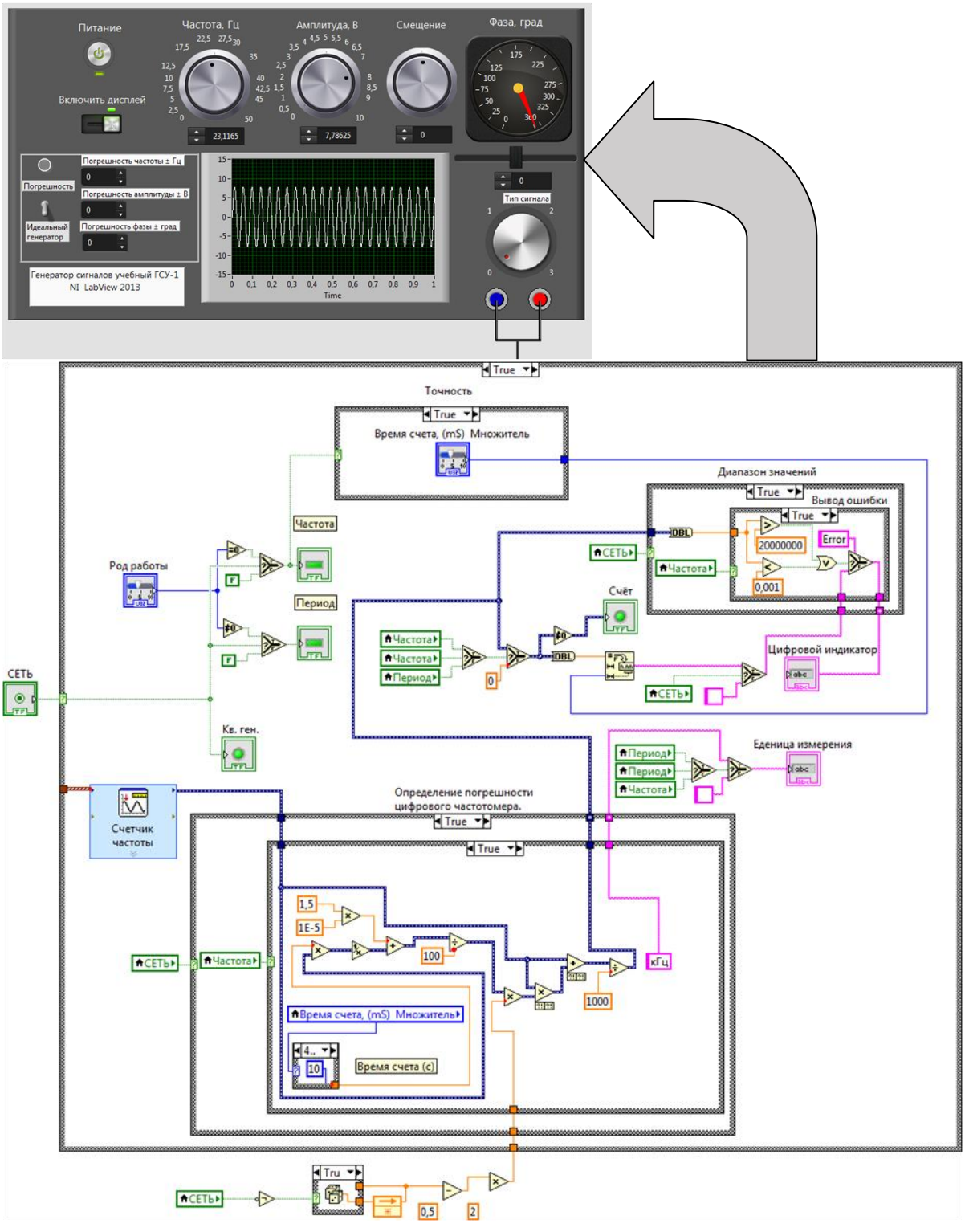
Основы моделирования в LabVIEW



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НОВГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ЯРОСЛАВА МУДРОГО

Основы моделирования в LabVIEW

Учебное пособие

Великий Новгород
2017

УДК 621.3
ББК 32.973

Печатается по решению
РИС НовГУ

Жукова, И.Н.,

Б19 Основы моделирования в LabVIEW: Учебное пособие / Авт. сост. И.Н. Жукова, С.А. Проторгуев НовГУ им. Ярослава Мудрого, Великий Новгород, 2017 г. – 73 с.

В учебном пособии описана среда графического программирования LabVIEW, некоторые базовые функции для создания приложений, тестирования и измерения, сбора данных, управления приборами, регистрации данных, анализа измерений.

Данное пособие предназначено для общего ознакомления со средой LabVIEW.

«Основы моделирования в LabVIEW» предоставляет список лекций, которые сопровождены упражнениями. Упражнения помогают на практике изучить основные темы.

Учебное пособие отвечает требованиям образовательного стандарта и предназначено для подготовки бакалавров по направлению 11.03.01 - «Радиотехника».

УДК 621.3
ББК 32.973

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Материал пособия «Основы моделирования в LabVIEW» знакомит студентов с теоретическими и практическими основами моделирования и создания виртуальных приборов в среде LabVIEW.

Все теоретические вопросы изложены в пособие в сжатой, последовательной форме, что облегчает усвоение материала студентами, систематизирует их знания.

Представленное пособие – «Основы моделирования в LabVIEW» обеспечивает преподавание дисциплины «Проектирование радиотехнических систем в среде LabVIEW» при подготовке бакалавров по направлению 11.03.01 – «Радиотехника».

1 ЗНАКОМСТВО СО СРЕДОЙ LABVIEW

LabVIEW - это среда создания приложений для задач сбора, обработки, визуализации информации от различных приборов и лабораторных установок. В среде LabVIEW также возможна разработка автономных моделей радиотехнических систем и устройств различного назначения.

Работа в LabVIEW может быть разнообразной. От создания простого сумматора до автономного полноценного прибора.

Отмеченные вопросы рассматриваются в настоящем учебном пособии.

1.1 Создание нового виртуального прибора

Программа в среде LabVIEW называется Virtual Instrument (VI) – виртуальный прибор (ВП), чтобы подчеркнуть, что она предназначена для управления лабораторным оборудованием и приборами и способна выполнять некоторые их функции.

Для ознакомления со средой LabVIEW:

- Запустим программу LabVIEW 12.0, при этом откроется стартовое окно (рис.1.1).
- Для создания нового проекта выберем **Create Project.**
- Для создания нового виртуального прибора выберем пункт **BlankVI** (пустой виртуальный прибор (ВП)).

При этом откроются два окна – **Untitled Front Panel** (рис.1.2) и **Untitled Block Diagram** (рис.1.3). Слово **Untitled** – "не имеющий наименования" указывает на то, что мы еще не назвали наш виртуальный прибор и не сохранили его на диск компьютера.

Если окно **Block Diagram** скрыто, то его можно вызвать через меню **Window →Show Block Diagram** или нажав клавиши <Ctrl>+<E>.

- Сохраним новый ВП. Для первого сохранения ВП в меню **File** любого из этих окон выберем пункт **Save**.
- В открывшемся диалоговом окне выберем или создадим новую папку для наших ВП, введем имя файла – Example01 и подтвердим сохранение, нажав **OK**. Файл сохранится с расширением **.vi** от Virtual Instrument – Виртуальный прибор.

Чтобы не потерять сделанную работу, время от времени следует сохранять ВП в процессе его редактирования. Для этого можно использовать меню **File → Save**, или сочетание клавиш <Ctrl>+<S>.

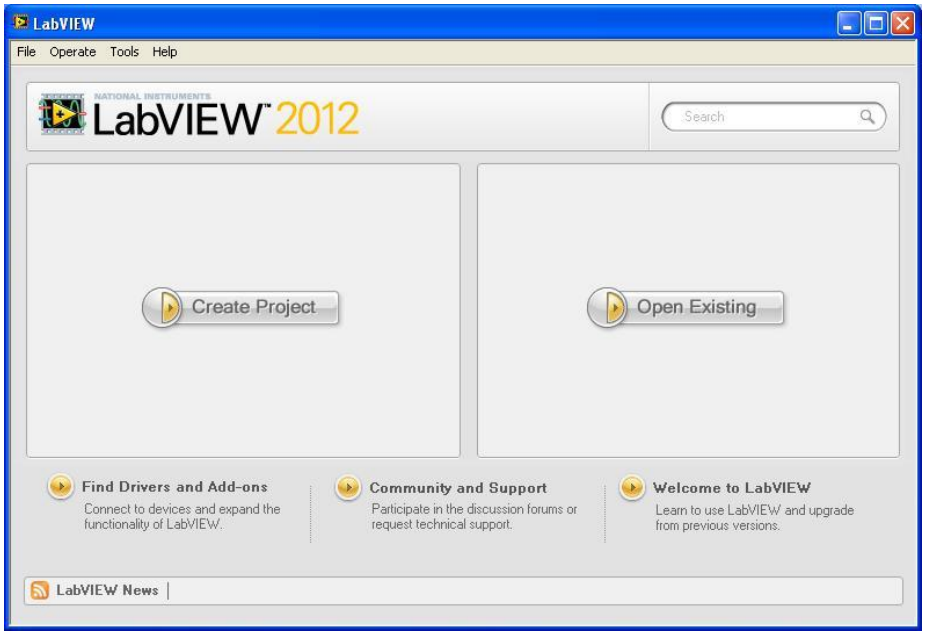


Рис. 1.1. Стартовое окно LabVIEW 12.0

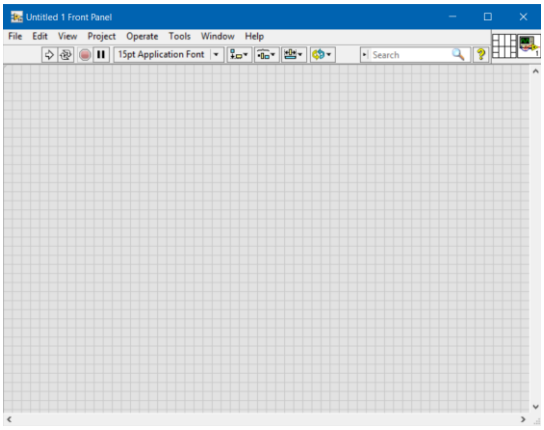


Рис.1.2. Окно **Front Panel**

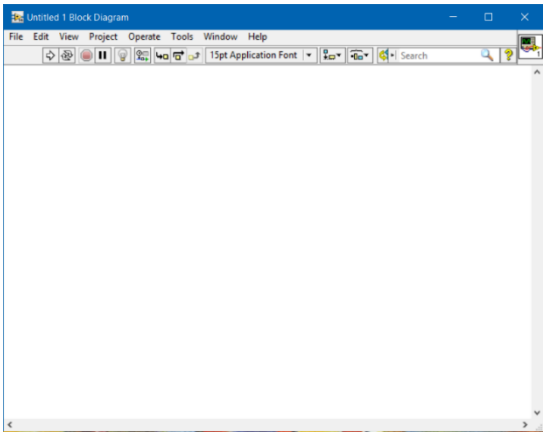


Рис.**1.3.** Окно **Block Diagram**

Передняя панель (**Front panel**) виртуального прибора служит для размещения на ней элементов управления и индикаторов – то есть элементов, при помощи которых этот виртуальный прибор будет взаимодействовать с человеком (или с другим виртуальным прибором, как мы увидим немного позже).

1.2 Главное меню

Главное меню в верхней части окна ВП содержит пункты общие с другими приложениями, такие как **Open**, **Save**, **Copy**, **Paste**, а также специфические пункты меню LabVIEW. Некоторые из них содержат сведения о «горячих» клавишах вызова этих пунктов. Внимание. Во время выполнения ВП некоторые пункты главного меню недоступны.

