

МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Д.С. Лежин

**Автоматизация испытаний и экспериментальных
исследований**

Электронное учебное пособие

САМАРА

2012

УДК 629.192 (035)
ББК 30.14
К 93

Автор: **Лежин Дмитрий Сергеевич**

Рецензент:

К. Е. Климентьев;
доцент кафедры информационных систем и технологий

Редакторская обработка О. В. Нагурная
Компьютерная верстка О. В. Нагурная
Доверстка Д. С. Лежин

Лежин, Д. С. Автоматизация испытаний и экспериментальных исследований
[Электронный ресурс] : электрон. учеб. пособие / Д. С. Лежин; Минобрнауки России, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т). - Электрон. текстовые и граф. дан. (1,021 Мбайт). - Самара, 2012. –
1 эл. опт. диск (CD-ROM)

Приведены общие алгоритмы автоматизации эксперимента.

Рассматриваются основные понятия, связанные с формулированием цели, постановкой задачи, подбором датчиков, разработкой и отладкой программ.

Учебное пособие предназначено для подготовки магистров 2 факультета 2 курса 3 семестра по специальности 160700.68 "**Двигатели летательных аппаратов**"

Подготовлено на кафедре конструкции и проектирования двигателей летательных аппаратов СГАУ.

© Самарский государственный
аэрокосмический университет, 2012

Содержание

Предисловие	4
1. Решение задач	4
2. Введение в LabVIEW	8
2.1. Виртуальные приборы	8
2.2. Составные части ВП	8
2.3. Начало разработки ВП	11
2.4. Project Explorer	13
2.5. Лицевая панель	15
2.6. Блок-диаграммы	19
2.7. Поток данных	26
3. Создание ВП	26
3.1. Проектирование лицевой панели	26
3.2. Документирование кода	28
3.3. Цикл While	29
3.4 Цикл For	29
3.5. Синхронизация	29
3.6. Передача данных между итерациями	30
3.7. Графическое представление данных	30
3.8. Структура выбора Case	31
3.9 Узел формул	31
4. Объединение данных	32
4.1. Массивы	32
4.2. Кластеры	33
4.3. Определение типов	34
5. Отладка ВП	34
5.1. Исправление неработоспособных ВП	34
5.2. Методы отладки	34
5.3. Неопределенные или непредвиденные данные	36
5.4. Обработка ошибок	37
6. Основы измерений	38
6.1. Компьютерные измерительные системы	38
6.2. Принципы измерений	38
6.3. Повышение качества измерений	39
7. Сбор данных	40
7.1. Аппаратура	40
7.2. Программная архитектура	40
7.3. Аналоговый ввод	40
7.4. Аналоговый вывод	41
7.5. Счетчики	42
7.6. Цифровой ввод-вывод	42
8. Анализ и сохранение результатов	42
8.1. Анализ и обработка числовых данных	42
8.2. Чтение и запись данных	43

Предисловие

LabVIEW Laboratory Virtual Instrument Engineering Workbench (Среда разработки лабораторных виртуальных приборов) представляет собой среду графического программирования, которая широко используется в промышленности, образовании и научно-исследовательских лабораториях в качестве стандартного инструмента для сбора данных и управления приборами. LabVIEW - мощная и гибкая программная среда, применяемая для проведения измерений и анализа полученных данных.

LabVIEW находит применение в самых разнообразных сферах человеческой деятельности. В соответствии со своим названием он первоначально использовался в исследовательских лабораториях, да и в настоящее время является наиболее популярным программным пакетом как в лабораториях фундаментальной науки так и в отраслевых промышленных лабораториях. Все более широкое применение LabVIEW находит и в образовании в университетских лабораторных практикумах - особенно по предметам электротехники, механики и физики. Лучшим способом объяснить причины столь широкого распространения пакета LabVIEW будет обобщение способов его использования. Во всех видах человеческой деятельности существуют области, где не обойтись без определенных видов измерений - очень часто это температурные измерения, например в печах, холодильниках, парниках, технологических помещениях. Кроме температуры, часто измеряют давление, силу, пространственное смещение, механическое напряжение, *pH* и т.д. Сейчас персональные компьютеры проникли практически во все сферы жизнедеятельности. LabVIEW ускоряет внедрение компьютера в измерительные системы - и не только потому, что облегчает проведение измерений, он также дает возможность проанализировать измеренные величины, отобразить их на графиках и в отчетах и при желании опубликовать.

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

1. Решение задач

Необходимость автоматизации лабораторной установки или стенда может возникать на различных жизненных циклах их существования. Наиболее правильный вариант предусматривает решение задачи автоматизации на самом раннем этапе – когда стенд или установка еще не существуют в металле, а имеются лишь какие-то задумки, воплощенные (или частично воплощенные) на бумаге. Но в жизни более часто встречается вариант, когда стенд или установка уже существуют и работают, при этом они исправно выполняют свои функции, причем вполне добротное «железо» установки укомплектовано давно морально (а зачастую и физически) устаревшими средствами измерения и контроля, используются примитивные средства управления установкой. Получение результата на таких установках сопряжено с большим объемом работ и огромной трудоемкостью. Задача автоматизации на таких стендах часто заключается не только в замене устаревшего оборудования управления и контроля на современное, но и в замене алгоритмов управления и сбора информации, последующей обработке полученной информации и формировании на основании полученной информации управляющего решения.

Все эти современные технологии имеют универсальный механизм реализации, заключающийся в том, что основным «мозговым» центром, куда стекается вся собираемая

со стенда информация, происходит ее обработка, формируются управляющие сигналы, является компьютер, включаемый в состав стенда.

Соответственно, задача автоматизации стенда распадается на ряд типовых процедур:

1. получение компьютером информации со стенда об интересующих параметрах
2. выполнение определенных действий с полученной информацией – расчет каких-либо характеристик
3. формирование управляющего воздействия в виде некоторого сигнала и его передача на стенд
4. преобразование сигнала управляющего воздействия в некоторое конкретное физическое действие (включение или отключение какого-то элемента, переключение на другой режим и т.д.).

Типовыми указанные процедуры названы по той причине, что, несмотря на все многообразие физических параметров, алгоритмы работы с ними ничем не отличаются.

1. Так, по пункту 1, получение компьютером информации со стенда (имеется в виду измерение различных физических и электрических параметров) заключается в создании цепочки:

первичный преобразователь – устройство согласования сигнала – АЦП – компьютер

Для электрических параметров эта цепочка выглядит еще проще:

электрический параметр – АЦП – компьютер

В некоторых случаях (особенно это относится к случаям, когда необходимо без дополнительных затрат на разработку методик и аттестацию АЦП обеспечить метрологическую законность измерений) цепочка измерения какого-то физического параметра может иметь вид:

первичный преобразователь – измерительный прибор – КОП – компьютер

Так, например, сигнал с метрологически аттестованной системы виброизмерений может быть заведен на поверенный осциллограф. Учитывая то, что большинство современных приборов (осциллографы, вольтметры, частотомеры и т.д.) имеют разъем связи с компьютером, по которому можно не только передавать измеряемую информацию, но и производить с компьютера управление прибором – переключение режимов работы прибора, шкал измеряемых диапазонов, и др., может быть обеспечена связь указанного прибора с компьютером по т.н. каналу общего пользования (КОП).

2. Одним из важных поводов, побуждающих к проведению автоматизации какого-либо стенда, является желание мгновенного получения результатов. Это могут быть какие-либо характеристики, рассчитываемые по измеряемым параметрам

Например, перепад давлений может быть измерен непосредственно датчиком дифференциального давления, а, при использовании двух датчиков, может быть получен расчетным способом, путем вычитания показаний указанных датчиков. Аналогично, в процессе испытаний могут быть получены достаточно сложно рассчитываемые характеристики. В некоторых случаях не требуется получение характеристик в численном виде – результатом испытаний должен стать вывод о пригодности испытуемого объекта. Другими словами, требуется определить, вписываются ли характеристики испытуемого объекта в требуемый диапазон. Эта задача также может быть решена на этапе расчета характеристик.

3. Формирование управляющего воздействия может быть не привязано к измеряемым параметрам и характеристикам. Одним из вариантов управления, повышающих удобство работы с установкой, является внедрение ручного управления с компьютера. В этом случае на экране компьютера эмулируется панель управления стенда с необходимыми органами управления. Управление производится переключением органов управления на экране с помощью компьютерной мыши. Более сложной задачей является

внедрение автоматизированного управления. В этом случае требуемые переключения органов управления организованы в виде циклограммы по временным переходам. Наконец, управляющее воздействие может формироваться в зависимости от измеряемых параметров. Например, нагреватель должен отключиться при достижении определенной температуры. С точки зрения реализации, данная цепочка представляет собой следующий вид:

управляющий сигнал с компьютера – ЦАП – исполнительный орган стенда

в случае необходимости получения аналогового вывода, либо:

управляющий сигнал – цифровой выход (или порт компьютера) – исполнительный орган стенда

в случае цифрового вывода.

4. Полученный на выходе ЦАП или на цифровом выходе управляющий сигнал может быть преобразован в управляющее воздействие непосредственно на стенде разнообразными способами. Если это аналоговый сигнал и его необходимо использовать для плавного изменения какого-либо органа управления, наиболее простой способ – на этапе проектирования подобрать требуемый регулятор. Современные регуляторы (регуляторы давления, расхода, температуры) имеют опцию управления по аналоговому сигналу. При таком решении полученный аналоговый сигнал заводится непосредственно на подобный регулятор. При работе с цифровыми сигналами (кнопки включения – выключения, клапаны открытия – закрытия) сигнал с цифрового выхода чаще всего подается на соответствующий исполнительный орган через реле.

В соответствии с изложенными типовыми процедурами можно сформулировать обобщенный алгоритм автоматизации установки или стенда.

1. ЦЕЛЬ

Любая работа по автоматизации эксперимента начинается с формулирования основной цели автоматизации. Цели могут быть самыми разнообразными. При создании стендов для испытаний серийной продукции стенд должен отрабатывать определенную циклограмму, в ходе которой измеряются параметры испытуемого изделия, по этим параметрам проводится расчет характеристик и делается заключение о пригодности испытуемого изделия. Часто потребность в автоматизации обуславливается необходимостью одновременного измерения большого числа параметров. Одной из целей автоматизации является желание совмещения измерения и обработки параметров. Наконец, автоматизация дает возможность замены большого числа приборов (часто морально устаревших), как осциллографы, вольтметры, тензостанции и т.д. одним или несколькими компактными модулями АЦП-ЦАП и компьютером. Современные комплексные решения по автоматизации (как программно – аппаратный комплекс фирмы NI) достигли такого уровня, что автоматизированное решение на его базе (создание виртуального прибора для какого – то набора датчиков) получается проще, чем реализация с использованием набора специальных и универсальных приборов для регистрации сигналов с первичных преобразователей.

Цель определяет способ реализации решения по автоматизации. Так, если автоматизируется стенд для испытания серийной продукции, надо иметь в виду, что работу на этом стенде будет проводить не высококвалифицированный специалист и особое внимание следует уделить разнообразным защитам, исключающим возможность ошибки со стороны оператора в процессе работы, защищенности самого программного продукта и операционной системы. При автоматизации какого – либо однократного эксперимента на создание защит не стоит тратить время, не столь требовательно можно отнестись и к лицевой панели, и к созданию алгоритма обработки измеряемых параметров. Основное внимание следует обратить на корректность измерения параметров. Все это позволит существенно снизить трудоемкость автоматизации.

Если целевая функция заключается не в измерении параметров или расчете характеристик, а в получении определенного результата в ходе испытания, например,

выполнение балансировки ротора, определение собственных частот и т.д., сначала требуется определиться с методикой, характеристиками и параметрами.

2. АЛГОРИТМ

На данном этапе определяется перечень необходимых для измерения параметров, перечень необходимых для расчета характеристик и расписывается полный алгоритм по получению этих характеристик. Часто получить необходимые параметры можно несколькими разными алгоритмами. В этом случае выбор алгоритма проводится совместно с решениями по выбору датчиков.

3. АНАЛИЗ МАТЕРИАЛЬНОЙ ЧАСТИ

Работы по автоматизации развиваются двумя основными ветвями. Первая ветвь предусматривает работы по автоматизации перед началом проектирования и создания стенда или установки, т.е. работа «с нуля», когда установки еще не существует. Часто можно столкнуться с другой ветвью автоматизации. Имеется стенд или установка, «железо» у которых в хорошем состоянии, а измерительная и управляющие части морально и физически устарели и нуждаются в замене. Это и является поводом для автоматизации. Специфика данной ветви заключается в том, что существующая установка дает меньше свобод для размещения датчиков, появляются некоторые ограничения по управляющим органам.

В любом случае, при выполнении работ по автоматизации необходимо провести анализ имеющейся материальной части (стенда или установки, их комплектующих, элементов и т.д.). В случае, если задача автоматизации поставлена на этапе проектирования, анализируется техническое задание, эскизный проект, имеющаяся конструкторская документация.

4. ПОДБОР ДАТЧИКОВ....

Ассортимент средств автоматизации, датчиков, АЦП-ЦАП сейчас чрезвычайно широк. Для сокращения времени разработки следует отдавать предпочтение устройствам сбора информации NI. Вопрос выбора компонентов схемы автоматизации, в частности датчиков, сугубо индивидуален для каждой задачи. При выборе датчиков необходимо обратить внимание на датчики со встроенными предусилителями. Такие усилители имеют многие датчики давления, температуры, вибродатчики. Эти датчики имеют встроенный предусилитель сигнала и выдают на выход нормированный сигнал напряжения непосредственно на устройство сбора. Их преимущество заключается в большей помехозащищенности участка от датчика до устройства сбора, особенно при большой его длине.

В настоящем пособии уделено некоторое внимание особенностям и способам регистрации некоторых наиболее распространенных физических параметров.

5. ЭКРАН

После того, как все предыдущие шаги успешно выполнены, производится создание передней панели. На нее выводится вся информация об измеряемых параметрах в наиболее удобочитаемом виде (цифровое табло, график, рисованная форма, и т.д.), а также необходимые органы управления установкой.

5. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

В данном пособии особое внимание уделено программному обеспечению фирмы NATIONAL INSTRUMENTS.

LabVIEW — это язык программирования, который можно использовать для решения самых разных задач. Чтобы создавать решения на основе LabVIEW, важно иметь общие навыки решения задач. Для решения задач при помощи компьютерных программ программисты пользуются методиками разработки программного обеспечения (ПО). Следование определенной методике помогает программисту создавать код, который имеет больший потенциал для успешного решения задачи по сравнению с кодом, написанным без плана. Методика также помогает сделать код более читаемым, масштабируемым и

модифицируемым. В данном разделе в общих чертах описывается стратегия разработки ПО, которую можно применять для решения своих измерительных и аналитических задач.

2. Введение в LabVIEW

Большую роль в успехе автоматизации эксперимента имеет знание языка программирования LabVIEW. В данном разделе приводятся основные приемы работы в среде LabVIEW, включая использование меню, инструментальных панелей, палитр, инструментов, справки и стандартных диалогов. Даны основы запуска виртуальных приборов (ВП) и общее представление о лицевой панели и блок-диаграмме.

1. Виртуальные приборы (ВП)
2. Составные части ВП
3. Начало разработки ВП
4. Project Explorer
5. Лицевая панель
6. Блок-диаграмма
7. Поиск элементов управления, ВП и функций
8. Выбор инструмента
9. Поток данных
10. Создание простого ВП
11. Справочные средства LabVIEW

Окна лицевой панели и блок-диаграммы каждого ВП содержат в правом верхнем углу иконку (показана слева).

Иконка — это графическое представление ВП; она может содержать текст и изображения. Если ВП используется как подВП, иконка идентифицирует его на блок-диаграмме. По умолчанию иконка содержит число, которое показывает, сколько новых ВП вы открыли после запуска Lab VIEW.

Чтобы использовать ВП в качестве подВП, нужно построить соединительную панель (показана слева). Соединительная панель представляет собой набор терминалов, соответствующих элементам управления и индикаторам данного ВП. Это аналог списка параметров функции в текстовых языках программирования. Откройте соединительную панель, щелкнув правой кнопкой на иконке в верхнем правом углу окна лицевой панели. К соединительной панели нет доступа через иконку окна блок-диаграммы.

2.1 Виртуальные приборы

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

2.2. Составные части ВП

ВП состоит из трех основных частей: лицевой панели, блок-диаграммы и иконки с соединительной панелью.

Лицевая панель (Front panel) – это пользовательский интерфейс ВП. Лицевая панель конструируется из элементов управления и индикаторов, которые представляют собой интерактивные входные и выходные терминалы ВП, соответственно.

