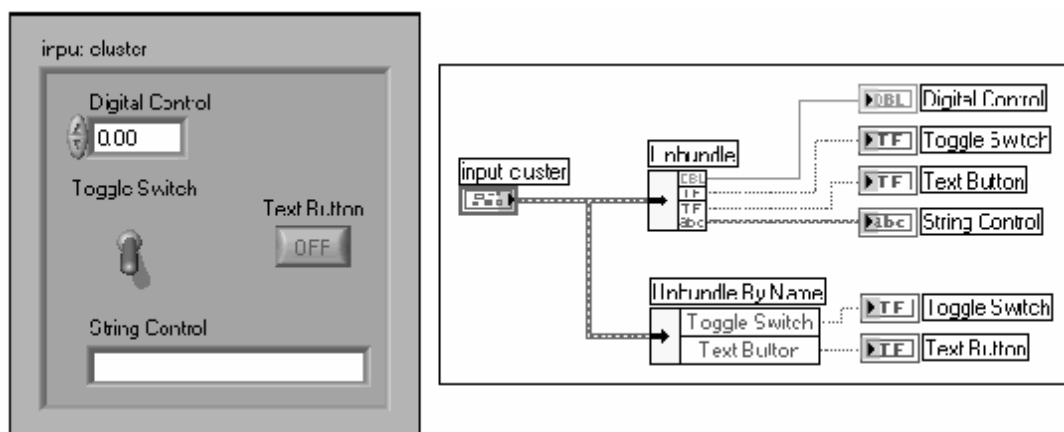


Рис. 3.10. Использование функции Unbundle by Name.

3.2.3. Кластеры ошибок

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

При программировании любых операций ввода/вывода стоит подумать о возможном появлении ошибок. Почти все операции ввода/вывода возвращают информацию об ошибке. Чтобы правильно обрабатывать ошибки, в ВП нужно особо тщательно выполнять проверку для таких операций ввода/вывода, как файловые и последовательные операции, операции работы с приборами, операции получения данных, а также процессы передачи информации. Проверка на ошибки в ВП может помочь определить следующие проблемы:

- Неправильная инициализация связи с внешним устройством или запись в него некорректной информации.
- Внешнее устройство не включено или не работает.
- При переустановке системного программного обеспечения или по другим причинам был изменен путь к необходимым файлам.

Обработка ошибок

В LabVIEW не реализована автоматическая обработка ошибок. Это сделано для того, чтобы можно было самостоятельно выбирать метод, которым обрабатываются ошибки. Например, если для ВП истекло время ожидания ввода/вывода, можно сделать так, чтобы не прекращалась работа всего приложения. Можно также заставить ВП повторить попытку через некоторое время. Процесс обработки ошибок в LabVIEW происходит на блок-диаграмме.

Существует два способа возврата ошибок в ВП и функциях: с помощью числа, обозначающего код ошибки и с помощью кластера

ошибок. Как правило, функции используют число – код ошибки, а ВП принимают на вход и выдают на выходе информацию об ошибках в виде кластера. Обработка ошибок в LabVIEW также построена на модели поточного программирования. Как и другие данные, информация об ошибках проходит через ВП. Для передачи информации об ошибках через ВП необходимо использовать входной и выходной кластеры ошибок, а также включить в конце ВП обработчик ошибок для определения того, были ли сбои в процессе работы ВП.

При выполнении ВП LabVIEW следит за появлением ошибок, и как только где-нибудь происходит сбой, составляющие части ВП перестают выполняться и только передают ошибку дальше, на выход. Для обработки появляющихся в ВП ошибок в конце потока выполнения обычно используется приведенный на рисунке 3.11 простой обработчик ошибок **Simple Error Handler**, который находится на палитре **Functions»Programming»Dialog & User Interface**. Подсоедините кластер ошибок к полю входных данных «**Error In**» (по умолчанию ошибки нет).

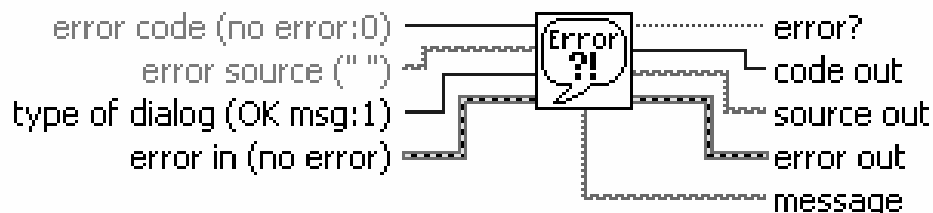


Рис. 3.11. Простейшая функция обработки ошибок.

На рисунке 3.12 приведены компоненты кластеров ошибок, расположенных на палитре **Controls»Modern»Array, Matrix & Cluster**. Кластер ошибок и проводники, соединяющие кластеры ошибок, обозначаются желтым цветом.

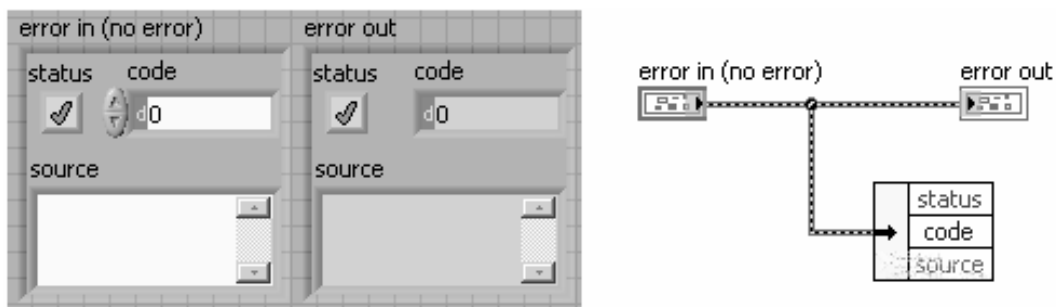


Рис. 3.12. Кластеры ошибок.

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

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

Объяснение ошибки

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

Использование цикла While при обработке ошибок

Кластер ошибок может быть подсоединен к терминалу условия цикла **While** для остановки цикла (рис. 3.13). Когда кластер ошибок подсоединен к терминалу условия, на терминал подаются только значения параметра **status** – TRUE или FALSE. При возникновении ошибки выполнение цикла **While** прекращается.

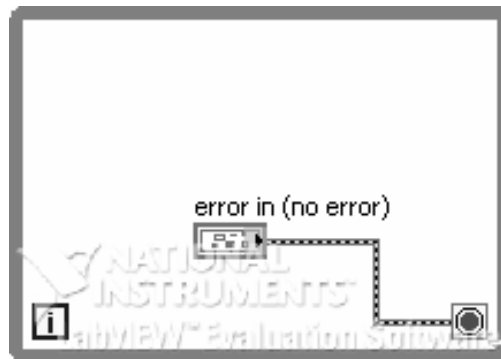


Рис. 3.13. Остановка цикла по статусу ошибки.

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

3.3. Графическое отображение данных

3.3.1. Использование графика Диаграмм

График Диаграмм (**Waveform Chart**) – специальный элемент отображения данных в виде одного и более графиков. График Диаграмм расположен на палитре **Controls»Modern»Graph**. На рисунке 3.14 показан пример Графика Диаграмм с двумя графиками: экспериментальные данные и их бегущее среднее значение.

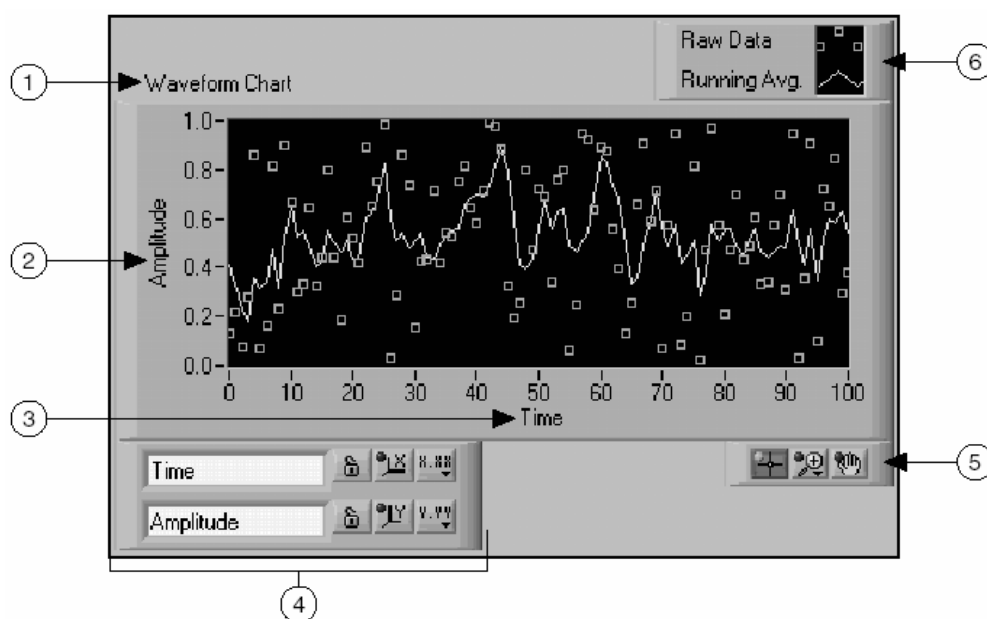


Рис. 1.14. Элементы и обозначения на графике Диаграмм.

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

График Диаграмм использует три различных режима отображения данных: **strip chart**, **scope chart** и **sweep chart** (рис. 3.15). Режим по умолчанию – **strip chart**.

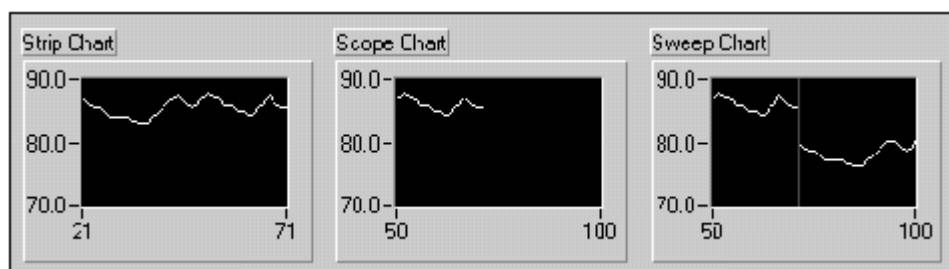


Рис. 3.15. Режимы обновления графика Диаграмм.

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

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

Соединение графиков

Для создания диаграмм достаточно соединить поле вывода скалярной величины с терминалом данных графика Диаграмм. В примере на рисунке 3.16, а) тип данных на терминале графика Диаграмм соответствует входному типу данных (Double).

График Диаграмм может отображать несколько графиков. Для объединения отображаемых данных используется функция **Bundle**, расположенная в палитре **Functions»Programming»Cluster & Variant**. Например, блок-диаграмма, показанная на рисунке 3.16, б), с помощью функции **Bundle** объединяет выходные данные трех подпрограмм ВП для последующего отображения на графике Диаграмм.

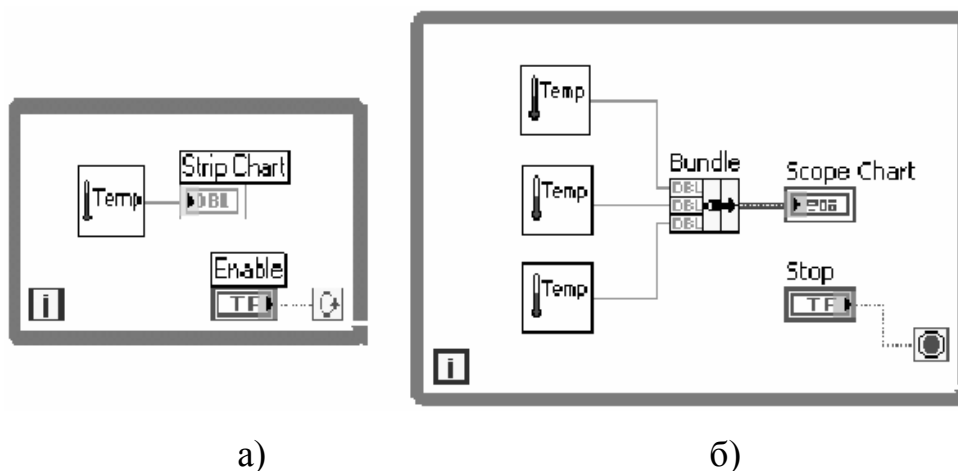


Рис. 3.16. Объединение графиков Диаграмм.

Терминал данных графика Диаграмм имеет кластерный тип данных в соответствии с полем вывода функции **Bundle**. Для увеличения количества полей ввода данных функции **Bundle** необходимо с помощью инструмента ПЕРЕМЕЩЕНИЕ изменить количество ее терминалов.

3.3.2. График Осциллограмм и двухкоординатный график

Осциллограмм

С помощью графиков в виде осциллограмм ВП обычно отображает накопленные в массив данные. На рисунке 3.17 показаны элементы графика.

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

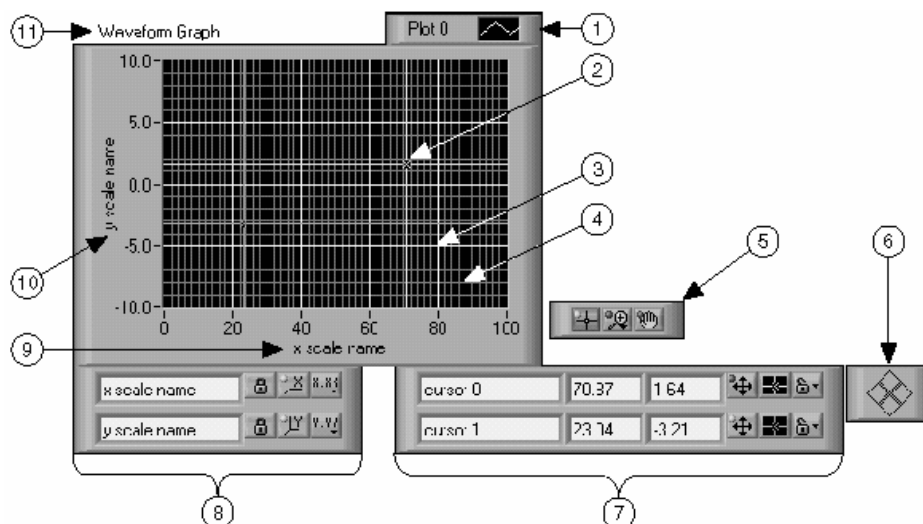


Рис. 3.17. Элементы и обозначения на графике Осциллограмм.

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

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

Одиночный график Осциллограмм

Одиночный график Осциллограмм работает с одномерными массивами и представляет данные массива в виде точек на графике с приращением по

оси X равным 1 и началом в точке $x = 0$. Графики также отображают кластеры с установленным начальным значением x , Δx и массивом данных по шкале y . В качестве примера можно рассмотреть ВП **Waveform Graph VI** в разделе библиотеки примеров (*examples\general\graphs\gengraph.llb*).

График множества Осциллограмм

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

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

В качестве примера можно рассмотреть график **(Y) Multi Plot 1**, открыв ВП **Waveform Graph VI** из библиотеки примеров (*examples\general\graphs\gengraph.llb*).

Графики множества Осциллограмм отображают кластеры, состоящие из начального значения x , Δx и двумерного массива данных по шкале y . График представляет данные по шкале y в виде точек с приращением Δx по оси x и началом в точке $x=0$. В качестве примера можно рассмотреть график **(Xo, dX, Y) Multi Plot 3**, открыв ВП **Waveform Graph VI** из библиотеки примеров (*examples\general\graphs\gengraph.llb*).

Графики множества Осциллограмм отображают также и кластеры с установленным начальным значением x , Δx и массивом данных, содержащим кластеры. Каждый кластер содержит массив точек, отображающих данные по шкале y . Для создания массива кластеров следует использовать функцию **Bundle**, которая объединяет массивы в кластеры.

Далее с помощью функции **Build Array** создается массив кластеров. Можно также использовать функцию **Build Cluster Array**, которая создает массив кластеров с определенными полями ввода данных. В качестве примера можно рассмотреть график **(Xo, dX, Y) Multi Plot 2**, открыв ВП **Waveform Graph VI** из библиотеки примеров (*examples\general\graphs\gengraph llb*).

Двухкоординатные графики множества Осциллограмм

Двухкоординатные графики множества Осциллограмм работают с массивами осциллограмм, в которых осциллограмма данных является кластером, содержащим массивы значений x и y . Двухкоординатные графики множества Осциллограмм воспринимают также массивы множества осциллограмм, где каждая осциллограмма представляет собой массив точек.

Каждая точка – это группа данных, содержащая значения по x и y . В качестве примера можно рассмотреть ВП **XY Graph VI** из библиотеки примеров (*examples\general\graphs\gengraph llb*).

3.3.3. График интенсивности

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

На рисунке 3.18 изображен массив размера 4x3, визуализированный на графике интенсивности. График отображает транспонированный массив.

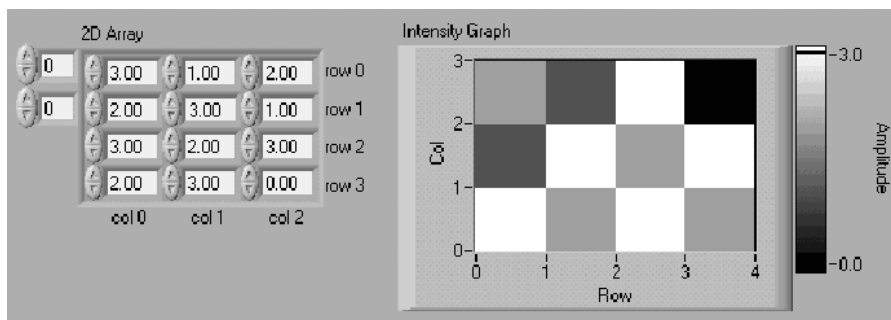


Рис. 3.18. Пример визуализации массива с помощью графика интенсивности.

Настройки графиков и таблиц интенсивности

Графики и таблицы интенсивности имеют много общих свойств с графиками Диаграмм и Осциллограмм, которые можно отобразить или спрятать, выбрав пункт контекстного меню **Visible Items**. Так как в графиках и таблицах интенсивности появляется третье измерение, то необходим дополнительный элемент – элемент управления цветовой шкалой, который определяет диапазон и способ цветового отображения данных. На рисунке 3.19 показаны составляющие части графика интенсивности.

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

или использовать инструмент ВВОД ТЕКСТА для ввода нового значения в текстовое поле маркера.

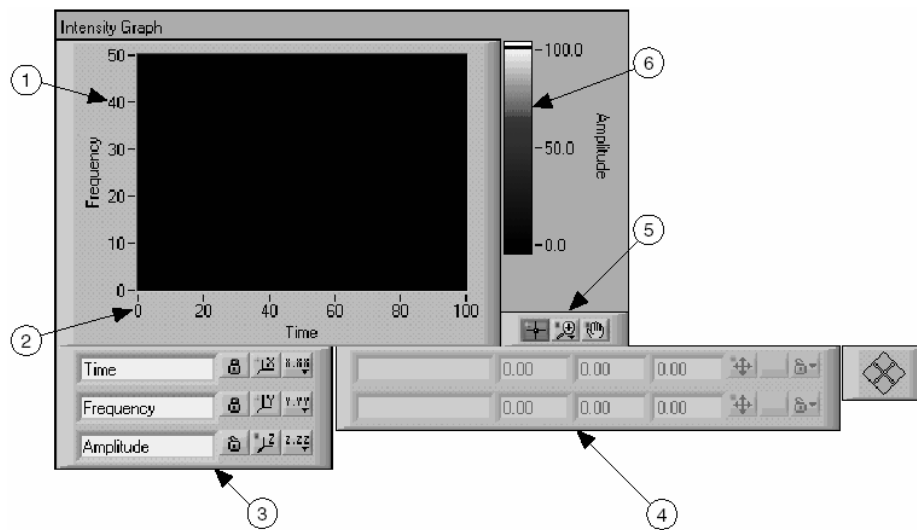


Рис. 3.19. Элементы и обозначения на графике Интенсивности.

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

4. Работа со строковыми данными и файлами

4.1. Строки. Функции работы со строками

Строки – это последовательность отображаемых и неотображаемых ASCII символов. Строки обеспечивают не зависящий от платформы формат обмена данными. Некоторые из наиболее распространенных строковых приложений включают в себя:

- Создание простых текстовых сообщений.
- Передача числовых данных в приборы в виде строк символов и преобразование строк в числовые данные.
- Сохранение числовых данных на диск. Чтобы сохранять числовые данные в виде файла ASCII, необходимо перед записью преобразовать их в строки.
- Диалоговые окна инструкций и подсказок.

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

4.1.1. Создание строковых элементов управления и отображения данных

Для работы с текстом и метками используются строковые элементы управления и отображения данных, расположенные в палитре **Controls»Modern»String & Path**. Создание и редактирование текста в строке производится с помощью инструментов **УПРАВЛЕНИЕ** и **ВВОД ТЕКСТА**. Для изменения размера строкового объекта на лицевой панели используется инструмент **ПЕРЕМЕЩЕНИЕ**. Для экономии места на лицевой панели можно использовать полосу прокрутки. Для этого необходимо щелкнуть правой кнопкой мыши по строковому объекту и выбрать в контекстном меню пункт **Visible Items»Scrollbar**.

Тип отображения строкового объекта выбирается в его контекстном меню.

Таблица 4.1. Типы отображения и примеры заполнения строки.

Тип отображения	Описание	Пример текста
Режим стандартного отображения (Normal Display)	Отображает стандартные ASCII коды, используя шрифт элемента управления. Управляющие коды для печати выводятся на экран в виде квадратов.	There are four display types. \ is a backslash
Режим отображения с обратным слэшем непечатаемых управляющих кодов (' Codes Display)	Выводит символ \ для всех непечатаемых управляющих кодов	There\sare\sfour\s display\stypes.\n\ \\sis\sasbackslash
Режим скрытого отображения текста (Password Display)	Выводит символ * для всех кодов текстового пространства	***** ***** *****
Режим отображения 16-тиричных ASCII кодов (Hex Display)	Выводит значение ASCII кода для каждого символа	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

Элемент управления Таблица, расположенный в палитре **Controls»Modern»List & Table** предназначен для создания таблиц на лицевой панели. Каждая ячейка находится в строке и столбце таблицы. Поэтому таблица отображает двумерный массив строк. На рисунке 4.1 показана таблица и ее составные части.

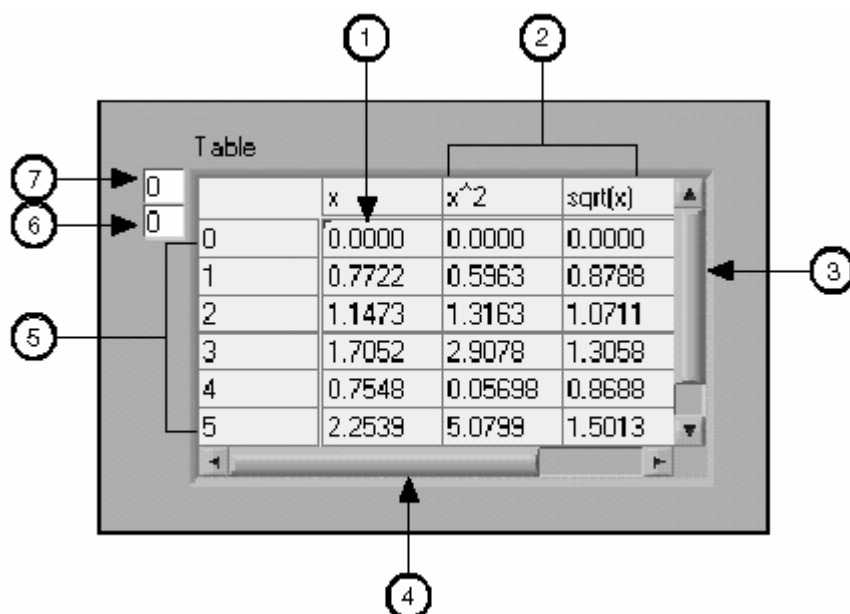


Рис. 4.1. Компоненты элемента управления Таблицы.