- Пункт меню **File** используется для открытия, закрытия, сохранения и печати ВП.
- Пункт меню **Edit** используется для поиска и внесения изменений в компоненты ВП.
- Пункт меню **View** используется для вывода различных инструментов на экран.
- Пункт меню **Project** используется для практически как пункт меню **File.**
- Пункт меню **Operate** используется для запуска, прерывания выполнения и изменения других опций ВП.
- Пункт меню **Tools** используется для связи с приборами и DAQ устройствами, сравнения ВП, формирования приложений и конфигурации LabVIEW.
- Пункт меню **Window** используется для отображения окон LabVIEW и палитр.
- Пункт меню **Help** используется для получения информации о палитрах, меню, инструментах, ВП и функциях, для получения пошаговой инструкции использования LabVIEW и информации о компьютерной памяти.

1.3 Лицевая панель

Лицевая (передняя) панель имитирует панель реального физического прибора. На ней располагаются управляющие и измерительные элементы виртуального прибора. Пример лицевой панели представлен на рис. 1.4.

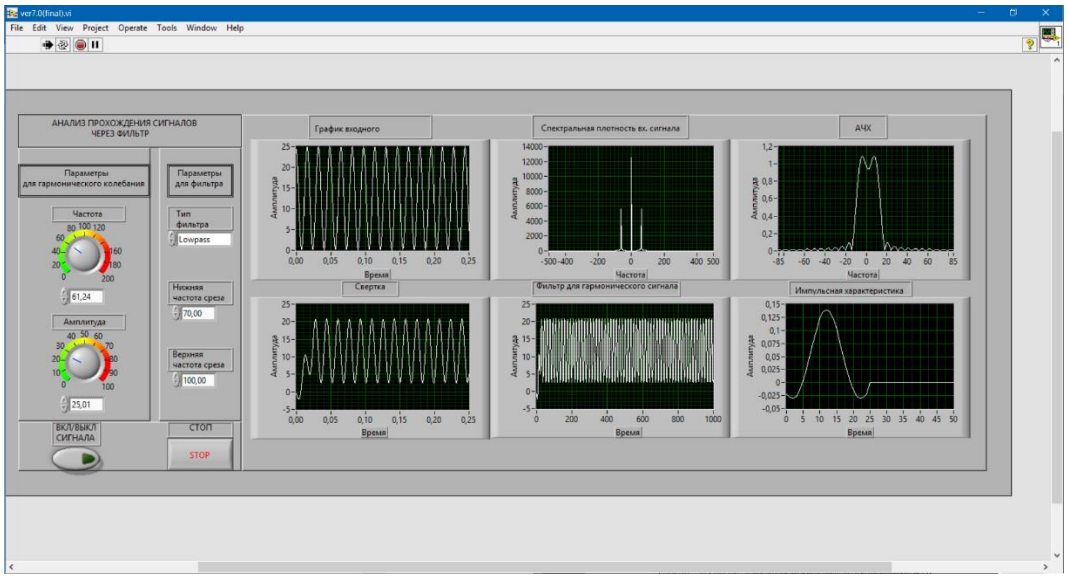


Рис.1.4. - Лицевая панель ВП

1.3.1 Палитра элементов лицевой панели

Лицевая панель создается с использованием палитры элементов под общим названием **Controls**, которая вызывается нажатием правой клавиши мыши на свободное поле лицевой панели. Эти элементы могут быть либо средствами ввода данных - элементами собственно управления **(Controls)**, либо средствами отображения данных - элементами отображения **(Indicators)**.

По умолчанию палитра элементов появляется в экспресс-виде (рис. 1.5) и содержит лишь наиболее часто используемые элементы.

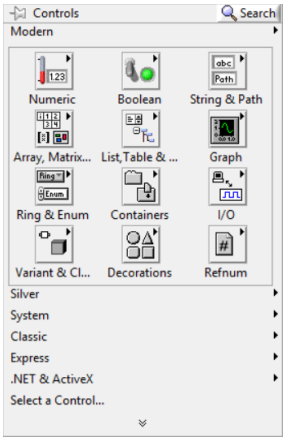


Рис. 1.5. – Палитра элементов лицевой панели

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

1.3.2 Инструментальная панель лицевой панели

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

Рис.1.6. - Инструментальная панель

 Кнопка запуска **Run** – запускает ВП

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

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

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

 Во время выполнения ВП появляется кнопка **Abort Execution**[1]. Эта кнопка используется для немедленной остановки выполнения ВП.

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

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

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

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

 Меню **Reorder** используется при работе с несколькими объектами, которые накладываются друг на друга. Выделив один из объектов с помо-

---

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

щью инструмента ПЕРЕМЕЩЕНИЕ, в меню **Reorder** следует выбрать его порядок отображения на лицевой панели.

 Кнопка **Search** используется для поиска элементов.

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

1.4 Блок-диаграмма

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

1.4.1 Палитра функций блок-диаграммы

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

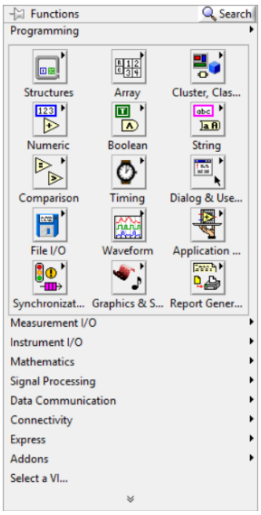


Рис.1.7. – Палитра функций блок-диаграммы

1.4.2 Инструментальная панель блок-диаграммы

При запуске ВП на блок-диаграмме появляется, показанная на рис. 1.8, инструментальная панель:

Рис.1.8 - Инструментальная панель

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

Кнопка **Retain Wire Values** предназначена для сохранения данных прошедших по проводникам. Включив ее, можно посмотреть значения данных в любом проводнике ВП в любой момент времени.

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

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

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

Кнопка **Clean Up Diagramm** позволяет автоматически развести функции для удобного чтения всего прибора.

Кнопка **Warning** появляется, когда есть потенциальная проблема с блок-диаграммой, но она не запрещает выполнение ВП. Кнопку **Warning** можно активизировать, войдя в пункт главного меню **Tools → Options → Debugging → Environment.**

Имея понятия об этих основных функциях и инструментах можно с легкостью начинать изучения среды LabVIEW для разработки виртуальных приборов.

2 ПЕРВЫЕ ШАГИ В LABVIEW

2.1 Блок схема. Элементы управления и индикаторы

Для создания нового ВП выберем пункт *BlankVI* (пустой виртуальный прибор (ВП)).

Рассмотрим процесс создания ВП на примере выполнения небольшого задания.

Задание:

Рассчитать значения токов I_1, I_2, I_3 в резистивной цепи (рис.2.1) при изменении сопротивлений R_1, R_2, R_3. Значение ЭДС E является постоянной величиной и задаётся самостоятельно пользователем.

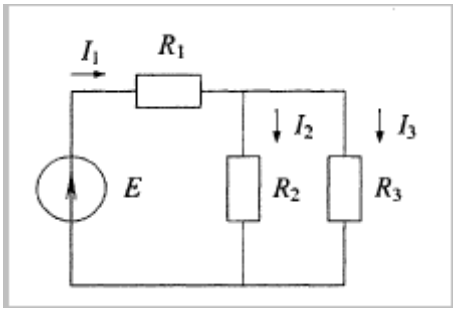


Рис.2.1. – схема электрической цепи

Решение:

1) Запишем значения ЭДС E и сопротивлений R_1, R_2, R_3, которые также выбираются пользователем.
2) Выведем уравнения токов согласно 1-му и 2-му законам Кирхгофа:

$$I_1 = \frac{E}{\frac{R_1 + R_2 * R_3}{R_2 + R_3}} \quad (2.1)$$

$$I_2 = \frac{R_3 * I_1}{R_2 + R_3} \quad (2.2)$$

$$I_3 = \frac{R_2 * I_1}{R_2 + R_3} \quad (2.3)$$

$$R = \frac{R_1 + R_2 * R_3}{R_2 + R_3} \quad (2.4)$$

Для создания будущего ВП перейдём в переднюю панель (*Front panel*) виртуального прибора размещения на ней элементов управления.

Переместим указатель мыши на рабочую (клетчатую) поверхность передней панели и нажмем *правую* кнопку мыши.

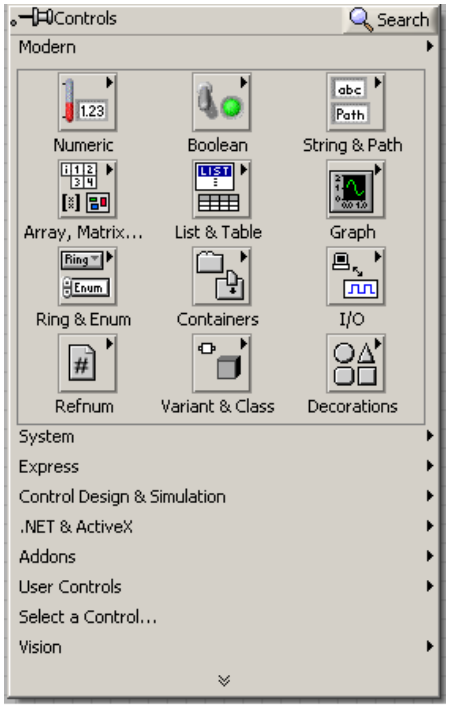


Рис.2.2. - Меню **Controls** и раздел **Modern**

Рис.2.3. - Меню **Numeric**. Элемент управления. **Numeric Control** выделен рамкой

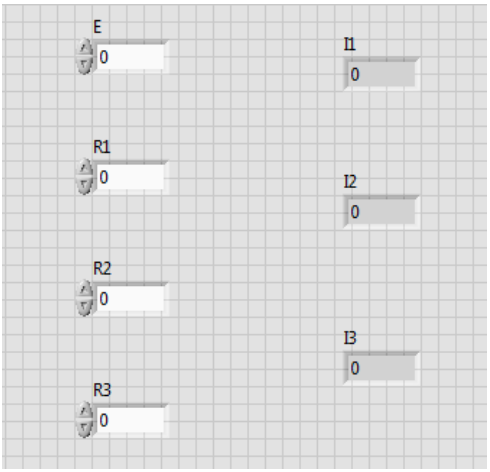


Рис.2.4. - Элементы управления и индикатор на передней панели

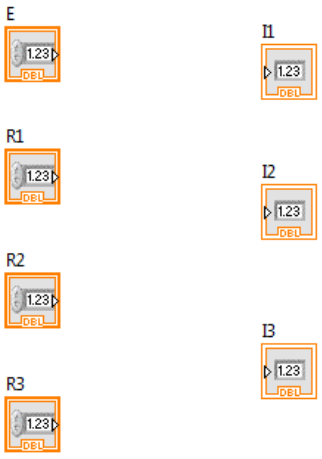


Рис.2.5. - Терминалы элементов управления и индикатора на блок схеме

- В появившемся меню элементов управления **Controls** из раздела **Modern** (современные) выберем пункт **Numeric** (числовые, цифровые) (рис.2.2).
- В раскрывшемся меню **Numeric** выберем **Numeric control** (числовой элемент управления) (рис.2.3). Указатель мыши примет вид руки, держащей пунктирные прямоугольники – контур элемента управления.
- Переместим указатель на переднюю панель, и установим элемент, нажав один раз на *левую* кнопку мыши.
- Два раза щелкнем *левой* кнопкой мыши на надпись **Numeric** над элементом управления и введем текст: *E*.
- Добавим на переднюю панель еще три элемента управления и назовем их: R_1; R_2; R_3;
- Аналогичным образом добавим на переднюю панель индикаторы (***Controls → Modern →Numeric →Numeric Indicator***). Назовем их I_1; I_2; I_3.
- Теперь передняя панель выглядит так, как показано на рис.2.4 – на ней расположены элемента управления, используя которые можно ввести в компьютер значения сопротивлений - R_1, R_2, R_3, значение источника напряжения E и индикаторы с результаты токов в контурах. А на блок схеме появились соответствующие им *терминалы* (рис.2.5), при помощи которых введенные E и R_1, R_2, R_3 передаются в программу, а результат выполнения программы выводится на индикаторы I_1, I_2, I_3.