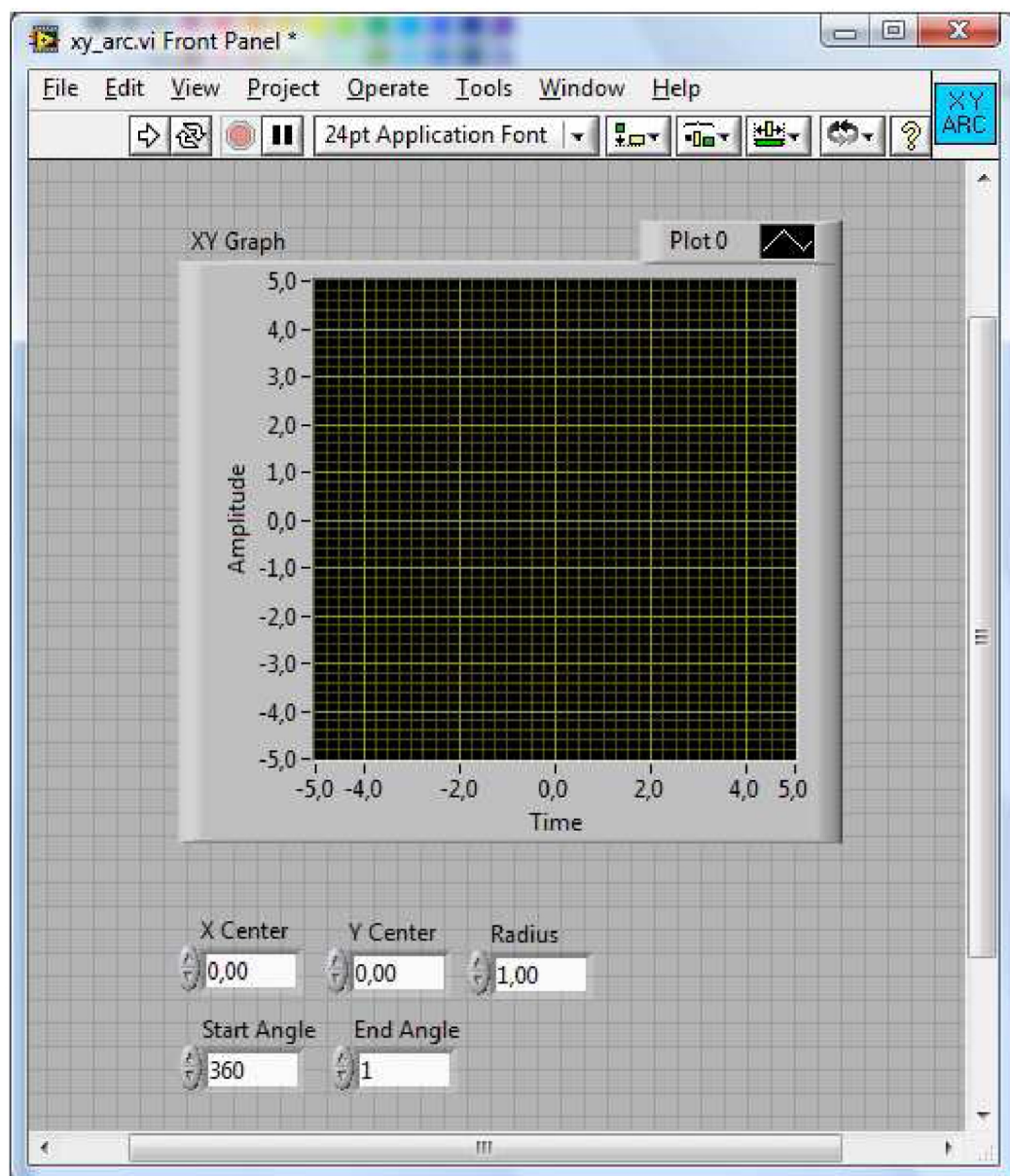


Рис Лицевая панель ВП

Блок диаграмма (Block diagram) – Содержит графический исходный код, который определяет функционирование ВП.

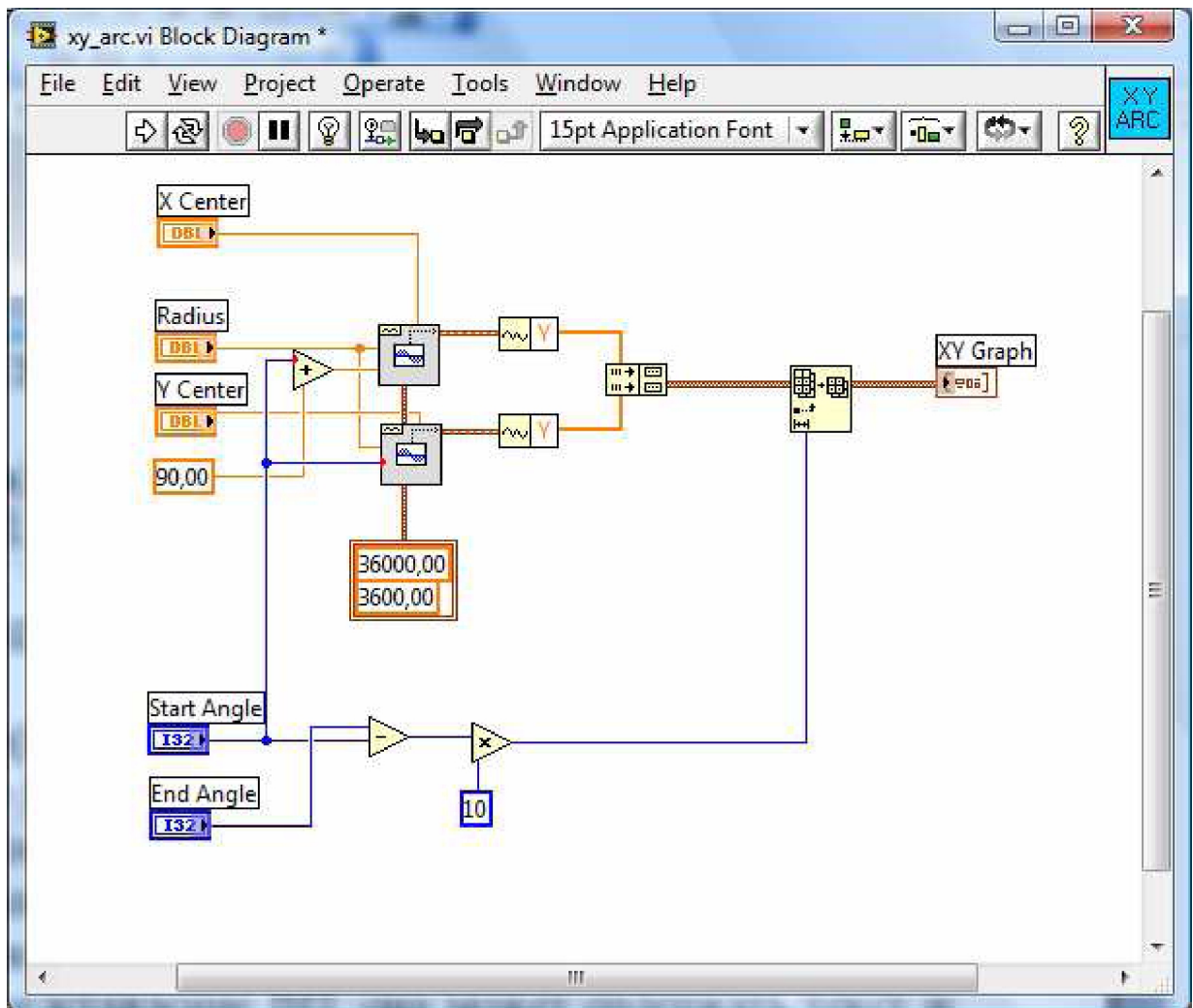
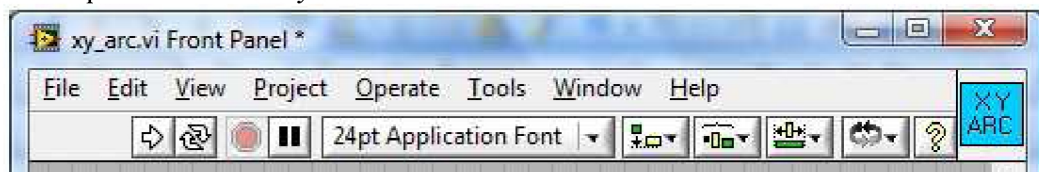


Рис. Блок-диаграмма ВП

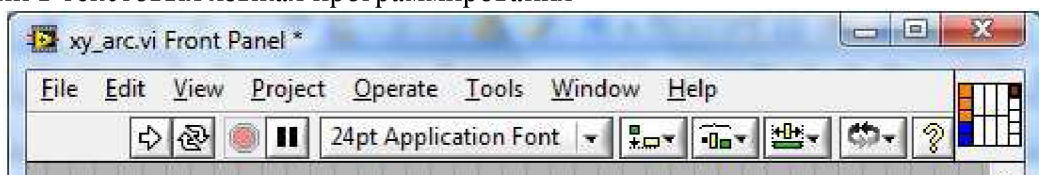
Иконка и соединительная панель (Icon and connector pane).

ВП можно использовать в качестве ПодВП (subVI). ПодВП- это ВП, вызываемый из другого ВП, подобно функции в текстовом языке программирования. Чтобы ВП мог использоваться в качестве подВП, он должен иметь иконку и соединительную панель.

Окна лицевой панели и блок-диаграммы каждого ВП содержат в правом верхнем углу иконку. Иконка – это графическое представление ВП; она может содержать текст и изображения. Если ВП используется как подВП, иконка идентифицирует его на блок-диаграмме. По умолчанию иконка содержит число, которое показывает, сколько новых ВП были открыты после запуска LabVIEW.



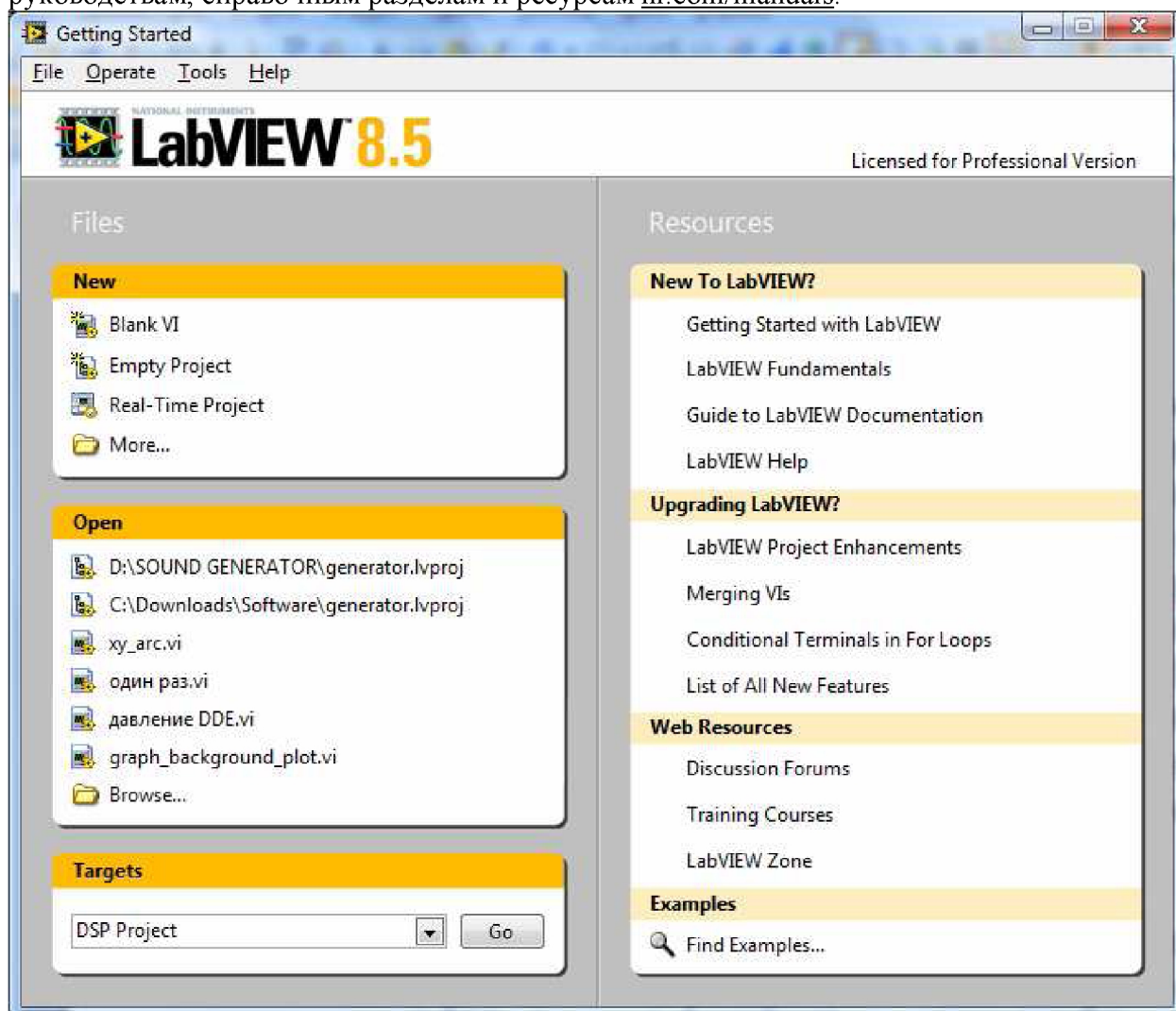
Соединительная панель представляет собой набор терминалов, соответствующим элементам управления и индикаторам данного ВП. Это аналог списка параметров функции в текстовых языках программирования



Соединительная панель открывается щелчком правой кнопки мыши на иконке в верхнем правом углу окна лицевой панели. К соединительной панели нет доступа через иконку окна блок-диаграммы.

2.3. Начало разработки ВП

При запуске LabVIEW появляется окно **Getting Started**. Его можно использовать для создания новых ВП и проектов, выбора одного из недавно открывавшихся файлов LabVIEW, поиска примеров и справочной информации. Это окно также дает доступ к информационным ресурсам, которые помогают изучать LabVIEW: специализированным руководствам, справочным разделам и ресурсам ni.com/manuals.



Окно **Getting Started** исчезает, когда открывается существующий или создается новый файл. Это окно можно снова вызвать, выбрав пункт меню **View»Getting Started Window**. Можно настроить LabVIEW так, чтобы при запуске вместо этого окна открывался новый, пустой ВП. Для этого выбирается пункт меню **Tools»Options**, затем **Environment** в списке **Category** и ставится отметка **Skip Getting Started window on launch**.

Создание/открытие ВП или проекта.

Работу в LabVIEW можно начать с пустого ВП (проекта), открыть существующий ВП (проект) и модифицировать его, или открыть шаблон, на основе которого будет создан новый ВП (проект).