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

Для инициализации значений ячеек таблицы используется инструмент УПРАВЛЕНИЕ или ВВОД ТЕКСТА, с помощью которых достаточно ввести текст в выделенную ячейку.

Таблица – это двумерный массив строк. Таким образом, для использования таблицы в качестве элемента отображения данных, необходимо двумерный массив чисел преобразовать в двумерный массив строк. Заголовки строк и столбцов таблицы, как в таблице символов, автоматически не отображаются. Необходимо создать одномерный массив строк, содержащий заголовки строк и столбцов таблицы.

4.1.2. Использование некоторых функций обработки строк

Для редактирования и управления строками на блок-диаграмме следует пользоваться функциями обработки строк, расположенными в палитре **Functions»Programming»String**. Некоторые из функций работы со строками рассмотрены ниже:

- **String Length** – выдает количество символов в строке, включая пробелы. Например, функция **String Length** выдает значение 10 для приведенного ниже текста:

«Это пример»

- **Concatenate Strings** – объединяет строки и одномерные массивы строк в отдельную строку. Для увеличения полей ввода данных функции следует изменить ее размер. Например, объединение предыдущей строки со следующим массивом строк

«объединения »	«строки и »	«массива строк»
----------------	-------------	-----------------

Даст результат, показанный на рисунке 4.2 (т.е. «Это пример объединения строки и массива строк»).

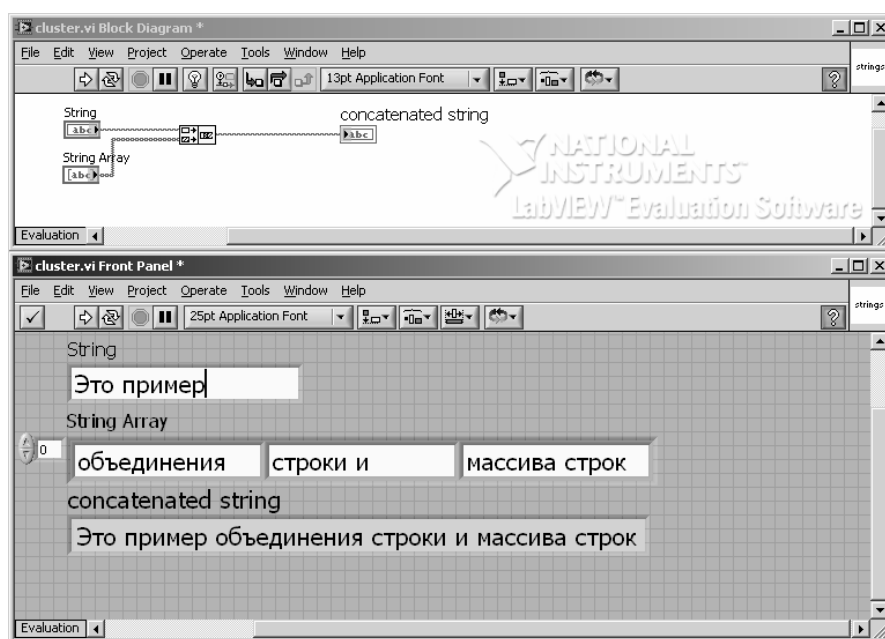


Рис. 4.2. Функция Concatenate Strings.

- **String Subset** – выдает подстроку определенной длины **length**, начиная со значения **offset** (смещение). Смещение первого элемента в строке равно 0. Например, если на поле ввода данных функции подать предыдущую строку, то функция **String Subset** при **offset = 4** и **length = 6** выдаст значение «пример».
- **Match Pattern** – ищет повторяющуюся последовательность, поданную на поле ввода данных **regular expression**, в строке начиная со значения смещения **offset**, и, если находит соответствие, разбивает строку на три подстроки. Если соответствие не найдено, поле вывода данных **match substring** является пустым, а значение поля вывода данных **offset past match** (смещение повторяющейся последовательности в строке) равно -1 . Например, на поле **regular expression** (шаблон подстроки) подается значение «:», а строка на входе «*VOLTS DC: +1.22863E+1*». Функция **Match Pattern** выдаст величины **before substring** (перед подстрокой) «*VOLTS DC*», **match substring** (шаблон подстроки) «:» и **after substring** (после подстроки) «*+1.22863E+1*», а также **offset past match**, равный 9.

4.1.3. Преобразование числовых данных в строку

Для преобразования числовых данных в строковые используются ВП **Build Text Express** и функция **Format Into String** (конвертирование в строку). Обе эти функции имеют входные и выходные кластеры ошибок.

При недостатке места на блок-диаграмме лучше использовать функцию **Format Into String**.

Экспресс-ВП **Build Text Express VI**

Экспресс-ВП **Build Text**, расположенный в палитре **Functions»Express»Output** производит объединение входных строк. Если

входные величины имеют нестроковый тип данных, то они преобразуются в строку в соответствии с настройками этого экспресс-ВП.

При помещении Экспресс-ВП **Build Text** на блок-диаграмму появляется диалоговое окно настроек **Configure Build Text**. В примере на рисунке 4.3 значение напряжения подается на вход экспресс-ВП и преобразуется к формату данных с плавающей запятой с 4-мя числами после запятой. Затем это значение добавляется к концу строки **Voltage is** (Напряжение равно).

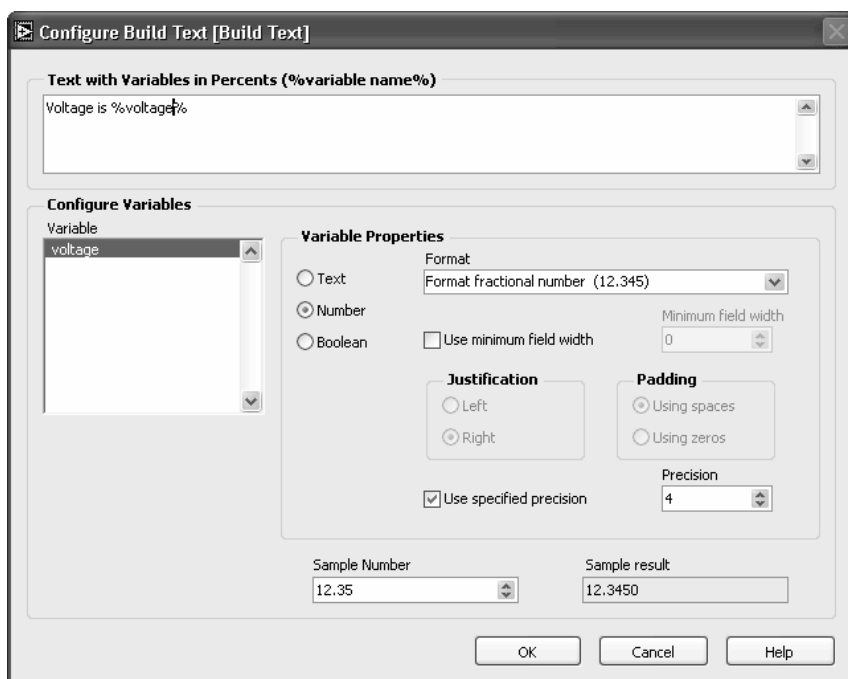


Рис. 4.3. Применение Экспресс-ВП Build Text Express VI.

При такой настройке экспресс-ВП на диаграмме выглядит, как показано на рисунке 4.4. Для наблюдения за выходной строкой используется отладочный индикатор. Любые значения, подаваемые на поле ввода данных **Beginning Text**, будут присоединяться к началу текста из диалогового окна настроек.

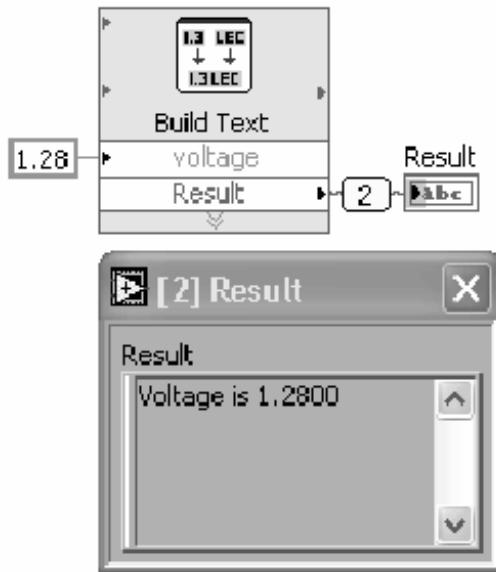


Рис 4.4. Вид Экспресс-ВП Build Text Express VI на диаграмме.

Функция **Format Into String**

Функция **Format Into String** преобразует параметры любого формата, такие как числовые данные, в строку. Для увеличения количества параметров следует изменить размер функции.

В приведенном на рисунке 4.5 примере функция **Format Into String** выдает указанную на отладочном индикаторе строку при значениях полей **format string** (формате строки) «%.4f», **input string** (входной строке) «*Voltage is*» (учитывая пробел в конце), и параметре «1.28».

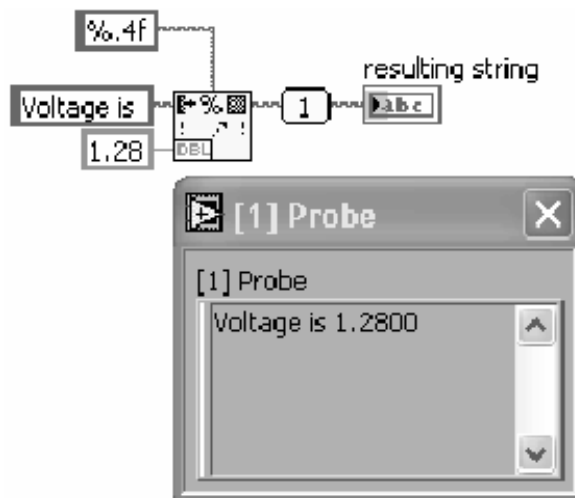


Рис. 4.5. Пример использования функции Format Into String.

В формате строки символ «%» указывает начало формата строки, «4» после точки определяет точность представления числа, показывая количество знаков после запятой, а «f» указывает тип данных с плавающей запятой. Для создания и редактирования формата строки следует щелкнуть правой кнопкой мыши по функции и выбрать пункт контекстного меню **Edit Format String**. Рисунок 4.6 показывает вид диалогового окна **Edit Format String** из предыдущего примера.

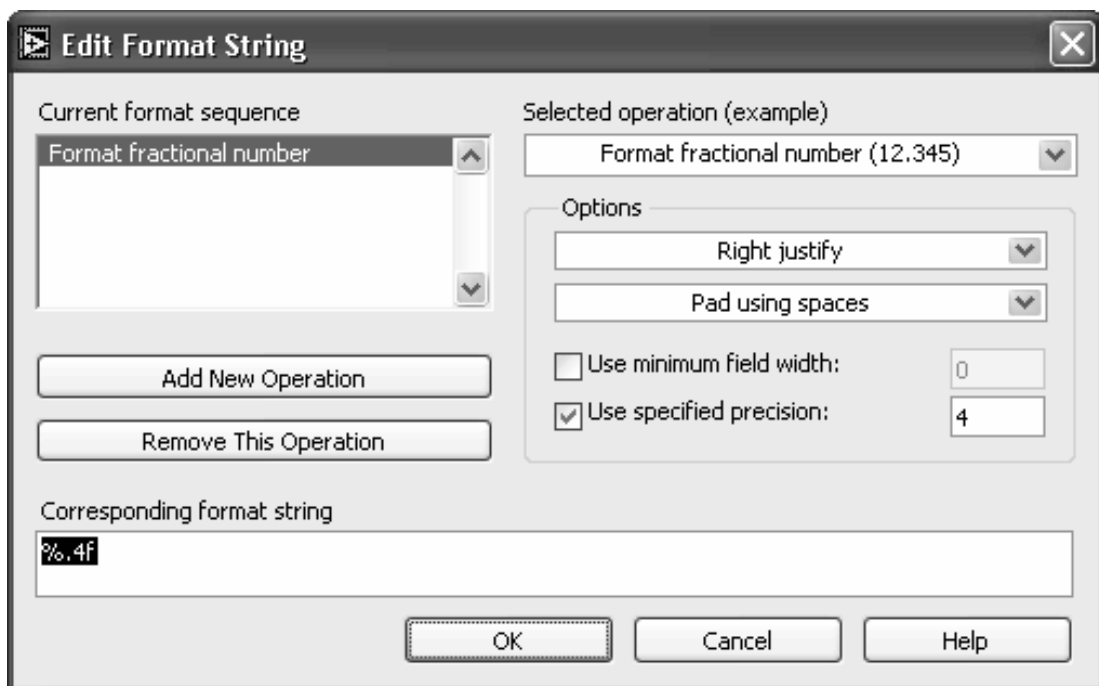


Рис. 4.6. Редактирование формата строки.

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

Преобразование строк в числовые данные

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

Просмотр и конвертирование строки

Функция **Scan From String** преобразует строку, содержащую допустимые числовые символы, такие как «0» – «9», «+», «-», «e», «E» и разделитель «.», в данные числового формата. Функция начинает просмотр строки, подаваемой на поле ввода данных **input string** с номера символа, задаваемого на поле **initial search location**. Функция может просматривать входящую строку различных типов данных, таких как числовые или логические данные, основываясь на формате строки. Для увеличения количества полей вывода данных следует изменить размер функции. Например, при значениях на полях ввода данных **format string** – «%f», **initial search location** – «8», **input string** – «VOLTS DC+1.28E+2» функция выдает результат 128.00, как показано на рисунке 4.7.

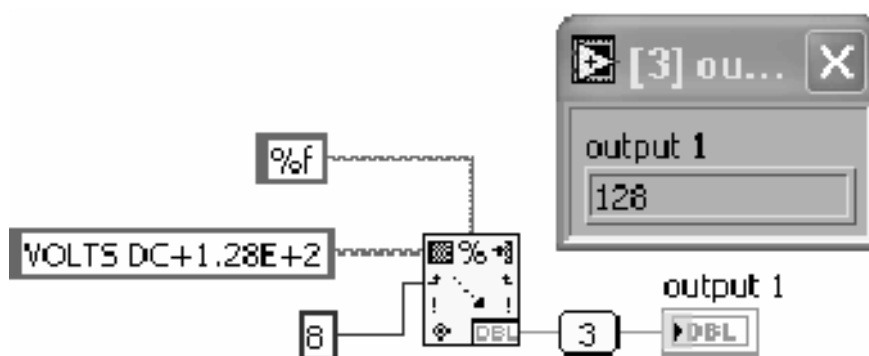


Рис. 4.7. Функция Scan From String.

В формате строки символ «%» – указывает начало формата строки, а символ «f» – указывает тип данных с плавающей запятой. Для создания и редактирования формата строки следует щелкнуть правой кнопкой мыши по функции и выбрать пункт контекстного меню **Edit Scan String**.

4.2. Функции файлового ввода/вывода

Функции файлового ввода/вывода производят файловые операции записи и считывания данных. Функции файлового ввода/вывода расположены в палитре **Functions»Programming»File I/O** и предназначены для:

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

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

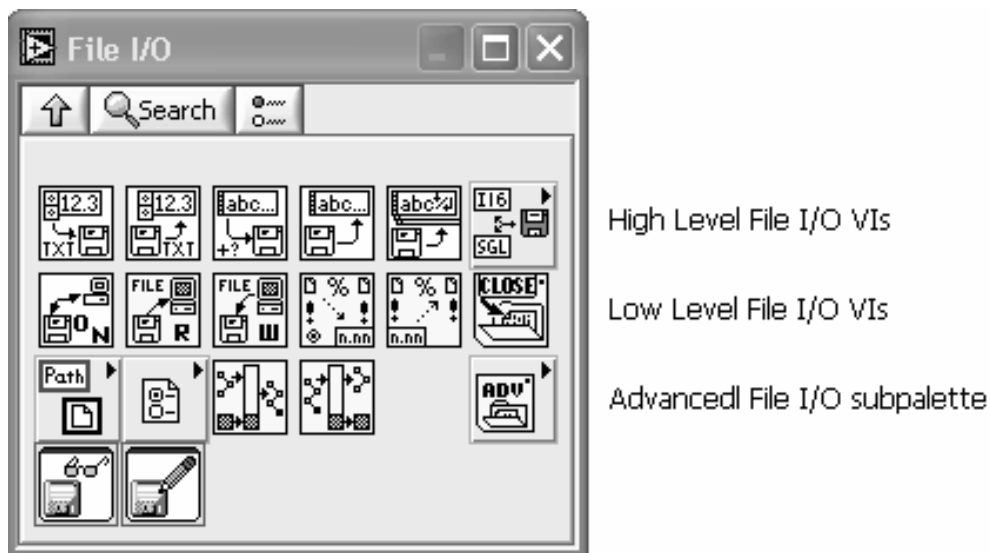


Рис. 4.8. Палитра функций файлового ввода/вывода.

Функции файлового ввода/вывода высокого уровня

Функции файлового ввода/вывода высокого уровня расположены в верхней строке палитры **Functions»Programming»File I/O**. Они предназначены для выполнения основных операций по вводу/выводу данных. Более подробную информацию можно получить в разделе 4.4. *Использование файлового ввода/вывода высокого уровня.*

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

Функции файлового ввода/вывода низкого уровня

Функции файлового ввода/вывода низкого уровня расположены в средней строке палитры **Functions»Programming»File I/O**.

Дополнительные функции работы с файлами (**Advanced File I/O**) расположены в палитре **Functions»Programming»File I/O»Advanced File Functions** и предназначены для управления отдельными операциями над файлами.

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

Основы файлового ввода/вывода





Стандартные операции ввода/вывода данных в/из файл(а) состоят из следующей последовательности действий:

1. Создание или открытие файла. Указание месторасположения существующего файла или пути для создания нового файла с

помощью диалогового окна LabVIEW. После открытия файл LabVIEW создает ссылку на него.

2. Произведение операций считывания или записи данных в/из файл(а).
3. Закрытие файла.
4. Обработка ошибок.

Таблица 4.2. ВП и функции основных операций файлового ввода/вывода.

	<p>Open/Create/Replace File – открывает, перезаписывает существующий файл или создает новый. Если file path (путь размещения файла) не указан, ВП выводит на экран диалоговое окно, в котором можно создать новый или выбрать уже существующий файл.</p>
	<p>Read File – считывает данные из файла, определяемого по ссылке refnum, и выдает данные на поле вывода data, на поле count подается значение количества считываемых данных. Считывание данных начинается с места, определяемого элементами pos mode и pos offset, и зависит от формата файла.</p>
	<p>Write File – записывает данные в файл, определяемый по ссылке refnum. Запись начинается с места, определяемого полями ввода данных pos mode и pos offset для файла потока байтовых данных и указателем конца файла для файла протоколированных данных.</p>
	<p>Close File – закрывает указанный в ссылке refnum файл.</p>

Обработка ошибок

Подпрограммы ВП и функции низкого уровня содержат информацию об ошибках. Для их обработки используются подпрограммы обработки ошибок, такие как **Simple Error Handler VI** (ВП Простой обработчик ошибок), расположенный в палитре **Functions»Programming»Dialog & User Interface**.



Рис. 4.9. ВП Простой обработчик Ошибок.

Поля ввода **error in** и вывода **error out** информации об ошибках используются в каждом ВП для обмена информацией об ошибках между ВП.

Во время работы ВП LabVIEW проверяет наличие ошибок в каждом узле. Если LabVIEW не находит ошибок, то узел выполняется нормально. Если LabVIEW обнаруживает ошибку в одном узле, то его выполнение прерывается, а информация об ошибке передается следующему узлу. Следующий узел поступает так же, и в конце выполнения LabVIEW сообщает об ошибках.

Сохранение данных в новом или уже существующем файле

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

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

Программный доступ к файлу экономит время. В операционной системе Windows путь файла состоит из имени (литеры) локального или сетевого диска, двоеточия, обратного слэша, разделяющего директории, и имени файла. Например, c:\testdata\test1.dat – есть путь к файлу test1.dat в папке testdata на диске C.

В приведенном на рисунке 4.10 примере показано, как записать строку данных в файл при программном указании пути и имени файла. Если файл уже существует, то он перезаписывается, если нет – то создается новый файл.

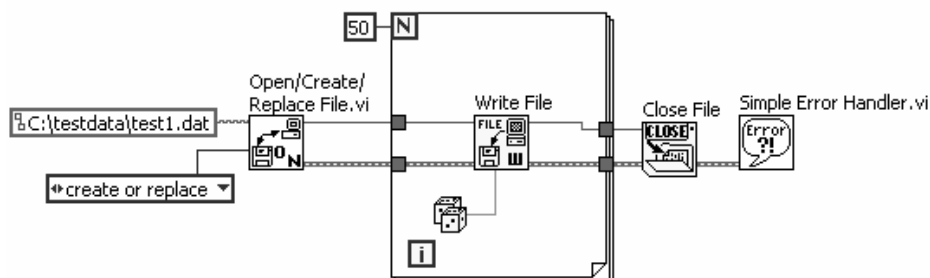


Рис. 4.10. Запись строки данных в файл по указанному пути.

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

При открытии файла, устройства или сетевого соединения LabVIEW создает ссылку на объект. Все операции с открытыми объектами выполняются с использованием ссылок.

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

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

4.3. Форматирование строк таблицы символов

Для того чтобы записать данные в файл формата электронной таблицы, необходимо переформатировать строковые данные в строку таблицы, содержащую разделители, такие как символ табуляции. Во многих приложениях символ табуляции разделяет столбцы, а символ **end of line** (конец строки) разделяет строки.

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

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

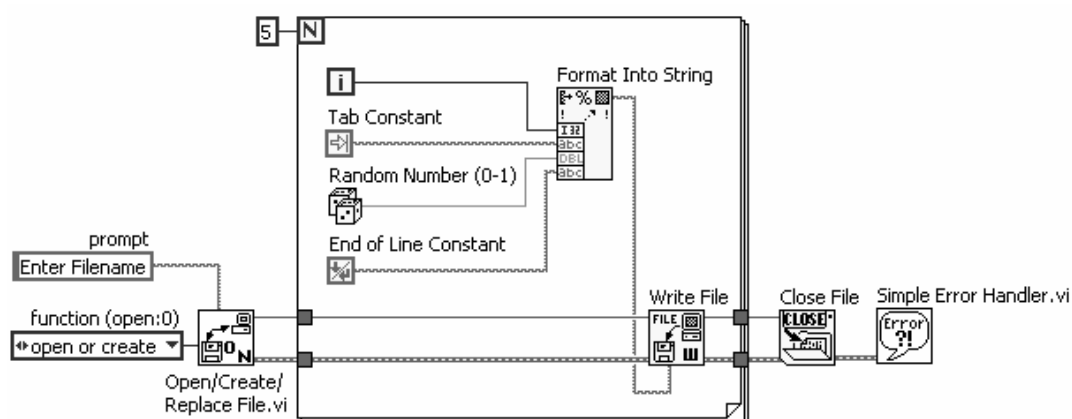


Рис. 4.11. Запись в файл с применением функции Format Into File.