Обратите внимание, что терминалы элементов управления находятся слева, а индикатора – справа. Они расположены так, потому что общепринятое направление передачи данных и сигналов на блок схемах – *слева направо,* и *сверху вниз.* Этого правила следует обязательно придерживаться при оформлении программ в среде LabVIEW! Выходы терминалов элементов управления (*маленькие треугольники*) расположены *справа* на иконках этих терминалов, а вход индикатора – *слева* (рис.2.5).

В среде LabView все процедуры и функции представлены в виде иконок. Переместим указатель на переднюю панель, и установим элемент, нажав один раз на *левую* кнопку мыши. В появившемся меню элементов управления **Functions** из раздела **Mathematics** выберем пункт **Script & Formulas** и нажмём на **Formula Node**. Далее записываем уравнение токов (2.1 – 2.4) их в окно **Formula Node** как показано на рис.2.6.

- Щелкните левой кнопкой мыши и подведите курсор к I_1, I_2, I_3. При этом он примет вид катушки с проводом (**Wiring tool**– инструмент для соединения проводами). За курсором потянется пунктирная линия – набросок будущего провода. Его направление можно задавать, нажимая на *левую* кнопку мыши. Нажатие на *правую* кнопку *отменяет* рисование провода.
- Нажмите на рамку окна **Formula Node** правой кнопкой мыши и выберите пункт **Add Input** для создания входных данных (E; R_1, R_2, R_3) слева и аналогично создайте окна для выходных данных (I_1, I_2, I_3) пунктом **Add Output** справа.

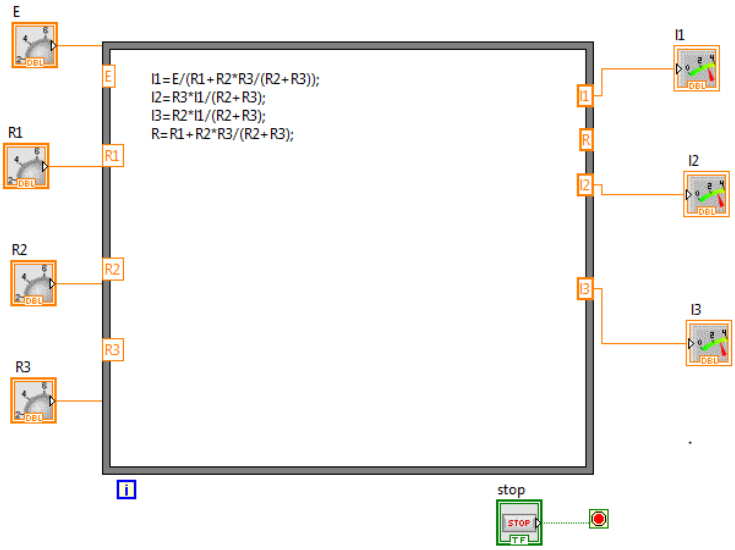


Рис.2.6. - команда **Formula Node**

Соединяя контакты проводами, следует руководствоваться следующими правилами:

1. Передача данных в блок-схеме осуществляется слева направо.
2. Проводники ВСЮДУ на виду – их нельзя перекрывать иконками или другими проводниками.
3. В одной точке могут соединяться НЕ БОЛЕЕ ТРЕХ проводников.
4. Блок схема должна полностью помещаться на экране монитора.

После того, как редактирование завершено, необходимо сохранить программу, используя меню **File → Save**, или сочетание клавиш <Ctrl>+<s>. Теперь программа полностью готова, и ее можно запустить. Для этого нажмем кнопку циклического выполнения программы (**Run Continuously***)*

- Убедимся в ее работоспособности, изменяя при помощи мыши и клавиатуры значения в полях ввода элементов управления и наблюдая как изменяется значение индикатора.
- Чтобы остановить выполнение программы, нажмем кнопку прервать выполнение *(***Abort Execution***)*

При запуске наша программа однократно опрашивает состояние элементов управления, передает их значения операции вычисления, а затем результат сложения индикатору и завершается. Для того чтобы она непрерывно отслеживала изменения состояния элементов управления, мы использовали возможности среды программирования – режим **Run Continuously**, в котором программа запускается снова и снова сразу же после ее завершения. Теперь сделаем так, чтобы программа самостоятельно периодически опрашивала элементы управления, складывала полученные числа и отображала результат вычислений на индикаторах. Для этого поместим нашу программу в цикл **While Loop** (while – пока, в то время как; loop – петля). (В дальнейшем понятию цикл будет посвящена отдельная глава, здесь большинство понятий примем как факт).

- Щелкнем правой кнопкой мыши блок-схеме и выберем в **меню *Functions → Programming → Structures → While Loop***
- Курсор мыши примет вид маленького пунктирного прямоугольника с черным левым верхним уголком
- Поместим курсор левее и выше группы терминалов и процедур, которую мы хотим заключить в цикл, нажав и удерживая левую кнопку, обведем группу пунктирным прямоугольником.
- После того, как мы отпустим левую кнопку мыши, программа будет выглядеть так, как показано на рис.2.7.

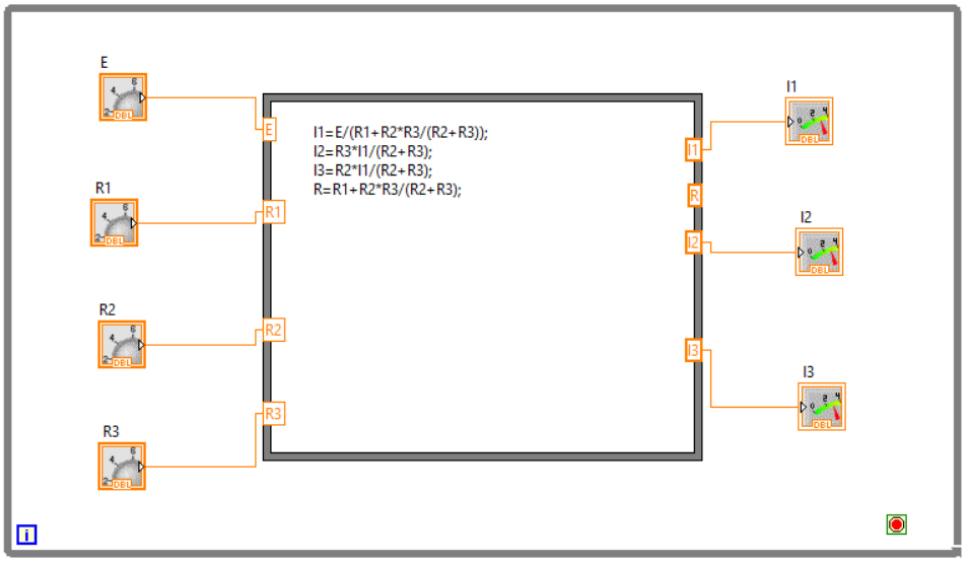


Рис.2.7. - Цикл **While Loop**

- Если действие не получилось с первого раза, можно его отменить, нажав клавиши <Ctrl>+<z>. После того, как мы поместили цикл **While Loop** на блок-схему, кнопка циклического выполнения заблокировалась, а на кнопке запуска программы появилось изображение сломанной стрелочки. Это произошло потому, что среда програм-

мирования LabVIEW проверяет программу непосредственно в процессе ее составления. Она обнаружила ошибку и запретила запуск программы. Выясним причину ошибки.

- Нажмем на кнопку со сломанной стрелкой **List Errors** (составить список ошибок), откроется окно списка ошибок (рис.2.8)

Окно списка ошибок состоит из трех частей:

Items with errors (объекты с ошибками), в этом окне указан наш виртуальный прибор *Example01.vi*;

1 errors and warnings (1 ошибка(и) и(или) предупреждение(ия)), это перечень ошибок в котором указан тип ошибки – **Block Diagram Errors** (ошибки в блок схеме), и ее причина – **While Loop: conditional terminal is not wired** (Цикл While: не подключен провод к терминалу условия);

Details (подробности) – объяснение ошибки, выделенной в списке. В нашем случае **The conditional terminal is not wired to anything and must be wired to a boolean data source such as a button or the result of a comparison.** (Терминал условия ни с чем не соединен и должен быть соединен к источнику данных логического (Булева) типа, такому как кнопка или результат сравнения).

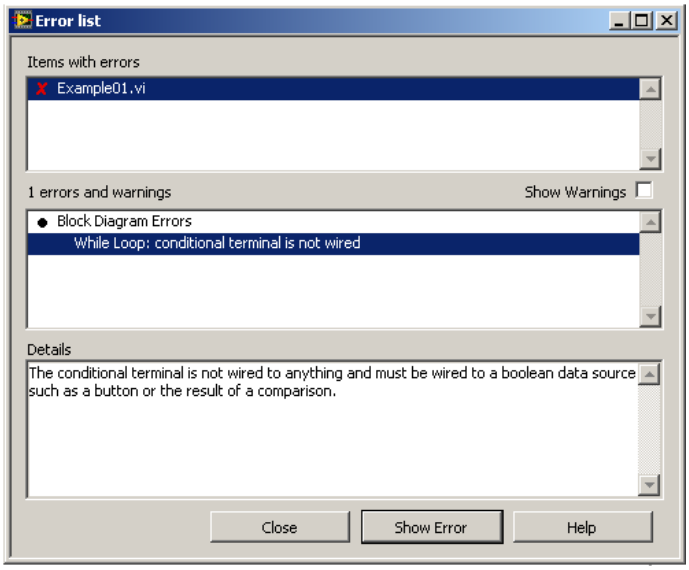


Рис.2.8. - Окно списка ошибок

Чтобы выяснить, в каком именно месте блок-схемы обнаружена ошибка, нажмем кнопку **Show Error** (Показать ошибку). При этом окно блок-схемы станет активным, а цикл **While Loop** будет кратковременно выделен черным цветом.

Для того чтобы получить информацию о неизвестном элементе блок-схемы, можно воспользоваться окном контекстной помощи, которое открывается при нажатии клавиш <Ctrl>+<H>. В этом окне появляется описание каждого элемента блок схемы, на который помещен указатель мыши. Более подробное описание каждого элемента блок схемы, а также ссылки на примеры его использования можно найти в справочной системе (**Help**) среды LabVIEW.

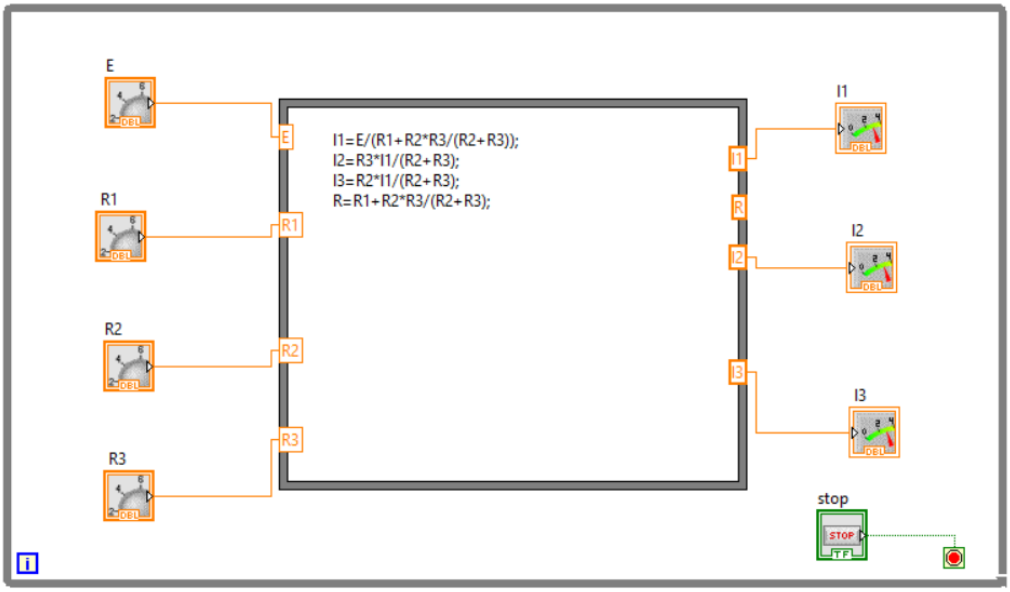


Рис.2.9. - Терминал кнопки **Stop** на блок-схеме

Исправим ошибку на блок-схеме. Для этого подключим к терминалу условия цикла кнопку, прерывающую его выполнение.