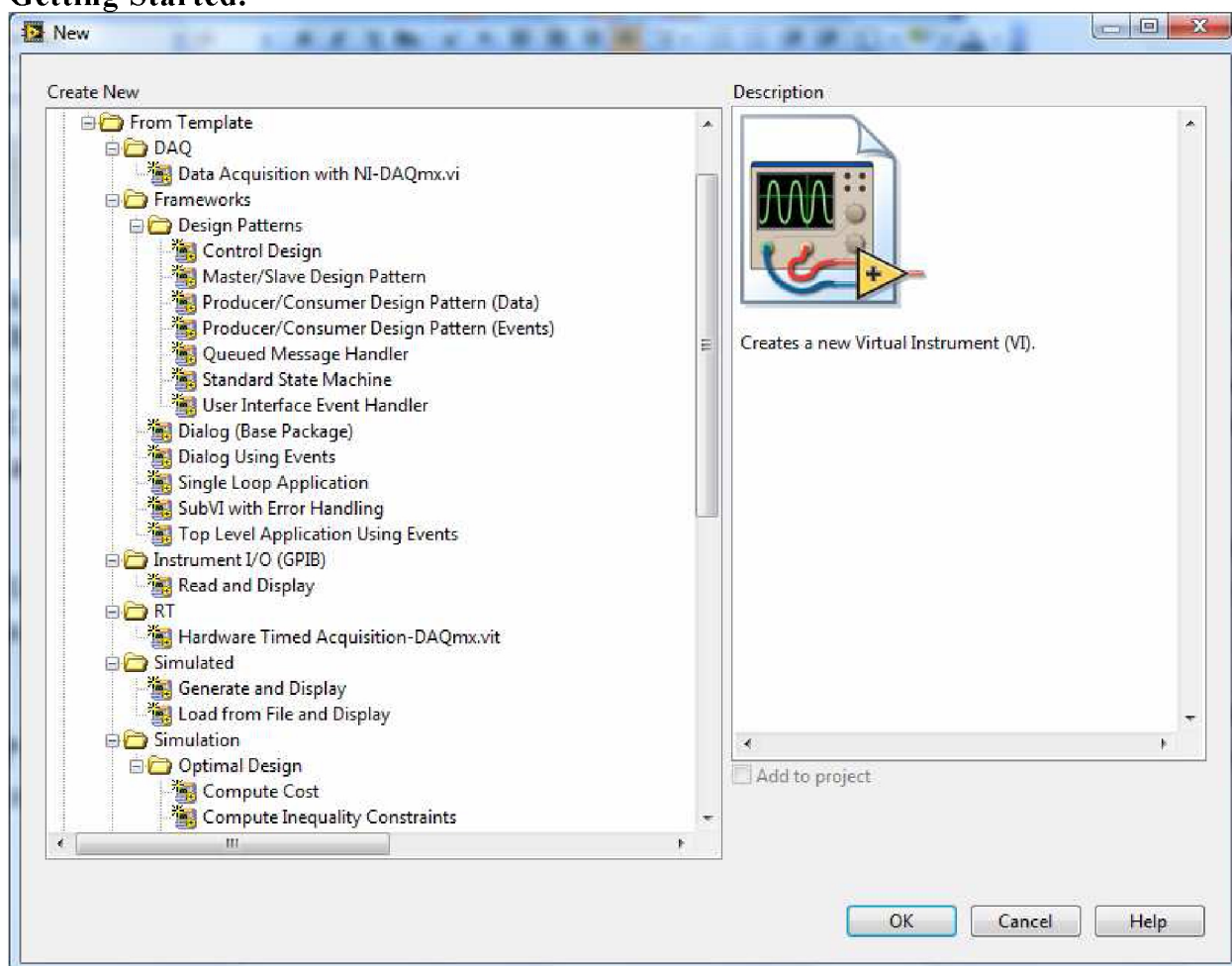
Создание «с чистого листа»

Чтобы открыть новый проект из окна **Getting Started**, выбирается вариант **Empty Project**. Откроется новый безымянный проект, к которому можно добавить файлы, а затем сохранить.

Чтобы открыть новый, пустой ВП, который не связан ни с каким проектом, выбирается вариант **Blank VI** в окне **Getting Started**.

Создание ВП или проекта по шаблону

Пункт меню **File»New** открывает диалог **New**, где перечислены встроенные шаблоны ВП. Для вызова этого диалога можно также использовать ссылку **New** в окне **Getting Started**.



Открытие существующего ВП

Выбирается вариант **Browse** в окне **Getting Started**, чтобы перейти в каталог с существующим ВП и открыть его.

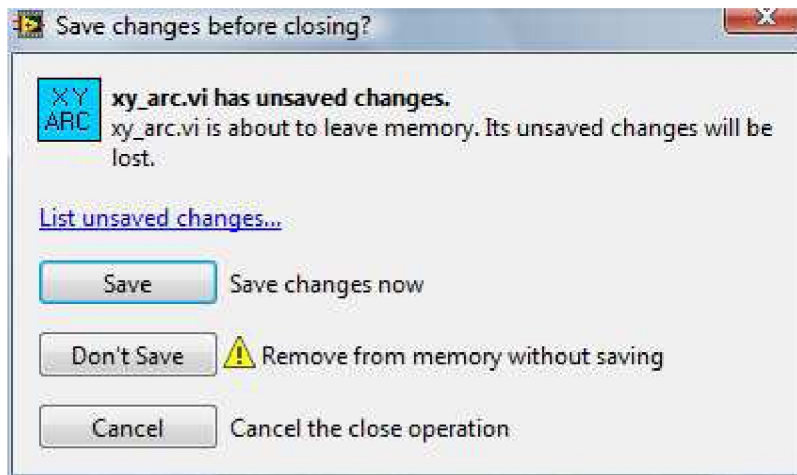
В секции **Loading** последовательно перечисляются подВП, загружаемые в память, а также показывается, сколько подВП уже загружено. Загрузку можно в любой момент отменить, щелкнув на кнопке **Stop**.

Если LabVIEW не может сразу найти подВП, он начинает поиск по всем каталогам, указанным в списке путей поиска. Вы можете отредактировать этот список, выбрав пункт меню **Tools»Options**, а затем категорию **Paths**.

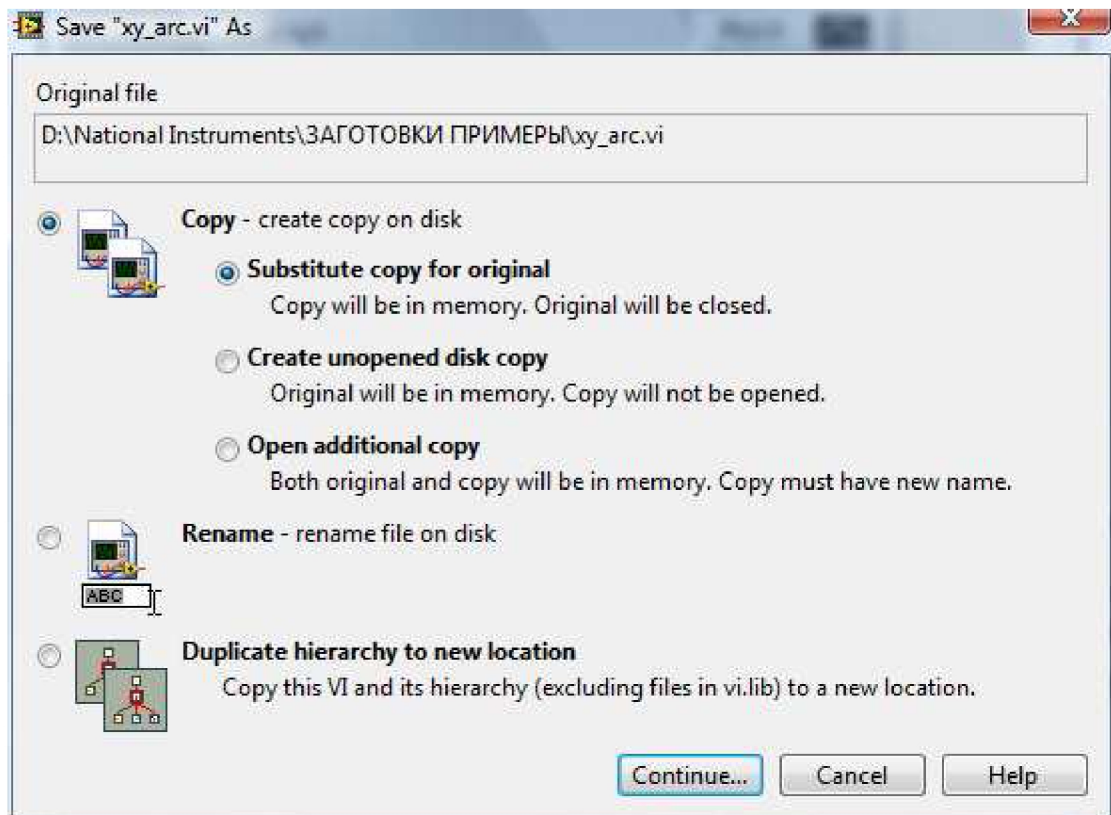
Можно заставить LabVIEW пропустить подВП, щелкнув на кнопке **Ignore SubVI**. Вы также можете найти отсутствующий ВП самостоятельно, нажав кнопку **Browse**.

Сохранение ВП

Чтобы сохранить новый ВП, выберите пункт меню **File»Save**.



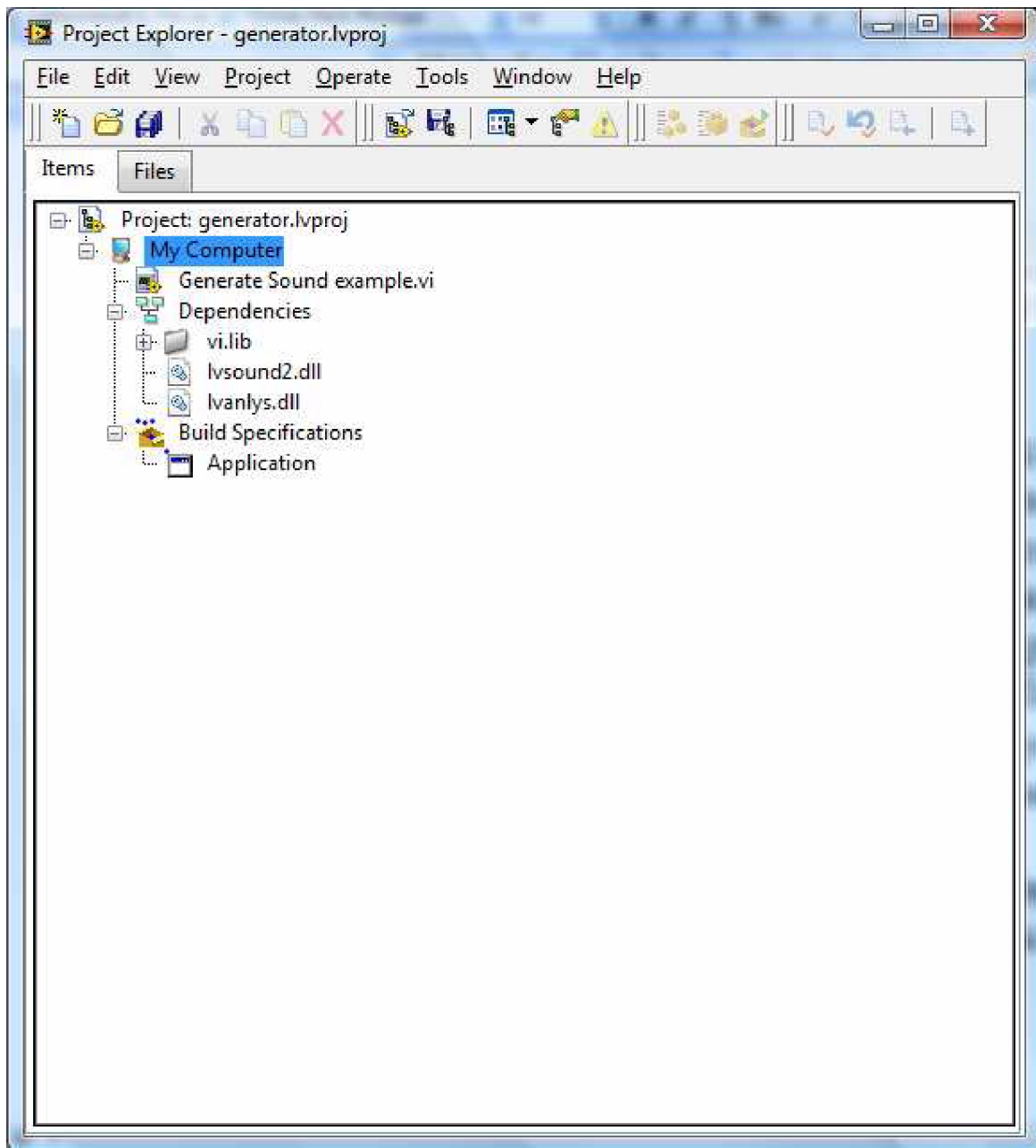
Если вы уже сохраняли ВП, выберите **File»Save As**, чтобы вызвать диалог **Save As**.



С его помощью вы сможете создать копию ВП или удалить исходный ВП, заменив его новым.

2.4. Project Explorer

Проект (Project) используется для группирования файлов LabVIEW и файлов других приложений, создания спецификаций сборки, а также развертывания или загрузки файлов на целевые системы. Когда вы сохраняете проект, LabVIEW создает файл проекта (.lvproj), который содержит ссылки на все файлы, составляющие данный проект, информацию о конфигурации, сборке, развертывании и т.д. Проекты необходимы для построения приложений и разделяемых библиотек.



Окно **Project Explorer** используется для создания и редактирования проектов LabVIEW. Оно вызывается через пункт меню **File>>New Project**, по умолчанию содержит следующие элементы:

Project: <имя_проекта> - корневой элемент, служит контейнером для всех остальных элементов данного окна. Метка корневого элемента включает имя проекта.

- **My Computer** – представляет локальный компьютер как целевую систему проекта.

- **Dependencies** (зависимости) – содержит элементы, необходимые для ВП на данной целевой системе.

- **Build Specifications** (спецификации сборки) – содержит сборочную конфигурацию для распространения исходного кода, а также для других типов сборки, доступных в библиотеках и модулях LabVIEW.

Целевой системой (Target) называется любое устройство, на котором можно запустить ВП.

Для выполнения операций над проектом используются кнопки инструментальных панелей Standard, Project, Build Specifications и Source Control. Вы

можете скрывать или отображать инструментальные панели, используя меню **View >> Toolbars**.

2.5 Лицевая панель

Когда Вы открываете новый или существующий ВП, появляется его лицевая панель. Она представляет собой пользовательский интерфейс ВП.

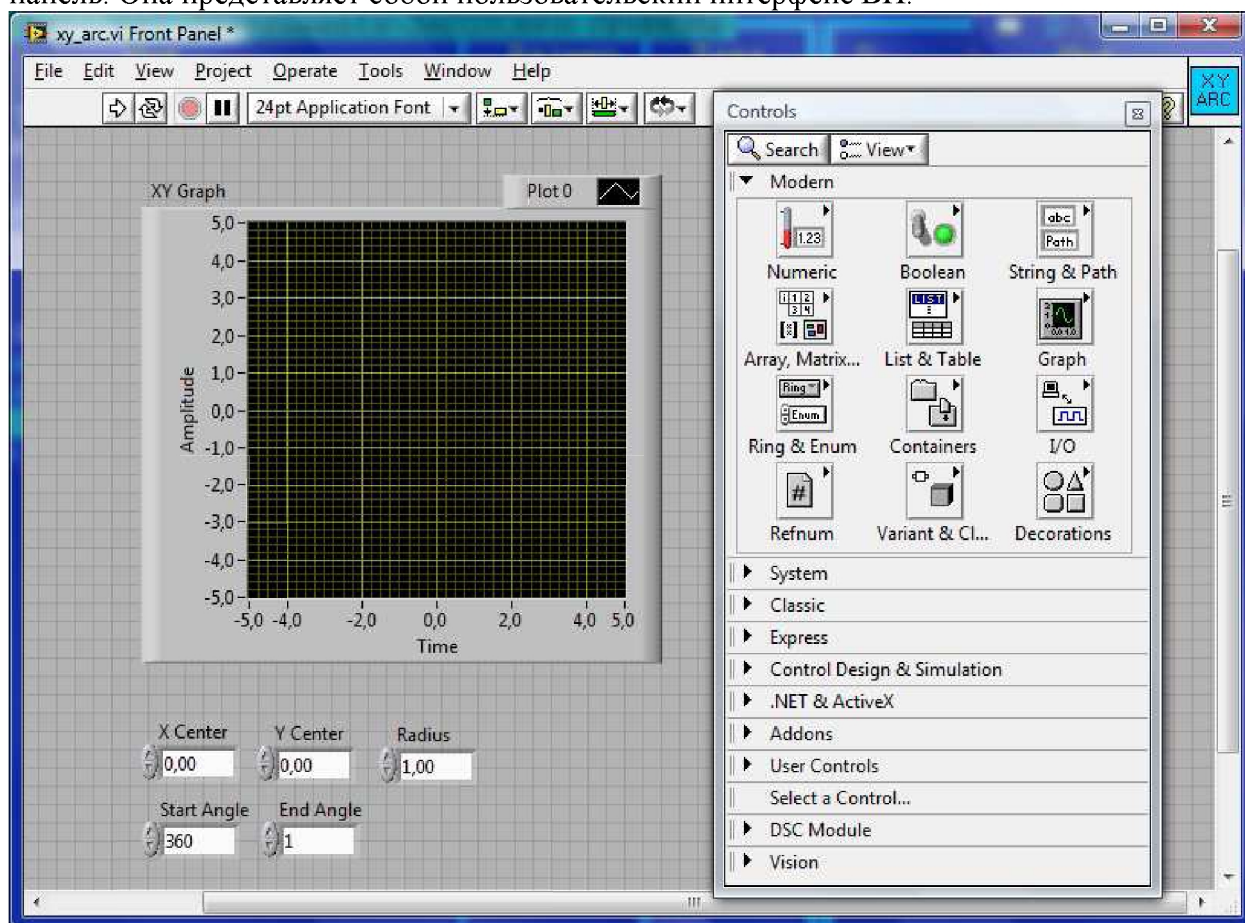


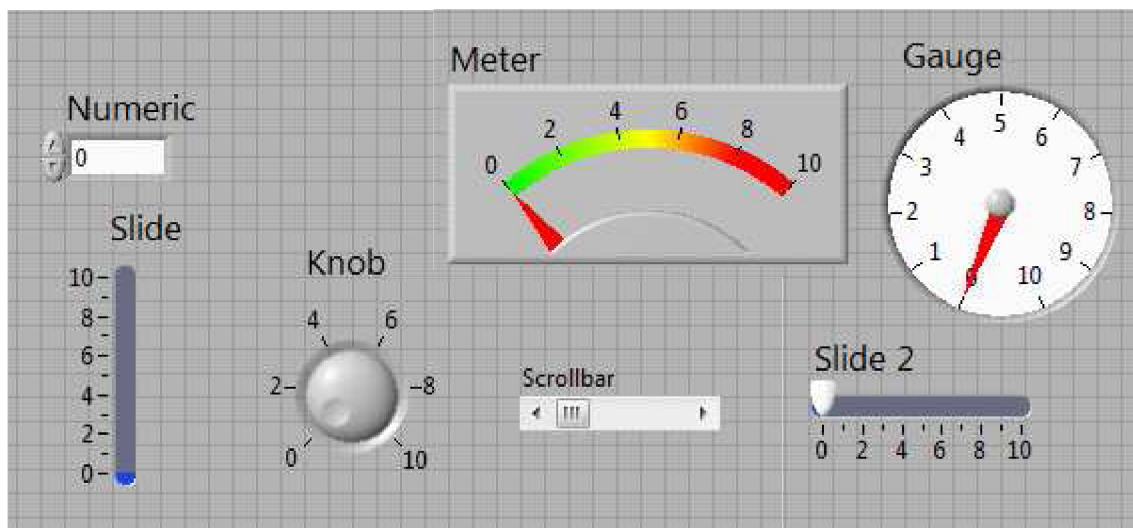
Рис лицевой панели

Элементы управления и индикаторы

Лицевая панель конструируется из элементов управления и индикаторов, которые являются интерактивными входными и выходными терминалами ВП, соответственно. Элементы управления (controls) – это кнопки, клавиши, переключатели и другие входные устройства. Индикаторы (indicators) – это графопостроители, лампочки и другие элементы отображения. Элементы управления имитируют входные устройства прибора и поставляют данные на блок-диаграмму ВП. Индикаторы имитируют выходные устройства прибора и отображают данные, которые получает или генерирует блок-диаграмма.