На рисунке 4.11 представлена блок-диаграмма, на которой подпрограмма ВП **Open/Create/Replace File VI** открывает файл. Цикл **For** выполняется пять раз. Функция **Format Into String** преобразует значения счетчика итераций и случайное число в строку. Также указываются символы **Tab constant** (табуляции) и **End of Line Constant** (конца строки) для создания двух столбцов и одной строки таблицы символов. По окончании пяти итераций цикла файл закрывается и ВП проверяет наличие ошибок.

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

```
0→0.798141¶  
1→0.659364¶  
2→0.581409¶  
3→0.526433¶  
4→0.171062¶
```

Можно открыть данный текстовый файл в любом редакторе электронных таблиц для отображения на экране следующей таблицы (рис. 4.12).

	A	B
1	0,659364	
2	0,581409	
3	0,526433	
4	0,171062	
5		
6		
7		





Рис. 4.12. Считывание результата в редакторе таблиц.















4.4. Функций файлового ввода/вывода высокого уровня











Функции файлового ввода/вывода высокого уровня расположены в верхней строке палитры **Functions»Programming»File I/O**. Они предназначены для выполнения действий файлового ввода или вывода данных следующих типов:

- Символов в/из текстовых файлов.
- Строк из текстовых файлов.
- Одномерных или двумерных массивов числовых данных одинарной точности в/из файла электронной таблицы.
- Одномерных или двумерных массивов числовых данных одинарной точности или целочисленных 16-разрядных в/из бинарного файла.

Таблица 4.3. Функции файлового ввода/вывода высокого уровня

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

	Read from Binary File – читает файл, записанный в бинарном формате. Данные могут быть целочисленного типа или числовыми данными одинарной точности с плавающей точкой.
	Write to Binary File – записывает файл в бинарном формате. Данные могут быть целочисленного типа или числовыми данными одинарной точности с плавающей точкой.
	New Zip File – создает новый пустой файл с расширением *.zip.
	Add File to Zip – добавляет файлы в zip файл.
	Close Zip File – закрывает файл с расширением *.zip.
Дополнительные функции работы с файлами (Advanced File I/O) расположены в палитре Functions»Programming»File I/O»Advanced File Functions .	
	Get File Position – возвращает текущее значение положения метки в файле по ссылке на файл.
	Get File Size – возвращает размер файла в байтах.
	Set File Position – устанавливает значение положения метки в файле по ссылке на файл.
	Set File Size – устанавливает размер файла в байтах.
	Get Type and Creator – возвращает тип файла, поданного на вход ВП и название программы, в которой файл создан. В случае, если ВП не удалось распознать файл, то он возвращает «????».
	Set Type and Creator – устанавливает тип файла, поданного на вход ВП и название программы, в которой файл создан.
	Move – перемещает файл или папку из одного места на диске в другое.
	Copy – копирует файл или папку из одного места на диске в другое.
	Delete – удаляет файл или папку на диске.

	File/Directory Info – возвращает информацию о файле или папке, поданной на вход, включая размер файла или папки, дату последнего изменения и т.п.
	File/Directory Info – возвращает информацию о файле или папке, поданной на вход, включая размер файла или папки, дату последнего изменения и т.п.
	Get Volume Info – возвращает информацию об объеме свободного и занятого места на диске.
	List Folder – возвращает два массива строк, перечисляющих все файлы и папки, обнаруженные по указанному пути.
	Check if File or Folder Exists VI – проверяет, существует ли файл или папка на диске по указанному пути.
	Compare Two Paths VI – сравнивает между собой два пути и возвращает общую часть пути, различающуюся часть пути и переменную логического типа, говорящую одинаковые пути или нет.
	Generate Temporary File Path VI – генерирует путь к файлу с расширением .tmp. Данный виртуальный прибор только создает путь, но не создает файл.
	Get File Extension VI – возвращает расширение файла, поданного на вход.
	MD5Checksum File VI – вычисляет для файла MD5 message-digest. MD5 message-digest –это 128-битное число, отображающееся прописными символами в шестнадцатеричном формате.
	Recursive File List VI – возвращает список содержимого папки или библиотеки LLB.

5. Настройка ВП

5.1. Настройка внешнего вида лицевой панели

Для того чтобы пользователю было удобно работать с ВП, существует возможность настройки внешнего вида лицевой панели. Например, можно убрать меню и инструментальную панель, т.к. при работе ВП они не нужны.

Для настройки внешнего вида и поведения ВП следует выбрать пункт меню **FileVI»Properties**. Можно также щелкнуть правой кнопкой мыши по иконке в правом верхнем углу лицевой панели и выбрать пункт **VI Properties** из контекстного меню. Диалоговое окно **VI Properties** не доступно во время работы ВП.

Для выбора настроек следует использовать выпадающее меню **Category** в диалоговом окне **VI Properties**. Можно выбрать следующие категории настроек.

- **General** – общие настройки, такие как путь, по которому ВП был сохранен, номер версии, история ВП, а также все изменения, которые были сделаны с момента последнего сохранения. На этой странице также можно отредактировать иконку ВП.
- **Documentation** – эта страница используется для добавления описания ВП и ссылки на справочный файл.
- **Security** – на этой странице можно защитить с помощью пароля или заблокировать доступ к ВП.
- **Window appearance** – эта страница используется для настройки различных свойств окна.
- **Window Size** – на этой странице устанавливается размер окна.
- **Execution** – эта страница используется для настройки вариантов выполнения ВП.

Внешний вид окна

Для настройки того, как будет выглядеть окно при выполнении ВП, выберите пункт **Window appearance** из выпадающего меню **Category** в диалоговом окне **VI Properties**. По умолчанию название окна ВП совпадает с названием самого ВП.

Название окна можно изменить, чтобы сделать его более понятным, чем имя файла, под которым сохранен ВП. Это удобно при локализации ВП, так как название окна может быть переведено на другой язык. Для того чтобы изменить название окна, следует снять галочку напротив надписи **Same as VI Name** и ввести новое название в поле **Window Title**.

При настройке внешнего вида окна можно выбрать несколько стилей оформления, описанных ниже. При выборе какого-либо стиля его графическое представление отображается справа.

- **Top-level Application Window** – отображается название и меню, скрыты полосы прокрутки и инструментальная панель, позволяет пользователю закрыть окно, разрешается использование контекстных меню во время работы ВП, запрещено изменять размер окна, при вызове приложения отображается лицевая панель.
- **Dialog** – ВП ведет себя подобно диалоговому окну в операционной системе, т.е. пользователь не может взаимодействовать с другими окнами LabVIEW, пока это окно ВП открыто. Однако это не означает, что другие окна или приложения не могут отображаться поверх данного окна. (UNIX). Нельзя заставить окно оставаться поверх всех других окон. Диалоговые окна остаются поверх других окон, не имеют ни меню, ни полос прокрутки, ни инструментальной панели. Их можно закрыть, но изменить их размер нельзя. Разрешено использование контекстных меню во время работы, при вызове отображается лицевая панель. Если какой-нибудь логический элемент

лицевой панели ассоциирован с клавишей <Enter> или <Return>, LabVIEW отображает его границу более темным цветом.

- **Default** – стиль окна такой же, как и у окон среды LabVIEW.
- **Custom** – стиль определяется пользователем, для определения стиля окна следует нажать кнопку **Customize**.

Размер окна

Для настройки размера лицевой панели и объектов, расположенных на ней, выберите пункт **Window appearance** из выпадающего меню **Category** в диалоговом окне **VI Properties**. На странице расположены следующие настройки:

- **Minimum Panel Size** – устанавливает минимальный размер лицевой панели. Если на странице **Window appearance** пользователю разрешено изменять размер окна, то минимальные высота и ширина задаются на этой странице.
- **Size the Front Panel to the Width and Height of the Entire Screen** – автоматически изменяет размер окна так, чтобы оно отображалось на весь экран при запуске ВП. Старые положение и размер окна не сохраняются, поэтому при переходе обратно в режим редактирования окно остается на новом месте.
- **Maintain Proportions of Window for Different Monitor Resolutions** – изменяет размер окна лицевой панели так, чтобы оно занимало примерно столько же места на мониторе при изменении разрешения. Например, если ВП разрабатывался на мониторе с разрешением 1024*768, то для нормального отображения ВП на мониторе с разрешением 800*600 следует использовать эту опцию.
- **Scale All Objects on Front Panel as the Window Resizes** – автоматически изменяет размеры всех объектов на лицевой панели пропорционально размеру окна лицевой панели. Размер текста при

этом не изменяется, т.к. размеры шрифтов фиксированы. Эту опцию следует использовать, если пользователю разрешено изменять размер окна лицевой панели.

5.2. Отображение лицевых панелей подпрограмм ВП во время работы

Размер единственной лицевой панели накладывает ограничение на максимально возможное количество используемых элементов. Для решения этой проблемы используемые Виртуальные Приборы следует организовать таким образом, чтобы самый верхний ВП предоставлял возможность управлять основными параметрами, а подпрограммы ВП – второстепенными.

Как правило, при запуске подпрограммы ВП его лицевая панель не отображается. Для отображения лицевой панели при запуске отдельного экземпляра подпрограммы ВП следует воспользоваться диалоговым окном настройки узла подпрограммы ВП **SubVI Node Setup**. Для отображения лицевой панели при запуске всех экземпляров подпрограмм ВП следует воспользоваться диалоговым окном свойств ВП **VI Properties**.

Отдельный экземпляр

Для отображения лицевой панели при запуске отдельного экземпляра подпрограммы ВП следует щелкнуть правой кнопкой мыши по подпрограмме ВП и из контекстного меню выбрать пункт **SubVI Node Setup**. В открывшемся диалоговом окне отметьте пункты **Show Front Panel when called** и **Close afterwards if originally closed**. Диалоговое окно также содержит следующие элементы:

- **Open Front Panel when loaded** – отображает лицевую панель при загрузке подпрограммы ВП или ВП, который ее вызывает.
- **Show Front Panel when called** – отображает лицевую панель при запуске подпрограммы ВП.

- **Close afterwards if originally closed** – если опция **Show Front Panel when called** включена и подпрограмма ВП была закрыта до вызова, то лицевая панель закрывается по завершению работы подпрограммы ВП.
- **Suspend when called** – приостанавливает выполнение подпрограммы ВП и ожидает взаимодействия с пользователем. Эту опцию можно также включить, выбрав **Operate»Suspend when called**.

Все экземпляры

Для отображения лицевой панели при запуске всех экземпляров подпрограмм ВП следует выбрать пункт **File»VI Properties**. В диалоговом окне **VI Properties** выберите пункт **Window appearance** из выпадающего меню **Category**, нажмите кнопку **Customize** и отметьте пункты **Show Front Panel when called** и **Close afterwards if originally closed**.

5.3. Назначение и использование «горячих» клавиш

Для перемещения фокуса ввода во время работы ВП от одного элемента управления к другому можно воспользоваться клавишей **<Tab>**. Фокус ввода также можно установить, щелкнув правой кнопкой мыши по элементу управления. Пока элемент управления обладает фокусом ввода, значение в него можно ввести с клавиатуры. Если элемент управления цифровой или текстовый, то LabVIEW выделяет текст, который можно отредактировать. Если элемент управления логический, то поменять его значение можно нажатием клавиш пробел или **<Enter>**.

Для элементов управления можно назначать «горячие» клавиши, то есть пользователь может передвигаться по лицевой панели, используя другие клавиши. Щелкните правой кнопкой мыши по элементу управления и выберите пункт контекстного меню **Advanced»Key Navigation** для вызова диалогового окна **Key Navigation**. Пункт контекстного меню

Advanced»Key Navigation для элементов отображения не активен, так как вводить данные в элементы отображения невозможно.

Выберите «горячие» клавиши для нужного элемента управления в секции **Key Assignment**. Названия элементов управления на лицевой панели, перечисленные в списке **Current Assignments**, соответствуют собственным меткам элементов управления.

Для того чтобы пользователь не имел доступа к элементу управления с помощью клавиши <**Tab**>, следует отметить пункт **Skip this control when tabbing**.

5.4. «Нередактируемые» ВП

Некоторые настройки ВП приводят к тому, что редактировать ВП становится трудно. Например, можно отметить настройку **Run When Opened** и убрать меню и инструментальную панель. Если настроить ВП так, что он будет закрываться и выходить из LabVIEW по окончании работы, то остановить и отредактировать такой ВП не получится.

Редактировать такой ВП будет очень трудно.

Для выхода из среды LabVIEW можно использовать функцию **Quit LabVIEW**, расположенную в палитре **Functions»Programming»Application control**. Эта функция прекращает работу всех ВП и завершает работу текущей сессии LabVIEW. Эта функция имеет только одно поле ввода данных. Если оно подключено, то сессия LabVIEW заканчивается только когда входное значение равно TRUE. Если же поле ввода не подключено, то сессия LabVIEW заканчивается при выполнении данного узла.

Для того чтобы избежать описанных в примере ситуаций, перед редактированием свойств ВП следует создавать резервную копию ВП в новой папке. Для этого выберите из меню **File»Save with Options**.

Для сохранения ВП со всей его иерархией следует выбрать опцию **Development Distribution**. В сохраняемые файлы также можно включить файлы *vi.lib*. После создания резервной копии измените свойства ВП. При возникновении каких-либо проблем можно вернуться к резервной копии.

При выборе опции **Remove diagrams** удаляется исходный код ВП.

Использовать эту опцию следует только в том случае, когда точно известно, что редактировать ВП больше не потребуется. Перед тем, как сохранять ВП без блок-диаграмм, следует сделать резервную копию с блок-диаграммами.

6. Сбор данных и управление в LabVIEW

6.1. Конфигурация системы сбора данных

Среда LabVIEW включает в себя набор подпрограмм ВП, позволяющих конфигурировать, собирать и посылать данные на DAQ-устройства. Часто DAQ-устройства могут выполнять разнообразные функции: **аналого-цифровое преобразование (A/D), цифро-аналоговое преобразование (D/A), цифровой ввод/вывод (I/O)** и управление счетчиком/таймером. Каждое устройство имеет свой набор возможностей и скорость обработки данных. Кроме этого, DAQ-устройства разрабатываются с учетом аппаратной специфики платформ и операционных систем. Для получения дополнительной информации о DAQ-устройствах используйте документ National Instruments Product Catalog на web-сайте **ni.com/catalog**.

Компоненты DAQ-системы

На рисунке 6.1 продемонстрированы два варианта компоновки DAQ-системы. В варианте «А» DAQ-устройство встроено в компьютер, а в варианте «В» DAQ-устройство является внешним. С внешним устройством можно построить DAQ-систему на базе компьютера без доступных слотов расширения, например, с использованием портативных компьютеров. Компьютер и DAQ-модуль связываются между собой через аппаратные интерфейсы, такие как параллельный порт, последовательный порт и сетевые карты (Ethernet). Практически эта система является примером удаленного управления DAQ-устройством.

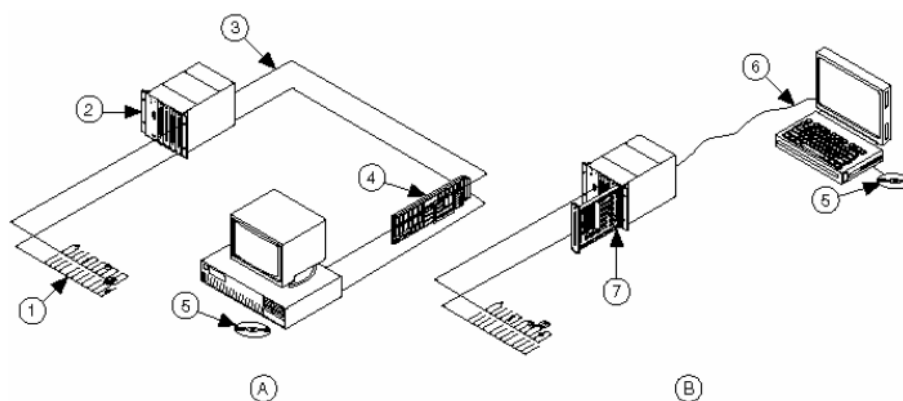


Рис. 6.1. Варианты компоновки DAQ-системы.

(А) – встраиваемое DAQ-устройство;

(В) – внешнее DAQ-устройство.

1 – Датчики.

2 – Модуль согласования сигналов.

3 – Согласованные сигналы.

4 – Встроенное DAQ-устройство.

5 – Программное обеспечение.

6 – Связь с параллельным портом.

7 – Внешний DAQ-модуль.

Основной задачей, решаемой **DAQ**-системами, является задача измерения или генерации физических сигналов в реальном времени. Перед тем как компьютерная система измерит физический сигнал, датчик или усилитель должен преобразовать физический сигнал в электрический, например, ток или напряжение. Встроенное **DAQ**-устройство часто рассматривается как полная **DAQ**-система, хотя практически это только один из компонент системы. В отличие от самостоятельных устройств измерения, не всегда возможно соединение напрямую источника сигналов с встроенным **DAQ**-устройством. В этих случаях необходимо использовать дополнительные модули согласования сигналов перед тем, как **DAQ**-устройство преобразует их в цифровой формат. Программные средства **DAQ**-систем включают в себя: сбор данных, анализ данных и представление результатов.

DAQ-устройства производства компании NI поставляются в комплекте с драйверами NI-DAQ. NI-DAQ взаимодействует и управляет измерительными устройствами National Instruments, включая такие DAQ-устройства, как многофункциональные устройства ввода/вывода сигналов (МІО) серии Е, SCXI модули согласования сигналов и модули переключения сигналов. NI-DAQ является расширенной библиотекой функций, которые можно вызывать из среды создания приложений, например, LabVIEW, для программирования всех возможностей измерительного устройства NI.

Программирование измерительного устройства NI возможно как в программных пакетах National Instruments: LabVIEW, LabWindows/CVI и Measurement Studio, так и в любой среде программирования, поддерживающей вызовы динамических библиотек (DDL) с использованием ANSI C интерфейса. Использование любого программного обеспечения NI существенно уменьшает время, затраченное на создание приложений сбора данных:

- LabVIEW обеспечивает сбор данных с помощью LabVIEW DAQ-комплекса виртуальных приборов для программирования измерительных устройств NI.
- LabWindows/CVI имеет полную встроенную поддержку ANSI C окружения, программирование измерительных устройств NI производится с помощью библиотеки сбора данных LabWindows/CVI Data Acquisition library.
- Инструменты программирования Measurement Studio предназначены для создания тестовых программ и приложений сбора данных в среде Microsoft Visual Studio .NET. Measurement Studio имеет поддержку Visual C#, Visual Basic .NET и Visual C++ .NET.

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

- Стандартный NI-DAQ.
- NI-DAQmx.
- NI-SWITCH.



Рис. 6.2. Структура типичной системы сбора данных и управления.

На рисунке 6.2. приведена структура типичной системы сбора данных и управления. На нижнем уровне этой системы измерительные датчики преобразуют разнообразные измеряемые сигналы (температура, давление, освещенность и пр.) в электрические. Далее в случае необходимости эти сигналы нормализуются в соответствии с диапазонами допустимых параметров входных каскадов устройств сбора данных. В случае необходимости производится также дополнительное согласование выхода

датчика с входом устройства сбора. Все устройства сбора данных работают под управлением соответствующих приложений, взаимодействующих с программными интерфейсами, драйверами и алгоритмами NI-DAQmx, стандартного NI-DAQ и NI-SWITCH.

Стандартный NI-DAQ

Стандартный NI-DAQ является обновлением предыдущей версии 6.9.x NI-DAQ. **Стандартный NI-DAQ** включает в себя те же ВП/функции и работает аналогично NI-DAQ версии 6.9.x, однако внесенные изменения позволяют использовать NI-DAQ и NI-DAQmx совместно в разрабатываемом приложении. В **Стандартном NI-DAQ** исключена поддержка некоторых измерительных устройств по сравнению с версией 6.9.x. Список устройств, поддерживаемых **Стандартным NI-DAQ**, приведен в документации NI-DAQ. **NI-DAQmx** является следующим поколением драйверов NI-DAQ. Он обладает новыми функциями и инструментами управления измерительными устройствами. **NI-DAQmx** имеет много новых особенностей и преимуществ по сравнению с предыдущей версией NI-DAQ:

- DAQ Configuration Assistant – помощник настройки DAQ-устройств с помощью нового графического интерфейса позволяет конфигурировать настройки, каналы и задания измерения DAQ-устройств в LabVIEW, LabWindows/CVI и Measurement Studio.
- Увеличилось быстродействие ряда операций, в частности однократного аналогового ввода/вывода, более эффективно организована многозадачность.
- Программный интерфейс для создания DAQ-приложений стал более простым и интуитивно понятным.
- Появились дополнительные возможности программного интерфейса **NI-DAQmx** для LabVIEW, включая узлы атрибутов для сбора данных

и улучшенную поддержку типов данных waveform для операций аналогового и цифрового ввода/вывода.

- Разработаны единые программные интерфейсы и функциональность для ANSI C, LabWindows/CVI и Measurement Studio, включая интерфейсы .NET и C++.

NI-SWITCH

NI-SWITCH является IVI-совместимым драйвером устройств, поддерживающим все модули переключения сигналов компании NI. NISWITCH имеет интерактивную программную лицевую панель, которую можно использовать для поиска неисправности приложений. В этом курсе описан программный интерфейс NI-DAQmx.

Настройка аппаратных средств DAQ-устройств

Перед использованием ВП **Сбора Данных** необходимо настроить DAQ-устройство, установленное в компьютер.

Windows

Windows Configuration Manager хранит информацию обо всем установленном на компьютер аппаратном обеспечении, включая устройства National Instruments DAQ. Если у вас есть Plug & Play (PnP) устройства, например, карта MIO серии E, Windows Configuration Manager автоматически его определит и настроит. Если устройство не поддерживает PnP, необходимо его настроить вручную с помощью раздела Add New Hardware (Установка и удаление устройств) в Control Panel (Панели Управления).

Аппаратную конфигурацию Windows можно проверить с помощью Device Manager (Менеджера Устройств), который находится в **Start»Settings»Control Panel»System»Device Manager** (Пуск»Настройки»Панель Управления»Система»Диспетчер Устройств).

В разделе **Data Acquisition Devices** отображаются все DAQ-устройства, установленные на компьютер. Дважды щелкните мышью по устройству для отображения диалогового окна с закладками. На закладке **General** (Общие) приводится общая информация об устройстве. На закладке **Resources** (Ресурсы) перечислены системные ресурсы, используемые устройством: номера прерываний (**IRQ**), каналы прямого доступа к памяти (**DMA**) и базовые адреса ввода/вывода для программно-конфигурируемых устройств. На закладке **NI-DAQ Information** указан тип шины данного DAQ-устройства. Производитель и номер версии драйвера DAQ-устройства приведены на закладке **Driver** (Драйвер).

Среда LabVIEW устанавливает утилиту конфигурации **Measurement & Automation Explorer** для детальной настройки параметров конфигурации каналов устройств. Эту утилиту необходимо запускать после установки DAQ-устройства на компьютер. Утилита конфигурации считывает информацию из реестра Windows, записанную **Диспетчером устройств (Device Manager)**, и присваивает логическое имя для каждого DAQ-устройства. По логическому имени среда LabVIEW распознает DAQ-устройство. Запуск конфигурационной утилиты происходит двойным щелчком левой кнопки мыши по ее иконке на рабочем столе операционной системы или выбором пункта главного меню **Tools»Measurement & Automation Explorer** непосредственно в среде LabVIEW. Начальное окно конфигурационной утилиты показано на рисунке 6.3. **Measurement & Automation Explorer** также используется для конфигурации устройств стандарта **SCXI**.