- Наведем *указатель* мыши на терминал условия цикла, и, нажав *правую кнопку мыши*, выберем в появившемся меню режим работы терминала **Stop if True**. Убедимся, что иконка терминала приобрела соответствующий вид.
- Наведем указатель мыши на *контакт* терминала, и когда курсор превратится в *катушку с проводом* (**Wiring tool**), нажмем правую кнопку мыши
- В появившемся меню выберем **Create Control** (создать элемент управления)
- В результате на передней панели появится кнопка управления **Stop**, а ее терминал на блок схеме автоматически соединится с терминалом условия цикла (рис.2.8)
- Теперь изменим вид элементов управления на передней панели. Поместим указатель мыши на элемент *E* и нажмем *правую кнопку мыши*.
- В появившемся меню выберем **Replace** (заменить) → **Modern** → **Numeric** → **Knob** (шарообразная ручка). Элемент управления при-

мет вид, показанный на рисунке справа. Передвинем надпись *E* при помощи мыши.

- Изменим атрибуты ручки. Нажмем правую кнопку мыши и в выпадающем меню выберем **Visible Items** (Видимые объекты**)** → ***Ramp***.
- Для удобства точной установки ручки выберем в том же меню Digital Display (цифровой дисплей).
- Аналогично изменим остальные элементы управления, чтобы передняя панель выглядела так, как показано на рис.2.10.

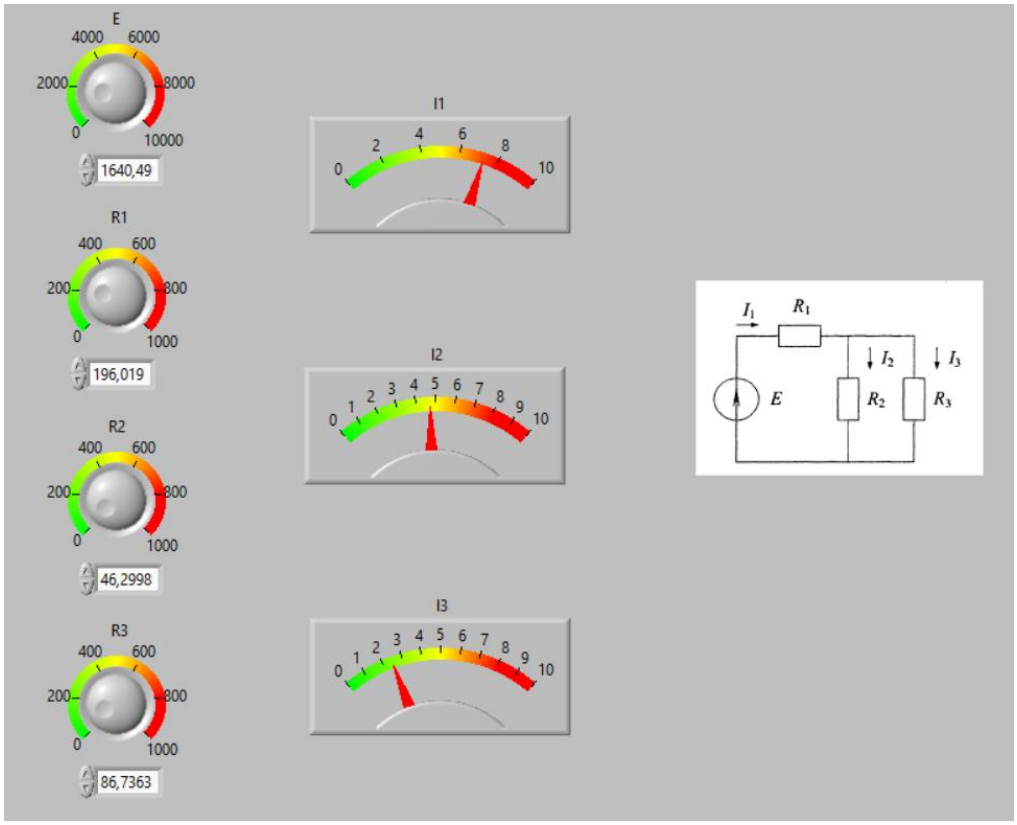


Рис.2.10. - Передняя панель с измененными элементами управления

- Элементы управления и индикаторы можно перемещать при помощи мыши и изменять их размеры «растягивая» их за уголки.
- После того, как редактирование завершено, необходимо сохранить программу, используя меню **File** → **Save**, или сочетание клавиш <Ctrl>+<s>.
- Теперь программа полностью готова, и ее можно запустить. Для этого нажмем кнопку выполнения программы ***(*Run*)***
- Вращая и передвигая ручки элементов управления при помощи мыши, убедимся в работоспособности программы.
- Чтобы изменить диапазон вводимых или выводимых значений, достаточно поместить указатель мыши на *крайнюю цифру* шкалы и, нажав левую кнопку мыши, ввести новое значение.

- Не останавливая программу, перейдем в окно блок схемы и нажмем кнопку **Highlight Execution** (подсветить выполнение)
- На проводниках появятся движущиеся точки, которые показывают, в каком порядке выполняется программа.
- *ОБЯЗАТЕЛЬНО отключите* режим **Highlight Execution**, иначе программа будет выполняться очень *медленно*, так как это демонстрационный режим, специально предназначенный для отладки программ.
- Перейдем на переднюю панель, и остановим выполнение программы, нажав кнопку **Stop**.
- Сохраним и запустим программу.

3 ЦИКЛЫ

3.1 Цикл For

Цикл **For** выполняет участок программы, расположенный в поддиаграмме цикла определенное количество раз. Выберите его в палитре **Functions** (Функций).

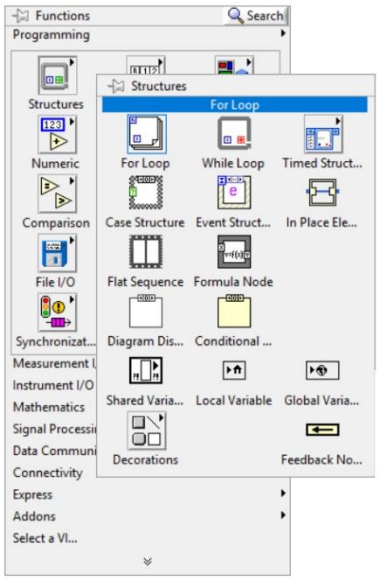


Рис.3.1. – Цикл For Loop

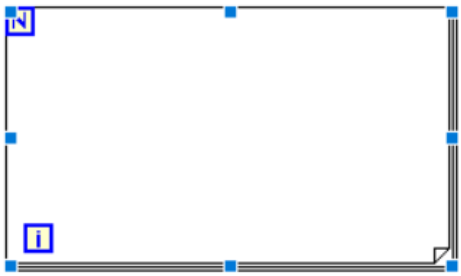


Рис.3.2. – Терминалы цикла

При этом изменится изображение курсора. Выделите область блок-диаграммы, в которой вы хотите разместить эту структуру. В процессе выделения не отпускайте кнопку мыши. Отмеченная область выделяется штриховым контуром. Выбрав область, отпустите кнопку мыши. Структура окажется на блок-диаграмме. Если в выделенной области находились другие объекты блок-диаграммы, они помещаются в тело цикла. Добавить новый объект внутрь структуры можно простым помещением его в область структуры. Кроме основной рамки цикла в нем присутствуют два терминала (рис.3.2):

[N] - терминал общего числа итераций, определяет общее число итераций;

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

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

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

- Непосредственно присоединить проводник к терминалу общего числа итераций [N].
- Присоединить к одному из входных терминалов массив. В этом случае структура сама разберет массив на элементы.

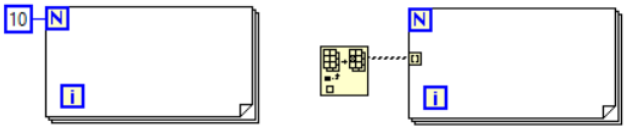


Рис 3.3. – Число итераций

Цикл на рис. 3.3 выполнится ровно 10 раз, терминал счетчика итераций [i] будет принимать значения от 0 до 9. Цикл на рисунке справа выполниться столько раз, сколько элементов содержится в массиве, другими словами по итерации для каждого элемента массива. В случае если к терминалу [N] ничего не присоединено и нет разбираемых (indexing) массивов, LabVIEW выдаст сообщение об ошибке.

3.2 Построение частотных характеристик последовательного колебательного контура

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

Рассматривается последовательное соединение катушки, индуктивность которой L, конденсатора ёмкостью C и резистора с сопротивлением

R, образующих последовательный колебательный контур Цепь подключена к источнику синусоидальной ЭДС с частотой f_0. Требуется построить зависимости тока в цепи и напряжений на катушке и конденсаторе от частоты при неизменном напряжении *U*.

Для этого выводятся формулы значений тока цепи и напряжений на катушке индуктивности и конденсаторе в цепи

$$Z = \sqrt{R^2 + (\omega * L - \frac{1}{\omega * C})} \tag{3.1}$$

$$I = \frac{U}{Z} \tag{3.2}$$

$$U_L = \omega * L * I \tag{3.3}$$

$$U_C = \frac{1}{\omega * C} \tag{3.4}$$

Далее требуется записать полученные выражения (3.1 – 3.4) в окно **Formula Node** аналогично тому, как это делали в прошлой главе. После этого подключаем индикаторы регулирования на вход вычислений, для того, чтобы задавать значания велечин. На выходе окна **Formula Node** ставим массив, для объединения данных и вывод на один график. Таким образом, мы получаем блок-схему, которая изображена на рис. 3.4. Лицевая панель ВП представлена на рис. 3.5.

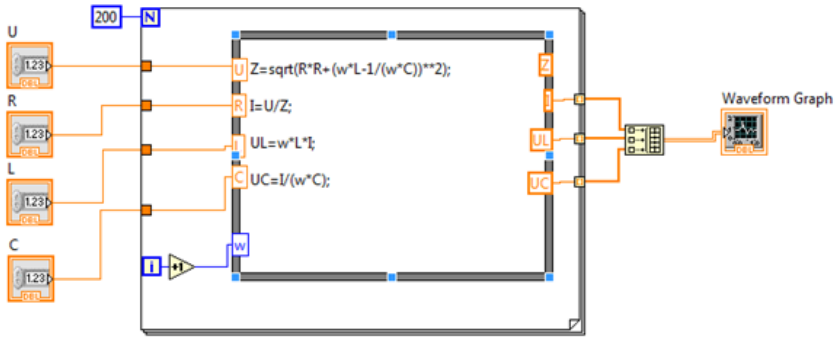


Рис. 3.4. – Блок-схема для расчёта характеристики тока и напряжений

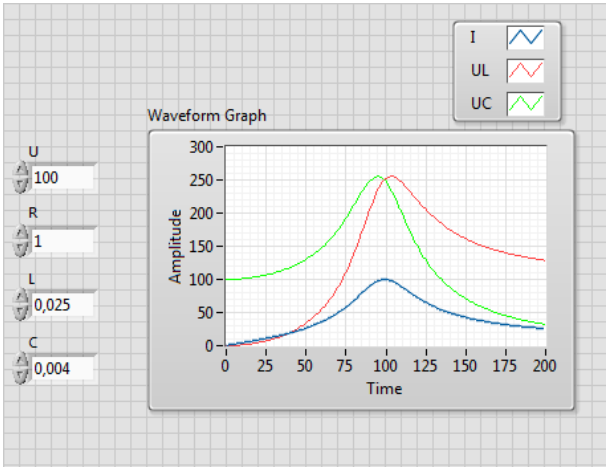


Рис.3.5. – Лицевая панель прибора

3.3 Цикл по условию (While)

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

Элементы цикла While:

 **Loop Iteration** (Повторение цикла). Имеет один контакт, расположенный *справа* – выход, значение которого соответствует количеству выполненных повторений цикла. *Синий цвет* терминала и контакта показывает, что выходное значение представлено целым числом.