Числовые элементы управления и индикаторы

Числовой тип данных представляет числа различных типов, например, целые или вещественные.



Такие объекты, как стрелочный индикатор(meter) и регулятор(dials) также представляют числовые данные.

Логические элементы управления и индикаторы

Логический, или булев, тип данных представляет данные, имеющие только два значения: TRUE и FALSE, ON и OFF. Для ввода и отображения логических (TRUE и FALSE) значений используются логические элементы управления и индикаторы.

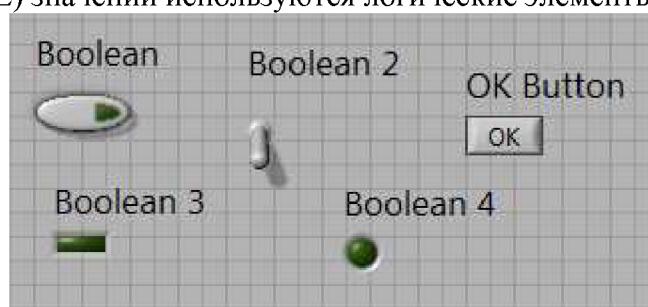


Рис логического элемента.

Строковые элементы управления и индикаторы(String Control and Indicators)

Строковый тип данных представляет собой последовательность ASCII-символов. Строковые элементы управления используются для получения от пользователя текста, например, имени или пароля. Строковые индикаторы, наоборот, выводят текст пользователю. Наиболее часто используемые строковые объекты – это таблицы и окна ввода текста.

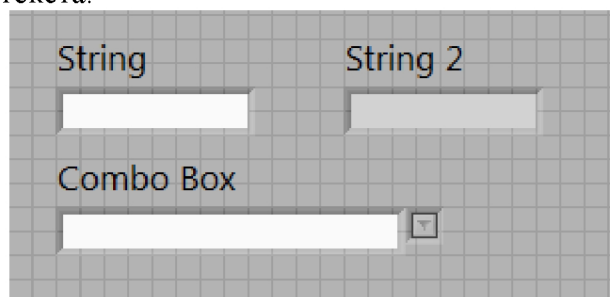
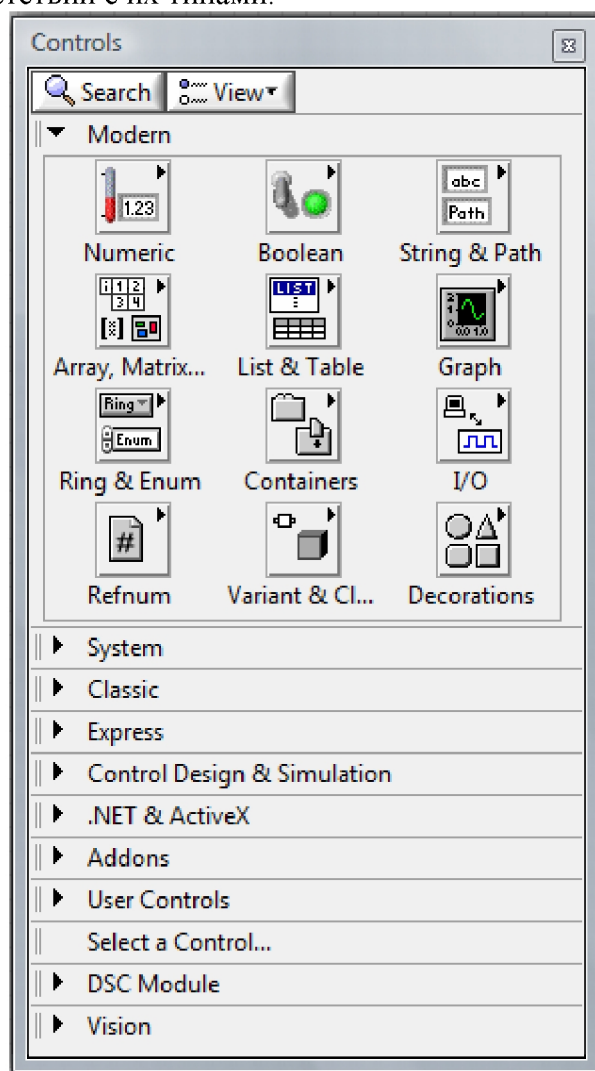


Рис строковых элементов управления

Палитра Controls (Элементы управления)

Палитра **Controls** содержит элементы управления (controls) и индикаторы (indicators), используемые при создании лицевой панели. Чтобы открыть эту палитру из лицевой панели, выберите пункт меню **View>>Controls Palette**. Палитра разбита на

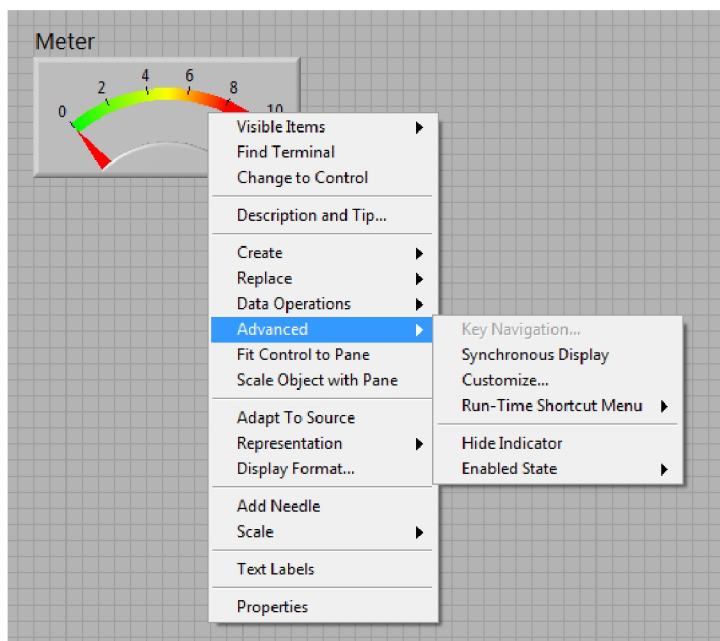
категории, вы можете отобразить некоторые или все категории, в зависимости от своих потребностей. Элементы управления и индикаторы размещены в подпалитрах в соответствии с их типами.



Палитру **Controls** можно переместить в любое место на экране. LabVIEW запоминает положение и размер палитры **Controls**, поэтому после следующего запуска LabVIEW палитра появится в том же месте и сохранит свои прежние размеры. Чтобы отобразить или скрыть категории (подпалитры), нажмите кнопку **View**, а затем отметьте нужное в разделе **Always Visible Categories** (Постоянно видимые категории).

Контекстные меню (Shortcut menu)

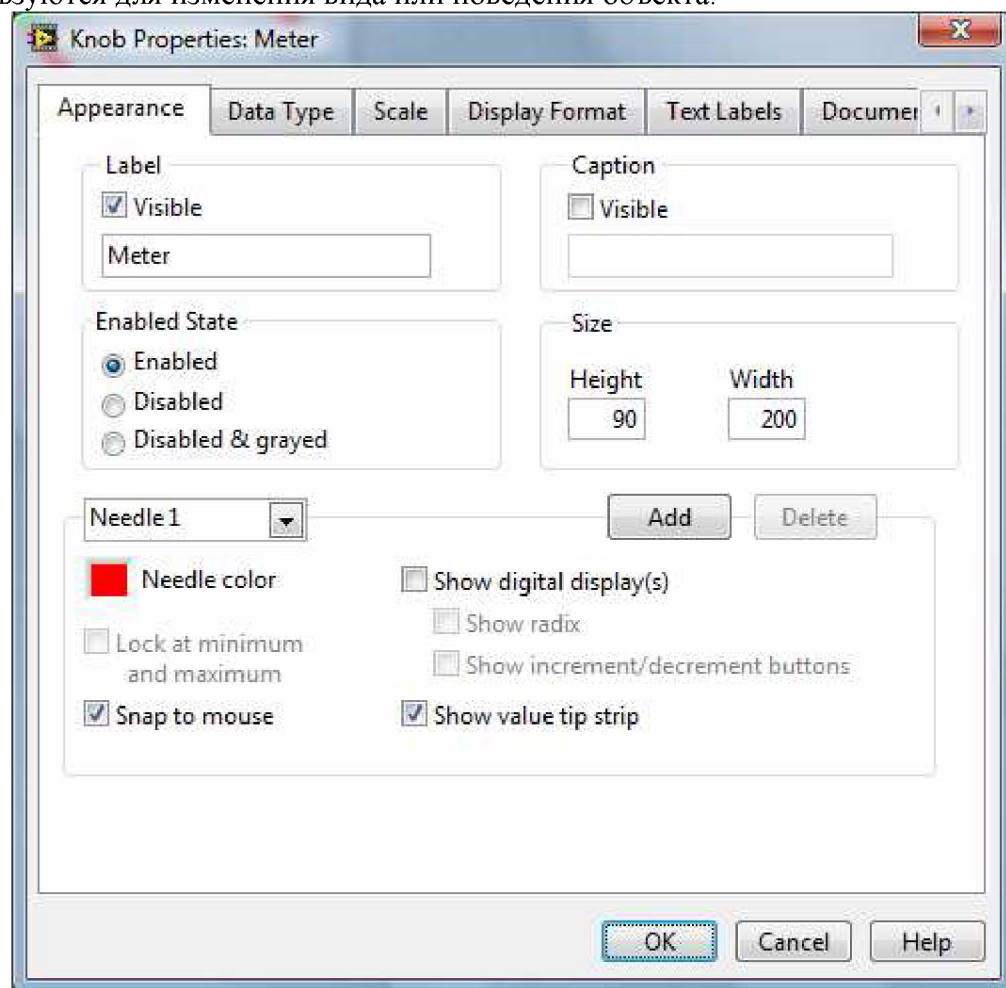
Каждый объект LabVIEW имеет свое контекстное меню.



При создании ВП контекстные меню для изменения внешнего вида или поведения объектов лицевой панели и блок-диаграммы. Для вызова контекстного меню щелкните правой кнопкой мыши на объекте лицевой панели или блок-диаграммы.

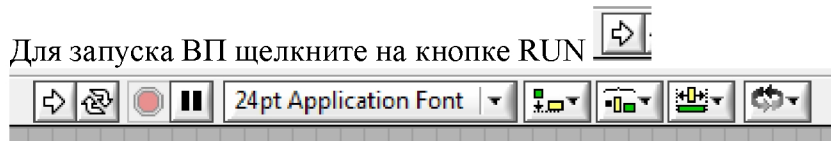
Диалоги свойств

С объектами лицевой панели также связаны диалоги свойств, которые используются для изменения вида или поведения объекта.



Для открытия диалога щелкните правой кнопкой на объекте и выберите Properties в контекстном меню.

Инструментальная панель лицевой панели.



2.6. Блок-диаграммы

К объектам блок- диаграммы относятся терминалы, подВП, функции, константы, структуры, а также проводники, передающие данные между объектами.

Терминалы

Терминалы представляют типы данных элементов управления или индикаторов. Вы можете конфигурировать элементы управления и индикаторы лицевой панели так, что на блок-диаграмме они будут представлены терминалами либо в виде иконок, либо в виде терминалов типов данных. По умолчанию объекты лицевой панели имеют вид терминалов-иконок. Терминалы являются входными и выходными портами, через которые идет обмен информацией между лицевой панелью и блок- диаграммой. Данные, которые Вы вводите в элементы управления лицевой панели (**a** и **b** на рис. ???) поступают на блок-диаграмму через терминалы элементов управления. Данные затем поступают на функции Add (Сложение) и Subtract (Вычитание). Когда функции Add и Subtract завершат свои внутренние вычисления, на их выходах будут выработаны новые значения. Данные поступают на индикаторные терминалы, где они покидают блок-диаграмму, поступают на лицевую панель и появляются на индикаторах лицевой панели (**a+b** и **a-b** на рис. ???).