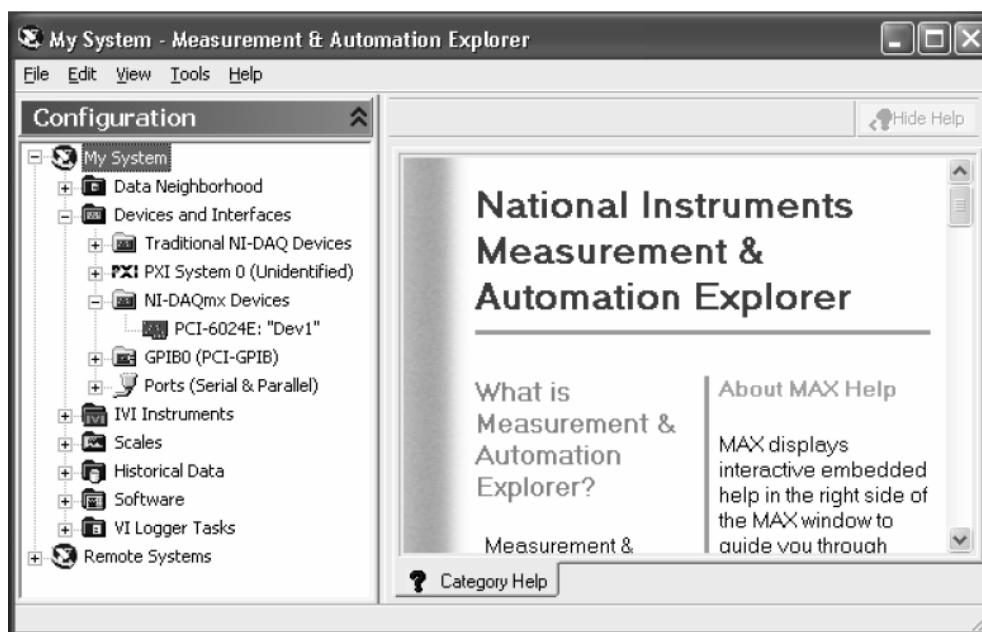


Рис. 6.3. Начальное окно конфигурационной утилиты **Measurement & Automation Explorer**.

Конфигурационная утилита определяет все аппаратные средства фирмы National Instruments, включая **GPIB** интерфейс. Дополнительная информация о **GPIB** интерфейсе приведена в Уроке 11 *Управление измерительными приборами*. Параметры устройства можно также установить с помощью утилит-конфигураций, входящих в комплект поставки устройств. **Measurement & Automation Explorer** позволяет сохранить логическое имя устройства и параметры конфигураций в реестр Windows. Windows автоматически находит и настраивает **DAQ**-устройства, удовлетворяющие стандарту **PnP**, например, карту **PCI-6024E**.

Настройка каналов и заданий

В стандартном NI-DAQ можно создавать *виртуальные каналы*, которые включают в себя совокупность настроек физического канала DAQ, типа измерений и информацию о нормировке значений. В стандартном NI-DAQ и NI-DAQ более ранних версий понятие виртуальных каналов используется в качестве соответствия физических каналов проводимым

измерениям. Понятие *NI-DAQmx channels* аналогично виртуальным каналам стандартного NI-DAQ.

NI-DAQmx также включает в себя *задания*, являющиеся частью программного интерфейса API. *Задание* – это совокупность настроек одного или нескольких каналов, синхронизации, временных и других параметров. *Задание* описывает измерение или генерацию сигналов, которые необходимо выполнить. Каналы, созданные в рамках задания, являются локальными. Каналы, определенные вне заданий, являются глобальными и могут быть использованы отдельно. Настройка виртуальных каналов является необязательной в стандартном NI-DAQ и более ранних версиях, но в NI-DAQmx выполнение этой процедуры необходимо для проведения измерений. В стандартном NI-DAQ конфигурирование виртуальных каналов производилось в MAX. В NI-DAQmx настройка виртуальных каналов возможна как в MAX, так и в самом приложении, причем как в рамках *задания*, так и отдельно.

6.2. Сбор данных в LabVIEW

Подпрограммы сбора данных размещены в палитрах **Functions»Measurements I/O»Data Acquisition** и **Functions»Measurements I/O»DAQmx – Data Acquisition**. В разделе **Data Acquisition** собраны стандартные виртуальные приборы сбора данных, а в **DAQmx – Data Acquisition** – ВП для работы с NI-DAQmx.

В палитре **DAQmx – Data Acquisition** содержатся все необходимые подпрограммы для осуществления операций аналогового и цифрового ввода/вывода и работы со счетчиками/таймерами. Виртуальные приборы собраны таким образом, что большинство задач могут быть решены с их использованием. Можно настроить ВП для выполнения специфического действия с помощью узла Атрибутов. Многие задания, не требующие расширенных возможностей синхронизации, могут быть выполнены с помощью экспресс-ВП **DAQmx Assistant**. В этом курсе описано, как использовать экспресс-ВП **DAQmx Assistant** для выполнения операций сбора данных. Для получения дополнительной информации об использовании NI-DAQmx обратитесь к справочному пособию NIDAQmx или изучите расширенный курс *LabVIEW Data Acquisition and Signal Conditioning*.

Экспресс-ВП **DAQmx Assistant** позволяют просто осуществить настройку DAQ-устройства. При добавлении экспресс-ВП **DAQmx Assistant** на блок-диаграмму появляется диалоговое (рис. 6.4) окно, в котором осуществляется конфигурация задания – провести определенные измерения. В процессе создания локального задания указывается необходимый тип измерения.

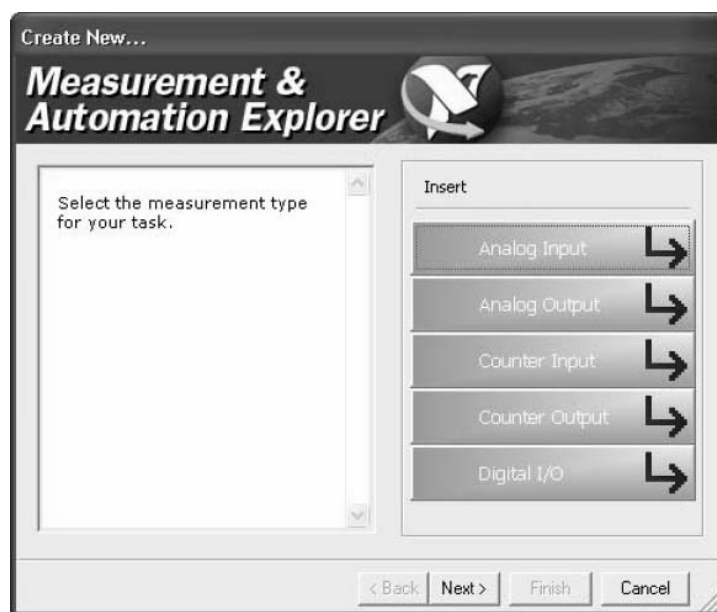


Рис. 6.4. Окно начальной настройки экспресс-ВП DAQmx Assistant.

Достаточно один раз создать задание, чтобы информация о нем сохранилась в экспресс-ВП DAQmx Assistant. Впоследствии можно сконфигурировать экспресс-ВП DAQmx Assistant заново, дважды щелкнув по нему мышью.

6.2.1. Выполнение операций аналогового ввода

Используйте операции аналогового ввода для осуществления аналого-цифрового преобразования (рис. 6.5). Существует несколько типов измерений входного сигнала: напряжение, температура, деформация, ток, сопротивление или частота.



Рис. 6.5. Выбор типа измерений входного сигнала.

Каждый тип измерений имеет собственные параметры, например, величина сопротивления для измерения тока или калибровка датчика деформаций для их измерений.

Установка временного такта выполнения заданий

При выполнении операций аналогового ввода данных задание может формулироваться по-разному: получение 1 значения, получение n значений или непрерывный сбор данных.

Получение одного значения

Получение одного значения является операцией по вызову. Другими словами NI-DAQmx оцифровывает одно значение с входного канала и немедленно возвращает его величину. Выполнение этой операции не требует наличия буфера и аппаратного контроля временного такта.

Например, для периодического контроля уровня жидкости в резервуаре необходимо получать по одному значению. Вы можете подключить датчик, генерирующий разное напряжение в зависимости от

уровня жидкости, к одному из каналов DAQ-устройства и производить контроль, периодически оцифровывая по одному значению.

Получение n значений

Один из методов получения n значений – n раз получить по одному значению. Однако получение по одному значению из одного или нескольких каналов снова и снова является неэффективным и занимает много времени. Более того, отсутствует контроль времени между последовательными операциями получения значений. Вместо этого необходимо использовать аппаратное задание временного такта выполнения операций, в процессе которого используется буферизация полученных данных в компьютерной памяти, что приводит к более эффективному процессу сбора данных. С программной точки зрения необходимо включить режим аппаратного задания временного такта выполнения операций и задать частоту оцифровки **sample rate** и ограниченный по времени режим работы **sample mode (finite)**. Возможна оцифровка нескольких значений из одного канала или из нескольких.

С помощью NI-DAQmx можно осуществлять сбор данных с нескольких каналов. Например, необходимо контролировать уровень жидкости в резервуаре и ее температуру. В этом случае нужно иметь два датчика, подключенных к разным каналам DAQ-устройства.

Непрерывный сбор данных

Если необходимо отображать, обрабатывать и производить запись данных по мере их поступления, лучше использовать режим непрерывного сбора данных. Для этого устанавливается режим работы **sample mode (continuous)**.

Синхронизация заданий

Когда устройство, управляемое NI-DAQmx, работает, оно производит операции. Два наиболее часто встречающихся действия – обработка

значения и начало сбора данных. Каждое производимое действие NI-DAQmx вызвано чем-либо или имеет причину. Причины, приводящие к действиям NI-DAQmx, называются синхронным запуском.

Start Trigger

Сигнал, запускающий сбор данных.

Reference Trigger

Сигнал, устанавливающий реперную точку в наборе входных значений. Данные, полученные до этой точки, называются **pretrigger data**. Данные после реперной точки являются **posttrigger data**.

6.2.2. Запись полученных данных в файл

Часто бывает необходимо производить запись данных, полученных с помощью DAQ-устройства. При планировании сохранения данных в файл необходимо учесть следующие важные моменты:

- Не все приложения анализа данных используют LabVIEW. Подумайте, какое приложение будет использоваться для обработки сохраненных данных.
- Формат записи данных в файл определяет приложение, которое будет их обрабатывать. Поскольку LabVIEW обладает стандартными файловыми операциями, которые присутствуют и в других языках программирования, то существует полный доступ к формату записываемой информации.

LabVIEW может создавать LabVIEW Measurement File – текстовый ASCII файл, который может быть открыт в любом редакторе электронных таблиц или в текстовом редакторе. Файл формата LabVIEW Measurement File просто создается LabVIEW, легко открывается как в LabVIEW, так и в других приложениях.

Экспресс-ВП **Write LabVIEW Measurement File**, расположенный в палитре **Functions»Express»Output**, производит запись данных в файл формата LabVIEW Measurement File. При помещении экспресс-ВП **Write LabVIEW Measurement File** на блок-диаграмму появляется диалоговое окно, в котором указывается, как сохранить файл.

Экспресс-ВП **Read LabVIEW Measurement File**, расположенный в палитре **Functions»Express»Input**, производит чтение данных из файла формата LabVIEW Measurement File. Чтение данных производится по одному значению, поэтому необходимо помещать этот экспресс-ВП внутрь цикла.

6.2.3. Выполнение операций аналогового вывода

Аналоговый выход используется для вывода сигналов обратной связи, управляющих измерительной системой. Иногда генерация аналоговых сигналов оказывается необходимой для связи системы с дополнительным оборудованием. Для вывода аналогового сигнала производится цифро-аналоговое преобразование. Сигнал на выходе может быть представлен в виде напряжения или тока.

Для осуществления вывода напряжения или тока должно быть установлено соответствующее DAQ-устройство, позволяющее генерировать сигналы заданной формы.

Генерация данных

При выполнении операций аналогового вывода данных задание может формулироваться по-разному: генерация 1 значения, генерация n значений или непрерывная генерация данных.

Генерация одного значения

Генерация одного значения используется в том случае, когда уровень сигнала более важен, чем скорость его обновления. Например, используйте

генерацию одного значения, если необходимо сгенерировать постоянный сигнал. В этом случае возможно использование программных средств управления временными задержками.

Выполнение этой операции не требует наличия буфера и аппаратного контроля временного такта. Например, необходимо сгенерировать известное напряжение для эмуляции работы устройства, в этом случае можно использовать метод генерации одного значения.



Рис. 6.6. Настройка формата выходного сигнала.

Генерация n значений

Один из методов генерации n значений – n раз сгенерировать по одному значению. Однако генерация по одному значению из одного или нескольких каналов снова и снова является неэффективной и занимает много времени. Более того, отсутствует контроль времени между последовательно сгенерированными значениями. Вместо этого необходимо использовать аппаратное задание временного такта выполнения операций, в процессе которого используется буферизация генерируемых данных в

компьютерной памяти, что приводит к более эффективному процессу генерации.

Возможно использование программного и аппаратного задания временного такта выполнения операций. В случае программного управления моментом генерации значений, временные задержки определяются программой и операционной системой, а не измерительным устройством. В случае аппаратного управления временным тактом генерация данных производится по TTL-сигналу внутреннего таймера DAQ-устройства. Аппаратный таймер работает намного быстрее программных циклов. Также аппаратный таймер более точен по сравнению с программными циклами.

С программной точки зрения необходимо включить режим аппаратного задания временного такта выполнения операций и задать частоту оцифровки **sample rate** и ограниченный по времени режим работы **sample mode (finite)**. Возможна генерация значений на одном канале или нескольких.

Режим генерации n значений имеет смысл использовать при создании изменяющегося в конечный интервал времени сигнала, например, фрагмента переменного тока.

Непрерывная генерация данных

Непрерывная генерация данных аналогична генерации n значений с отличием в том, что для остановки непрерывной генерации должно произойти какое-то событие. Если необходимо непрерывно генерировать сигнал, для этого устанавливается режим работы **sample mode (continuous)**.

Синхронизация заданий

Когда устройство, управляемое **NI-DAQmx**, работает, оно производит операции. Два наиболее часто встречающихся действия – задание значения

и начало его генерации. Так же, как в случае сбора данных, при генерации используется синхронный запуск.

Start Trigger

Сигнал, запускающий генерацию данных. Некоторые устройства не поддерживают аппаратное управление тактом. Обратитесь к руководству пользователя DAQ-устройства для получения подробной информации.

Reference Trigger

Этот сигнал не поддерживается для аналогового выхода.

6.2.4. Информация о счетчиках

Счетчики – это цифровые временные устройства. Обычно счетчики используют для подсчета произошедших событий, измерения периода и частоты сигнала и генерации импульсов.

Счетчик состоит из четырех основных компонентов: регистр значений счетчика, источник, сигнал управления и выходной сигнал.

- **регистр значений счетчика** – содержит текущее значение счетчика. Значение, хранимое в регистре, можно узнать программно.
- **источник** – сигнал, вызывающий изменение значения счетчика, хранимого в регистре. Счетчик реагирует на возрастающий или спадающий фронт сигнала. Какой тип фронта сигнала вызывает изменение состояния счетчика, задается программно. Программно выбранный тип фронта сигнала называется активным. Когда на вход счетчика подается активный фронт сигнала, его значение изменяется на единицу. Программно задается и знак изменения значения счетчика – увеличивается оно или уменьшается.
- **сигнал управления** – входящий сигнал, определяющий, вызывает ли активный уровень изменение состояния счетчика. Счет может происходить при высоком или низком уровне этого сигнала или при

различных комбинациях возрастающих и/или спадающих фронтов сигнала управления. Сигнал управления формируется программно.

- **выходной сигнал** – сигнал, генерируемый импульсы или серию импульсов.

Увеличение значения счетчика, сконфигурированного для подсчета простых событий, происходит при поступлении на вход источника сигнала с активным фронтом. Чтобы счетчик считал события при поступлении активного фронта сигнала, он должен быть инициализирован. Разрядность счетчика определяет его разрешение, например, 24-битовый счетчик может подсчитать следующее число событий:

$$2^{(\text{Разрядность счетчика})} - 1 = 2^{24} - 1 = 16,777,215$$

Когда 24-разрядный счетчик достигает числа 16,777,215, это означает, что он достиг своего предельного значения. Следующее событие приведет к его переполнению и сбросу на 0.

6.2.5. Ввод и вывод цифровых сигналов

Измерение и генерация цифровых сигналов используется в большом количестве приложений, включая мониторинг систем безопасности. В основном, генерация и измерение цифровых сигналов применяется в лабораторных исследованиях, тестировании продуктов и промышленных процессах мониторинга и контроля.

Цифровой ввод/вывод представляет собой чтение или запись значения в цифровую линию или во весь цифровой порт, состоящий из совокупности линий.

Вы можете использовать цифровые линии **DAQ**-устройства для сбора цифровых значений. Этот сбор данных основывается на программном задании временного такта выполнения операций. На некоторых устройствах можно настраивать цифровые линии независимо для генерации

или сбора данных. Каждая цифровая линия является отдельным каналом в LabVIEW.

Вы можете использовать цифровые порты **DAQ**-устройства для сбора данных совокупности цифровых линий. Этот сбор данных основывается на программном задании временного такта выполнения операций. Вы можете настраивать цифровые порты независимо для генерации или сбора данных. Каждый цифровой порт является отдельным каналом в LabVIEW.

7. Работа с измерительным оборудованием

7.1. Управление измерительными приборами

7.1.1. Управление в LabVIEW измерительными приборами

Среда LabVIEW не накладывает ограничения на средства управления измерительными приборами, если они удовлетворяют промышленным технологическим стандартам управления. LabVIEW позволяет использовать различные коммуникационные интерфейсы, такие как последовательный и параллельный порты, **GPIB, VXI, PXI, Ethernet, SCSI** и **CAMAC**. Этот урок описывает два наиболее распространенных коммуникационных интерфейса: GPIB и последовательный порт.

Для выполнения урока понадобится следующая информация и перечень дополнительных аппаратных средств:

- Тип разъема у измерительного прибора.
- Нуль-модемный кабель с заданным числом контактных штырьков и типом разъемов «мама» или «папа».
- Техническая характеристика выхода измерительного прибора: уровень выходного сигнала, максимальная длина соединительного кабеля и наличие заземления.
- Используемые коммуникационные протоколы: ASCII-команды, двоичные команды и формат передаваемых данных.
- Существующие драйверы устройства.

7.1.2. Использование Instrument I/O Assistant

ВП **Instrument I/O Assistant**, расположенный в палитрах **Functions»Express»Input** и **Functions»Instrument I/O**, является экспресс-ВП среды LabVIEW. Этот экспресс-ВП позволяет легко проверять связь с измерительными приборами, а также разрабатывать последовательности

запросов, анализа и записи данных. Эти этапы могут быть сохранены как экспресс-ВП для непосредственного использования или конвертированы в подпрограмму ВП. **Instrument I/O Assistant** следует использовать, когда нет необходимых драйверов к измерительному прибору.

Для запуска **Instrument I/O Assistant** поместите этот экспресс-ВП на блок-диаграмму. Появится диалоговое окно настроек **Instrument I/O Assistant** (рис. 7.1). Если оно не появилось, следует нажать два раза на иконке **Instrument I/O Assistant**.

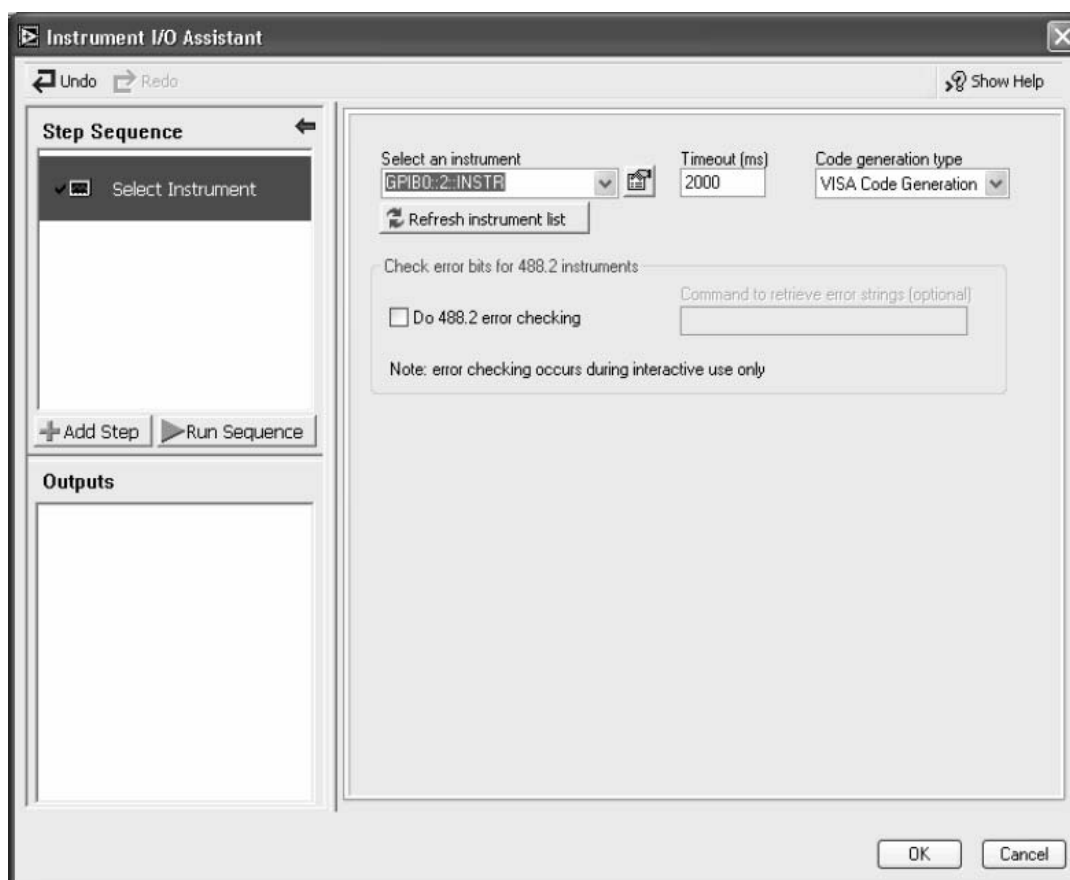


Рис. 7.1. Диалоговое окно настроек **Instrument I/O Assistant**.

Для настройки экспресс-ВП **Instrument I/O Assistant** следует выполнить следующие пункты.

1. Выберите измерительный прибор. Приборы, настроенные в **MAX** отображены в выпадающем списке **Select an instrument**.

2. Выберите пункт **Code generation type**. Генерация кода **VISA** обеспечивает большую гибкость и модульность, чем генерация кода **GPIB**.
3. Выберите один из следующих шагов связи с помощью кнопки **Add Step**:
 - **Query and Parse** – посылает устройству запрос типа ***IDN?** и анализирует возвращенную строку.
 - **Write** – посылает команду измерительному прибору.
 - **Read and Parse** – считывает и анализирует данные с измерительного прибора.
4. После составления требуемой последовательности операций нажмите кнопку **Run Sequence** для проверки созданной для экспресс-ВП последовательности.
5. Нажмите кнопку **OK** для выхода из диалогового окна настройки **Instrument I/O Assistant**.