Loop Condition (Условие цикла) терминал условия выхода из цикла. Имеет один контакт *слева* – вход. Он может работать в двух режимах:

 a) **Stop if True**

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

 б) **Continue if True**

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

Зеленый цвет терминала и контакта показывает, что формат данных, передаваемых ему, *булев* (логический).

3.4 Доступ к значениям предыдущих итераций цикла

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

Автоиндексирование в цикле по условию

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

Надо заметить, что в цикле **While** (в отличие от цикла **for**) по умолчанию автоиндексирование для массивов выключено, т.е. массив не разбирается на элементы, а поступает в каждую итерации целиком. Следует учесть, что цикл **While** может прекратить выполнение только при заранее определенном значении логической переменной, присоединенной к терминалу условия выхода из цикла, и не зависит от размера, разбираемого (**Indexing**) массива. Т.е. выполнение цикла может закончиться раньше или

позже, чем закончатся элементы массива. Очевидно, что в первом случае из массива будут извлечены не все элементы, а во втором, при попытке считывания после конца массива, в цикл будет поступать значение по умолчанию для данного типа (0 для чисел, пустой массив, пустая строка для массивов и строк соответственно и т.д.)

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

4 МАССИВЫ

Массив - это набор элементов определенной размерности. Массивы объединяют элементы одного типа данных. Элементами массива называют группу составляющих его объектов. Размерность массива - это совокупность столбцов (длина) и строк (высота), а также глубина массива. Массив может быть одномерным (вектор), двумерным (матрица) или многомерным, и содержать до 2^{31}-1 элементов в каждом направлении, насколько позволяет оперативная память. Данные, составляющие массив, могут быть любого типа: численные, логические или строковые. Массив также может содержать элементы графического представления данных и кластеры. Массивы удобно использовать при работе с группами данных одного типа и при накоплении данных после повторяющихся вычислений. Иными словами, массивы используются для работы с упорядоченным набором однотипных величин. Массив упрощает работу с ВП.

4.1 Создание массива элементов управления и индикации

Для создания массива элементов управления или индикации данных необходимо выбрать шаблон массива из палитры **Controls → Array, Matrix & Cluster** и поместить его на лицевую панель. Затем в шаблон массива поместить элемент управления или индикации данных (см. рис. 4.1). При этом терминал элемента на блок диаграмме приобретет цвет, соответствующий типу данных элементов массива.

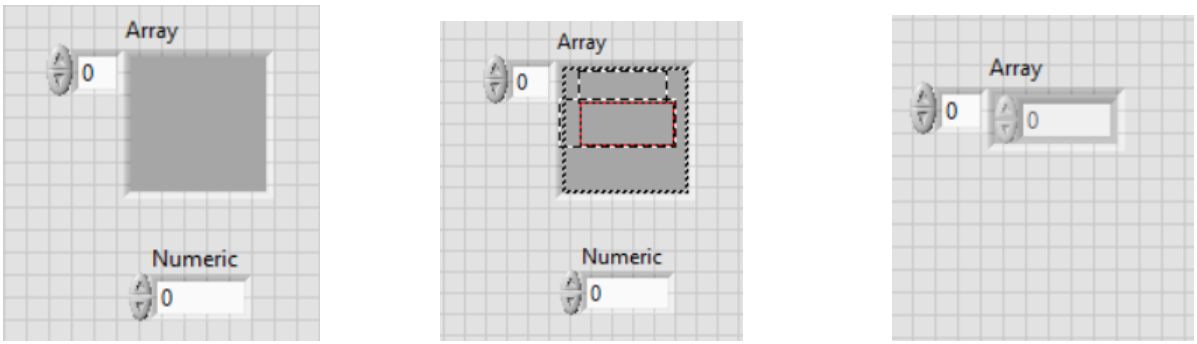


Рис.4.1. – Изменение индикации данных

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

На лицевой панели массив представляется двумя областями: зона индекса и зона видимости элементов. Сразу после создания массива виден

только один элемент. Для того, чтобы увидеть несколько элементов массива необходимо с помощью инструмента перемещение растянуть зону видимости элементов в горизонтальном или вертикальном направлении. На рис. 4.2 показан массив, состоящий из элементов управления **Controls → Boolean → Push Button**. Свойства элементов, входящих в массив, можно редактировать непосредственно в зоне видимости элементов, как если бы элемент управления или индикации находился вне массива. Например, для элементов управления изменить размер (рис. 4.3).

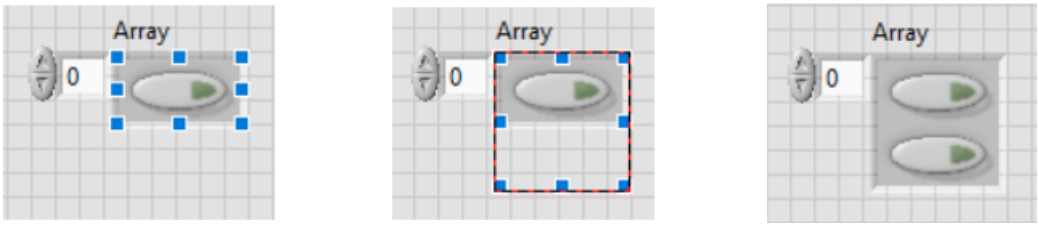


Рис.4.2. – Изменение числа элементов массива

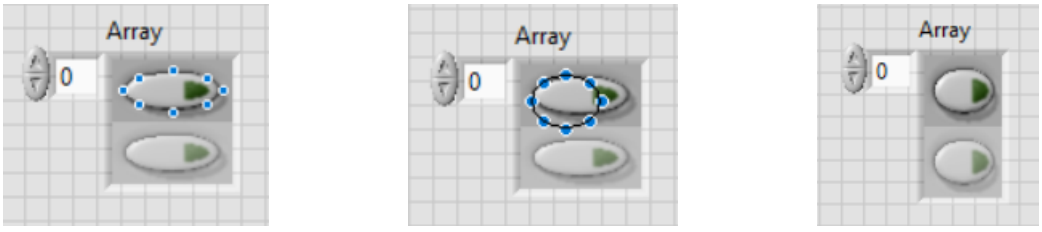


Рис.4.3. – Изменение размера массива

Обратите внимание на то, что у всех элементов массива различаются только их значения, а все свойства: размер, цвет, точность, представление и т.д. могут быть только одинаковыми. Изменяя свойство у одного из элементов массива, вы изменяете свойства всех элементов. В зоне индекса задается номер элемента массива, начиная с которого показываются элементы массива в зоне видимости элементов, т.е. индекс левого верхнего отображенного элемента. По умолчанию это значение 0. Это значит, что элементы массива показаны, начиная с нулевого элемента. Изменяя значение индекса можно наблюдать любой последовательный участок массива.

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

4.2 Двумерные массивы

Двумерный (2D) массив представляет собой прямоугольную таблицу (матрицу). Каждый элемент двухмерного массива характеризуется двумя индексами. Пример двухмерного массива размерностью 6 х 4 показан на рис. 4.4. Для увеличения размерности массива необходимо щелкнуть правой кнопкой мыши по элементу индекса и выбрать из контекстного меню пункт **Add Dimension**.

0	-1	-2	-3	-4	-5
1	0	-1	-2	-3	-4
2	1	0	-1	-2	-3
3	2	1	0	-1	-2

Рис.4.4. – Массив размерностью 6 х 4

Также можно использовать инструмент перемещение. Для этого надо просто изменить размер элемента индекса. Таким образом, можно увеличить размерность массива с одномерного до двумерного и выше, при этом зона видимости элементов становиться двумерной (рис. 4.5).

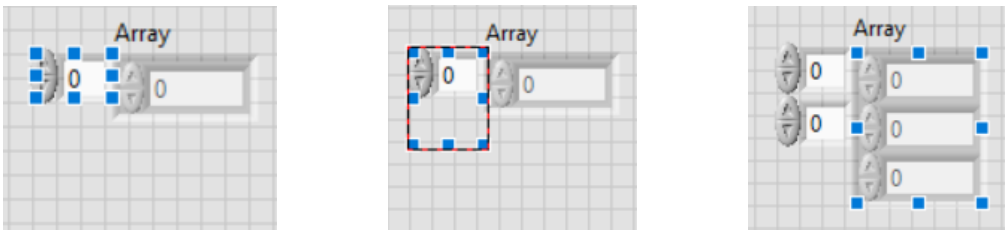


Рис.4.5. – Изменение размерности массива

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

5 ГРАФИЧЕСКОЕ ОТОБРАЖЕНИЕ ДАННЫХ

График диаграмм (**Waveform Chart**)- специальный элемент индикации в виде одного и более графиков. Для создания диаграмм достаточно соединить поле вывода скалярной величины с терминалом данных графика диаграмм. График диаграмм может отображать несколько графиков. Имеется два типа отображения данных кривые расположены друг под другом и все кривые на одном графике. График множества осциллограмм используется с целью экономии пространства на лицевой панели и для сравнения осциллограмм данных между собой. График осциллограмм и двухкоординатный график осциллограмм автоматически поддерживают режим отображения множества осциллограмм. Главные детали **Waveform Chart** отображены на рис.5.1.

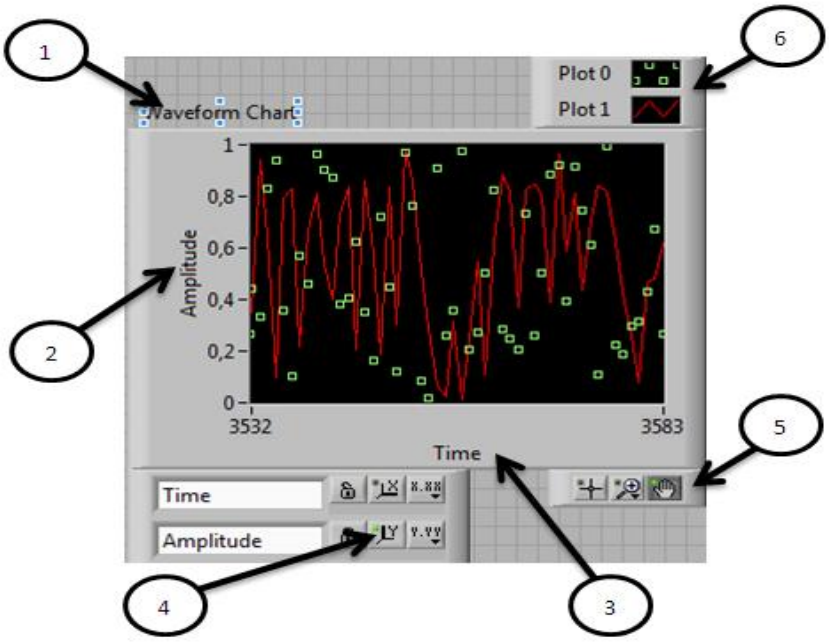


Рис.5.1. - Waveform Chart

1 - название(Label);
2 - шкала Y (Y scale);
3 - шкала X (X scale);
4 - панель управления шкалами (Scale legend);
5 - палитра инструментов для работы с графиком (Graph Palette);
6 - Панель управления графиком (Plot legend)

5.1 Одиночный график осциллограмм

Одиночный график осциллограмм работает с одномерными массивами и представляет данные массива в виде точек на графике, с приращением по оси X равным 1 и началом в точке x = 0. Графики также отображают кластеры, с установленным начальным значением *x,* Δ*x* и массивом данных по шкале *y*. Одиночный график показан на рис.5.2.