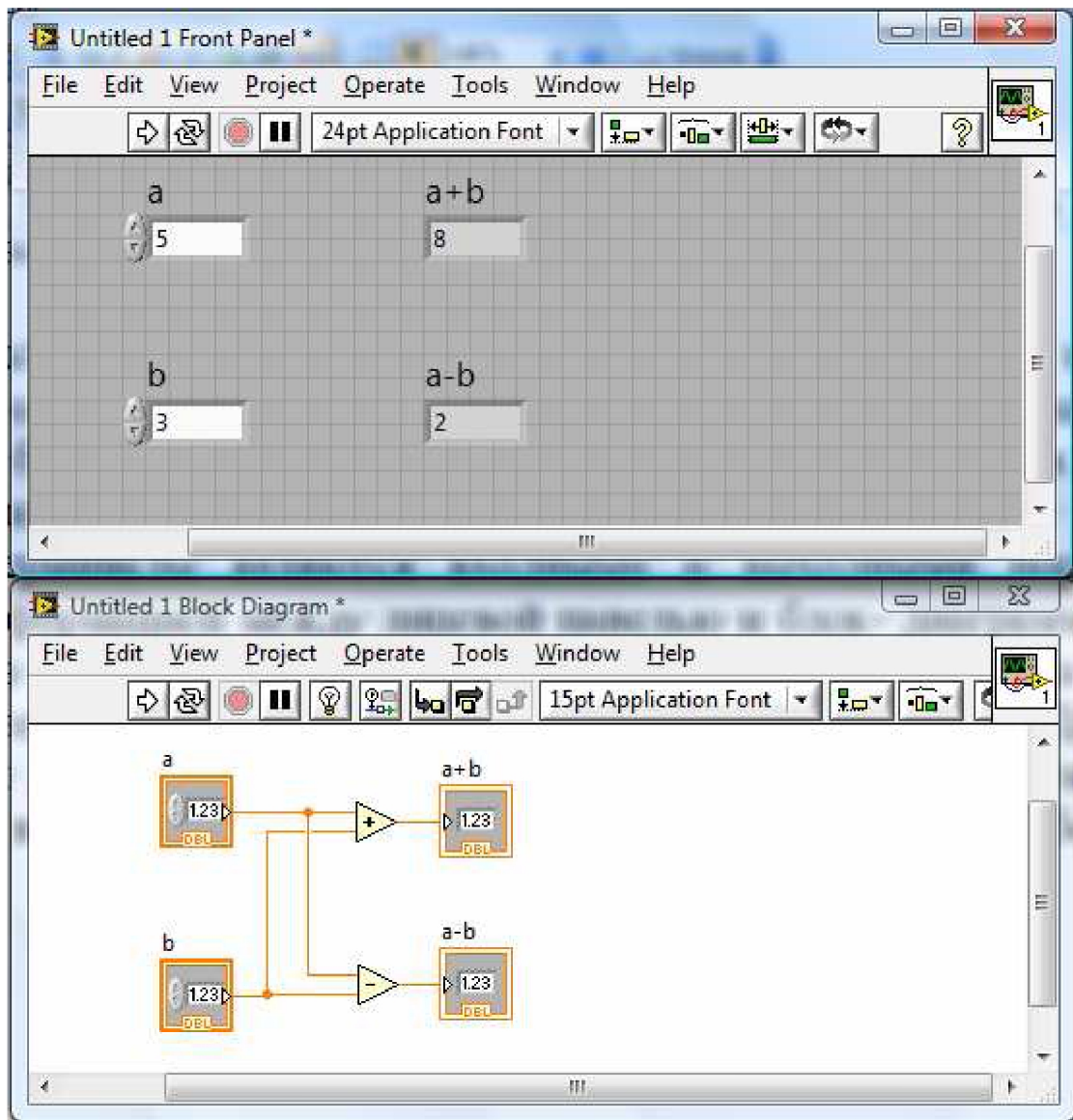


Рис. Терминала

Узлы блок-диаграммы

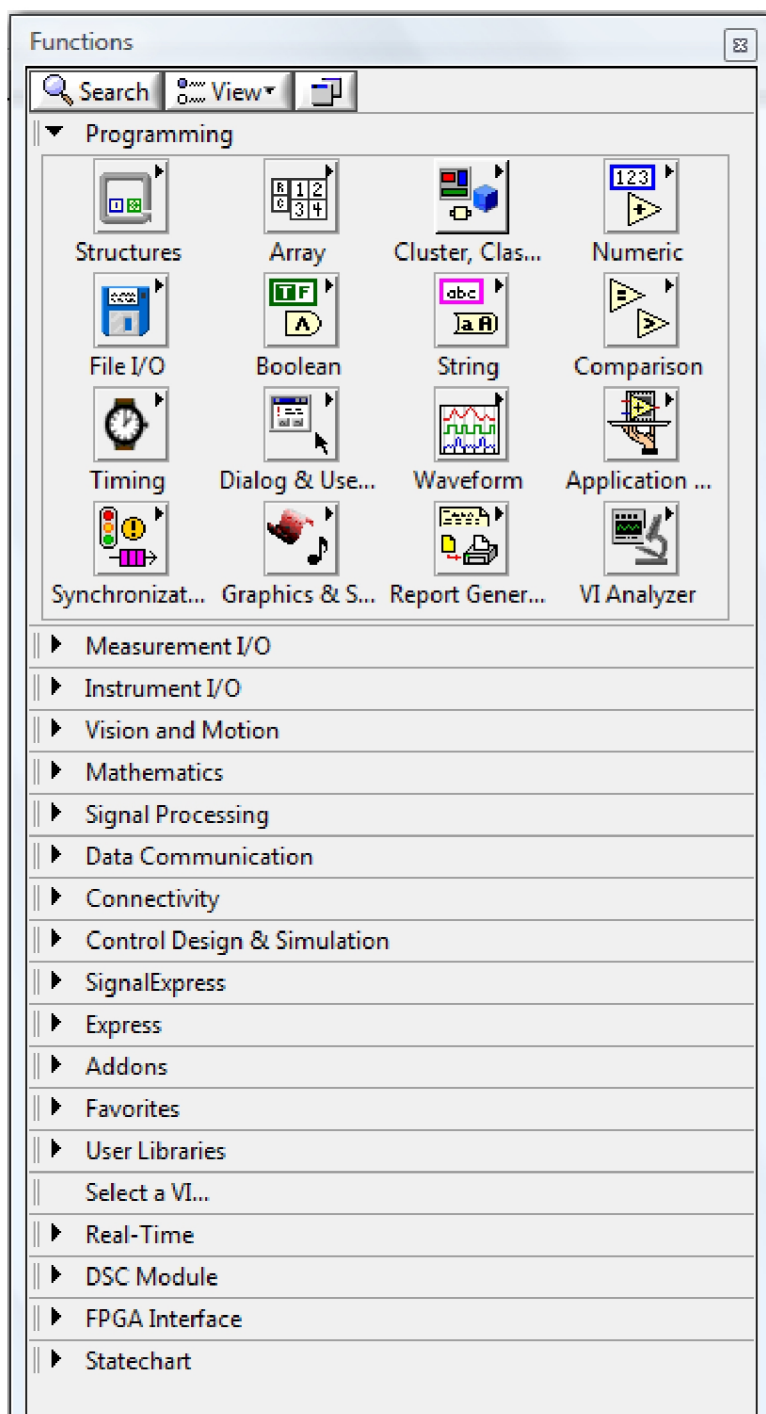
Узлы – это объекты блок-диаграммы, которые имеют входы и/или выходы и выполняют операции, когда ВП запущен. Они аналогичны операторам, функциям и подпрограммам в текстовых алгоритмических языках. Узлами могут быть функции, подВП или структуры. Структуры – это элементы управления выполнением программы, такие как структура выбора Case, цикл For или цикл While.

Функции

Функции – встроенные исполнительные элементы, сравнимые с оператором или функцией.

Обзор функций

Функции – это фундаментальные элементы LabVIEW. У них нет лицевых панелей или блок-диаграмм, но есть соединительные панели. Двойной щелчок на функции только выделяет ее. Иконки функций на палитре **Functions** имеют светло желтый фон и черные линии. Функции нельзя ни открывать, ни редактировать.



Палитра **Functions** содержит также ВП, которые поставляются вместе с LabVIEW. Используйте эти ВП как ВПП, когда Вы строите свои ВП для сбора данных, управления приборами, связи и т.п.

Числовые функции

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

Логические функции

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

- Преобразование значения TRUE (истина) в значение FALSE (ложь) и наоборот.
- Определение булевой величины, которую нужно возвращать, если получены две или более булевы величины.

- Преобразование булевой величины в число (1 или 0).
- Выполнение сложных операций над двумя и более булевыми величинами.

Строковые функции

Используйте строковые (String) функции для выполнения следующих задач:

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

Функции над массивами

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

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

Кластерные функции

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

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

Функции сравнения

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

Временные и диалоговые функции

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

- Манипуляция со скоростью выполнения операций.
- Извлечение информации о времени и дате из часов вашего компьютера.
- Создание диалоговых окон для выдачи пользователям инструкций.

Палитра Time & Dialog содержит также ВП для обработки ошибок.

Функции файлового ввода/вывода

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

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

Палитра File I/O содержит также ВП, которые выполняют общие задачи файлового ввода/вывода.

Функции работы с осциллограммами

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

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

Функции управления приложениями

Используйте функции управления приложениями (Application Control) для программного управления ВП и LabVIEW-приложениями на Вашем локальном компьютере или через компьютерную сеть.

Дополнительные функции

Используйте дополнительные (Advanced) функции для вызова программного кода из библиотек, таких как динамически связываемые библиотеки (dynamic link libraries – DLL), чтобы манипулировать данными LabVIEW для использования их в других приложениях, для создания и манипуляции регистрационными ключами Windows и для вызова сегментов кода на текстовых языках программирования.

Добавление терминалов у функций

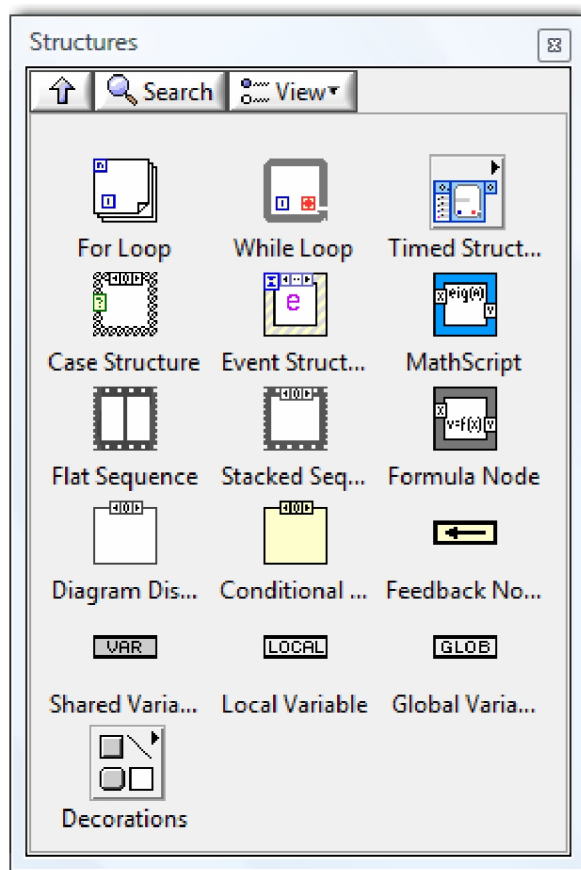
Вы можете изменять количество терминалов у некоторых функций. Например, для построения массива из 10 элементов, Вы можете увеличить до 10 количество терминалов у функции Build Array. Вы можете добавлять терминалы для растягиваемых (expandable) ВП и функций, используя инструмент Positioning для оттягивания нижней или верхней границы функции вниз или вверх, соответственно. Вы можете также использовать инструмент Positioning для удаления терминалов у растягиваемых ВП или функций, но Вы не сможете удалить уже подсоединенные терминалы. Вы можете также добавлять или удалять терминалы, щелкая правой кнопкой по терминалам функции и выбирая **Add Input** (добавить вход), **Add Output** (добавить выход), **Remove Input** (удалить вход) или **Remove Output** (удалить выход) из контекстного меню. Для разных функций Вы можете добавлять терминалы для входов, выходов или для элементов управления ссылочными номерами (refnum). Пункты контекстного меню **Add Input** и **Add Output** добавляют терминалы сразу же после того, как Вы щелкните правой кнопкой по терминалу. Пункты контекстного меню **Remove Input** и **Remove Output** удаляют терминалы после того, как Вы щелкните правой кнопкой по терминалу. Если для удаления подсоединенных терминалов Вы используете пункты контекстного меню, то LabVIEW удаляет терминалы и разрывает проводники.

ПодВП

Виртуальный прибор, вызываемый с блок-диаграммы другого виртуального прибора, называется виртуальным подприбором (ВПП). ВПП соответствует подпрограмме в тестовых языках программирования. Узлу ВПП соответствует вызов подпрограммы. Этот узел не является самим ВПП, также как оператор вызова подпрограммы не является самой подпрограммой. Блок-диаграмма, которая содержит несколько одинаковых узлов ВПП, несколько раз вызывает один и тот же ВПП. Элементы управления и индикаторы ВПП получают данные от блок-диаграммы вызывающего ВП и возвращают в нее результаты. Щелкните на палитре **Functions** иконку **Select a VI**, найдите нужный файл ВП, сделайте по нему двойной щелчок и поместите этот ВП на блок-диаграмму, чтобы создать ВПП, который вызывается из данного ВП. Перед тем, как использовать ВП в качестве ВПП, Вы должны установить соединительную панель.

Структуры

Это – элементы управления процессами, такие как структуры последовательности Flat Sequence и Stacked Sequence, структура выбора Case, циклы For Loops и While Loops.



Проводники

Проводники(wires) служат для передачи данных между объектами блок-диаграммы. Каждый проводник имеет единственный источник данных, но его можно соединить со многими ВП и функциям, которые читают эти данные. Проводники имеют различные цвета, стили и толщину, в зависимости от их типа данных. Поврежденные проводники имеют вид пунктирных черных линий с красным символом **X** посередине. Стрелки с боков красного символа **X** указывают направление потока данных, а цвет стрелок указывает тип данных, проходящих по проводнику. Когда Вы перемещаете инструмент Wiring над узлом ВП или функции, около не присоединенных терминалов появляются обрывки проводников. Они показывают тип данных каждого терминала. Появляются также краткие подсказки (tip strip) с именами терминалов. После того, как Вы подсоедините терминал, обрывок проводника для такого терминала перестанет появляться при перемещении инструмента Wiring над его узлом. Сегмент (segment) проводника это одна горизонтальная или вертикальная часть проводника. Изгиб (bend) проводника – это место, где сливаются два сегмента. Точка, в который два или более сегмента сливаются, есть соединение (junction). Одна ветвь (branch) проводника содержит все сегменты проводника от соединения до соединения, от терминала до соединения или от терминала до терминала, если между ними нет соединений. На рис. 5-1 показаны сегмент, изгиб и соединение проводника.

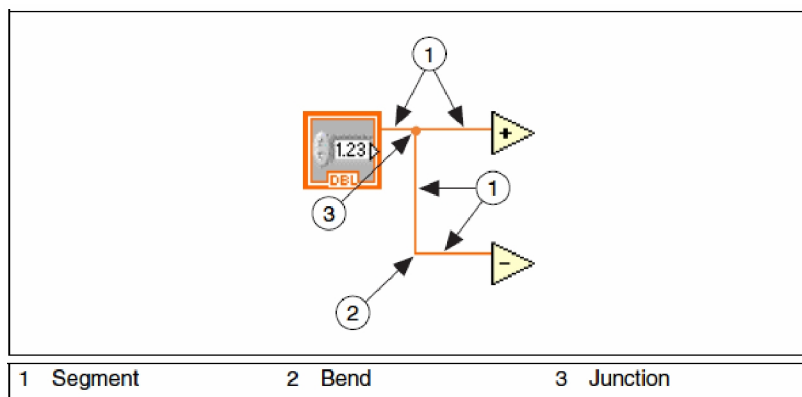


Рис пример блок-диаграммы и соответствующей лицевой панели.

В следующей таблице показаны наиболее распространенные типы проводников.

Тип	Скаляр	1D массив	2D массив	Цвет
Числовой				Оранжевый (число с плавающей точкой) Синий (целое число)
Логический				Зеленый
Строковый				Розовый

Автоматическое соединение объектов

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

Если Вы переместите селектированный объект близко к другому объекту на блок-диаграмме, то LabVIEW начертит временные проводники, чтобы показать Вам правильные соединения. Когда Вы отпускаете кнопку мыши, чтобы поместить объект на блок-диаграмму, LabVIEW автоматически подсоединит проводники. Включение/выключение автоматического соединения проводников осуществляется нажатием клавиши пробела при перемещении объекта с помощью инструмента Positioning. По умолчанию автоматическое соединение активно, когда Вы выбираете объект из палитры **Function**, или, когда Вы копируете объект уже имеющийся на блок-диаграмме, нажимая клавишу <Ctrl> и перемещая этот объект. Автоматическое соединение по умолчанию отключено, когда Вы используете инструмент Positioning для перемещения уже имеющегося на блок-диаграмме объекта. Вы можете отключить автоматическое соединение, выбирая **Tools»Options** и затем **Block Diagram** из выпадающего меню. В диалоговом окне нужно снять птичку с опции **Enable auto wiring**.

Соединение объектов вручную

Используйте инструмент Wiring для соединения вручную терминалов одного узла блок-диаграммы с терминалами другого узла блок-диаграммы. Курсорной точкой этого инструмента является конец распушенной катушки с проводом. Когда Вы перемещаете инструмент Wiring над терминалом, этот терминал мерцает. При этом появляется также подсказка с указанием имени терминала. Если подключение к терминалу будет давать поврежденный проводник(broken wire), то курсор перестанет иметь вид катушки и превратится в текстовое предупреждение. Вы можете создавать поврежденные проводники, но их нужно исправить прежде, чем Вы сможете запускать ВП.

Для точного определения того, как правильно подсоединять проводники, пользуйтесь окном контекстной помощи **Context Help**. Когда Вы перемещаете курсор над

ВП или функцией, в окне **Context Help** появляется перечень всех терминалов ВП или функции. Окно **Context Help** не показывает терминалы для растягиваемых (expandable) ВП и функций, таких, например, как функция Build Array. Чтобы отобразить необязательные терминалы соединительной панели, щелкните кнопку **Show Optional Terminals and Full Path** в окне **Context Help**.

Палитра Functions (Функции)

Палитра **Functions** доступна только на блок-диаграмме. Палитра **Functions** содержит ВП и функции, которые Вы можете использовать для построения блок-диаграммы. ВП и функции размещены на подпалитрах в соответствии с типами ВП и функций. Какие ВП и функции размещены на палитре **Functions** и их внешний вид зависит от текущих настроек внешнего вида палитры. Для отображения палитры **Functions** выберите **Window»Show Functions Palette** или щелкните правой кнопкой мыши на рабочем пространстве блок-диаграммы. Палитру **Functions** можно переместить в любое место на экране. LabVIEW запоминает положение и размер палитры **Functions**, поэтому после следующего запуска LabVIEW палитра появится в том же месте и сохранит свои прежние размеры.

Вы можете изменять способ представления палитры **Functions**.