LabVIEW добавит поля ввода/вывода в ВП **Instrument I/O Assistant** на блок-диаграмме в соответствии с данными, которые будут приходиться с измерительного прибора.

Для просмотра кода, созданного **Instrument I/O Assistant**, щелкните правой кнопкой мыши по иконке **Instrument I/O Assistant** и выберите из контекстного меню пункт **Open Front Panel**. Эта опция преобразует экспресс-ВП в подпрограмму ВП. Для просмотра созданного кода перейдите на блок-диаграмму. После того, как экспресс-ВП был преобразован в подпрограмму ВП, преобразовать его назад невозможно.

7.1.3. Архитектура программного обеспечения виртуальных интерфейсов

Virtual Instrument Software Architecture (VISA) – это низкоуровневые функции, которые используются в Виртуальных Приборах драйверов измерительных приборов для связи с программными драйверами.

В 1993 году компания National Instruments подписала соглашение с компаниями GenRad, Racal Instruments, Tektronix и Wavetek о поддержке единого стандарта **VXIplug&play**. Цель альянса – сделать программное обеспечение для систем **VXI** совместимым независимо от производителя и сократить время разработки приложений.

Для этих целей был разработан единый стандарт для драйверов интерфейса и интерфейса ввода/вывода. Термин **VXIplug&play** охватывает как аппаратную, так и программную стандартизацию. В попытке стандартизации программных средств члены альянса **VXIplug&play** выработали следующий блок требований:

- Максимальная легкость в использовании и исполнении.
- Долговременная поддержка совместимости базовых компонентов.
- Открытая архитектура для других фирм.
- Максимальная многоплатформенная совместимость.
- Максимальная расширяемость и модульность архитектуры.
- Максимальная многократность использования программных средств.
- Стандартизация использования программных элементов в системе.
- Рассмотрение драйверов интерфейса как части интерфейса.
- Адаптация к уже утвержденным стандартам.
- Максимальная совместная поддержка пользователей.

VISA – это язык программирования ввода/вывода **VXIplug&play**, который явился результатом усилий альянса по стандартизации. **VISA** сам

по себе не обеспечивает программную совместимость интерфейсов измерительных приборов. Это высокоуровневый блок **API**, который вызывает низкоуровневые функции драйвера. **VISA** позволяет управлять **VXI**, **GPIB**, последовательным портом и другими интерфейсами, основанными на компьютере. **VISA** выполняет соответствующие вызовы функции драйвера в зависимости от типа используемого интерфейса. В случае возникновения проблем отладки **VISA** необходимо вспомнить об этой иерархии. Возникновение очевидных проблем использования **VISA**, как правило, всегда связано с проблемами драйверов, функции которых **VISA** вызывает.

В среде LabVIEW **VISA** является единственной библиотекой функций, которая используется для связи с **GPIB**, последовательным портом, **VXI** и измерительными приборами, соединенными с компьютером. Нет необходимости использовать разные палитры ввода/вывода для программирования измерительного прибора. Например, некоторые измерительные приборы предоставляют выбор типа интерфейса. Драйвер, написанный с помощью функций, размещенных в палитре **Functions»Instrument I/O»GPIB**, не сможет работать с интерфейсом последовательного порта. **VISA** решает эти проблемы предоставлением единого набора функций для работы с интерфейсами любого типа. Поэтому все драйверы интерфейса в LabVIEW используют **VISA** как язык ввода/вывода.

Терминология программирования VISA

Функции, применяемые к ресурсам, называются операциями. Ресурсы имеют свойства или атрибуты, содержащие информацию, относящуюся к ресурсам. При программировании с использованием **VISA** употребляется терминология, похожая на терминологию Виртуальных Приборов драйверов измерительных приборов:

- **Resource** – любое устройство системы, включая последовательный и параллельный порты.
- **Session** – для связи с ресурсом необходимо открыть **VISA**-сессию, что эквивалентно открытию канала связи. При открытии сессии для ресурса среда LabVIEW возвращает номер сессии **VISA**, который является уникальным логическим идентификатором устройства. Этот номер сессии необходимо использовать во всех последующих функциях **VISA**.
- **Instrument Descriptor** – точное имя ресурса. Дескриптор указывает тип интерфейса (**GPIB**, **VXI**, **ASRL**), адрес устройства (логический или первичный) и тип **VISA**-сессии (**INSTR** или **Event**).

Дескриптор интерфейса похож на телефонный номер, где ресурс – это абонент, а сессия – это телефонная линия. Каждый вызов использует свою собственную линию, и при пересечении таких линий происходит ошибка.

Таблица 7.1. Синтаксис для описания дескриптора интерфейса в **VISA**.

Интерфейс	Синтаксис
Асинхронный последовательный порт (Asynchronous serial)	ASRL[<i>board</i>][:INSTR]
<i>GPIB</i>	GPIB[<i>board</i>]:: <i>primary address</i> [:: <i>secondary address</i>][:INSTR]
VXI интерфейс через встроенный или MXIbus контроллер	VXI[<i>board</i>]:: <i>VXI logical address</i>][:INSTR]
GPIB-VXI контроллер	GPIB-VXI[<i>board</i>]:: <i>GPIB-VXI primary address</i>]:: <i>VXI logical address</i>][:INSTR]

Можно использовать имя, заданное в конфигурационной утилите **MAX**, вместо дескриптора. (**MacOS**) Отредактируйте файл *visaconf.ini* для создания имени **VISA**. (**UNIX**) Используйте утилиту **visaconf**.

В случае, когда **Instrument I/O Assistant** не используется для автоматического создания кода, можно самостоятельно написать ВП для связи с измерительным прибором. Наиболее часто используемые функции **VISA** – это функции **VISA Write** и **VISA Read**. Большинству измерительных приборов необходимо получить информацию в виде запроса или команды до того, как они начнут передавать информацию. Таким образом, после функции **VISA Write** обычно стоит функция **VISA Read**.

7.1.4. Драйверы измерительных приборов

Драйвер измерительного прибора – это набор модульных программных функций, которые используют команды или протокол измерительного прибора для проведения стандартных операций. Драйвер измерительного прибора также вызывает нужные функции и Виртуальные Приборы.

Драйверы устройств из библиотеки **LabVIEW** устраняют необходимость изучать сложные низкоуровневые команды программирования для каждого отдельного измерительного прибора. Библиотека **LabVIEW** драйверов устройств содержит драйверы для множества программируемых приборов, использующих **GPIB, VXI, PXI** интерфейсы или последовательный порт.

Драйверы измерительных приборов принимают, анализируют и масштабируют строки отклика измерительных приборов в данные, используемые в приложениях. Драйверы измерительных приборов упрощают приложения диагностики, т.к. драйверы содержат в одной библиотеке все вводы/выводы прибора отдельно от какого-либо другого кода. При замене оборудования изменить приложения становится проще, поскольку весь код, относящийся к определенному оборудованию, заключен в драйверах.

Библиотека LabVIEW драйверов измерительных приборов расположена на LabVIEW CD. Драйверы также можно скачать с сайта компании National Instruments по адресу www.ni.com/idnet. Для установки драйверов следует их разархивировать и поместить полученную директорию в `\labview\instr.lib`. При последующем запуске LabVIEW Виртуальные Приборы драйверов измерительных приборов будут находиться на палитре **Functions»Instrument I/O»Instrument Drivers. Getting Started Example (Начальный пример)**. Все драйверы измерительных приборов содержат пример, который может быть использован для проверки связи с прибором. Этот пример обычно называется **Getting Started Example**. Укажите верный GPIB-адрес (или имя ресурса VISA) измерительного прибора, как он был настроен в MAX.

7.1.5. Использование ВП драйвера устройства

Драйверы создаются под определенные измерительные приборы и делают необязательным для пользователя точное знание команд стандарта IEEE 488.2, которые требуются для управления прибором.

Компоненты драйвера устройства

Все драйверы устройств в библиотеке имеют одинаковую базовую иерархию. Иерархия, последовательность использования ВП и обработка ошибок устроены так же, как и в ВП сбора данных, ВП файлового ввода/вывода, TCP/IP и т.д.

На рисунке 7.2 показана иерархия драйвера устройства.

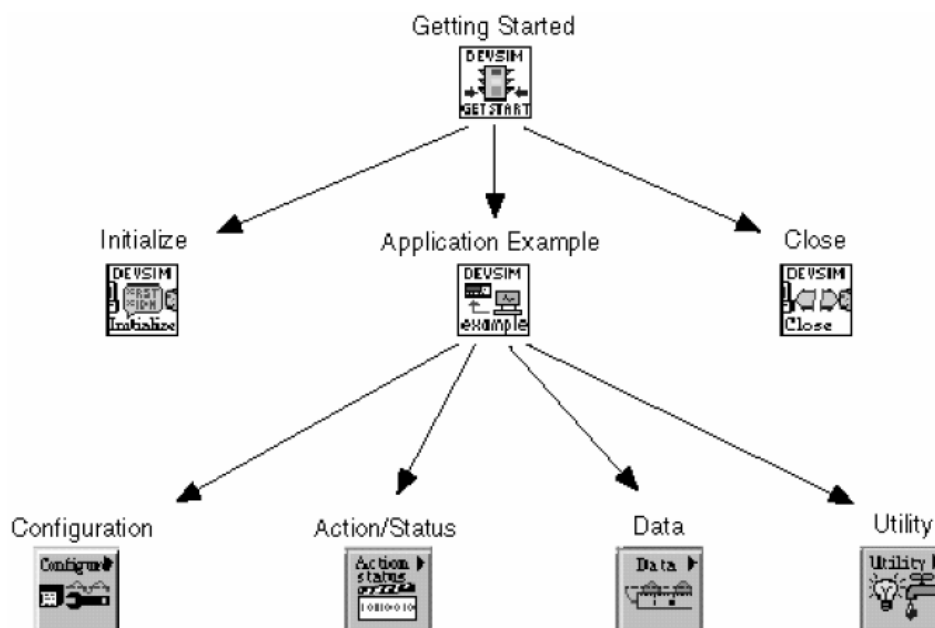


Рис. 7.2. Иерархия драйвера устройства.

Высокоуровневые функции созданы на основе низкоуровневых функций. Низкоуровневые функции обеспечивают расширенный контроль над интерфейсом, однако высокоуровневые функции просты в использовании и имеют удобные лицевые панели. Драйверы устройств содержат Виртуальные Приборы следующих категорий:

- **Initialize** – инициализирует каналы связи с устройством. Этот ВП также осуществляет операции идентификационного запроса, операции сброса или любые другие операции по сбросу значений установок устройства к значениям по умолчанию.
- **Configuration** – настраивает устройство для выполнения операций, таких как установка частоты триггера.
- **Action/Status** – включает в себя два типа ВП. ВП типа **Action VIs** осуществляют начало и остановку тестирования и измерительных операций. ВП типа **Status VIs** предназначены для получения текущего состояния устройства или статуса выполняемой операции.

Например, ВП типа **Action VIs** – ВП **Acquire Single Shot**. ВП типа **Status Vis** – ВП **Query Transfer Pending**.

- **Data** – выполняет операции передачи данных к/из измерительного прибора, такие как чтение измеренной осциллограммы (оцифрованный сигнал) из устройства или загрузка осциллограммы в устройство.
- **Utility** – выполняет широкий класс функций, таких как сброс, самотестирование, запрос ошибки или проверка состояния запроса.
- **Close** – разрывает канал связи с устройством и освобождает использованные для него ресурсы.

Все драйверы интерфейсов от компании National Instruments включают в себя следующие функции: инициализация, закрытие, сброс, самотестирование, проверка состояния запроса, обнаружение ошибки запроса и сообщение об ошибке.

Примеры приложений

Среда LabVIEW также включает примеры приложений, которые показывают, как использовать Виртуальные Приборы – компоненты драйвера для решения стандартных задач. Как правило, к этим задачам относятся: настройка, переключение и получение результатов измерения из измерительного прибора. Эти приложения не выполняют операции инициализации или закрытия драйвера устройства. Кроме того, эти ВП не претендуют на вариант полного приложения, а только демонстрируют возможности драйвера интерфейса и являются введением в разработку собственных драйверов интерфейса.

Поля ввода/вывода Виртуальных Приборов драйвера устройства

Так как иерархия ВП всех драйверов устройств одинакова, они имеют похожие поля ввода/вывода данных.

Имя ресурса или дескриптор устройства

При инициализации канала связи с устройством необходимо знать дескриптор этого устройства (**instrument descriptor**) или имя ресурса (**resource name**). Ресурс – это интерфейс или измерительный прибор, а идентификатор ресурса – точное имя и положение ресурса в данном формате: *Interface Type[board index]::Address::INSTR*. Необязательные параметры показаны в квадратных скобках [].

Например, GPIB::2::INSTR – дескриптор GPIB прибора по адресу 2. ВП **VISA resource name control**, размещенный в палитре **Controls»Modern»I/O**, аналогичен ВП **DAQ channel name control**, но ориентирован на управление измерительными приборами.

Для получения информации о доступных ресурсах и адресах устройств можно использовать конфигурационную утилиту **MAX** для настройки имени **VISA (VISA alias)**. Наличие имени упрощает связь с устройством, потому что, работая с именем, не нужно запоминать тип интерфейса и адрес устройства. Можно ввести имя вместо дескриптора устройства в поле имени ресурса **VISA**, например, можно ввести строку **devsim** вместо строки **GPIB::2::INSTR**.

Сессии VISA

После инициализации устройства ВП **Initialize VI** возвращает номер сессии **VISA**. Сессия **VISA** является связующим звеном с ресурсом, таким как измерительный прибор. Нет необходимости отображать это значение, однако каждый раз, когда происходит связь с устройством, необходимо соединять поля **VISA session** Виртуальных Приборов драйвера устройства. После завершения связи с устройством необходимо использовать ВП **Close VI** для закрытия всех ссылок и ресурсов, использованных во время связи.

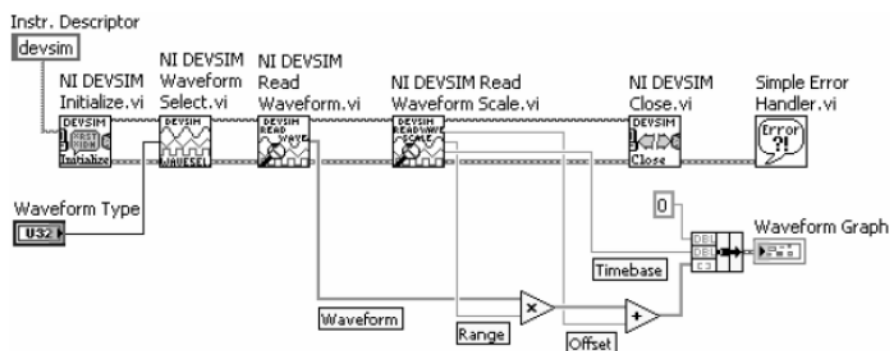


Рис. 7.3. Пример приложения драйвера интерфейса.

Блок-диаграмма, показанная на рисунке 7.3, инициализирует измерительный прибор под именем **devsim**. С помощью ВП настройки выбирается тип осциллограммы, затем используются два ВП: для чтения осциллограммы и ее масштабирования. После этого производится закрытие измерительного прибора и проверка статуса ошибки. Каждое приложение, которое использует драйвер интерфейса, применяет подобную последовательность выполнения событий.

7.2. Работа с GPIB приборами

Стандарт ANSI/IEEE Standard 488.1-1987, известный также как **General Purpose Interface Bus (GPIB)**, описывает стандартный интерфейс для связи между измерительными и управляющими приборами (сканеры, пленочные регистраторы и т.д.) различных производителей. Он включает в себя информацию об электрических, механических и функциональных спецификациях интерфейса. Интерфейс **GPIB** – это цифровой 8-битный параллельный коммуникационный интерфейс со скоростью передачи данных 1 Мбайт/с и выше. Он использует 3 линии синхронизации данных. Шина **GPIB** поддерживает один системный контроллер, обычно компьютер, и может управлять дополнительно 14-ю измерительными приборами. Стандарт ANSI/IEEE Standard 488.2-1992 расширяет

возможности IEEE 488.1 и описывает протокол связи, общие форматы данных и управляющих команд для стандартных устройств.

Интерфейс **GPIB** часто вводится в измерительные приборы широкого класса производителей с целью обеспечения возможности их тестирования. Интерфейс традиционно используется для самостоятельных настольных измерительных приборов с ручным управлением.

GPIB является 24-проводной параллельной шиной, состоящей из 8-ми линий данных, 5-ти линий управления шиной (**ATN, EOI, IFC, REN,** и **SRQ**), 3-х линий синхронизации и 8-ми заземляющих линий. В интерфейсе **GPIB** использована побайтовая асинхронная схема передачи данных, т.е. байты целиком последовательно передаются через шину на скорости, которая определяется скоростью самого медленного участника передачи. Поскольку в качестве единицы данных используется 1 байт, то передаваемые сообщения кодируются как символьные строки ASCII.

7.2.1. Адресация в интерфейсе GPIB

Каждый **GPIB** -измерительный прибор и **GPIB** -интерфейсная плата имеют уникальный **GPIB** -адрес в диапазоне от 0 до 30. Адрес 0 обычно присваивается **GPIB** -интерфейсу. Измерительные приборы, связанные с **GPIB** -интерфейсом, могут иметь адреса от 1 до 30. Каждое **GPIB** -устройство может быть передатчиком (**talker**) – источником сообщения, слушателем (**listener**) – устройством, принимающим данные, или контроллером (**controller**). Контроллер, обычно компьютер, управляет потоком информации, передаваемым по шине. Он определяет коммуникационные связи и посылает **GPIB** -команды измерительным приборам. Виртуальные Приборы **GPIB** автоматически оперируют адресацией и большинством других управляющих команд шины.

7.2.2. Остановка передачи данных

Прервать передачу **GPiB**-данных можно следующими тремя методами:

- Аппаратная линия **GPiB (EOI)** изменяет уровень при передаче последнего байта данных. Этот метод считается наиболее предпочтительным.
- Установка управляющего символа **end-of-string (EOS)** в конце строки передаваемых данных. Некоторые измерительные приборы используют этот метод вместо или в дополнение к аппаратному обозначению окончания передачи данных на линии **EOI**.
- Слушатель считывает заданное количество переданных байтов и останавливает чтение, когда счетчик достиг предельного значения. Этот метод часто используется как метод прерывания по умолчанию, потому что передача прекращается в соответствии с результатом логической операции ИЛИ параметров **EOI** и **EOS** (если этот способ задействован), логически умноженной на параметр счетчика байтов. Таким образом, количество байтов на счетчике устанавливается равным или превышающим число байтов, которые нужно прочесть.

7.2.3. Ограничения

Для достижения высокой скорости передачи, заложенной в **GPiB**-интерфейсе, следует ограничить количество измерительных приборов, присоединенных к шине, и физическое расстояние между устройствами.

Ниже перечислены типичные ограничения:

- Максимальное расстояния между двумя соседними измерительными приборами должно быть 4 м, а среднее расстояние должно составлять 2 м.
- Максимальная длина кабеля – 20 м.

- Максимальное количество измерительных приборов, соединенных с каждой шиной не должно превышать 15, при этом по крайней мере две трети приборов должны быть включены.
- Для достижения максимальной скорости передачи данных необходимо выполнять следующие условия:
- Все измерительные приборы в системе должны быть включены.
- Длина кабеля должна быть как можно короче, максимум длины кабеля для каждой системы составляет 15 м.
- Измерительные приборы должны располагаться через каждый метр длины кабеля.

Если необходимо превысить перечисленные ограничения, то при увеличении длины кабеля используется устройство **bus extender**, а при увеличении количества измерительных приборов – устройство **bus expander**. Эти устройства можно заказать на сайте компании National Instruments.

Для получения дополнительной информации о **GPIB** используйте ссылку на Web-сайт: ni.com/support/gpibsupp.htm.

7.2.4. Архитектура программных средств

Архитектура программных средств LabVIEW для управления измерительными приборами посредством **GPIB**-интерфейса аналогична архитектуре программных средств **DAQ**. В **GPIB**-систему включен комплект драйверов, эти драйверы также входят в комплект поставки LabVIEW CD, и большинство из них можно скачать по адресу ni.com/support/gpib/versions.htm. Всегда необходимо устанавливать новейшие версии драйверов, если нет особых указаний в инструкции к конкретному **GPIB**-интерфейсу или в руководстве к версии LabVIEW.

Конфигурационная утилита **MAX (Measurement & Automation Explorer)** используется для настройки и тестирования аппаратных средств **GPiB**. **MAX** взаимодействует с установленными вместе с драйвером средствами диагностики и настройки, а также с реестром и Диспетчером устройств ОС Windows. Программный драйвер является динамической библиотекой (**DLL**) и включает в себя все функции прямой связи с **GPiB**-интерфейсом. Виртуальные Приборы и функции раздела **Instrument I/O** работают напрямую с драйверами устройств.

7.2.5. Программные средства настройки

Утилита настройки **MAX** предназначена для управления аппаратными и программными средствами компании National Instruments. Утилита позволяет проводить диагностику системы, добавлять новые каналы, интерфейсы и виртуальные каналы, просматривать устройства и измерительные приборы, подсоединенные к системе.

Запуск утилиты **MAX** осуществляется двойным щелчком левой кнопкой мыши по ее иконке на рабочем столе или выбором в главном меню среды LabVIEW пункта **Tools»Measurement & Automation Explorer**.

Панель **Configuration** утилиты **MAX** включает в себя несколько секций под элементом **My System**:

- **Data Neighborhood** – секция предназначена для создания и тестирования виртуальных каналов, имен и дескрипторов каналов и измерений, настраиваемых в секции **Devices and Interfaces**.
- **Devices and Interfaces** – секция предназначена для настройки ресурсов и других физических свойств устройств и интерфейсов, а также для просмотра атрибутов, например, серийных номеров, одного или многих устройств.

- **IVI Instruments** – секция предназначена для создания имен виртуальных инструментов **IVI**, изменения их параметров и перемены мест инструментов **IVI**.
- **Scales** – секция предназначена для проведения простых операций масштабирования данных, таких как преобразование температуры из единиц напряжения Вольты, зарегистрированных температурным датчиком, в градусы шкалы Цельсия.
- **Historical Data** – секция используется для доступа к базам данных и записанным данным.
- **Software** – секция предназначена для определения установленных драйверов и приложений от National Instruments, а также их версий.
- **VI Logger Tasks** – секция используется для создания, изменения, запуска и просмотра задач **VI Logger**.

На рисунке 7.4 показана панель утилиты **MAX** после нажатия кнопки **Scan for Instruments** на инструментальной панели.