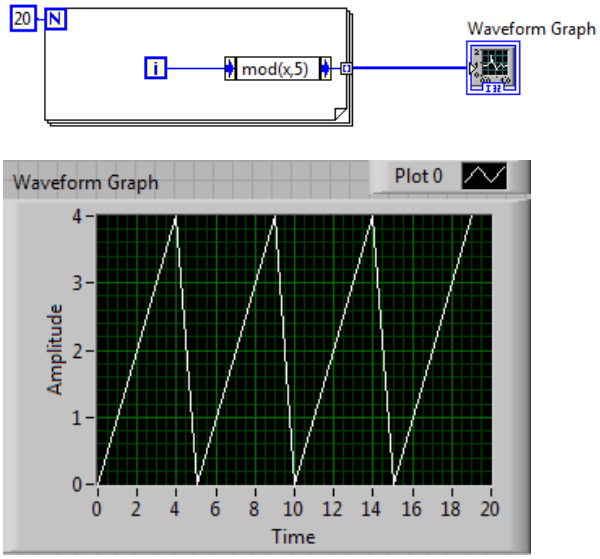


Рис. 5.2. - Одиночный график осциллограмм

5.2 График множества осциллограмм

График множества осциллограмм работает с двумерными массивами данных, где каждая строка массива есть одиночная осциллограмма данных и представляет данные массива в виде точек на графике, с приращением по оси X равным 1 и началом в точке x = 0. Графики множества осциллограмм так же отображают кластеры, состоящие из начального значения *x,* Δ*x* и двумерного массива данных по шкале *y*. График представляет данные по шкале *y* в виде точек с приращением Δ*x* по оси *x* и началом в точке *x = 0*. Пример показан на рис. 5.3.

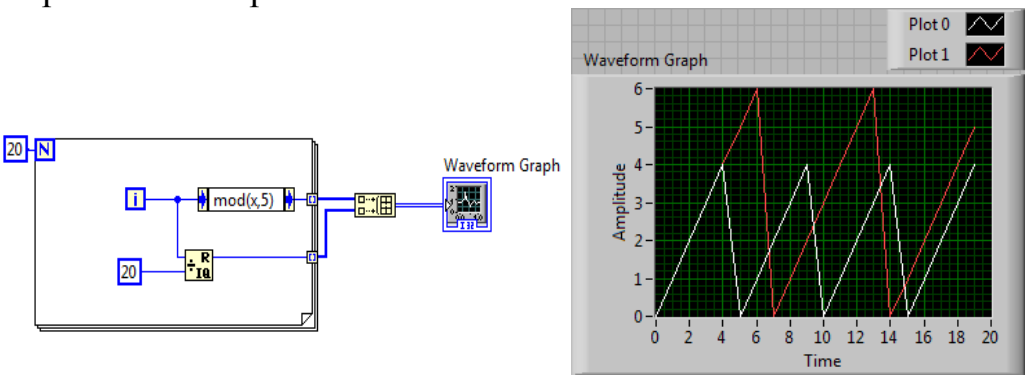


Рис. 5.3. - График множества осциллограмм

5.3 Одиночные двухкоординатные графики осциллограмм

Одиночный двухкоординатный график осциллограмм работает с кластерами, содержащими массивы *x* и *y*. Двухкоординатный график осциллограмм также воспринимает массивы точек, где каждая точка является кластером, содержащим значения по шкалам *x* и *y*. Обе показанные на рис. 5.4 блок диаграммы при выполнении выводят одинаковые графики. Обратите внимание на различие типов данных: кластер из двух одномерных массивов и массив из кластеров, содержащих пару численных значений.

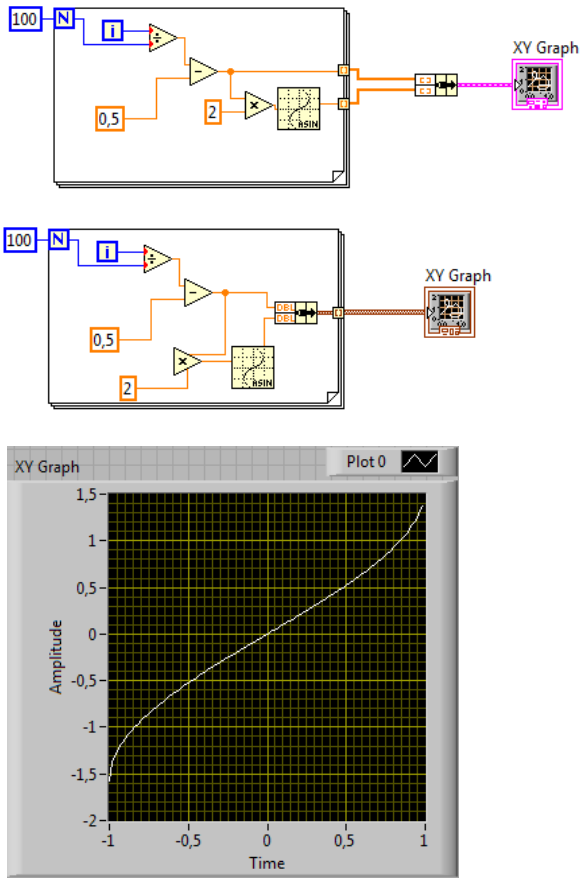


Рис.5.4. - Одиночные двухкоординатные графики осциллограмм

5.4 Двухкоординатные графики множества осциллограмм:

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

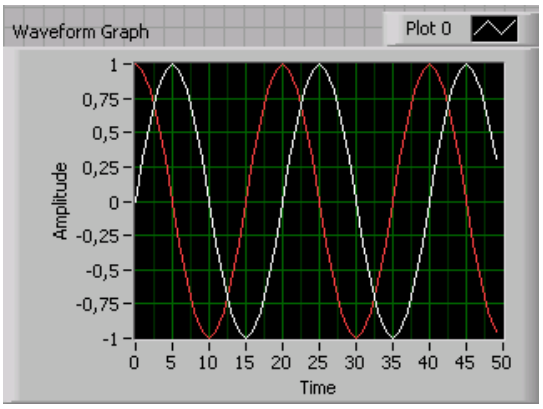


Рис. 5.5. - Двухкоординатные графики множества осциллограмм

5.6 Графики интенсивности

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

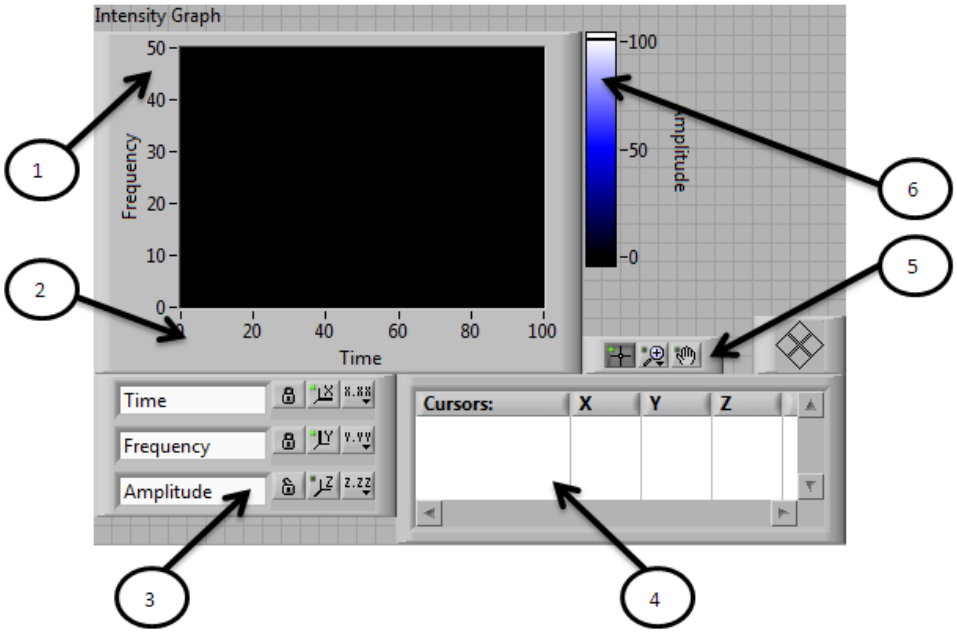


Рис. 5.6. - График интенсивности:

1 – шкала Y (Y scale); 2 - шкала X (X scale); 3 - панель управления шкалами (Scale legend); 4 - панель управления курсорами (Scale legend); 5 - палитра инструментов для работы с графиком (Graph Palette); 6 - шкала Z (цветовая шкала) (Z scale (color ramp))

5.7 Масштабирование графиков

Рассмотрим масштабирование графиков на примере генерирования сигнала с использованием блока **Express VI Simulate Signal**.

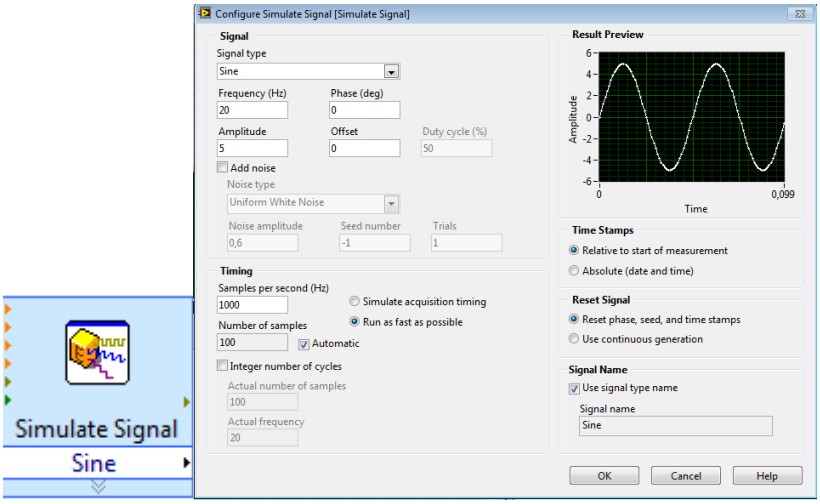


Рис. 5.7. – Настройки Simulate Signal

Данный блок имитирует различные виды сигналов (синусоидальный, прямоугольный, треугольный, пилообразный) и шумовой сигнал. При размещении иконки **Simulate Signal** на блок-диаграмме появляется диалоговое окно, с помощью которого можно задать все необходимые параметры сигнала и шума. В строке "**Signal type**" укажите форму сигнала, ниже его частоту, фазу, амплитуду и смещение по амплитуде. Поставив флажок "**Add noise**" можно к выбранному сигналу добавить шум, задав его тип и амплитуду. Также можно изменять количество отсчетов в секунду "**Samples per second**" и общее количество отсчетов "**Number of samples**". Если же Вам нужно изменить установленные ранее пара метры или значения, двойного нажатия левой кнопки мыши на иконке блока достаточно для повторного открытия диалогового окна.

Подключив **Waveform Graph**, увидим вполне ожидаемую картину: синусоиду (см. рисунок 5.8). При этом код исполняемой программы будет иметь очень простой и компактный вид.

Следует также отметить, что для изменения, например, частоты сигнала нет необходимости каждый раз обращаться к блок диаграмме и вызывать диалоговое окно конфигурации. Для этого нужно просто растянуть **Express VI** "вниз" и соединить регулятор (**Control**) со входом "**frequency**" нашего VI.