2.7. Поток данных

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

Visual Basic, C++, JAVA и большинство других текстовых языков программирования в процессе исполнения программы следуют модели потока управления (control flow). В модели потока управления порядок выполнения программы определяется последовательным порядком программных элементов. Поскольку в LabVIEW порядок исполнения элементов блок-диаграммы определяет поток данных, а не последовательный порядок команд, можно создавать блок диаграммы с параллельными операциями. Например, Вы можете одновременно запустить два цикла While Loops и отображать результаты их выполнения на лицевой панели. LabVIEW является многозадачной и многопоточной (multithreaded) системой, которая выполняет одновременно несколько исполняемых потоков и виртуальных приборов.

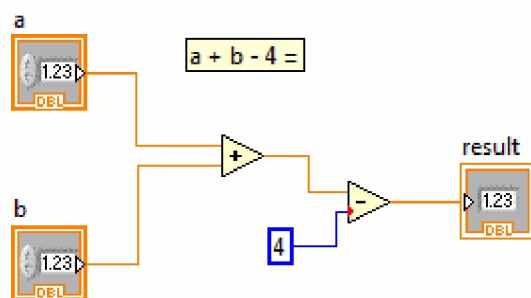


Рис Примера потока данных

3.Создание ВП

3.1 Проектирование лицевой панели

На этапе проектирования определяются входные и выходные данные задачи. Эта информация позволяет сразу переходить к проектированию лицевой панели.

Источником входных данных могут служить:

1. физическое устройство, например, плата сбора данных или мультиметр
2. файл
3. элементы управления

Выходные данные можно отображать на индикаторах или записывать в файл, выводить на физическое устройство, т.е. генерировать сигнал.

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

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

Использование текстовых меток

Для идентификации объектов на лицевой панели и блок-диаграмме используйте метки (**labels**). В LabVIEW имеется два типа меток – собственные метки (owned labels) и свободные метки (free labels). Собственные метки принадлежат отдельному объекту, перемещаются вместе с ним и характеризуют только его. Вы можете перемещать собственную метку независимо, но если Вы переместите помеченный объект, то метка переместится вместе с таким объектом. Можно скрыть (hide) собственные метки, но нельзя копировать или удалять их независимо от объекта. Для числовых элементов и индикаторов Вы можете также отобразить метку единиц измерения (unit label), выбирая из контекстного меню **Visible Items»Unite Label**.

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

Заголовки (Captions)

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

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

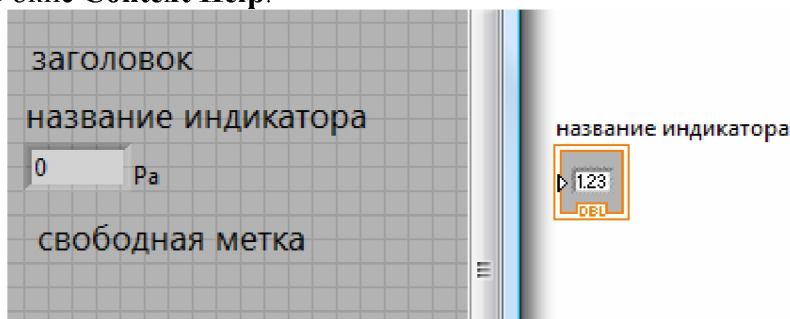


Рис Пример подписи и метки

Установка комбинаций клавиш для элементов управления

Вы можете назначить комбинации клавиш для элементов управления таким образом, что пользователь сможет перемещаться по лицевой панели без мыши. Щелкните правой кнопкой элемент управления и выберите **Advanced»Key Navigation** из

контекстного меню для отображения диалогового окна **Key Navigation** (Клавиши перемещения).

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

Раскрашивание объектов

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

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

Выравнивание и распределение объектов

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

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

Для отключения сетки или настройки ее параметров выберите из главного меню **Tools»Options** и затем категорию **Alignment Grid**. Чтобы выровнять объекты после того, как Вы их разместите, селектируйте эти объекты и затем вызовите спадающее меню нажатием кнопки **Align Objects** на панели инструментов. Чтобы равномерно распределить объекты, селектируйте их и затем вызовите спадающее меню нажатием кнопки **Distribute Objects** на панели инструментов.

Масштабирование объектов лицевой панели

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

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

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

3.2. Документирование кода

Профессиональные разработчики, которые занимаются сопровождением и модификацией ВП, знают цену хорошей документации. Документируйте блок-диаграмму, чтобы облегчить модификацию кода в будущем. Документируйте лицевую панель, чтобы было понятно назначение ее объектов в целом.

К средствам документирования лицевой панели относятся всплывающие подсказки, описания, свойства ВП, а также хороший дизайн.

3.3 Цикл While

Цикл While подобен конструкциям Do или do-while в текстовых языках программирования. Поддиаграмма выполняется до тех пор, пока не будет выполнено условие выхода.

Структура цикла **While** находится на палитре **Functions>>All Functions>>Structures**. Выберите цикл в палитре, а затем растяните курсором прямоугольник выделения вокруг той части блок-диаграммы, которую нужно повторять. После отпускания кнопки мыши выделенный участок будет заключен в рамку цикла. Для добавления к циклу других объектов блок-диаграммы просто перетаскивайте их внутрь рамки.

Цикл While всегда выполняется как минимум один раз.

Цикл While выполняется до тех пор, пока терминал условия – condition terminal (входной терминал цикла) не получит заданное логическое значение. По умолчанию терминал условия работает в режиме **Stop if True** (завершить по значению **True** Истина). Вы можете изменить поведение и внешний вид терминала условия, щелкнув на нем правой кнопкой и выбрав в контекстном меню пункт **Continue if True**

(продолжить по значению **True**).

В режиме **Continue if True** диаграмма цикла будет выполняться до тех пор, пока на терминал условия не поступит значение FALSE. Изменить условие можно также инструментом УПРАВЛЕНИЕ(Positioning), щелкнув им на терминале.

Терминал итерации – iteration terminal (выходной терминал), показанный слева, содержит значение, равное количеству завершенных итераций (повторов). Подсчет итераций всегда начинается с нуля. При этом в процессе выполнения первой итерации на выходе терминала итераций будет значение 0.

Туннели

Туннели (**tunnels**) используются для ввода и вывода данных из структуры. На блок-диаграмме туннель имеет вид сплошного квадрата на границе цикла. Его цвет совпадает с цветом проводника, подведенного к туннелю. Данные выводятся из цикла после его завершения. Если туннель используется для ввода данных в цикл, то выполнение цикла не начинается до поступления данных в туннель.

3.4 Цикл For

Структура For (цикл с заданным числом итераций), показанная слева, выполняет поддиаграмму заданное число раз. Цикл For находится на палитре **Functions>>All Functions>>Structures**. Можно также превратить существующий цикл While в цикл For. Для этого щелкните правой кнопкой на рамке цикла While и выберите в контекстном меню пункт **Replace with For Loop** (Заменить циклом For). Терминал количества итераций (входной терминал) показывает, сколько раз нужно повторять диаграмму цикла.

Счет итераций всегда начинается с нуля. Во время первой итерации терминал возвращает 0.

В отличие от цикла While, цикл For выполняется строго определенное количество раз. Цикл While, напротив, завершается только при подаче логического значения на терминал условия.

Преобразование числовых типов данных

3.5. Синхронизация ВП

После завершения текущей итерации цикла сразу же начинается выполнение следующей итерации, пока не будет выполнено условие завершения. Однако на практике часто требуется контролировать частоту итераций. Например, если считывание данных должно выполняться каждые 10 секунд, то необходим какой-то способ синхронизации цикла с часами реального времени.

Даже если вам не нужно привязывать выполнение к какой-то частоте, желательно предоставлять процессору время для завершения других задач, например обслуживания пользовательского интерфейса. В этом разделе описываются некоторые методы управления временем выполнения цикла.

Функции ожидания

Функция ожидания, помещенная внутрь цикла, приостанавливает ВП на определенное время. Это позволяет процессору обработать другие задания. Функция ожидания использует миллисекундный таймер операционной системы.

Функция `Wait Until Next ms Multiple` следит за счетчиком миллисекунд и ждет, пока его значение не станет кратно заданной величине. Используйте эту функцию для синхронизации операций. Помещенная внутрь цикла, она позволяет управлять частотой выполнения. Чтобы функция работала эффективно, время выполнения кода должно быть меньше времени, заданного для функции. Время выполнения первой итерации цикла не определено.

Функция `Wait (ms)` ждет до тех пор, пока счетчик миллисекунд не увеличится на заданную величину. Эта функция гарантирует, что период выполнения цикла будет не меньше, чем заданное вами число миллисекунд.

3.6. Передача данных между итерациями

При программировании циклов часто приходится обращаться к данным предыдущей итерации. Например, если в каждой итерации требуется считывать какие-то данные и проводить усреднение по пяти итерациям, то необходимо запоминать данные четырех предыдущих итераций. Для передачи данных от одной итерации к другой используются сдвиговые регистры (`shift registers`).

Сдвиговые регистры аналогичны статическим переменным в текстовых языках программирования.

Для создания сдвигового регистра щелкните правой кнопкой на правой или левой стороне рамки цикла и выберите в контекстном меню **Add Shift Register** (Добавить сдвиговый регистр). Сдвиговый регистр передает данные любого типа и автоматически настраивается на тип первого объекта, который с ним соединен. Данные, подаваемые на терминалы сдвигового регистра, должны быть одного типа. К циклу можно добавить более одного сдвигового регистра. Если внутри цикла есть несколько операций, использующих значения предыдущей итерации, используйте набор регистров для хранения данных этих нескольких процессов.

Инициализация сдвигового регистра позволяет задать значение, которое будет передано на первую итерацию цикла. Для инициализации соедините элемент управления или константу с левым терминалом регистра.

Стек сдвиговых регистров.

Стек сдвиговых регистров обеспечивает доступ к данным нескольких предыдущих итераций. Эти данные передаются на следующие итерации. Для создания стека сдвиговых регистров щелкните правой кнопкой на левом терминале и выберите в контекстном меню **Add Element** (добавить элемент).

Стек сдвиговых регистров может присутствовать только с левой стороны цикла, поскольку назначение правого терминала – передавать на следующую итерацию данные только текущей итерации.

3.7. Графическое представление данных

Waveform Chart

Индикатор Waveform Chart – это специальный числовой индикатор, который графически отображает поступающие данные (как правило, с постоянным периодом выборки) предназначены для отображения одной или нескольких кривых. Обновление данных на Waveform Chart может выполняться различными способами. Для выбора режима обновления щелкните на индикаторе правой кнопкой и выберите в контекстном меню Advanced>>Update Mode. Доступны следующие режимы:

Strip Chart (Ленточная диаграмма). Данные непрерывно прокручиваются в окне справа налево. Новые точки появляются справа, старые смещаются к левому краю окна. (Аналог – самописец)

Scope Chart (Циклическая развертка). Отображается фрагмент сигнала фиксированной длины, развернутый слева направо. Каждая следующая точка появляется справа от предыдущей. Этот режим соответствует осциллографу.

Sweep Chart (Непрерывная развертка). Аналогична циклической развертке, за исключением того, что старые данные не исчезают по окончании хода развертки, а постепенно заменяются новыми. Новые данные отделяются от старых данных вертикальной линией.

Соединения на диаграммах.

Скалярный выход можно непосредственно соединять с Waveform Chart. Терминал индикатора автоматически примет вид входного значения. На индикаторе Waveform Chart можно отображать несколько кривых одновременно. В этом случае данные подаются на вход при помощи функции **Bundle (Объединить)**, расположенной на палитре **Cluster**.

Waveform Graph

Виртуальные приборы, в которых используется индикатор Waveform Graph, обычно записывают данные в массив, а затем отображают их на экране.

На палитре графических индикаторов (**Graph**) расположены Waveform Graph и XY Graph. На индикаторе Waveform Graph можно построить только однозначную функцию вида $y=f(x)$, причем значения аргумента x должны быть эквидистантными. На индикатор XYGraph можно вывести произвольный набор точек.

3.8. Структура выбора Case

Структура Case состоит из двух или более поддиаграмм, называемых вариантами (Cases). В каждый момент времени на экране видна только одна поддиаграмма, и каждый раз, когда в ходе выполнения программы происходит обращение к структуре, выполняется только один вариант. Структура Case аналогична конструкциям switch или if...then...else в текстовых языках программирования.

Метка селектора варианта в верхней части структуры содержит селекторное значение, соответствующее видимой в данный момент поддиаграмме, и стрелки прокрутки с обеих сторон.

Стрелки используются для просмотра существующих вариантов.

Входное значение (селектор) может иметь целочисленный, логический, строковый или перечислимый тип. Если терминал селектора имеет логический тип, то структура содержит два варианта: TRUE и FALSE. Для целочисленного, строкового и перечислимого типов количество вариантов может быть любым.

Для значений, которые не соответствуют ни одному из вариантов, создается вариант по умолчанию (default case).

Входные и выходные туннели

В структуре Case можно создавать входные и выходные туннели. Входные значения будут доступны всем вариантам, но варианты не обязаны использовать каждый вход. При этом вы обязаны подать значение в выходной туннель в каждом из вариантов.

3.9. Узел формул.

Когда в LabVIEW требуется использовать сложное уравнение, не обязательно соединять на блок-диаграмме множество разных арифметических функций. Уравнение можно записать в привычном математическом виде и затем интегрировать в приложение.

Узел формул (Formula node) предназначен для выполнения математических операций на блок-диаграмме в удобной текстовой форме. Он позволяет избежать обращения к внешнему коду или другим приложениям, а также избавляет от необходимости графически конструировать уравнение из отдельных функций. Кроме текстовых выражений, в узле формул можно использовать оператор if, циклы while, for и do, знакомы программистам на C.

Для создания входных и выходных терминалов узла формул нужно щелкнуть правой кнопкой на рамке узла, выбрать в контекстном меню **Add Input** или **Add Output** и ввести имя переменной. После этого можно вводить уравнение в рабочую область структуры. Каждая инструкция должна оканчиваться точкой с запятой (;). Узлы формул можно использовать для принятия решений. На приведенной ниже блок-диаграмме показаны два различных варианта конструкции if...then. Эти две структуры выдают одинаковый результат. Узел формул может выполнять много различных операций.

4. Объединение данных

Иногда бывает выгодно сгруппировать связанные друг с другом данные. Для этого используются массивы и кластеры. Массив (**array**) представляет собой структуру, содержащую данные одного типа, а кластер (**cluster**) – структуру с данными разных типов. Для создания нестандартных массивов и кластеров применяются определения типов (**type definitions**).

4.1. Массив

Массив состоит из элементов и имеет измерения (dimensions). Элементы – это те данные, которые хранятся в массиве. Измерение(dimensions) – это длина, высота или глубина массива. Массив может иметь одно или более измерений и содержать до $(2^{31}-1)$ элементов на каждое измерение (если нет ограничений со стороны оперативной памяти).

Массивы можно создавать из чисел, логических значений, файловых путей, строк, сигналов (waveform) и кластеров. Нельзя создавать массивы, элементы которых сами являются массивами. Однако можно создать многомерный массив, а также массив кластеров, в котором каждый кластер содержит один или более массивов. Нельзя также создавать массивы многостраничных элементов управления, элементов .NET или ActiveX, диаграмм или графиков XY. Элементы массива упорядочены. Для быстрого доступа к любому элементу используются индексы (номера элементов). Индексация начинается с нуля.

Чтобы создать на лицевой панели массив элементов управления или индикаторов, поместите на панель оболочку массива. Затем перетащите в нее элемент управления или индикатор, который может представлять число, логическое значение, строку, путь, ссылку (refnum) или кластер.

В двумерном массиве (2D) элементы расположены в виде матрицы. Чтобы создать на лицевой панели многомерный массив, щелкните правой кнопкой на поле индекса и выберите в контекстном меню пункт **Add Dimension**.

Инициализация массива не является обязательной. При инициализации определяется число элементов в каждом измерении и содержимое каждого элемента.

Чтобы создать массив, состоящий из констант, выберите константный массив на палитре функций, поместите оболочку массива на блок-диаграмму, а затем вставьте в оболочку строковую, числовую или кластерную константу.

При передаче массива внутрь цикла For или While, а также при выводе массива из цикла можно сделать так, чтобы в каждой итерации использовался только один элемент массива, индекс которого совпадает с номером итерации. Такой режим называется автоиндексацией.

Если включить автоиндексацию для массива, поданного на входной терминал цикла For, то на терминал количества итераций автоматически поступает размер массива. Поскольку циклы For часто используются для поэлементной обработки массивов, LabVIEW по умолчанию включает автоиндексацию для всех массивов, передаваемых в цикл For. Выключите автоиндексацию, если вам не нужно обрабатывать массивы поэлементно.