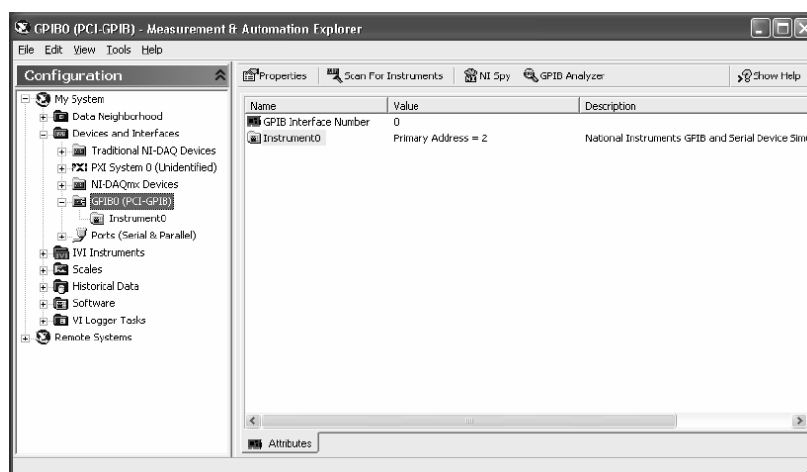


Рис. 7.4. Панель утилиты **MAX** в режиме **Scan for Instruments**.

Раздел **Remote Systems** на панели **Configuration** позволяет просматривать и настраивать удаленные системы, такие как контролеры **RT Series PXI Controllers**. Настройка объектов, перечисленных в **MAX**,

осуществляется щелчком правой кнопки мыши по имени объекта и выбором пунктов контекстного меню.

7.3. Работа с RS-232 приборами

7.3.1. Последовательная связь

Последовательная связь – наиболее распространенное средство передачи данных между компьютером и периферийными устройствами, такими как программируемые измерительные приборы или даже другие компьютеры. Последовательная связь использует передатчик для передачи данных последовательно бит за битом по однопроводной линии к приемнику. Этот метод можно использовать, когда скорость передачи мала или необходимо передать данные на большие расстояния.

Распространенность последовательной передачи определяется тем, что большинство компьютеров имеют один или более последовательных портов и не требует никаких дополнительных аппаратных средств, кроме нуль-модемного кабеля (рис. 7.5).

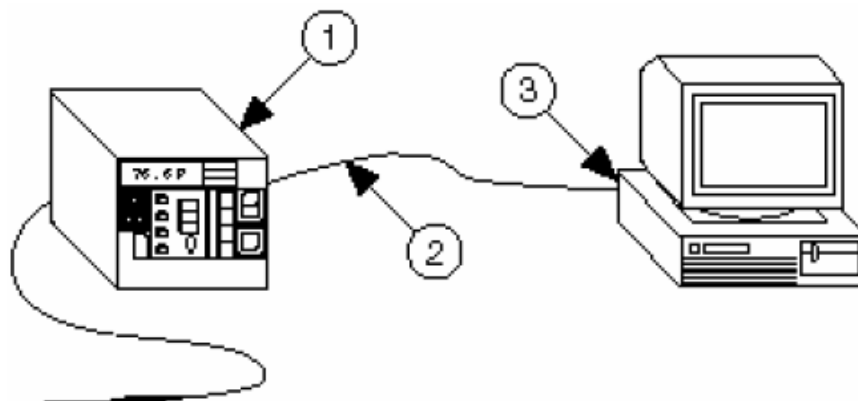


Рис. 7.5. Соединение по последовательному интерфейсу.

- 1 – Измерительный прибор с RS-232 интерфейсом;
- 2 – Кабель RS-232 (нуль-модемный кабель);
- 3 – Последовательный порт.

7.3.2. Настройка последовательного порта

Для использования последовательной связи требуется задание четырех параметров:

- Скорость передачи (baud rate).
- Количество битов данных, кодирующих передаваемый символ.
- Наличие бита четности (parity bit).
- Количество стоп-битов (stop bits).

Каждый передаваемый символ упаковывается в кадр символа, который состоит из одиночного стартового бита (**start bit**), за ним следуют биты данных (**data bits**), бит четности (если установлен) и заданное количество стоповых битов. На рисунке 7.6 показан кадр символа буквы **m**.

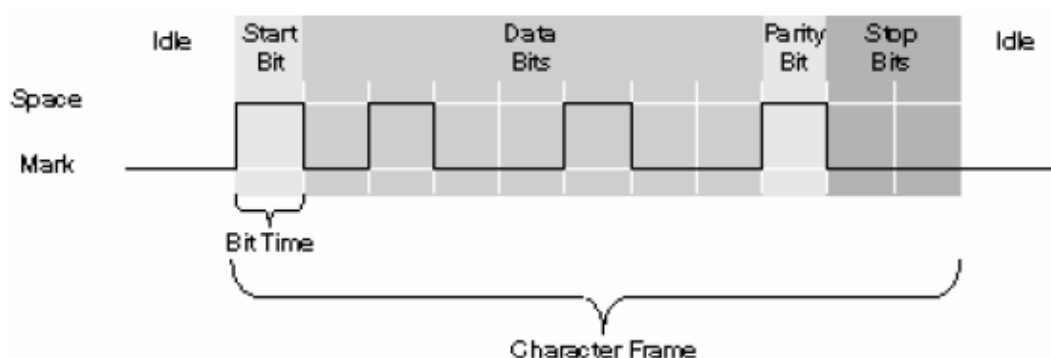


Рис. 7.6. Пример кодировки данных.

Скорость передачи показывает, как быстро поток данных перемещается между устройствами, использующими последовательный порт. Последовательный порт **RS-232** использует только два уровня напряжения, которые называются **MARK** (ЛОГИЧЕСКАЯ ЕДИНИЦА) и **SPACE** (ЛОГИЧЕСКИЙ НОЛЬ). В такой двухуровневой схеме передачи скорость передачи равна максимальному количеству информационных битов, включая управляющие биты, в секунду. **MARK** – отрицательное значение напряжения, **SPACE** – положительное.

На рисунке 7.6 было показано, как идеальный сигнал выглядит на осциллографе. Таблица истинности для RS-232 выглядит следующим образом:

Уровень сигнала $> +3 \text{ V} = 0$

Уровень сигнала $< -3 \text{ V} = 1$

Выходной уровень сигнала обычно находится в диапазоне $+12 - -12 \text{ В}$. «Мертвая зона» (от -3В до $+3 \text{ В}$) необходима для отсечки шума коммуникационной линии.

Стартовый бит сигнала является началом каждого кадра символа. Он передается изменением уровня от отрицательного (**MARK**) напряжения к положительному (**SPACE**). Его длительность характеризует скорость передачи. Если устройство передает на скорости 9600 бод, длительность стартового бита и всех последующих бит составляет 0,104 мс. Полный кадр символа (11 бит) передается примерно за 1,146 мс.

Биты данных передаются в обратном порядке с использованием отрицательной логики. Сначала передаются младшие биты (LSB – least significant bit), затем старшие биты (MSB – most significant bit). Для интерпретации битов, данных в кадре символа, необходимо читать кадр справа налево, причем **1** соответствует отрицательному значению напряжения, а **0** – положительному. На рисунке биты символа можно прочесть как **1101101** в двоичном формате или **6D** в шестнадцатеричном формате. Это значение по таблице ASCII соответствует символу **m**.

Если установлен бит четности, то он следует за битами данных в кадре символа. Бит четности также передается в инвертированной логике: **1** – отрицательное значение напряжения; **0** – положительное. Бит четности является простым средством обнаружения ошибок передачи. Заранее определяется, какой будет четность передачи – четной или нечетной.

Если выбрана нечетная схема, то передатчик устанавливает бит четности так, чтобы сделать число, состоящее из битов данных и бита четности, нечетным. Приведенная в примере операция передачи данных использует нечетную схему. Количество битов данных равно 5, т.е. уже нечетное число, поэтому бит четности равен 0. Конец кадра символа состоит из 1, 1,5 или 2 стоповых битов. Эти биты всегда представлены отрицательным уровнем напряжения. Если передачи символов больше не происходит, то линия связи остается в отрицательном уровне напряжения (**MARK**). Изменение напряжения к положительному уровню (**SPACE**) показывает начало передачи следующего кадра символа.

Максимальная скорость передачи

Зная структуру кадра символа и значение скорости передачи, можно подсчитать максимальную частоту передачи в символах за 1 с для заданных параметров связи. Эта частота является отношением скорости передачи к общему количеству битов в кадре символа. В предыдущем примере общее количество битов в кадре символа было равным 11. Если скорость передачи установить равной 9600 бод, то частота передачи будет равна $9600/11 = 872$ символов в секунду. Полученное значение является максимальной частотой передачи символов. Аппаратные средства интерфейса могут не достигнуть этого значения по разным причинам.

7.3.3. Обзор аппаратных средств

Существует много различных стандартов последовательной связи, наиболее распространенные из которых перечислены ниже.

RS-232

Стандарт RS-232 разработан **Electronic Industries Association (EIA)** и другими заинтересованными организациями в целях описания последовательного интерфейса между **Data Terminal Equipment (DTE)** и

Data Communications Equipment (DCE). Стандарт RS-232 включает описание характеристики электрического сигнала (уровня напряжения), механические спецификации (разъемы), функциональное описание линий (функции каждого электрического сигнала) и несколько способов соединения терминала с модемом. Наиболее часто встречающееся расширение этого стандарта называется RS-232C. Этот стандарт описывает (с различным уровнем надежности) использование последовательного порта для связи между компьютером и принтером, модемом и другими периферийными устройствами. Стандарту RS-232 соответствуют последовательные порты IBM-совместимых ПК.

RS-449, RS-422, RS-423

Стандарты RS-449, RS-422 и RS-423 являются дополнительными стандартами **EIA** последовательной связи. Эти стандарты являются производными от RS-232. Стандарт RS-449 был издан в 1975 году и предназначался для замены стандарта RS-232, однако его поддержали мало производителей. Стандарт RS-449 включает в себя две разновидности, названные RS-422 и RS-423. Если стандарт RS-232 описывает одноуровневую передачу, при которой уровень сигнала измеряется относительно общего заземления, то стандарт RS-422, напротив, описывает уровни сигналов друг относительно друга (относительная передача). Если в стандарте RS-232C приемник оценивает, насколько значение отрицательного уровня напряжения относительно земли достаточно, чтобы быть логической 1, то в RS-422 приемник оценивает отрицательный уровень сигнала относительно другого сигнала. Это делает стандарт RS-422 более защищенным от помех, нечувствительным к интерференции и ориентированным на большие расстояния передачи. Последовательные порты компьютера **Macintosh** соответствуют стандарту RS-422, стандарт RS-423 является модернизацией стандарта RS-422 и предназначен для

совместимости со стандартом RS-232. С помощью специального внешнего кабеля интерфейс RS-422 преобразуется к RS-423 и позволяет осуществлять связь с интерфейсом RS-422 на расстоянии более 15 м.

Кабель RS-232

Устройства, использующие последовательный порт, можно разделить на две категории: **DCE** и **DTE**. Категория **DCE** – это устройства, такие как модем, плоттер и т.д. Категория **DTE** – это последовательные порты другого компьютера. Разъемы RS-232 последовательного порта существуют в двух вариантах: 25- и 9-штырьковые (**D-Type 25-pin connector** и **D-Type 9-pin connector**). Оба варианта на компьютере являются «папами» и требуют разъема типа «мама» на приборе. На рисунке 7.7 показан внешний вид 9-штырькового разъема.

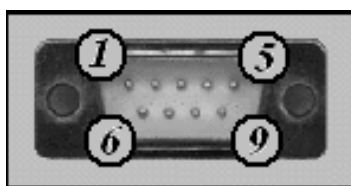


Рис. 7.7. Внешний вид разъема DB-9M.

Вариант 9-штырькового разъема (DB-9) обычно используется в лабораторном оборудовании небольших размеров. Он является компактной версией стандарта RS-232. DB-9 используют для передачи и приема 3-й и 2-й штырьки соответственно. 25-штырьковый разъем (DB-25), напротив, использует для передачи и приема штырьки 2-й и 3-й. Необходимо обращать внимание на эти различия при определении категории прибора – **DTE** и **DCE**.

Таблица 7.2. Описание контактов разъема RS-232 DB-9M.

Функция (Function)	Сигнал (Signal)	Номер штырька (PIN)	Категория DTE	Категория DCE
Обмен данных (Data)	TxD	3	Output	Input
	RxD	2	Input	Output
Управление интерфейсом (Handshake)	RTS	7	Output	Input
	CTS	8	Input	Input
	DSR	6	Input	Output
	DCD	1	Output	Output
	DTR	4		Input
Общий (Common)	Com	5	–	–
Другие (Other)	RI	9	Input	Output

25-штырьковый разъем (DB-25) соответствует полной версии стандарта RS-232. Внешний вид разъема показан на рисунке 7.8.

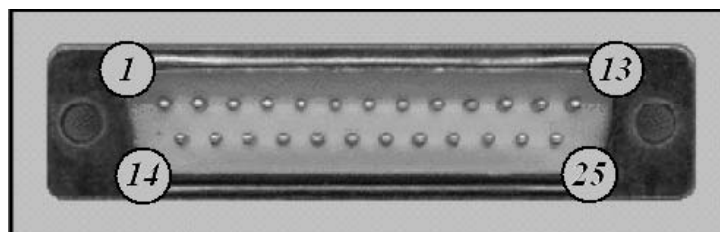


Рис. Рис. 7.8. Внешний вид разъема DB-25M.

Таблица 7.3. Описание контактов разъема RS-232 DB-25M.

Функция (Function)	Сигнал (Signal)	Номер штырька (PIN)	Категория DTE	Категория DCE
Обмен данных (Data)	TxD	2	Output	Input
	RxD	3	Input	Output
Управление интерфейсом (Handshake)	RTS	4	Output	Input
	CTS	5	Input	Output
	DSR	6	Input	Output
	DCD	8	Input	Output
	DTR	20	Output	Input
Общий (Common)	Com	7	–	–

7.3.4. Обзор программных средств

ВП и функции для работы с последовательным портом размещены на палитре **Functions»Instrument I/O»Serial**.

Функции **VISA (VISA Write и VISA Read)**, которые использовались для связи по GPIB-интерфейсу, также могут быть использованы для управления последовательной связью, как и любым другим интерфейсом. Так как последовательный порт использует много параметров, помимо этих функций необходимо дополнительно использовать ВП **VISA Configure Serial Port VI**. ВП **VISA Configure Serial Port VI** инициализирует последовательный порт, определенный полем **VISA resource name**, установленными значениями. Поле **timeout** устанавливает время простоя последовательного порта. Поля **baud rate**, **data bits**, **parity** и **flow control** определяют параметры настройки последовательного порта. На поля **error in** и **error out** подается кластер ошибок, содержащий информацию об ошибке.

Пример на рисунке 7.9 показывает, как осуществляется посылка управляющей команды ***IDN?** измерительному прибору через последовательный порт COM2.

ВП **VISA Configure Serial Port VI** открывает связь через последовательный порт COM2 и устанавливает его параметры: 9600 бод, 8 бит данных, проверка на нечетность (**odd parity**), один стоповый бит и программное управление передачей **XON/XOFF**. Функция **VISA Write** посылает управляющую команду. Функция **VISA Read** принимает 200 байт в буфер чтения, и ВП **Simple Error Handler VI** проверяет состояние кластера ошибок.

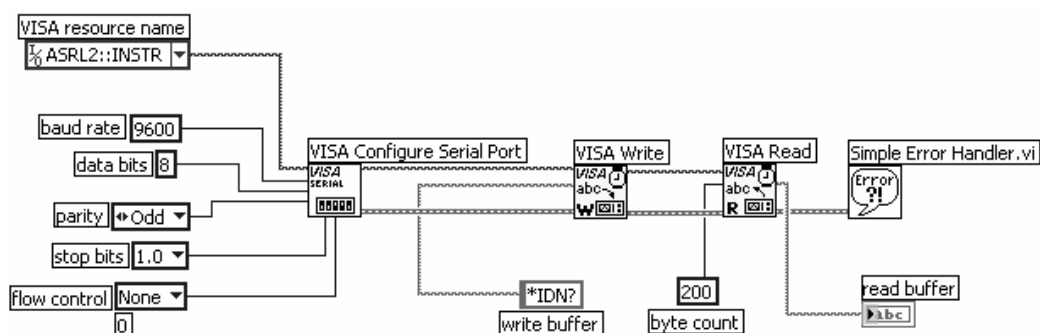


Рис. 7.9. Пример передачи команды через порт COM2.

ВП и функции, размещенные на палитре **Functions»Instrument I/O»Serial**, можно использовать для связи через параллельный порт. В поле **VISA resource name** должен быть описан дескриптор параллельного порта LPT. Например, с помощью конфигурационной утилиты **MAX** можно назначить порту LPT1 имя ресурса **VISA – ASRL10::INSTR**.

7.3.5. Передача сигнальных данных

Многие измерительные приборы возвращают сигнальные данные в виде ASCII-строк или в двоичном формате. Необходимо принять во внимание то, что передача сигнальных данных в бинарном формате

происходит быстрее и требует меньшего количества памяти, чем передача сигнальных данных в виде ASCII-строк. Кодирование сигнальных данных в бинарный формат требует меньшего количества байт, чем кодирование в ASCII-строки.

Сигнальные данные в формате ASCII

Например, рассматриваемый сигнал состоит из 1024 значений, которые находятся в диапазоне от 0 до 255. Использование ASCII-кодировки требует 4 байта для представления каждого значения (3 байта для числа и 1 байт для разделителя, такого как запятая). Таким образом, потребуется 4096 (4*1024) байт плюс заголовок и плюс сопроводительные символы, указывающие на тип представления осциллограммы данных в виде ASCII-строки. Ниже показан пример данных в формате ASCII-строки.

CURVE	{12,28,63,...1024 значения}	CR
Заголовок	значения	сопроводительный
(6 байт)	(по 4 байта на каждое значение)	символ
		(2 байта)

Для преобразования данных в формате ASCII-строки необходимо использовать ВП **Extract Numbers VI**, размещенный в палитре **Functions»User Libraries»Basics I Course**, как показано на блок-диаграмме.

Сигнальные данные в формате однобайтового целого

В этом формате сигнальные данные требуют только 1024 байта. В сумме: (1*1024) плюс заголовок плюс сопроводительный символ, представленный в двоичном формате. Для преобразования сигнальных данных в двоичный формат кодов символов требуется 1 байт на значение, т.е. каждое значение представлено как беззнаковое байтовое значение. Пример демонстрирует осциллограмму данных в формате 1-байтовых целых.

CURVE %	{MSB}{LSB}	{A,a...1024 значения}	{Chk} CR
Заголовок (7 байт)	счетчик (4 байта)	значения (по 1 байту на каждое значение)	сопроводительный символ (3 байта)

Сигнальные данные в формате двухбайтового целого

Каждое значение в этом формате представлено, как двухбайтовое целое и его преобразование можно осуществить с помощью функции **Type Cast**, размещенной в палитре **Functions»Programming»Numeric»Data Manipulation**.

Например, **GPiB**-осциллограф передает сигнальные данные в формате двухбайтового целого. Данные состоят из 1024 значений, каждое из которых представлено 2 байтами. Поэтому в этом формате сигнальные данные представлены 2048 байтами. Ниже показан формат сигнальных данных, состоящий из 4-байтового заголовка, данных и 2-байтового сопроводительного символа **linefeed**.

ДАТА	{(HB1,LB1),...1024 значения}	CR	CF
Заголовок (4 байта)	значения (по 2 байта на каждое значение)		сопроводительный символ (2 байта)

На блок-диаграмме (рис. 7.10, а) показано использование функции **Type Cast** для преобразования сигнальных данных в формате двухбайтового целого в массив 16-битовых (двухбайтовых) целых.

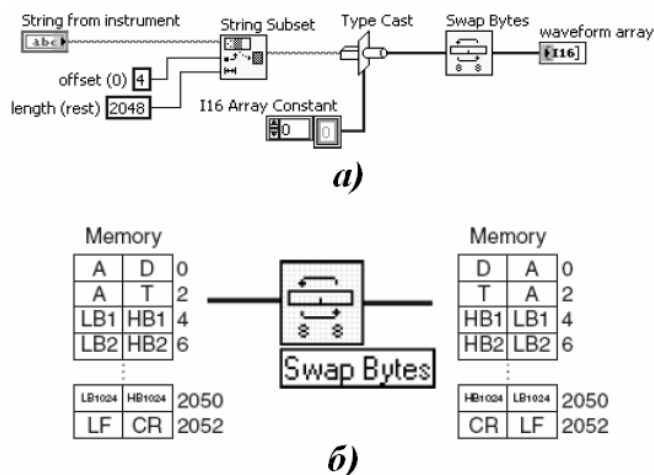


Рис. 7.10. Использование функции **Type Cast**.

Для перестановки старших и младших байтов значений необходимо использовать функцию **Swap Bytes**, размещенную на палитре **Functions»Programming»Numeric»Data Manipulation. GPIB** – 8-битовая шина, в каждый момент времени по ней передается только один байт. От измерительного прибора данные идут в обратной последовательности, поэтому и требуется функция **Swap Bytes**. Измерительный прибор передает вначале старший байт, затем младший байт, а приемник-интерфейс старший байт принятого значения физически размещает по адресу младшего, младший байт – по адресу старшего. Поэтому и требуется их перестановка с помощью функции **Swap Bytes**, как показано на рисунке 7.10, б).

Используемая литература:

1. Тревис Дж. LabVIEW для всех. – М.: «ДМК», 2004 г. – С. 544.
2. Суранов А.Я. LabVIEW 8.20. Справочник по функциям. – М.: «ДМК», 2007 г. – С. 536.
3. Поллак Б.П., Точилин Д.А., Л. И. Пейч Л.И. LabVIEW для новичков и специалистов. – М.: «Горячая Линия – Телеком», 2003 г. – С. 384.
4. Папуловский В.Ф., Мошкин В.В., Бессонов А.С., Батоврин В.К. LabVIEW. Практикум по основам измерительных технологий. Учебное пособие для вузов – М.: «ДМК», 2005 г. – С. 208.
5. Бессонов А.С., Мошкин В.В., Батоврин В.К. LabVIEW. Практикум по электронике и микропроцессорной технике. – М.: «ДМК», 2005 г. – С. 182.
6. Загидуллин Р.Ш. LabView в исследованиях и разработках. – М.: «Горячая линия – Телеком», 2005 г. – С. 352.
7. Линдваль В.Р., Щербаков Г.И., Евдокимов Ю.К. LabVIEW для радиоинженера. От виртуальной модели до реального прибора. – М.: «ДМК», 2007 г. – С. 400.
8. Нестеренко А., Федосов В. Цифровая обработка сигналов в LabVIEW. – М.: «ДМК» 2007 г. – С. 472.

Интернет-ресурсы:

LabVIEW user manuals. ni.com/manuals

ОПИСАНИЕ КУРСА И ПРОГРАММА

ОБЩЕЕ ОПИСАНИЕ (КОНЦЕПЦИЯ) КУРСА

Цель курса – обеспечение базовой подготовки в области использования среды графического программирования LabVIEW; введение в теорию и методику современного сбора данных; получение практических навыков в области современных методов получения и обработки экспериментальных данных с использованием новейших цифровых технологий; приобретение студентами базовых знаний в области автоматизации физического эксперимента.

Содержание курса – Среда графического программирования. Введение в LabVIEW. Виртуальные приборы (ВП). Последовательность обработки данных. Типы и проводники данных. Редактирование и отладка ВП. Подпрограммы ВП. Циклы. Структуры принятия решений. Использование узла Формулы. Массивы. Кластеры. Кластеры ошибок. Графическое отображение данных. Работа со строковыми данными. Файловый ввод/вывод. Настройка ВП. Организация системы сбора данных и управления в LabVIEW. Ввод аналогового сигнала. Генерация аналогового сигнала. Работа с цифровыми сигналами. Управление измерительными приборами. Архитектура программного обеспечения виртуальных интерфейсов (VISA). Драйверы измерительных приборов. Работа с GPIB приборами. Работа с RS-232 приборами.