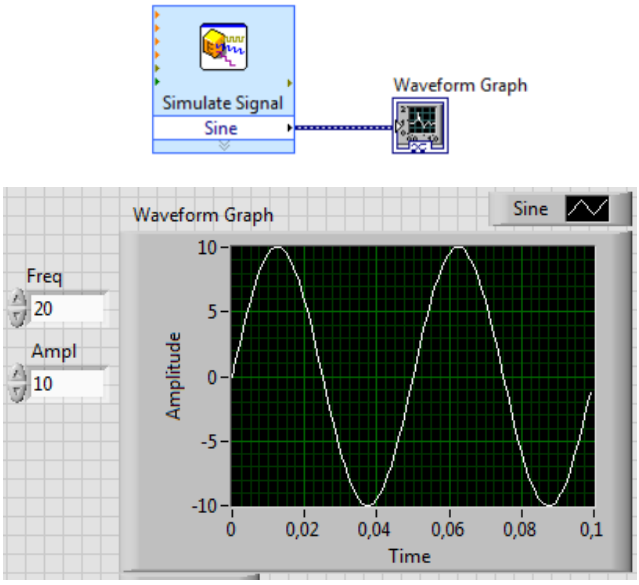


Рис. 5.8.

Для выбора определённого участка на графике надо включить **Graph Palette** (Для этого нажимаем правой кнопкой на графике в открывшем меню выбираем **Visible Items→ Graph Palette**). Рядом появится  Это палитра графиков. С графической палитрой вы можете перемещать курсоры, масштабировать и панорамировать дисплей. Графическая палитра появится со следующими кнопками, в порядке слева направо:

Масштаб - увеличение и уменьшение масштаба отображения. Используйте раскрывающееся меню, которое появляется при нажатии этой кнопки, чтобы выбрать метод масштабирования.

Курсор **Movement Tool** - перемещает курсоры на дисплее.

Инструмент «Панорамирование» (см. рисунок 5.9) отображает график и перемещает его по экрану.

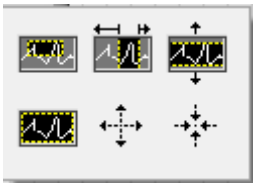
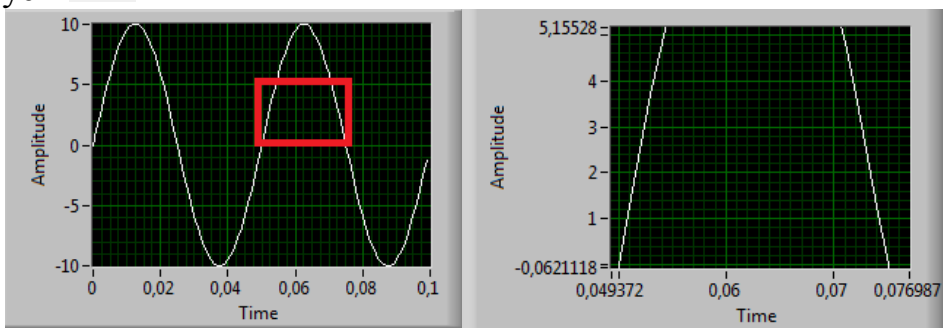


Рис. 5.9. – Инструмент «Панорамирование»

Примеры панорамирования приведены на рисунке 5.10. Область для панорамирования выделена на графиках слева красным цветом. На графиках справа отображены результаты выполнения операций. Для возвращения графика в исходное состояние используйте  .

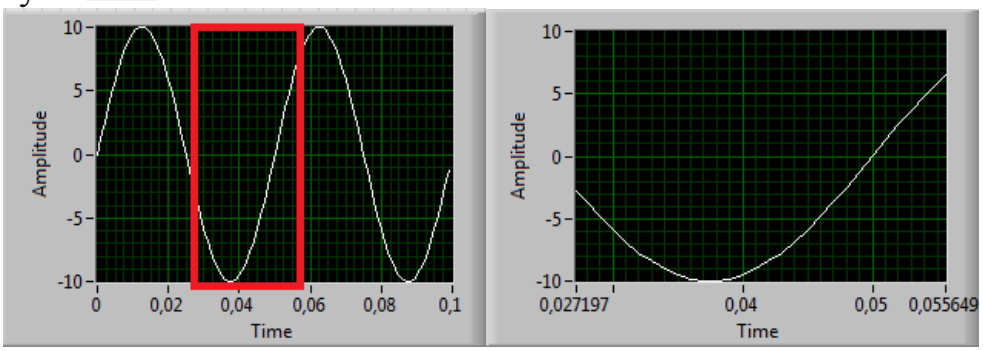
Функции  ,  используются для увеличения, уменьшения графика в определённой точки.

Используем 



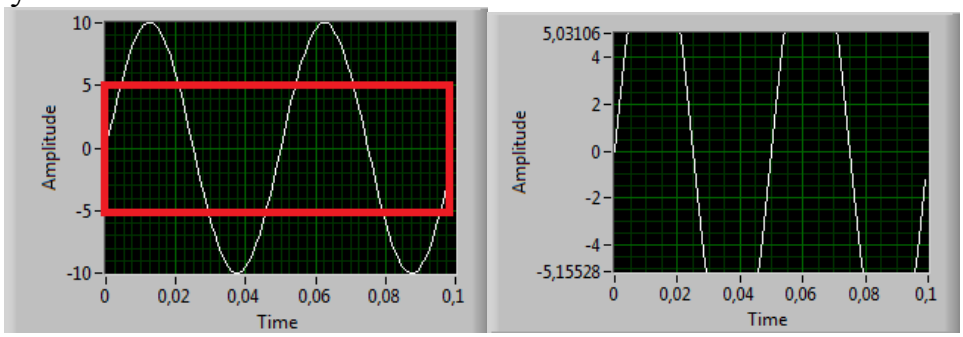
а

Используем 



б

Используем 



в

Рис. 5.10. – Выделение области график:
а – произвольная форма;
б – вертикальная форма;
в – горизонтальная форма

6 СТРУКТУРЫ

Алгоритм работы программы задается с помощью различных структур. Рассмотрим только основные структуры, а именно: циклы **For Loop**, структуру ветвления **Case**, и структуру **Formula Node**.

С помощью структур можно осуществить повторение отдельных частей программы, выполнение той или иной части программы в зависимости от какого-либо условия, выполнение программы в строго определенном порядке.

Некоторые структуры соответствуют циклу с фиксированным числом итераций (цикл **For**), циклу по условию (цикл **While**), оператору импликации (**if then else**). Вызвать любую структуру можно из палитры **Functions → Structures**.

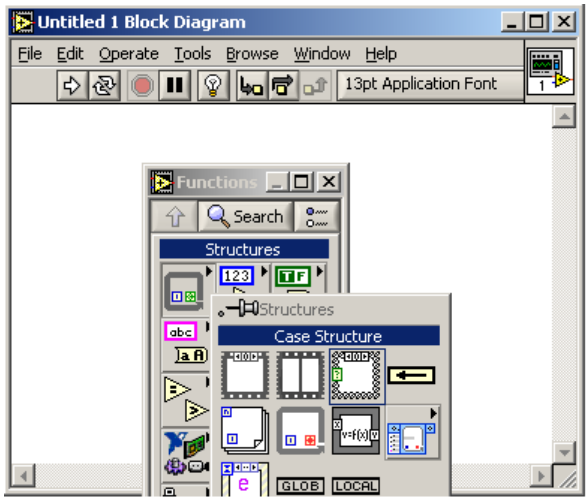


Рис.6.1. – Структура **Case**

Структура изображается в виде рамки, внутри которой содержится один или несколько участков программы. Каждый такой участок программы называется поддиаграммой. По краям структуры могут быть помещены входные и выходные терминалы. Можно наложить структуру на уже существующий участок программы или наоборот сначала поместить структуру, а за тем создавать элементы внутри нее. Контекстное меню структуры вызывается при нажатии правой кнопки мыши на рамке структуры. Общими для всех структур пунктами контекстного меню являются:
Auto Grow - если флажок установлен, то при помещении объектов во внутрь структуры, она будет соответственно увеличивать размер. **Remove**-удаление соответствующей структуры.

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

6.1 Цикл For Loop

Цикл For Loop служит для того, чтобы все объекты, размещенные внутри структуры, выполнялись циклически указанное число раз. Чтобы задать число повторений цикла, создайте константу привычным образом около символа N и установите число итераций. Путь для задания цикла **Programming → Structures → For Loop.** Положение цикла на панели инструментов показано на рис.6.2.

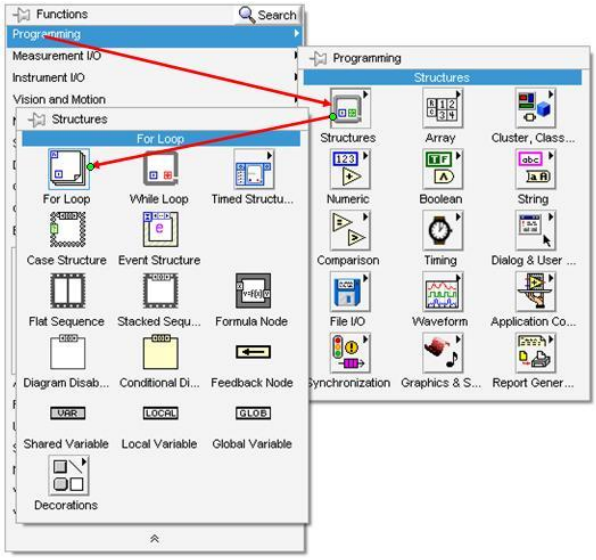


Рис.6.2. – Цикл **For loop**

6.2 Функция Select и принятие решений

Встречаются случаи, когда по ходу программы должно быть принято решение. Например, если происходит событие **A**, то необходимо сделать **B**, а если происходит **C** – то **D**. В этом случае необходимо использовать функцию **Select**, расположенную в палитре **Functions → Programming → Comparison**. Пример использования представлен на рисунке 6.3.

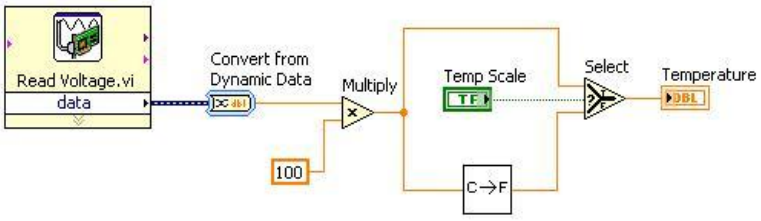


Рис.6.3. – Функция **Select**

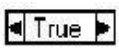
В зависимости от значения на логическом входе функция **Select** выбирает одно из двух значений. Если на логическом входе будет значение **TRUE**, то выходе функция выдаст значение, поданное на вход **t**, если же на логическом входе **FALSE**, то возвращается значение с поля **f.**

6.3 Структура Case

При необходимости принимать более сложные решения необходимо использовать структуру **Case.**

Case Structure аналогична операторам **case** или **if then else** в текстовых языках программирования. По умолчанию структура **Case** является логической и имеет два варианта - **ИСТИНА (TRUE)** и **ЛОЖЬ (FALSE)**, выбираемые с помощью терминала селектора структуры варианта. Структура автоматически преобразуется в числовую или строковую при подключении соответственно числовой или строковой переменной к терминалу селектора. В этом случае структура может иметь практически неограниченное количество вариантов, начиная с нулевого. С помощью строк **Add Case After** (Добавить вариант после) или **Add Case Before** (Добавить вариант перед) можно добавить новый вариант после или до текущего варианта. Одновременно можно наблюдать только один вариант (кадр) структуры. Переход между вариантами производится с помощью селектора структуры варианта, расположенного на верхней стороне рамки структуры или контекстного меню структуры. Для использования структуры **Case** необходимо отметить вариант по умолчанию (**Default**).