При автоиндексации выходного туннеля цикла выходной массив получает новый элемент на каждой итерации. Следовательно, размер автоиндексируемого выходного массива всегда равен числу итераций.

Чтобы включить или выключить автоиндексацию, щелкните правой кнопкой на выходном туннеле и выберите соответствующий пункт в контекстном меню. Для циклов While автоиндексация по умолчанию выключена.

4.2. Кластеры.

Кластеры позволяют сгруппировать элементы данных разных типов. Кластер аналогичен записи или структуре в текстовых языках программирования.

Объединение нескольких элементов данных в кластер упрощает соединения на блок-диаграмме и сокращает число терминалов соединительной панели в подВП.

У большинства кластеров проводники и терминалы на блок-диаграмме имеют розовый цвет. Кластеры числовых значений, иногда называемых точками, имеют коричневые проводники и терминалы.

Кластеры, в отличие от массивов, имеют фиксированный размер.

Чтобы создать на лицевой панели кластер из элементов управления или индикаторов, сначала поместите на панель оболочку кластера, затем перетащите в нее элемент управления или индикатор, который может представлять число, логическое значение, строку, путь, ссылку, массив или другой кластер.

Чтобы создать на блок-диаграмме кластер-константу, выберите кластерную константу на палитре функций, затем поместите на блок-диаграмму оболочку кластера и поместите в оболочку строковую, числовую или кластерную константу.

Чтобы создать на блок-диаграмме кластерную константу с теми же элементами, что и у кластера на лицевой панели, можно либо перетащить кластер с лицевой панели на блок-диаграмму, либо выбрать в контекстном меню кластера пункт **Create >> Constant**.

Элементы кластера логически упорядочены. Этот порядок не связан с положением элементов в оболочке. Первый помещенный в кластер объект имеет порядковый номер 0, второй – 1 и т.д. Если удалить один элемент, остальные будут автоматически перенумерованы.

Для создания и изменения кластеров используются функции из палитры **Cluster**. С их помощью можно выполнять следующие типичные операции:

1. Извлекать элементы данных из кластера.
2. добавлять элементы данных в кластер.
3. Разбирать кластер на отдельные элементы.

Для сборки кластера из элементов используются функции Bundle, для модификации – функции Bundle и Bundle By Name, а для разборки – функции Unbundle и Unbundle By Name.

Эти функции вызываются через контекстное меню терминалов кластеров на блок-диаграмме (пункт **Cluster Palette**).

Для доступа к элементам существующего кластера по именам используется функция Bundle By Name. Она работает аналогично Bundle, только вместо порядковых номеров элементов используются их метки. Доступ возможен только к тем элементам, для которых эти метки были созданы. Количество входов не обязано совпадать с количеством элементов выходного кластера.

Чтобы связать входной терминал функции с элементом кластера, щелкните инструментом УПРАВЛЕНИЕ на терминале и выберите элемент в раскрывающем меню.

Можно также щелкнуть на терминале правой кнопкой и выбрать элемент через пункт **Select Item** контекстного меню.

Для разделения кластера на отдельные элементы используется функция **Unbundle**. Она возвращает элементы кластера с указанными именами. Количество выходных терминалов не обязано совпадать с количеством элементов входного кластера.

Чтобы выбрать элемент кластера, который будет соответствовать выходному терминалу, щелкните на терминале инструментом **УПРАВЛЕНИЕ**. Можно также щелкнуть на терминале правой кнопкой и выбрать элемент через пункт **Select Item** контекстного меню.

В LabVIEW есть специальный кластер, называемый кластером ошибки. Он используется для передачи информации об ошибках, содержит следующие элементы:

1. **status** – логическое значение, в случае ошибки равно **TRUE**.
2. **code** – 32-битовое знаковое целое, содержит код ошибки.
3. **source** – строка, идентифицирующая источник ошибки.

4.3. Определения типов.

Определения типов часто используются при создании нестандартных массивов и кластеров. Для расширения стандартного набора объектов лицевой панели можно создавать заказные элементы управления и индикаторы, отличающиеся по внешнему виду от встроенных элементов управления и индикаторов LabVIEW. Заказной элемент можно сохранить в дисковом каталоге или библиотеке LLB, а затем использовать на других лицевых панелях.

5. Отладка ВП.

Для запуска ВП достаточно подать на входы всех его подВП, структур и функций данные правильного типа. Однако это не гарантирует, что ВП будет работать так, как ожидалось. LabVIEW позволяет управлять выполнением ВП и помогает выявлять ошибки в структуре блок-диаграммы или в передаваемых данных.

5.1. Исправление неработоспособных ВП.

Если ВП не запускается, это означает, что он содержит код, который невозможно выполнить. Кнопка **Run** при этом выглядит «сломанной». Если после всех соединений на блок-диаграмме кнопка не приняла нормальный вид, то ВП содержит ошибки.

Поиск причин неработоспособности ВП.

Чтобы узнать, почему ВП неработоспособен, щелкните на кнопке **Run** или выберите пункт меню **View>>Error list**. В окне **Error list** будут перечислены все ошибки.

Раздел **Items with errors** (Элементы с ошибками) содержит имена всех объектов в памяти (таких как ВП и библиотеки проектов), содержащих ошибки.

Раздел **errors and warnings** содержит ошибки и предупреждения для ВП, выбранного в разделе **Items with errors**.

Раздел **Details** содержит описания ошибок, а в некоторых случаях – рекомендации по их устранению.

Нажав кнопку **Help**, можно открыть раздел справочной системы LabVIEW с подробным описанием ошибки и пошаговыми инструкциями по ее исправлению.

После щелчка на кнопке **Show Error** или двойного щелчка на описании ошибки будет подсвечен тот участок блок-диаграмм или лицевой панели, где находится ошибка.

Типичные причины неработоспособности ВП.

Ниже перечислены типичные причины, по которым ВП может стать неработоспособным в процессе редактирования.

1. Блок-диаграмма содержит «висящий» (ни с чем не соединенный) или разорванный из-за несоответствия типов данных проводник.
2. Не соединен обязательный терминал блок-диаграммы.
3. ПодВП неработоспособен, или вы отредактировали его соединительную панель после помещения иконки на блок-диаграмму.

5.2. Методы отладки

Если ВП запускается, но выдает непредвиденные результаты, используйте перечисленные ниже методы для поиска и устранения ошибок в ВП или в потоке данных на блок-диаграмме.

Задействуйте параметры `error in` и `error out`, доступные в большинстве встроенных ВП и функций.

Чтобы исключить предупреждения, выберите пункт меню **View>>Error list** и поставьте отметку **Show Warnings**. Найдите причины предупреждений и устраните их.

Убедитесь, что проводники соединены с нужными терминалами. Трижды щелкнув на проводнике инструментом УПРАВЛЕНИЕ, вы увидите весь путь передачи данных.

Проверьте значения по умолчанию для всех функций и подВП на блок-диаграмме, пользуясь окном **Context Help**.

Используйте диалог **Find** для поиска подВП, текста и других объектов, подлежащих исправлению.

Чтобы найти ВП с отсутствующими соединениями, выберите пункт меню **View>>Browse Relationships>>This VI's Hierarchy**.

Используйте подсветку выполнения для наблюдения за движением данных по блок-диаграмме.

Выполните ВП в пошаговом режиме, чтобы просмотреть все выполняемые им действия.

Используйте инструмент ПРОСМОТР ДАННЫХ для проверки промежуточных значений и просмотра ошибок на выходах ВП и функций, особенно тех, которые выполняют ввод-вывод.

Используйте контрольные точки для приостановки выполнения с целью пошаговой отладки или установки пробников (probes).

Переводите подВП в состояние ожидания, чтобы отредактировать значения в элементах управления индикаторах, определить, сколько раз выполняется код, или вернуться к началу подВП.

Проверьте, нет ли на выходах функций или подВП неопределенных данных.

Сравните набор используемых подВП со списками **View>>Brows Relationships>> This VI's SubVIs** (ПодВП данного ВП) и **View>>Brows Relationships>> Unopened SubVIs** (Не открытые подВП), чтобы выявить лишние подВП. Откройте также окно VI Hierarchy (Иерархия ВП), чтобы увидеть все подВП данного ВП.

Подсветка выполнения

Кнопка **Highlight Execution**, включает анимацию выполнения блок-диаграммы. В этом режиме передача данных по блок-диаграмме от узла к узлу выглядит как движение пузырьков по проводникам. Анимация в сочетании с пошаговым выполнением позволяет увидеть, как меняются значения при проходе всего ВП от узла к узлу.

Пошаговое выполнение

Пошаговое выполнение ВП позволяет проверить каждое действие на блок-диаграмме. Для входа в пошаговый режим нажмите кнопку **Step Over** или **Step Into** на инструментальной панели блок-диаграмме. Задержав курсор на одной из кнопок **Step Over, Step Into** или **Step Out**, вы увидите подсказку с описанием следующего шага, который будет выполнен при щелчке на этой кнопке. ПодВП можно выполнять пошагово или в обычном режиме.

Инструмент ПРОБНИК

Инструмент ПРОБНИК используется для проверки промежуточных значений на проводниках при выполнении ВП. Этот инструмент полезен на сложных блок-диаграммах с сериями операций, каждая из которых может возратить неправильные данные. Используйте его совместно с подсветкой выполнения, пошаговым выполнением и контрольными точками, чтобы точно определить, где именно появляются неправильные

данные. При пошаговом выполнении или паузе в контрольной точке индикатор обновляется сразу, как только данные становятся доступны. Когда выполнение приостанавливается на каком-либо узле, вы можете щелкнуть инструментом на проводнике, по которому только что прошли данные, и увидеть их значение.

Типы пробников

1. **Универсальный пробник** используется для просмотра данных, передаваемых по проводнику. Щелкните правой кнопкой на проводнике и выберите в контекстном меню пункт **Custom Probe>>Generic Probe**, чтобы установить пробник.

2. **Просмотр данных с помощью индикаторов лицевой панели.** Для просмотра данных, передаваемых по проводнику, можно использовать индикатор лицевой панели (например, график), помещенный внутрь пробника. Щелкните правой кнопкой на проводнике, выберите **Custom Probe>>Control** в контекстном меню, а затем нужный индикатор. Можно также щелкнуть на иконке **Select a Control** в палитре **Control** и выбрать любой заказной элемент управления.

3. **Специализированные пробники.** Это ВП, которые отображают расширенную информацию о данных, передаваемых по проводнику. Например, ВП **Refnum Probe** (Пробник ссылок) возвращает имя ВП, путь к нему и шестнадцатеричное значение ссылки.

4. **Заказные пробники.** Для создания пробника заново или на основе существующего, используется мастер заказных пробников (**Custom Probe Wizard**). Для вызова мастера щелкните правой кнопкой на проводнике и выберите пункт контекстного меню **Custom Probe>>New**. Заказной пробник создают в тех случаях, когда нужен дополнительный контроль над способом отображения данных. Тип данных нового пробника будет соответствовать типу данных проводника, на котором вы щелкнули.

Контрольные точки

Инструмент **КОНТРОЛЬНАЯ ТОЧКА** позволяет установить контрольную точку на ВП, узле или проводнике блок-диаграммы, чтобы приостановить выполнение в этом месте.

Если контрольная точка установлена на проводнике, выполнение приостанавливается после передачи данных по проводнику. Если контрольная точка установлена на блок-диаграмме, приостановка произойдет после выполнения всех ее узлов. Когда ВП приостанавливается в контрольной точке, LabVIEW выводит на передний план блок-диаграмму и подсвечивает узел или проводник, содержащий эту точку. При наведении курсора на существующую контрольную точку черная область инструмента **КОНТРОЛЬНАЯ ТОЧКА** становится белой. Когда выполнение доходит до контрольной точки, ВП приостанавливается, и кнопка **Pause** становится красной. Вы можете предпринять следующие действия:

1. Начать пошаговое выполнение, используя соответствующие кнопки.
2. Просмотреть промежуточные значения на проводниках.
3. Изменить значения в элементах управления на лицевой панели.
4. Щелкнуть на кнопке **Pause**, чтобы продолжить выполнение до следующей контрольной точки или до конца программы.

Состояние ожидания.

ПодВП можно перевести в состояние ожидания, чтобы отредактировать значения элементов управления и индикаторов, узнать, сколько раз подВП был выполнен перед возвратом управления вызвавшему его ВП или выполнить подВП с самого начала. Можно сделать так, чтобы подВП входил в состояние ожидания при каждом вызове, или только при каком-то определенном вызове.

5.3. Неопределенные или непредвиденные данные

Неопределенные данные, к которым относятся **NAN** (Not a Number, не число) и **Inf** (Infinity, бесконечность), лишают смысла все последующие операции. Эти два

символьных значения, которые могут выдаваться при операциях с плавающей точкой, служат признаком некорректного вычисления или бессмысленного результата.

LabVIEW не отслеживает переполнение (overflow) или потерю значащих цифр (underflow) для целых чисел.

5.4. Обработка ошибок

Как бы вы ни были уверены в создаваемом ВП, невозможно предусмотреть все проблемы, с которыми может столкнуться пользователь.

Автоматическая обработка ошибок

Каждой ошибке соответствует числовой код и текстовое сообщение. По умолчанию LabVIEW обрабатывает все ошибки, произошедшие во время выполнения ВП, следующим образом: ВП переводится в состояние ожидания, подВП или функция, где произошла ошибка, подсвечивается, а на экран выводится диалог ошибки.

Чтобы запретить автоматическую обработку ошибок для текущего ВП, выберите пункт меню **File>>VI Properties**, а затем **Execution** в раскрывающемся меню **Category**.

Ручная обработка ошибок

Для управления ошибками предназначены ВП и функции обработки ошибок на палитре **Dialog & User Interface**, а также параметры **error in** и **error out**, присутствующие у большинства ВП и функций.

ВП и функции сообщают об ошибках либо при помощи числового кода, либо при помощи кластера ошибки. Как правило, числовые коды используются в функциях, а кластеры – в ВП, которые имеют для этого специальные входы и выходы.

При любых операциях ввода-вывода следует учитывать возможность ошибки. Почти все функции ввода-вывода возвращают информацию об ошибках. Наличие ошибок особенно важно проверять при работе с файлами, внешними приборами, устройствами сбора данных и при организации связи между компьютерами.

Кластеры ошибок

Элементы управления и индикаторы, представляющие кластеры ошибок, используются для передачи информации об ошибках между подВП. Кластеры **error in** и **error out** содержат следующую информацию:

status – логическое значение, в случае ошибки равно TRUE.

code – 32-битовое знаковое целое, идентифицирующее ошибку. Ненулевой код и **status**=FALSE означают предупреждение.

source – строка, идентифицирующая источник ошибки.

Обработка ошибок в LabVIEW соответствует модели потока данных. Информация об ошибках распространяется по ВП аналогично другим данным.

Объяснение ошибок

При возникновении ошибки щелкните правой кнопкой внутри кластера и выберите пункт **Explain Error** контекстного меню, чтобы открыть соответствующий диалог. Если ошибок нет, но есть предупреждения, контекстное меню будет содержать пункт **Explain Warning**.

Обработка ошибок при помощи структуры Case.

Кластер ошибки используется для выбора вариантов в структуре Case. При соединении кластера ошибки с терминалом селектора структуры Case, метка селектора автоматически показывает два варианта – Error и No Error, а рамка структуры меняет цвет на красный для варианта Error и зеленый для варианта No Error. Если с терминалом селектора соединен кластер ошибки, структура Case реагирует только на логический параметр **status**.

Обработка ошибок при помощи цикла While

Кластер ошибки можно соединить с терминалом условия цикла While. При этом на терминал передается только значение логического параметра **status**. Когда возникает ошибка, цикл останавливается.

Когда с терминалом условия соединен кластер ошибки, пункты контекстного меню **Stop if True** и **Continue if True** меняются на **Stop on Error** и **Continue while Error**.

6. Основы измерений

6.1. Компьютерные измерительные системы

Главная задача любой измерительной системы – это измерение и/или генерация реальных физических сигналов. Измерительные устройства помогают собирать, анализировать и представлять результаты измерений.

6.2. Принципы измерений

В этом разделе вводятся базовые понятия, с которыми нужно ознакомиться до того, как выполнять какие-либо измерения с использованием устройств DAQ и приборов.

Ввод сигналов

Ввод сигналов (signal acquisition) – это процесс преобразования физического сигнала в набор данных, с которыми может работать компьютер. Измерение начинается с преобразования физического сигнала в электрический при помощи измерительного преобразователя, или датчика. Датчики могут генерировать электрический сигнал в ответ на воздействие температуры, механической силы, звука и света.