Организационно-методическое построение курса.

Курс состоит из лекций и практических занятий (лабораторных работ), предусмотрено выполнение контрольного практического задания или курсовой работы (по выбору). Материал курса предполагает наличие у

слушателей базовых знаний в области программирования и создания алгоритмов. Естественным требованием является навык пользования ПК. В курсе используются материалы некоторых разделов высшей математики (математический анализ, дифференциальные уравнения), курсов общей физики (электричество, элементы теории цепей), входящих в учебный план обучения бакалавра классического университета по направлению подготовки – физика.

Лекции построены по принципу от простого к сложному и реализуют непрерывную подготовку в рамках учебной программы. Лекции проходят в дисплей-классе, оснащённом самым современным мультимедийным оборудованием. Студенты имеют возможность непосредственной работы с примерами работающих и отлаживаемых программ «по горячим следам» с возможностью применения как виртуальных моделей, так и реально действующего лабораторного оборудования.

Практические занятия проводятся в три этапа: 1) допуск к выполнению - проверка преподавателем самостоятельной работы студента, т.е. персональная проверка знаний вопросов связанных с тематикой предстоящей практической работы, подготовленных с использованием методических рекомендаций и предложенной литературы; 2) непосредственно разработка программы в соответствии с заданием, самостоятельная отладка и оптимизация кода, выполнение всех заданий и оформление результата, согласно методическим требованиям; 3) обсуждение полученных результатов.

Для контроля и закрепления студентами полученных знаний необходимо проведение обязательных практических занятий: лабораторные работы, минимальное количество - 7 в семестр (всего 14 работ за год); 1 контрольное практическое задание (в семестр, 2 за год) или курсовая работа (по выбору). Предусмотрены: промежуточная аттестация в

середине семестра, семестровый зачет по лабораторному практикуму и семестровый итоговый контроль – экзамен.

Освоив курс, студент должен:

Знать структуру и лексику графической среды программирования LabVIEW, уметь грамотно разделить комплексную задачу на ряд более простых и легко реализуемых фрагментов (процедур), уметь отлаживать взаимодействие процедур друг с другом, иметь представление об общих правилах разработки виртуальных интерфейсов, овладеть базовыми методами сбора (аналоговый и цифровой ввод/вывод, синхронизация) и обработки данных с применением современных DAQ-устройств.

Общие правила выполнения практических работ

В рамках читаемого курса в лаборатории практикума «Современные графические среды программирования» студенты выполняют лабораторные работы, количество которых определено учебным планом по дисциплине, а также выполняют самостоятельную исследовательскую работу - контрольное практическое задание или курсовую работу.

Лабораторное занятие - практическое учебное занятие, проводимое в учебных лабораториях с целью углубления знаний и приобретения навыков разработки приложений в области изучаемой дисциплины.

В качестве задания на лабораторную работу предлагается разработка простого приложения, выполняющего заданные функции. Объектом, с которым работает студент при проведении лабораторных занятий, может являться как конкретный физический объект (электрофизический стенд, измерительный модуль, измерительный прибор), так и модель измерительного прибора, математический метод и т.д.

Курсовая работа является самостоятельной учебной работой, в которой студентом раскрываются теоретические и практические проблемы выбранной темы.

В качестве заданий для курсовых работ предлагается разработка сложных приложений, выполняющих измерения и управляющих измерительной установкой, процессом измерений, сбором данных их обработкой, отображением и хранением с привлечением современных средств автоматизации измерений и математической обработки экспериментальных данных.

Объектом, с которым работает студент при выполнении курсовой работы, является существующий или вновь создаваемый измерительный стенд для исследований физических явлений или процессов и управления ходом эксперимента. Студент самостоятельно обеспечивает автоматизацию измерений при этом используемые диагностические методы и методы моделирования основаны как на знаниях полученных студентом в предшествующий период обучения, так и на знаниях получаемых самостоятельно. Обязательным требованием к выполнению курсовой работы является написание отчета, состоящего из подробного описания экспериментальной установки, описания алгоритма приложения, правил работы с интерфейсом программы и методических указаний по проведению работы на созданной установке. Результаты курсовой работы могут быть использованы при подготовке будущей дипломной работы.

Контрольное практическое задание – самостоятельная учебная работа, направленная на разработку приложения, обеспечивающего модернизацию и совершенствование работ лабораторного практикума.

Контрольное практическое задание представляет, заверченный материал, в котором представлены результаты самостоятельной работы студента по разработке и постановке новых упражнений лабораторного практикума и связаны с совершенствованием постановки эксперимента, разработки новых схем измерений, повышения точности измерений или обработки экспериментальных данных и содержат авторское видение и решение поставленной задачи.

Объектом исследований при выполнении контрольного практического задания являются существующие и вновь создаваемые лабораторные стенды практикума «Современные графические среды программирования».

Организация лабораторного практикума

Общие правила:

- Лабораторные работы выполняются студентами согласно установленного в начале семестра расписания.
- Лабораторная работа выполняется группой разработчиков, не превышающей 2 человека.
- Количество лабораторных работ, выполняемых за учебное занятие, не превышает одну работу.
- Перенос выполнения назначенной лабораторной работы допускается только в пределах расписания по согласованию с преподавателем.
- При обнаружении схожих программ (идентичных алгоритмов, одинаковых интерфейсов и т.п.) у различных групп разработчиков результаты работ аннулируются, а студенты переделывают работу в дополнительное время, в сроки, согласованные с преподавателем.

К выполнению работы не допускаются учащиеся, которые:

- не прошли аттестацию по технике безопасности;
- грубо нарушают правила внутреннего распорядка в лаборатории;
- не подготовились для выполнения лабораторной работы;
- опоздали к началу занятий;
- не прошли собеседование-защиту по предыдущей работе;
- пропустили два и более занятий без уважительной причины.

Организация лабораторных занятий включает:

- самостоятельную внеаудиторную подготовку студента в соответствии с методическими рекомендациями;

- первичный контроль преподавателем степени подготовленности каждого студента к выполнению лабораторной работы;
- выполнение всех заданий (упражнений) лабораторной работы в полном объеме;
- учет преподавателем текущего и итогового рейтингов каждого из студентов по результатам выполнения и защиты им отдельных лабораторных работ.

Студент имеет право:

- получить необходимые для выполнения лабораторной работы методические материалы в бумажном или электронном видах;
- проводить лабораторные работы по оригинальным методикам при предварительном согласовании их с преподавателем;
- выполнить лабораторную работу, пропущенную по уважительной причине, в часы, согласованные с преподавателем.

Студент обязан:

- Самостоятельно, согласно методическим рекомендациям, подготовиться к выполнению лабораторной работы, и получить допуск к ее выполнению по результатам краткого опроса в начале занятий;
- По выполнении экспериментальной части лабораторной работы студент предъявляет действующее приложение преподавателю. Оно сохраняется в виде электронного файла на рабочем компьютере преподавателя. Копия файла остается у студента;
- При пропуске занятия подготовиться к следующей по расписанию работе. Дату выполнения пропущенной работы необходимо согласовать с преподавателем.

Студент несет ответственность:

- за пропуск лабораторных занятий по неуважительной причине;
- за неподготовленность к выполнению работы;

- за несвоевременное выполнение работ и их защиту;
- за порчу имущества и нанесение материального ущерба лаборатории

Преподаватель несет ответственность:

- за качественную постановку и проведение лабораторных работ;
- за поддержание рабочей дисциплины и порядка в лаборатории;

Преподаватель имеет право:

- отстранять от проведения лабораторных работ студентов, нарушающих установленные выше правила;
- требовать от студентов пропустивших занятия разрешения из деканата факультета на посещение последующих лабораторных занятий;
- проводить перед началом лабораторных работ контрольный опрос студентов;
- вносить в случае необходимости (из-за отказа оборудования, измерительных или вычислительных средств и т. п.) частичные изменения в программу лабораторной работы.

Категорически запрещено:

- **Самостоятельно включать** экспериментальные стенды, без проверки преподавателем рабочей схемы измерений;
- **Использовать** для выполнения лабораторной работы приборы и устройства, не входящие в состав экспериментального стенда и не предусмотренные техническим заданием к выполнению;
- **Включать** не используемые в работе модули устройств, ручки и переключатели приборов;
- **Вскрывать** блок-схемы программно-аппаратных средств измерений и вносить в них изменения;
- **Включать измерительные стенды**, в которых предусмотрены системы охлаждения без их активации;

- **Несанкционированно подключать** к компьютерам дисплей-класса дополнительные устройства;
- **Несанкционированно работать** с локальной сетью лаборатории;
- **Вскрывать** измерительные модули и приборы.

Организация выполнения курсовых работ или контрольных практических заданий.

О необходимости выполнения курсовой работы студента информируют на первой лекции и предлагают либо выбрать тему работы из списка, либо заранее самостоятельно обдумать и предложить тему будущей работы.

Подготовка и защита курсовой работы, а также ее оценка в учебной программе каждого направления определяется большой ролью этого вида подготовки специалиста в общей системе учебных программ, нацеленных на фундаментальность и систематичность образования. Для завершения и защиты курсовой работы студенту предоставляется в конце семестра специально одна неделя, свободная от аудиторных занятий. В течение первой установочной недели, отведенной на выбор курсов, утверждается тема и научный руководитель курсовой работы. Темы курсовых работ находятся на портале кафедр. Студент, консультируясь с преподавателями кафедры, за которой закреплено руководство курсовыми работами, определяет тему своей курсовой работы.

Выполнение курсовых работ происходит по следующей схеме:

- выбор темы;
- поиск литературы и ее изучение;
- разработка кода программы, проведение необходимых расчетов и экспериментальных исследований;
- представление первого варианта программы научному руководителю;
- исправление и доработка приложения на основе замечаний руководителя;

- представление первого варианта отчета научному руководителю;
- исправление и доработка отчета на основе замечаний руководителя;
- представление окончательного варианта курсовой работы и ее защита перед учебной комиссией кафедры.

Отчетность (ведомость) по курсовой работе сдается в учебную часть в строго определенные сроки (первая учебная неделя).

Выполнение контрольных практических заданий происходит по аналогичной схеме, единственное отличие - сдача работы происходит курирующему преподавателю.

Выполнение экспериментальной части работы требует разработки программного, программно-аппаратного обеспечения, а также изготовления узлов, устройств или приспособлений для существующих в лаборатории по данному курсу или вновь создаваемых электрофизических стендов.

Разработка программного, программно-аппаратного обеспечения осуществляется в дисплей-классе центра прикладных информационных технологий университета, оснащенного всем необходимым оборудованием и программным обеспечением.

Часть работ (слесарных, электро и радиомонтажных), связанных с выполнением курсовой работы или контрольного практического задания, выполняется студентом самостоятельно в технологической лаборатории кафедры экспериментальной физики, оснащенной соответствующим инструментарием и станками. Узлы, требующие квалифицированного изготовления, могут быть изготовлены в научно-производственных или стеклодувных мастерских университета. В этом случае чертежи, самостоятельно подготовленные студентом, утверждаются преподавателем, и оформляется заказ от кафедры экспериментальной физики согласно установленных в университете правил для их изготовления.

Естественным требованием при выполнении экспериментальной части работы является наличие удостоверения о сдаче в начале учебного года минимума по технике безопасности, строгое выполнение правил ТБ и выполнение требований и правил эксплуатации энергоустановок. Надзор за выполнением указанных требований осуществляется курирующим преподавателем.

В процессе выполнения курсовой работы или контрольного практического задания текущее состояние работы еженедельно обсуждается с преподавателем, курирующим их выполнение.

Обработка результатов практической части работы, и подготовка отчета по курсовому проекту может быть выполнена студентом в дисплей-классе центра прикладных информационных технологий университета, оснащенном всеми необходимыми аппаратными и программными ресурсами.

Защита курсовых работы осуществляется перед учебной комиссией, назначаемой заведующим кафедрой и представляет собой публичное научное сообщение по результатам работы, должным образом оформленное с использованием мультимедийных средств. Лучшие работы могут быть рекомендованы в качестве докладов на ежегодную университетскую конференцию или к публикации.

Инновационная составляющая курса.

Реализация предлагаемого УМК будет способствовать повышению уровня подготовки бакалавров. Выпускники будут способны самостоятельно разрабатывать приложения средней сложности в области автоматизации физического эксперимента, осуществлять самостоятельно планирование и постановку физического эксперимента с использованием самого современного оборудования.

Подготовка УМК обусловлена необходимостью изучения студентами бакалавриата как базовых аспектов изучаемой дисциплины, так и современных подходов и решений применяемых при разработке автоматизированных методик для физических исследований. Особое внимание в УМК уделено практическому аспекту подготовки, столь необходимой современному специалисту.

Инновационность подачи учебного материала в курсе находится в русле активного внедрения цифровых технологий. С одной стороны в курсе излагаются базовые возможности среды LabVIEW с использованием наглядного иллюстративного материала и современных средств мультимедиа и визуализации. С другой стороны – лабораторные и самостоятельные работы предполагают применение новейших образцов современной измерительной техники, а так же управление работой научно-исследовательских установок на базе аппаратных и программных средств последнего поколения.

Освоение курса связано с выполнением самостоятельных научно-практических исследований в форме контрольного практического задания или курсовой работы (по выбору), согласно перечню предложенных актуальных тем, являющихся прологом выполнения дипломной работы.

В реализуемой магистерской программе все УМК построены по единой форме расположения и организации материала, позволяют соотносить их содержание в общем контексте подготовки специалистов. Использование единого подхода к представлению учебно-методической информации дает возможность отобразить, с одной стороны, существующие межпредметные

взаимосвязи, а с другой – динамику развития отдельных тем и их сочетание, что особенно актуально при выполнении самостоятельных научных исследований (курсовые и дипломные работы, магистерская диссертация). Задачи лабораторного практикума и предлагаемые темы курсовых работ позволяют получить практические навыки по разделам изучаемой дисциплины.

Предлагаемая для изучения курса литература в основе своей имеется в наличии в библиотечном фонде РУДН, в противном случае электронные версии доступны в локальной сети на сайте магистерской программы (<http://vlab.sci.pfu.edu.ru>).

Разрабатываемый УМК является плодом совместной работы коллектива авторов, состоящих из преподавателей Вуза и ведущих специалистов в области LabVIEW-программирования - Международного научно-учебного лазерного центра МГУ имени М.В. Ломоносова. Такой подход позволяет рассмотреть последние достижения в области создания приложений для автоматизации физических измерений, обеспечить проведение НИРС по самым актуальным и перспективным направлениям, относящимся к приоритетным направлениям развития науки, технологии и техники РФ, на современных электрофизических стендах и установках. Кроме того, применить последние разработки в сфере цифровых информационных технологий для постановки оригинальных экспериментальных работ лабораторного практикума, включая удаленный доступ.

Авторы курса выделили две основные цели его написания:

1. Ознакомление студентов с одной из самых перспективных сред программирования, являющейся «де-факто» международным стандартом при сборе данных и управлении экспериментом.
2. Практическая составляющая курса, нацеленная на изучение и освоение методов и принципов проведения экспериментальных исследований с использованием программно-аппаратных средств последнего поколения, а также современного лабораторного и измерительного оборудования.

Предлагаемый учебный курс является принципиально инновационным как в области содержания, так и технологии организации педагогического процесса.

Обязательная литература:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004, 544 с.
2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
3. Б.П. Поллак, Д.А. Точилин, Л.И. Пейч *LabVIEW для новичков и специалистов*. М.: "Горячая Линия - Телеком", 2003., 384 с.
4. В.Ф. Папуловский, В.В. Мошкин, А.С. Бессонов, Батоврин В.К. *LabVIEW. Практикум по основам измерительных технологий. Учебное пособие для вузов "ДМК"*, 2005, 208 с.
5. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Дополнительная литература:

1. А.С. Бессонов, В.В. Мошкин, В.К. Батоврин *LabVIEW: Практикум по электронике и микропроцессорной технике*. М.: "ДМК", 2005, 182 с.
2. Р.Ш. Загидуллин *LabView в исследованиях и разработках*. М.: "Горячая линия -Телеком", 2005, 352 с.
3. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007, 400 с.
4. А. Нестеренко, В. Федосов *Цифровая обработка сигналов в LabVIEW*. М.: "ДМК Пресс", 2007, 472 с.

Интернет ресурсы:

1. *LabVIEW user manuals*. <http://www.ni.com/manuals>
2. П.М. Михеев *Основы современных систем сбора данных*
<http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

3. Сайт автоматизированных лабораторий удаленного доступа Центра прикладных информационных технологий РУДН <http://vlab.sci.pfu.edu.ru/>
4. Сайт Центра прикладных информационных технологий РУДН <http://aitc.sci.pfu.edu.ru>

Условия и критерии выставления оценок:

От студентов требуется посещение лекций и лабораторных (практических) занятий, выполнение и сдача обязательного количества лабораторных работ (зачет по лабораторному практикуму) и самостоятельной контрольной практической работы или курсовой работы (по выбору), а также **сдача итогового экзамена**. Особо учитывается активная работа при выполнении самостоятельного контрольного практического задания (или курсовой работы), а также ритмичность и качество выполнения обязательных лабораторных работ.

Для успешного выполнения каждой лабораторной работы студент должен внимательно изучить учебно-методические материалы (пособия и литературу, рекомендованную для выполнения лабораторных работ), уметь изложить изученный материал и быть готовым к выполнению. Важным этапом в итоговой аттестации студента является обязательное выполнение и сдача самостоятельного контрольного практического задания (или курсовой работы). Студент не допускается к итоговому экзамену, если он набрал менее 50% баллов. При набранных 60% баллов студент **может** автоматически получить оценку «3». Более высокая оценка может быть получена только на итоговом экзамене.

Балльная структура оценки:

За выполнение обязательных лабораторных работ (7 работ в одного семестр) – 35 баллов. За каждую лабораторную работу – 5 баллов (1 балл – допуск к выполнению работы, 1 балл – выполнение, 3 балла – оформление и сдача работы). В итоговом подсчете на лабораторные работы – 35 баллов, зачет

по лабораторному практикуму ставится автоматически при сдаче обязательного минимума – 7 лабораторных работ в семестре.

Самостоятельная контрольная практическая работа – 25 баллов (всего) – одна.

Промежуточный контроль (рубежная аттестация) – 10 баллов.

Семестровый итоговый экзамен – 30 баллов.

Всего – 100 баллов за семестр.

При выборе (выполнении) курсовой работы по данному курсу, контрольное практическое задание снимается, максимальный балл за курсовую работу – 25 баллов засчитывается в итоговую сумму баллов.

Шкала оценок:

A (5+) - $93 \leq 100$ баллов;

B (5) - $84 \leq 92$ баллов;

C (4) - $74 \leq 83$ баллов;

D (3+) - $63 \leq 73$ баллов;

E (3) - $51 \leq 62$ баллов;

FX (2+) - $31 \leq 50$ баллов;

F (2) - $0 \leq 30$ баллов.

		Неуд		3		4	5	
кредит	Сумма	F	FX	E	D	C	B	A
	Баллов	2	2+	3	3+	4	5	5+
2	100	0≤30	31≤50	51≤62	63≤73	74≤83	84≤92	93≤100

Пояснение оценок:

A – выдающийся ответ

B – очень хороший ответ

C – хороший ответ

D – достаточно удовлетворительный ответ

E – отвечает минимальным требованиям удовлетворительного ответа

FX – означает, что студент может добрать баллы только до минимального удовлетворительного ответа

F – неудовлетворительный ответ (либо повтор курса в установленном порядке, либо основание для отчисления).

Основные правила проведения экспериментальных исследований в учебных лабораториях

Общие положения:

Экспериментальные исследования (лабораторные и курсовая работы, контрольные практические задания) выполняются студентами согласно установленного в начале семестра расписания. К выполнению работы не допускаются учащиеся, которые:

- не прошли аттестацию по технике безопасности и правил эксплуатации электроустановок;
- грубо нарушают правила внутреннего распорядка в лаборатории;
- не подготовились для выполнения экспериментальных исследований;

Техника безопасности

- Инструктаж по «Правила технической эксплуатации электроустановок и Правила техники безопасности при их эксплуатации» проводится преподавателем, ведущим занятие, совместно с представителями служб главного инженера университета.
- При положительных результатах тестовых заданий получают допуск (выдается службой главного энергетика РУДН) для работы в лаборатории с оборудованием до 1000 В.

Студент обязан:

- Строго выполнять правила внутреннего распорядка в лаборатории, бережно относиться к оборудованию и приборам лаборатории.

- Самостоятельно подготовиться к выполнению экспериментальных исследований, и получить разрешение на их выполнение у курирующего преподавателя с отметкой в лабораторном журнале;

Студент несет ответственность:

- за неподготовленность к выполнению работы;
- за порчу имущества и нанесение материального ущерба лаборатории

Преподаватель несет ответственность:

- за поддержание рабочей дисциплины и порядка в лаборатории;

Преподаватель имеет право:

- отстранять от проведения работ студентов, нарушающих установленные выше правила;

Требования к оформлению отчета самостоятельных работ (лабораторных, курсовых работ и контрольных практических заданий)

Структура отчета

Объем отчета по курсовой работе не должен превышать 18 стр. Текст набирается на компьютере и печатается на принтере. Требования по содержанию разделов и оформлению отчета изложены ниже.

Отчет (требования к содержанию разделов лабораторного отчета) включает:

- Титульный лист;
- Аннотация;

Представляет собой краткое (несколько предложений) содержание работы, включающее цель работы, объект исследований, используемый метод, диапазон варьируемых параметров эксперимента, основной результат, погрешности измерений.

1. Введение;

Содержит краткое теоретическое рассмотрение изучаемого явления и краткий обзор литературы по изучаемому вопросу;

2. Описание экспериментальной установки

В данном разделе необходимо дать подробное описание создаваемой лабораторной установки с указанием полных наименований промышленно изготовленных частей установки, а также с описанием конструкции узлов, созданных самостоятельно.

3. Описание алгоритма.

Раздел содержит подробное описание используемых методов, последовательности операций, способов и приемов, которые характеризуют алгоритм.

4. Описание интерфейса.

В данном разделе необходимо дать подробное описание органов управления и индикаторов виртуального интерфейса лабораторной установки, а так же правил работы с ними.

5. Методика проведения эксперимента

Раздел содержит методические указания по проведению измерений с помощью созданной лабораторной установки, описание упражнений, порядок действий оператора.

6. Выводы и заключение;

Отмечается суть выполненной работы, делаются выводы. Высказываются рекомендации для дальнейшего развития аппаратной базы стенда и критические замечания по улучшению методики проведения эксперимента.

7. Список используемой литературы.

В разделе указывается используемая при выполнении работы основная и дополнительная литература.

Общие положения по оформлению

Электронная копия отчета и ее бумажный вариант должны быть подготовлены с помощью пакета в MS Word. При оформлении границы полей, шрифты, параметры абзацев, вставки (номера страниц, рисунки, графики и т.п.) задаются с помощью панели инструментов или меню (Файл - Параметры страницы; Формат – Шрифт, Абзац, Регистр; Вставка – Номера страниц, Рисунок).

Титульные листы отчетов лабораторной работы и курсовой представлены в Приложении А. Шрифт - Times New Roman Cyr, размер - 14 пт, регистры – указаны в примере, выравнивание абзацев - по центру.

Разделы и подразделы должны иметь заголовки. Заголовки разделов располагают с левого края строки без точки в конце. Текст раздела отделяется от текста двумя межстрочными интервалами. Переносы в заголовках не допускаются. Каждый раздел рекомендуется начинать с новой страницы.