Физическая величина	Датчик
Температура	Термопара Резистивный датчик (RTD) Термистор Интегральный датчик
Интенсивность света	Фотоумножитель Фотоэлемент
Звуковое давление	Микрофон
Механическая сила и давление	Тензодатчик Пьезоэлектрический датчик Динамометрический датчик
Смещение	Потенциометр Линейный датчик на основе дифф. трансформатора (LVDT) Оптический кодер
Расход жидкости	Манометрический расходомер Турбинный расходомер Ультразвуковой расходомер
pH	pH-электроды

Формирование сигналов

Формирование сигнала (signal conditioning) – это предварительная обработка сигнала, направленная на повышение точности измерений, например, фильтрация, изоляция и т.д. Многие автономные приборы и устройства DAQ имеют встроенные схемы формирования сигналов. Для формирования сигналов можно также использовать внешнюю или специализированное устройство. National Instruments предлагает для этих целей устройства SCXI и другие.

Чтобы сигнал с датчика можно было надежно измерить, его необходимо согласовать со входом измерительного устройства.

Усиление

Усиление (amplification) – это самый распространенный способ формирования сигнала. Усиление электрических сигналов повышает точность оцифровки и уменьшает влияние шума.

Линеаризация

LabVIEW позволяет линейаризовать напряжение с датчика, чтобы оно было прямо пропорционально измеряемой величине. Встроенные функции масштабирования обеспечивают линейаризацию напряжений для тензодатчиков, резистивных датчиков, термопар и термисторов.

Возбуждение датчика

Для функционирования некоторых датчиков требуется сигнал возбуждения,, который может генерироваться системами формирования сигналов. Например, тензодатчики и резистивные датчики требуют подачи внешнего напряжения и тока, чтобы изменение их параметров можно было зарегистрировать.

6.3. Повышение качества измерений

Достижение предельной чувствительности

На величину наименьшего обнаружимого измерения напряжения влияют следующие факторы:

Разрешение и диапазон АЦП

Коэффициент усиления измерительного усилителя.

Комбинация разрешения, диапазона и усиления, определяющая такую характеристику, как шаг квантования.

Разрешение.

Разрешение АЦП определяется числом битов, используемых для представления аналогового сигнала. Чем выше разрешение АЦП, тем больше число уровней, на которые можно разбить входной диапазон, а следовательно, тем меньшее изменение напряжения можно обнаружить.

Входной диапазон

Диапазон определяется минимальным и максимальным уровнями аналогового сигнала, которые АЦП может оцифровать.

Усиление

Усиливая или ослабляя сигнал до его оцифровки сигнала, можно повысить точность представления. Усиление или ослабление изменяет эффективный входной диапазон АЦП, позволяя использовать максимальное число уровней для представления сигнала.

Диапазон, разрешение и усиление устройства DAQ определяют наименьшее обнаружимое изменение входного напряжения. Это изменение напряжения соответствует одному наименьшему значащему биту (LSB, least significant bit) цифрового значения и называется также шагом квантования (code width).

Шаг квантования

Шаг квантования – наименьшее изменение сигнала, которое может обнаружить система. Он вычисляется по следующей формуле:

$$C = D \cdot \frac{1}{2^R},$$

Где С – шаг квантования, D – входной диапазон устройства, R – разрешение в битах. Входной диапазон устройства определяется усилением и входным диапазоном АЦП. Чем меньше шаг квантования, тем более точно можно представить сигнал.

Повышение точности восстановления формы

Самый эффективный способ повысить точность восстановления формы сигнала – уменьшить шаг квантования и увеличить частоту выборки. Для достоверного измерения частоты сигнала частота выборки должна удовлетворить теореме Найквиста.

Теорема Найквиста формулируется следующим образом:

$$f_{\text{выборки}} > 2 \cdot f_{\text{сигнала}}$$

Теорема Найквиста утверждает, что для точного восстановления сигнала частота выборки должна как минимум в два раза превышать наивысшую частоту в спектре сигнала. В противном случае высокочастотные составляющие сигнала «отразятся» от

частоты Найквиста (равной половине частоты выборки) и наложатся на ту часть спектра, которая представляет интерес.

7. Сбор данных

7.1. Аппаратура

После преобразования неэлектрической физической величины в электрическую необходимо зарегистрировать полученный сигнал.

Типичная система сбора данных содержит следующие аппаратные средства: блок терминалов, кабель и устройство DAQ. Эта аппаратура позволяет превратить стандартный компьютер в систему измерений и автоматизации.

Блок терминалов и кабель

Блок терминалов предназначен для присоединения проводов или кабелей, по которым передаются сигналы. Он содержит винтовые или пружинные терминалы, а также разъем для подключения кабеля, соединяющего его с устройством DAQ. В блоке может быть 100, 68 или 50 терминалов. Выбор того или иного блока терминалов определяется тем, какое устройство DAQ у вас есть и сколько сигналов вы измеряете.

Блоки терминалов могут быть экранированными и неэкранированными. Экранированные блоки терминалов лучше защищают от шумов.

Кабель передает сигналы от блока терминалов к устройству DAQ. Разъемы кабеля могут иметь 100, 68 или 50 контактов. Выбор кабеля зависит от используемого блока терминалов и устройства DAQ. Кабели, как и блоки терминалов, бывают экранированные и неэкранированные.

Панель DAQ Signal Accessory

Панель DAQ Signal Accessory представляет собой блок терминалов, три разъема для подключения различных устройств DAQ.

Устройства DAQ

Большинство устройств DAQ состоят из четырех функциональных частей: аналоговый ввод, аналоговый вывод, цифровой ввод-вывод и счетчики.

Передавать данные от устройства DAQ в компьютер можно через различные интерфейсы:

1. плату DAQ с шиной PCI, устанавливаемую в настольный компьютер;
2. плату с интерфейсом PCMCIA для ноутбука;
3. устройство DAQ, подключаемое через интерфейс USB;
4. устройство в стандарте PXI/CompactPCI.

7.2. Программная архитектура

Для плат сбора данных National Instruments разработаны драйверы, осуществляющие связь между платой и прикладными программами. В настоящее время существует два типа драйверов: NI-DAQmx и Traditional NI-DAQ. Для взаимодействия с драйверами можно использовать LabVIEW. В дополнение, National Instruments предлагает утилиту Measurement & Automation Explorer (MAX), которая помогает конфигурировать платы сбора данных.

7.3. Аналоговый ввод

Аналоговый ввод – это процесс измерения аналогового сигнала и передачи результата измерения в компьютер для анализа, отображения или сохранения. Аналоговым называется сигнал, который изменяется непрерывно. Чаще всего аналоговый ввод используется для измерения напряжения или тока. Для выполнения аналогового ввода можно использовать много разных устройств, в том числе multifunctional устройства сбора данных (DAQ MIO), скоростные АЦП, цифровые мультиметры (DMM) и устройства сбора динамических сигналов (DSA, Dynamic Signal Acquisition).

Аналого-цифровое преобразование

Для ввода аналогового сигнала в компьютер требуется выполнить аналого-цифровое преобразование, которое переводит электрический сигнал в цифровые данные, доступные для компьютерной обработки. Устройство, которое преобразует уровень напряжения в серию нулей и единиц, называется аналого-цифровым преобразователем (АЦП). Каждый сигнал преобразования состоит из выборки мгновенного значения аналогового сигнала и последующей его оцифровки. Частота выборки определяется частотой тактового сигнала. Аналоговый сигнал на входе АЦП может иметь любое значение, и в этом смысле он имеет бесконечную точность. При преобразовании точное значение заменяется на приближенное; точность этого приближения зависит от разрядности АЦП, используемого метода преобразования и некоторых других факторов.

Управление задачами

На языке LabVIEW операция по обмену данными с устройством ввода-вывода называется задачей (task). Задача аналогового ввода может заключаться во взятии одной выборки, N выборок, или непрерывном вводе.

Однократная выборка

Однократная выборка является операцией по требованию (on demand). По запросу прикладной программы драйвер считывает одно значение из канала ввода и сразу же передает его программе.

Многократная выборка

Одним из способов получения серии выборок из одного или нескольких каналов является повторное выполнение однократных выборок. Однако такой подход неэффективен, т.к. требует больших затрат времени и ресурсов компьютера. Кроме того, нельзя точно контролировать интервал между выборками. Альтернативой служит аппаратная синхронизация, при которой данные накапливаются в буферной памяти и затем считываются за один раз прикладной программой. Для этого нужно использовать функцию синхронизации, установив режим выборки **finite** (конечный) и задав частоту выборки (sample rate), а также количество выборок. Как и при использовании других функций, данные можно считывать с одного или нескольких каналов.

Непрерывная выборка.

Если требуется просматривать, обрабатывать или записывать данные по мере их поступления, выборки должны производиться непрерывно. В подобных случаях используется режим **continuous** (непрерывный).

Синхронизация задач

Каждое действие, которое выполняется устройством ввода-вывода под управлением NI-DAQmx, должно быть инициировано. События, иницирующие выполнение того или иного действия (например, взятия очередной выборки или серии выборок), называются триггерами. Триггер запуска (start trigger) иницирует ввод данных. Опорный триггер (reference trigger) устанавливает опорную точку в серии выборок. В результате одна часть данных будет считана с опережением (pretrigger data), а другая с запаздыванием (posttrigger data).

7.4. Аналоговый вывод

Аналоговый вывод – это процесс генерации электрического сигнала под управлением компьютера. Аналоговый сигнал генерируется при помощи цифро-аналогового преобразователя (ЦАП). В задачах аналогового вывода возможна генерация напряжения или тока.

Для выполнения задачи по генерации напряжения или тока в компьютер должно быть установлено устройство, способное выдавать такой сигнал.

Однократное обновление

Если уровень сигнала не должен часто меняться, используйте режим однократного обновления ЦАП. Для управления моментом обновления используйте программную синхронизацию.

Генерация N точек

При аппаратной синхронизации данные записываются в буфер и затем выводятся независимо от программы. При программной синхронизации частота обновления определяется программой и операционной системой, а не измерительным устройством.

Непрерывная генерация

Непрерывная генерация похожа на генерацию N точек, за тем исключением, что прекращение генерации происходит при определенном событии. Если требуется генерировать непрерывный сигнал, установите режим **continuous**.

Цифро-аналоговое преобразование.

Цифро-аналоговое преобразование противоположно аналого-цифровому, источником данных является компьютер. Эти данные могут быть получены ранее в процессе аналогового ввода, или сгенерированы прикладной программой. ЦАП принимает данные и использует их для изменения напряжения на своем выходе.

ЦАП имеет свой тактовый генератор, который определяет моменты генерации новых значений. Функции этого генератора подобны функции генератора тактовой частоты АЦП.

7.5. Счетчики

Счетчик представляет собой цифровой таймер. Он используется для подсчета событий, измерения частоты и периода, измерения перемещений и генерации импульсов. При работе со счетчиками используется следующая терминология.

Счетный регистр (count register) хранит текущее значение счетчика. Этот регистр можно считывать программно.

Источник (source) – входной сигнал, который изменяет текущее значение счетного регистра.

Строб (gate) – входной сигнал, который определяет, будет ли активный фронт сигнала изменять счетчик.

Выход (output) – представляет собой импульс или серию импульсов, иначе называемую цугом (pulse train).

7.6. Цифровой ввод-вывод

Под цифровыми сигналами понимаются электрические сигналы, которые передают по проводам цифровые данные. Такие сигналы имеют только два состояния: включено и выключен, которым соответствуют высокий и низкий уровни, или 1 и 0. При передаче цифрового значения отправитель прикладывает к проводнику напряжение, а получатель определяет по этому напряжению, какое значение было передано.

Цифровые сигналы находят много применений:

1. Управление устройствами с конечным числом состояний, таким как переключатели и светодиоды, или определение их состояния;
2. Передача данных;
3. Для программирования устройств или осуществления связи между устройствами
4. Для синхронизации и запуска.

Для приема цифровых сигналов используются цифровые линии устройства DAQ. Цифровой ввод синхронизируется программно. Некоторые устройства позволяют независимо конфигурировать цифровые линии для ввода или вывода.

Цифровые линии можно логически объединять для приема или передачи многоразрядных цифровых значений. Такая группа линий называется портом. Синхронизация ввода-вывода через порты осуществляется программно. Порты можно независимо конфигурировать для ввода или вывода.

8. Анализ и сохранение результатов измерений

8.1 Анализ и обработки числовых данных.

Решение задач, которые включают в себя взаимодействие с физическими процессами, обычно начинается с ввода данных в программу. Чтобы извлекать из данных полезную информацию, управлять процессом и получать результаты, необходимо

обрабатывать и анализировать. LabVIEW ориентирован на прикладные разработки и содержит функции для анализа. Эти функции можно встраивать непосредственно в приложения, чтобы проводить измерения и быстрее получать результаты.

Функции анализа можно встраивать в приложения различными способами. Для выбора оптимального способа существуют определенные критерии, которые следует принимать во внимание.

Оперативный и отложенный анализ

Оперативный (встроенный, inline) анализ выполняется в том же приложении, которое выполняет сбор данных. Такой подход применяется в управляющих приложениях, где решения должны приниматься в ходе выполнения, и результаты анализа используются для воздействия на процесс (через изменение параметров или выполнение определенных действий). При оперативном анализе важно учитывать объем собранных данных и сложность процедур обработки. Необходимо найти правильный баланс, поскольку объем вычислений может стать настолько большим, что это отрицательно повлияет на быстродействие.

Отложенный (offline) анализ следует использовать в тех случаях, когда для принятия решений не требуется получать результаты в реальном времени. Основное назначение таких приложений – выявлять причинно-следственные связи путем корреляции множественных наборов данных. Эти приложения обычно импортируют данные из двоичных или ASCII-файлов.

Программный и интерактивный анализ

Процесс анализа измерений можно упростить путем создания интерактивных приложений с диалогами и интерфейсами, дающими возможность пользователям выполнять те или иные операции над заданными наборами данных. Чтобы такой подход был эффективен, необходимо хорошо знать, какая информация и какие типы анализа представляют интерес для пользователя.

Категории анализа

LabVIEW содержит сотни встроенных функций, которые относятся к различным областям анализа, и реализуют разнообразные методы извлечения информации из собранных данных. Все функции разбиты на несколько категорий: Measurement (Амплитуда и уровень),

Signal Processing (Обработка сигналов),

Mathematics (Общая математика),

Image Processing (Обработка изображений),

Control (Управление),

Simulation (Моделирование),

Application Areas (Прикладные области).

Полный список функций анализа LabVIEW можно посмотреть на ni.com/analysis.

8.2. Чтение и запись данных в файл.

Файловый ввод-вывод применяется для записи данных в файл или чтения из файла.

Типичная операция файлового ввода-вывода состоит из следующих шагов:

1. Создание или открытие файла.
2. Чтение из файла или запись в файл.
3. Закрытие файла.

LabVIEW может использовать или создавать файлы следующих форматов: двоичный (binary), ASCII, LVM (расширение .lvm), TDM.

Каталог LabVIEW Data

Для хранения файлов данных, сгенерированных LabVIEW (например, .lvm или .txt), можно использовать стандартный каталог LabVIEW Data. Это каталог, размещаемый в стандартном каталоге файлов операционной системы, помогает организовывать и находить файлы данных, генерируемые LabVIEW.

Потоковый вывод на диск

Функции файлового ввода-вывода можно использовать для потоковых операций, которые экономят память за счет меньшего числа обращений к операционной системе для открытия и закрытия файла. При потоковом выводе файлы остаются открытыми, пока выполняются множественные операции записи, например, в цикле. Чтобы не открывать один и тот же файл заново, в цикл нужно передать ссылку на этот файл. LabVIEW создает ссылку при открытии файла, устройства или сетевого соединения.

Высокоуровневый файловый ввод-вывод

Перечисленные ниже ВП файлового ввода-вывода относятся к высокоуровневым.

- **Write to Spreadsheet File.** Преобразует двумерный или одномерный массив чисел одинарной точности в текстовую строку и записывает ее в новый ASCII-файл или добавляет к существующему.

- **Read from Spreadsheet File.** Считывает заданное количество строк или столбцов из числового текстового файла, начиная с определенной позиции, и преобразует данные в двумерный массив чисел одинарной точности. ВП открывает файл перед чтением и закрывает его по окончании чтения. Этот ВП можно использовать для чтения файла электронной таблицы, сохраненного в текстовом формате.

- **Write to Measurement File.** Этот экспресс-ВП записывает данные в текстовый файл измерений (.lvm) или двоичный файл измерений (.tdm).

- **Read from Measurement File.** Этот экспресс-ВП считывает данные из текстового файла измерений (.lvm) или двоичного файла измерений (.tdm).

Старайтесь не использовать высокоуровневые ВП в циклах, поскольку они выполняют операции открытия и закрытия файла при каждом вызове.