Для нумерованных заголовков разделов отчета шрифт - Times New Roman, размер - 12 пт, Bold, выравнивание по левому краю.

Пример:

- Раздел 1,
 - Подраздел 1.1,
 - пункт 1.1.1,
 - подпункт 1.1.1.1.

Основной текст - формат А-4 (297x210), ориентация - книжная. Границы полей: верхнее, нижнее – 2 см; левое– 2.5 см, правое -1.5см, шрифт - Times New Roman, размер - 12 пт, интервал - полуторный (Word), автоматический перенос слов в границах полей, выравнивание по ширине. Формулы и символы набираются в редакторе формул Equation. Ссылки в тексте заключены в квадратные скобки.

Нумерация страниц (меню Вставка - Номера страниц). Положение - внизу страницы, выравнивание - по центру, без нумерации первой страницы - титульного листа.

Графический материал и таблицы

Иллюстрации: схемы, чертежи, графики, скриншоты, рисунки следует располагать по тексту непосредственно после первого упоминания или на следующей странице, если в указанном месте они не помещаются.

Все иллюстрации в тексте должны быть со ссылками. Допустима как сквозная нумерация рисунков, так и нумерация в пределах раздела («см. рис. 4» - при сквозной нумерации, либо «см. рис. 3.4» при нумерации в пределах

раздела). Номер следует размещать под иллюстрацией посередине после слова «Рис.».

Рисунки могут быть выполнены с помощью ЭВМ или от руки. На графиках, выражающих количественные зависимости (экспериментальные, расчетные), должна быть координатная сетка. Стрелки на осях координат в этом случае не ставятся. Цифры располагают ниже оси абсцисс и левее оси ординат. Обозначения физических величин и единиц измерения приводят через запятую с внешней стороны оси по центру или с противоположного относительно начала координат края. Масштаб координатной сетки (линейный, логарифмический масштаб или иной) выбирается из соображений удобства представления результатов. На одном графике допустимо представление семейства функциональных зависимостей. При этом кривые отображаются сглаженными линиями различного типа или цвета, допускается вводить обозначение параметра, при котором получены данные результаты. Экспериментальные точки отмечаются на графике символами.

В случае, если результаты измерений и расчетов целесообразно представлять в виде таблиц, все таблицы в тексте должны быть снабжены ссылками. Допустима как сквозная нумерация таблиц, так и нумерация в пределах раздела («см. табл. 4» - при сквозной нумерации, либо «см. табл. 3.4» при нумерации в пределах раздела). Таблицы следует располагать по тексту непосредственно после первого упоминания или на следующей странице, если в указанном месте они не помещаются. Номер таблицы следует размещать в правом верхнем углу после слова «Таблица» над заголовком таблицы. Если в работе одна таблица, её не нумеруют. Слово «Таблица» и заголовок начинаются с прописной буквы, точка в конце заголовка не ставится. Заголовки граф таблицы должны начинаться с прописных букв.

Математические символы и формулы

Уравнения и формулы в тексте располагаются в отдельной строке со свободными строками выше и ниже. Если уравнение не умещается в одну строку, оно должно быть перенесено после математических знаков (+, -, x) с их

обязательным повторением в новой строке. Пояснение значений, символов и числовых коэффициентов следует приводить непосредственно по тексту или под формулой в той же последовательности, как и в формуле. Значение каждого символа и числового коэффициента следует давать с новой строки, первую строку пояснения начинают со слова “где” без двоеточия. Формулы и уравнения в работе следует нумеровать в соответствии с выбранным способом нумерации сквозным или в пределах раздела.

Написание обозначений единиц физических величин

При написании числовых значений величин и их обозначений используются системы измерений СИ или СГС. Между последней цифрой числа и обозначением единицы физической величины следует оставлять пробел, исключение составляют знаки, поднятые над строкой. Пример: 15 В, 20 А, 12 Вт, 4,3 Дж, 35° , 1 10⁻³ Торр, 25% . Не допускается перенос обозначения единиц на следующую строку. Единицы измерений, названные в честь выдающихся ученых, обозначают с прописной буквы, например: В (Вольт), Гц (Герц), Па (Паскаль).

Обозначения единиц измерений величин, представимых в виде произведения или частного, следует отделять точкой или косой чертой например: А·м Вт/(м² К). Десятичные кратные и дольные единицы представляют в виде: кГц (килогерц), МВт (мегаватт), мВт (милливатт), мкс (микросекунда), мс (миллисекунда).

Список использованных литературных источников

Список использованных источников составляют в порядке появления ссылок в тексте или в алфавитном порядке. Ссылки следует приводить в форме указания порядкового номера по списку источников, выделенного квадратными скобками, например, [28]. При ссылке на формулу или рисунок и т.п. из первоисточника следует указывать номера страниц, например [18, с.704]. Допускается приводить ссылки на литературу в подстрочном примечании.

Примеры библиографических описаний:

Монография (учебник, справочник) центрального издательства при числе авторов не более трех:

1. Б.Б. Кадомцев Коллективные явления в плазме Москва.: Наука, 1988. 304 с.

Монография (учебник, справочник) центрального издательства при числе авторов больше трех и наличии редактора (редакторов):

2. Конструирование экранов и СВЧ-устройств / А.М. Чернушенко, Б.В. Петров, Л.Г. Малорацкий и др.; Под. ред. А.М. Чернушенко Москва: Радио и связь, 1990. 351 с.

Отдельный том многотомного издания:

3. Савельев И.В. Курс общей физики. Т.1. Механика. Молекулярная физика: Учеб. пособие для студентов вузов. 2-е изд., перераб. М.: Наука, 1982. 432 с.

Вузовские учебные пособия:

4. М.В. Кузелев, А.А. Рухадзе, П.С. Стрелков Плазменная релятивистская СВЧ-электроника: Учеб. пособие / Москва.: Издательство МГТУ им.Н.Э.Баумана, 2002. 543 с.

Периодические издания:

- 6.R. Benattar, C. Galas, P. Ney X-UV Index of refraction of dense and hot plasmas // Journal of X-ray Science and Technology. 1995. № 5. p.p. 249-260.
7. Взаимодействие электронного пучка с плазмой / И.Ф. Харченко, Я.Б. Файнберг, Р.Н. Николаев и др. // ЖЭТФ 1960. Т. 38, вып. 3. С. 685-692.

Материалы конференций:

9. А.С. Постникова, Б.В. Шишкин Система автоматизации для построения изображения объектов в терагерцовом диапазоне частот // Образовательные, научные и инженерные приложения в среде Labview и технологии National Instruments: Сборник трудов. междунар. науч.-практ. конф. / Москва. Издательство Российского университета дружбы народов, 2006. С. 259-262.

10. ECR plasmas and ECR Ion Sources / A.Girard, C.Lecot, G.Melin // 27th EPS Conference on Contr. Fusion and Plasma Phys. / Budapest, 2000. vol.24B(2000), p.p. 85-88.

Академическая этика

В курсовой работе и контрольных практических заданиях, используемые выдержки, идеи других авторов снабжаются сносками и отражаются в списке используемой литературы. Не допустимо включать в свою работу выдержки из работ без указания на это, пересказывать чужую работу близко к тексту без отсылки к ней, использовать чужие идеи без указания первоисточников, включая электронные версии, распространяемые в Интернет. Все случаи плагиата должны быть исключены. В конце работы, в соответствии с общими требованиями по оформлению отчетов самостоятельной работы, дается исчерпывающий список всех использованных источников.

ТЕМЫ ЛЕКЦИЙ И ЛАБОРАТОРНЫХ РАБОТ

7 СЕМЕСТР (сентябрь-январь)

Неделя 1: Лекция. Введение в LabVIEW.

Программная среда LabVIEW. Виртуальные приборы (ВП). Последовательность обработки данных. Организация программной среды LabVIEW. Встроенная Помощь среды LabVIEW и руководство пользователя.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004 г., 544 стр.
2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007г., 536 стр.
3. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005 г., 264 стр.

Дополнительная:

1. *LabVIEW user manuals. ni.com/manuals*
2. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007г., 400 стр.

Неделя 2: Лекция. Создание виртуальных приборов (ВП).

Компоненты ВП. Создание ВП. Типы и проводники данных. Редактирование ВП. Отладка ВП.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004, 544 с.
2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.

3. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Дополнительная:

1. *LabVIEW user manuals. ni.com/manuals*
2. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007, 400 с.

Неделя 3: Лекция. Подпрограммы ВП.

Подпрограммы ВП. Иконка ВП и соединительная панель. Использование подпрограмм ВП. Преобразование экспресс-ВП в подпрограмму ВП. Превращение выделенной секции блок-диаграммы ВП в подпрограмму ВП.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004, 544 с.
2. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Дополнительная:

1. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007, 400 с.

Неделя 4: Лекция. Циклы.

Цикл While (по условию). Цикл For (с фиксированным числом итераций). Организация доступа к значениям предыдущих итераций цикла.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004., 544 с.

2. *П.А. Бутырин Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций) М.: ДМК Пресс, 2005, 264 с.*

Дополнительная:

1. *Б. П. Поллак, Д. А. Точилин, Л. И. Пейч LabVIEW для новичков и специалистов. М.: "Горячая Линия - Телеком", 2003, 384 с.*

Неделя 5: Лекция. Структуры принятия решений.

Функция Select и принятие решений. Использование структуры Case. Использование узла Формулы.

Литература:

Обязательная:

1. *Дж. Тревис LabVIEW для всех. М.: "ДМК", 2004, 544 с.*
2. *П.А. Бутырин Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций) М.: ДМК Пресс, 2005, 264 с.*

Дополнительная:

1. *Б. П. Поллак, Д. А. Точилин, Л. И. Пейч LabVIEW для новичков и специалистов. М.: "Горячая Линия - Телеком", 2003, 384 с.*

Неделя 6 : Лекция. Массивы.

Создание массивов с помощью цикла. Использование функций работы с массивами. Полиморфизм.

Литература:

Обязательная:

1. *Дж. Тревис LabVIEW для всех. М.: "ДМК", 2004, 544 с.*
2. *В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов LabVIEW для радиоинженера. От виртуальной модели до реального прибора, М.: "ДМК Пресс", 2007, 400 с.*

Дополнительная:

1. Б. П. Поллак, Д. А. Точилин, Л. И. Пейч *LabVIEW для новичков и специалистов. М.: "Горячая Линия - Телеком", 2003, 384 с.*
2. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций) М.: ДМК Пресс, 2005, 264 с.*

Неделя 7 : Лекция. Кластеры.

Что такое кластеры? Использование функций работы с кластерами. Кластеры ошибок.

Литература:

Обязательная:

1. Б. П. Поллак, Д. А. Точилин, Л. И. Пейч *LabVIEW для новичков и специалистов. М.: "Горячая Линия - Телеком", 2003, 384 с.*
2. Дж. Тревис *LabVIEW для всех. М.: "ДМК", 2004, 544 с.*

Дополнительная:

1. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора, М.: "ДМК Пресс", 2007, 400 с.*
2. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций) М.: ДМК Пресс, 2005, 264 с.*

Неделя 8 : Лекция. Графическое отображение данных.

Использование графика Диаграмм для отображения потока данных. Использование графика Осциллограмм и двухкоординатного графика Осциллограмм для отображения данных. График интенсивности.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех. М.: "ДМК", 2004, 544 с.*

2. Р.Ш. Загидуллин *LabView в исследованиях и разработках*. М.: "Горячая линия -Телеком", 2005, 352 с.
3. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007, 400 с.

Дополнительная:

1. В.Ф. Папуловский, В.В. Мошкин, А.С. Бессонов, Батоврин В.К. *LabVIEW. Практикум по основам измерительных технологий. Учебное пособие для вузов "ДМК" · 2005, · 208 с.*
2. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Неделя 9 : Лекция. Работа со строковыми данными.

Строки. Функции работы со строками.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004 г., 544 стр.
2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.

Дополнительная:

1. *LabVIEW user manuals*. <http://www.ni.com/manuals>

Неделя 10 : Лекция. Файловый ввод/вывод.

Функции файлового ввода/вывода. Форматирование строк таблицы символов.

Использование функций файлового ввода/вывода высокого уровня.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004 г., 544 стр.

2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.

Дополнительная:

1. *LabVIEW user manuals*. <http://www.ni.com/manuals>

Неделя 11: Лекция. Настройка ВП.

Настройка внешнего вида лицевой панели. Отображение лицевых панелей подпрограмм ВП во время работы. Назначение и использование "горячих" клавиш. Редактирование ВП с некоторыми свойствами.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004 г., 544 стр.
2. Б. П. Поллак, Д. А. Точилин, Л. И. Пейч *LabVIEW для новичков и специалистов*. М.: "Горячая Линия - Телеком", 2003 г., 384 стр.
4. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.

Дополнительная:

1. *LabVIEW user manuals*. <http://www.ni.com/manuals>
2. Р.Ш. Загидуллин *LabView в исследованиях и разработках*. М.: "Горячая линия -Телеком", 2005, 352 с.

Неделя 12: Лекция. Организация системы сбора данных и управления в LabVIEW.

Введение и конфигурация. Сбор данных в LabVIEW.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004 г., 544 стр.
2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.

Дополнительная:

1. *LabVIEW user manuals.* <http://www.ni.com/manuals>
2. Р.Ш. Загидуллин *LabView в исследованиях и разработках.* М.: "Горячая линия -Телеком", 2005, 352 с.
3. А. Нестеренко, В. Федосов *Цифровая обработка сигналов в LabVIEW.* М.: "ДМК Пресс", 2007, 472 с.
4. П.М. Михеев *Основы современных систем сбора данных* <http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Неделя 13: Лекция. Ввод аналогового сигнала.

Выполнение операций аналогового ввода: однократный, конечная выборка, непрерывный ввод. Запись полученных данных в файл.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех.* М.: "ДМК", 2004 г., 544 стр.
2. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям.* М.: ДМК Пресс, 2007, 536 с.
3. А. Нестеренко, В. Федосов *Цифровая обработка сигналов в LabVIEW.* М.: "ДМК Пресс", 2007, 472 с.

Дополнительная:

1. *LabVIEW user manuals.* <http://www.ni.com/manuals>
2. Р.Ш. Загидуллин *LabView в исследованиях и разработках.* М.: "Горячая линия -Телеком", 2005, 352 с.
3. П.М. Михеев *Основы современных систем сбора данных* <http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Неделя 14: Лекция. Генерация аналогового сигнала. Выполнение операций аналогового вывода: однократный, конечная выборка, непрерывный вывод. Запись полученных данных в файл.

Литература:

Обязательная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
2. А. Нестеренко, В. Федосов *Цифровая обработка сигналов в LabVIEW*. М.: "ДМК Пресс", 2007, 472 с.
3. *LabVIEW user manuals*. <http://www.ni.com/manuals>

Дополнительная:

1. Р.Ш. Загидуллин *LabView в исследованиях и разработках*. М.: "Горячая линия -Телеком", 2005, 352 с.
2. П.М. Михеев *Основы современных систем сбора данных*
<http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Неделя 15 Лекция. Работа с цифровыми сигналами.

Информация о цифровых линиях ввода-вывода. Режимы работы счетчиков: измерение и генерация цифровых сигналов.

Литература:

Обязательная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
2. А. Нестеренко, В. Федосов *Цифровая обработка сигналов в LabVIEW*. М.: "ДМК Пресс", 2007, 472 с.
3. *LabVIEW user manuals*. <http://www.ni.com/manuals>

Дополнительная:

1. Р.Ш. Загидуллин *LabView в исследованиях и разработках*. М.: "Горячая линия -Телеком", 2005, 352 с.
2. П.М. Михеев *Основы современных систем сбора данных*
<http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Неделя 16 *Лекция. Управление измерительными приборами.*

Управление в LabVIEW измерительными приборами. Использование Instrument I/O Assistant. Архитектура программного обеспечения виртуальных интерфейсов (VISA). Драйверы измерительных приборов.

Литература:

Обязательная:

1. *А. Нестеренко, В. Федосов Цифровая обработка сигналов в LabVIEW. М.: "ДМК Пресс", 2007, 472 с.*
2. *Р.Ш. Загидуллин LabView в исследованиях и разработках. М.: "Горячая линия -Телеком", 2005, 352 с.*
3. *В.Ф. Папуловский, В.В. Мошкин, А.С. Бессонов, Батоврин В.К. LabVIEW. Практикум по основам измерительных технологий. Учебное пособие для вузов "ДМК", 2005, 208 с.*
4. *LabVIEW user manuals. <http://www.ni.com/manuals>*

Дополнительная:

1. *А.Я. Суранов LabVIEW 8.20. Справочник по функциям. М.: ДМК Пресс, 2007, 536 с.*
2. *П.М. Михеев Основы современных систем сбора данных <http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>*

Неделя 17: *Лекция. Работа с GPIB приборами. GPIB-интерфейс и его настройка.*

Литература:

Обязательная:

1. *В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов LabVIEW для радиоинженера. От виртуальной модели до реального прибора, М.: "ДМК Пресс", 2007, 400 с.*
2. *А. Нестеренко, В. Федосов Цифровая обработка сигналов в LabVIEW. М.: "ДМК Пресс", 2007, 472 с.*

Дополнительная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
2. *LabVIEW user manuals*. <http://www.ni.com/manuals>
3. П.М. Михеев *Основы современных систем сбора данных*
<http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Неделя 18 : Лекция. Работа с RS-232 приборами.

Последовательная связь. Настройка последовательного порта. Бинарный и ASCII форматы передачи данных. Передача сигнальных данных.

Литература:

Обязательная:

1. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007, 400 с.
2. А. Нестеренко, В. Федосов *Цифровая обработка сигналов в LabVIEW*. М.: "ДМК Пресс", 2007, 472 с.

Дополнительная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
2. *LabVIEW user manuals*. www.ni.com/manuals
3. П.М. Михеев *Основы современных систем сбора данных*
<http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

ЛАБОРАТОРНЫЕ РАБОТЫ ПО КУРСУ:

Введение в LabVIEW, работа со встроенными функциями и виртуальными интерфейсами:

- «Калькулятор»
- «Решение квадратных уравнений»
- «Количество сочетаний»
- «Игральная кость»
- «Умножение матриц»
- «Численные характеристики случайной величины»
- «Непрерывная диагностика температуры»
- «Работа с осциллограммами»
- «Диаграммы интенсивности»

Аннотация:

В работах изучаются основные встроенные функции среды LabVIEW. Студенты получают навыки создания ВП, подпрограмм ВП, виртуальных интерфейсов, использования логических, численных и графических индикаторов.

Литература:

Обязательная:

1. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004, 544 с.
2. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Дополнительная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.

Строки и работа с файлами:

- «Формат ASCII и преобразование строк»
- «Запись в файл текущих значений»
- «Сохранение данных в формате кластера»

Аннотация:

Изучаются функции работы со строками, процедуры высокого и низкого уровня для работы с файлами. Рассматриваются и сравниваются различные форматы хранения данных.

Литература:

Обязательная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
2. Дж. Тревис *LabVIEW для всех*. М.: "ДМК", 2004 г., 544 стр.

Дополнительная:

1. Б. П. Поллак, Д. А. Точилин, Л. И. Пейч *LabVIEW для новичков и специалистов*. М.: "Горячая Линия - Телеком", 2003, 384 с.

Работа с устройствами DAQ:

- «Считывание аналогового сигнала»
- «Генерация аналогового сигнала»
- «Работа с цифровыми каналами ввода/вывода»
- «Генерация цифровых последовательностей»

Аннотация:

Ознакомление с возможностями цифрового и аналогового ввода/вывода на базе устройств сбора данных (DAQ). Изучение основных функций для работы с DAQ-картами. Получение навыков непосредственного ввода опытных данных в компьютер.

Литература:

Обязательная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям*. М.: ДМК Пресс, 2007, 536 с.
2. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора*, М.: "ДМК Пресс", 2007, 400 с.

3. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Дополнительная:

1. *LabVIEW user manuals.* <http://www.ni.com/manuals>
2. П.М. Михеев *Основы современных систем сбора данных* <http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Работа с современными интерфейсами:

- «Связь с вольтметром по интерфейсу RS-232»
- «Работа с источником напряжения по интерфейсу GPIB»

Аннотация:

Ознакомление с современными коммуникационными возможностями лабораторного оборудования. Изучение основных функций для работы с интерфейсами RS-232 и GPIB.

Обязательная:

1. А.Я. Суранов *LabVIEW 8.20. Справочник по функциям.* М.: ДМК Пресс, 2007, 536 с.
2. В.Р. Линдваль, Г.И. Щербаков, Ю.К. Евдокимов *LabVIEW для радиоинженера. От виртуальной модели до реального прибора,* М.: "ДМК Пресс", 2007, 400 с.
3. П.А. Бутырин *Автоматизация физических исследований и эксперимента. Компьютерные измерения и виртуальные приборы на основе LabVIEW 7 (30 лекций)* М.: ДМК Пресс, 2005, 264 с.

Дополнительная:

1. *LabVIEW user manuals.* <http://www.ni.com/manuals>
2. П.М. Михеев *Основы современных систем сбора данных* <http://labview.ilc.edu.ru/LVcenter/Files/Course.pdf>

Календарный план курса

Виды и содержание учебных занятий				
Неделя	Лекции	Число часов	Лабораторные занятия	Число часов
1	Введение в LabVIEW.	2	ПТЭ ПТБ	2
2	Создание ВП.	2	Лабораторная работа №1	4
3	Подпрограммы ВП.	2		
4	Циклы.	2	Лабораторная работа №2	4
5	Структуры принятия решений.	2		
6	Массивы.	2	Лабораторная работа №3	4
7	Кластеры.	2		
8	Графическое отображение данных.	2	Лабораторная работа №4	4
9	Работа со строковыми данными.	2		
10	Файловый ввод/вывод.	2	Промежуточный контроль знаний	2
11	Настройка ВП.	2	Лабораторная работа №5	4
12	Организация системы сбора данных и управления в LabVIEW.	2		
13	Ввод аналогового сигнала.	2	Лабораторная работа №6	4
14	Генерация аналогового сигнала.	2		
15	Работа с цифровыми сигналами.	2	Лабораторная работа №7	4
16	Управление измерительными приборами.	2		
17	Работа с GPIB приборами.	2	Лабораторная работа №8	4
18	Работа с RS-232 приборами.	2		
19	Сдача контрольного практического задания (курсовой)			2
20	Итоговый контроль знаний			2

Примерный перечень тем курсовых работ и контрольных практических заданий.

Курсовые работы:

1. Система многоканального измерения температуры.
2. Управление двухканальным генератором сигнала.
3. Управление трехкоординатным приводом на основе шаговых двигателей.
4. Управление трехкоординатным сервоприводом.
5. Многоканальная система прецизионного измерения перемещения.
6. Многоканальная система прецизионного измерения усилия.
7. Генератор ШИМ.
8. Прецизионный мультиметр.

Контрольные практические задания:

1. Измерение биологических параметров человека.
2. Исследование взаимного влияния каналов в режиме многоканального сбора данных.
3. Создание программируемого источника питания.
4. Управление осциллографом по интерфейсам GPIB и RS-232.
5. Создание системы измерения АЧХ и ФЧХ систем.
6. Программирование контроллера шагового двигателя по интерфейсу I2C.
7. Спектральный анализ зашумленного сигнала.
8. Методы корреляционной обработки данных

Использованные литературные источники.

1. *Межгосударственный стандарт -ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе»*
2. *ГОСТ 8.417-81 "ГСИ. Единицы физических величин".*