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



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

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

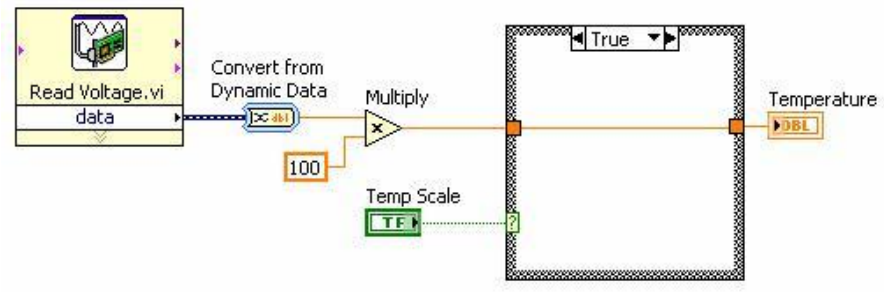


Рис.6.4. – Использование структуры **Case**

Для использования структуры **Case** необходимо отметить вариант по умолчанию**.** Вариант по умолчанию или поддиаграмма по умолчанию выполняется, если значение терминала варианта выходит за пределы диапазона или не существуют вариантов для возможных значений терминала варианта. Щелчок правой кнопки мыши на границе структуры **Case** позволяет добавлять, дублировать, перемещать и удалять варианты (поддиаграммы), а также отмечать вариант по умолчанию. В качестве примера использования структуры **Case** вместо функции **Select** приведена измененная блок-диаграмма **ВП Термометр**. На переднем плане структуры **Case** показан логический вариант **TRUE**. Определение варианта осуществляется либо выбором значения на селекторе структуры **Case**, либо вводом значения с помощью инструмента **ВВОД ТЕКСТА**

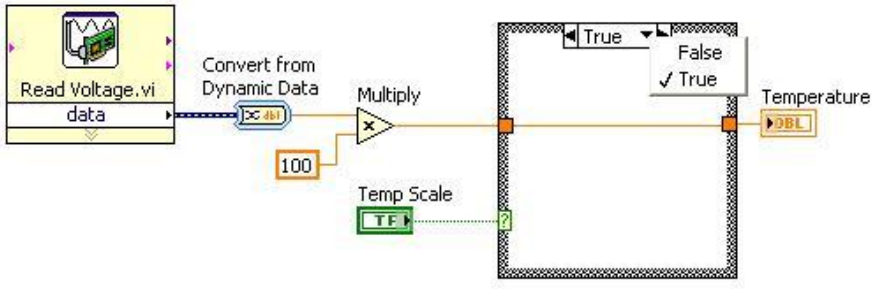


Рис.6.5. –Использование структуры **Case**

При выборе какого-либо варианта, он появляется на переднем плане, как показано на следующей блок-диаграмме.

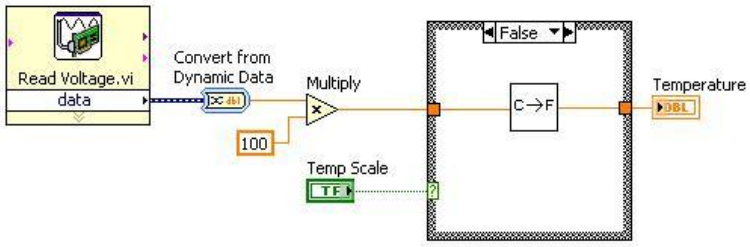


Рис.6.6. – Ввод текста в структуре case

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

6.4.1 Терминалы входа и выхода

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

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

6.4.2 Логическая структура Case

На рисунке 6.7 приведен пример логической структуры **Case**. Варианты структуры наложены друг на друга для упрощения иллюстрации.

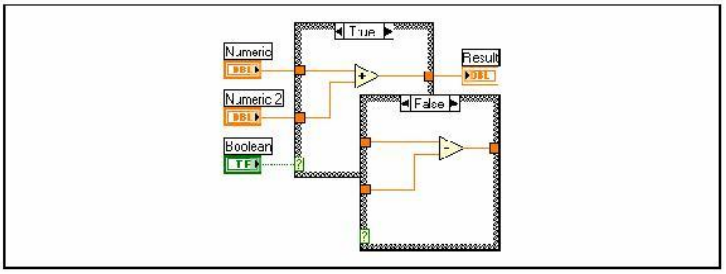


Рис.6.7. – Использование двух структур case

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

6.5 Структура Sequence

Sequence structure (структура последовательности) содержит одну или несколько поддиаграмм, которые выполняются последовательно одна за другой.

Flat Sequence и **Stacked Sequence** эти структуры практически ничем не отличаются. Когда нужно несколько кадров (до 6), тогда лучше использовать **Flat Sequence**, так как сразу видно всю блок диаграмму и она не скрыта за кадрами, да и передавать значения между кадрами проще. Когда вкладок больше или код внутри них очень объемный и не помещается на экран, тогда лучше использовать **Stacked Sequence**.

Используются структуры последовательности для управления порядком выполнения, когда естественная зависимость по данным отсутствует. Узел, который получает данные от другого узла, зависит от этого узла по данным и всегда выполняется после того, как этот узел завершит свое выполнение. Внутри каждого кадра структуры последовательности, как и на остальной блок-диаграмме, порядок выполнения узлов определяется зависимостью по данным.

Туннели на структурах **Stacked Sequence** могут иметь только один источник данных, в отличие от структур **Case**. Выход такого источника может поступать от любого кадра, но данные покидают структуру **Stacked Sequence** только тогда, когда будут выполнены все кадры, а не только этот конкретный кадр. Также как и в структуре **Case** данные от входных туннелей доступны во всех кадрах. Чтобы передать данные от одного кадра к любому другому кадру структуры **Stacked Sequence**, используйте терминал переменной последовательности (**sequence local terminal**), показанный слева. Стрелка, направленная наружу, появляется на терминале локальной переменной кадра, который содержит источник данных. Терминал на всех

последующих кадрах содержит входящую стрелку, показывающую, что этот терминал служит источником данных для этих кадров. На всех кадрах, предшествующих кадру с источником данных подключенному к терминалу локальной переменной, использовать этот терминал нельзя.

Путь к структуре отображен на рисунке 6.8: **Programming → Structures → Flat Sequence** и **Stacked Sequence.**

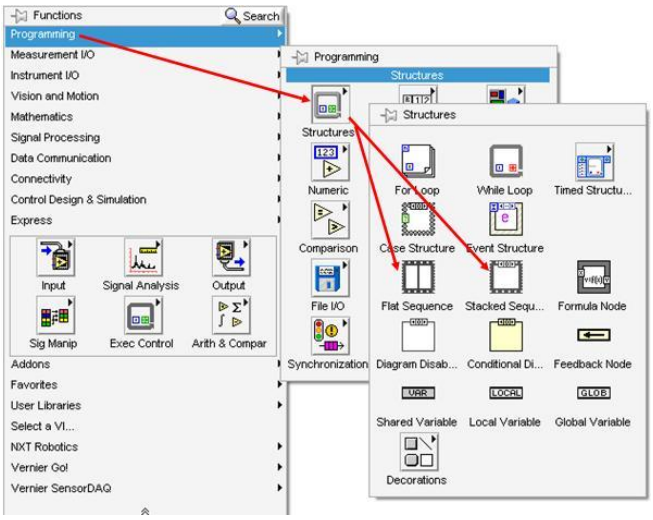


Рис.6.8. – Структура **Sequence**

6.6 Цикл Formula Node

Узел Формулы (**Formula Node**) используется для выполнения математических операций в текстовом виде на блок-диаграмме. Использовать узел Формулы удобно, когда выражения имеют много переменных, или они достаточно сложные. Для ускорения процесса создания алгоритма можно копировать и вставлять имеющиеся текстовые математические коды в узел Формулы вместо их воссоздания на блок-диаграмме. Создание терминалов входных и выходных данных узла Формулы осуществляется щелчком правой кнопки мыши по границе узла. В контекстном меню необходимо выбрать пункты **Add Input** или **Add Output**, а затем ввести переменные для входа и выхода**.** Далее вводится уравнение в рабочую область структуры. Каждое выражение должно заканчиваться разделителем **(;)**. Узел Формулы может также использоваться для принятия решений. На следующей блок-диаграмме показаны два эквивалентных способа применения операторов **if** – **then** в узле Формулы.

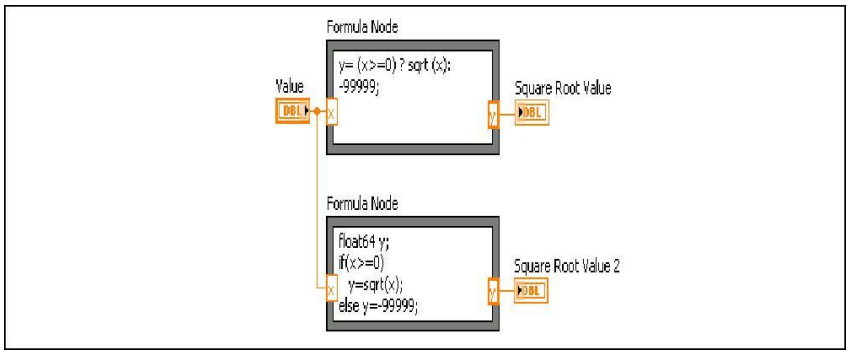


Рис.6.9. – Узел Формулы позволяет производить разнообразные математические операции

7 ФУНКЦИИ РАБОТЫ С ФАЙЛАМИ

Для чтения данных из файла и записи данных в файл в LabVIEW есть функции файлового ввода/вывода. Они расположены в палитре **Functions → Programming → File I/0** (см. рисунок 7.1) и предназначены для:

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

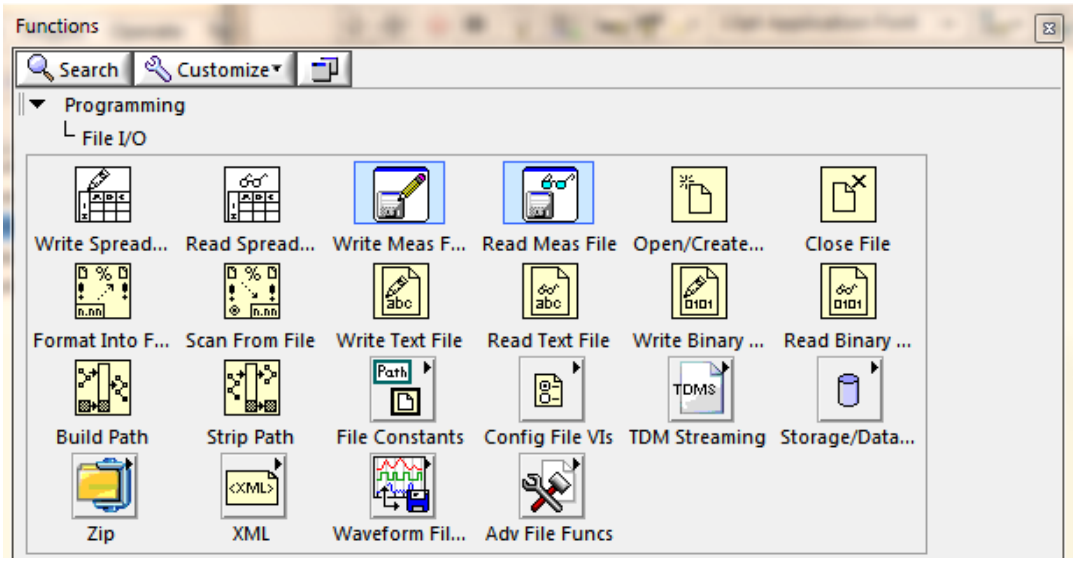


Рис.7.1. - Панель File I/0

7.1 Основы файлового ввода/вывода

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

1. Создание или открытие файла. Указание месторасположения существующего файла или пути для создания нового файла с помощью диалогового окна LabVIEW. После открытия файла LabVIEW создает ссылку (**refnum**) на него.

2. Выполнение операций чтения данных из файла или записи данных в файл.

3. Закрытие файла.

4. Обработка ошибок.

7.2 Функции файлового ввода/вывода низкого уровня

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

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

- **Open/Create/Replace File** - открывает, перезаписывает существующий файл, или создает новый. Если file path (путь размещения файла) не указан, ВП выводит на экран диалоговое окно, в котором можно создать новый или выбрать уже существующий файл. Структурная схема показана на рис. 7.2.

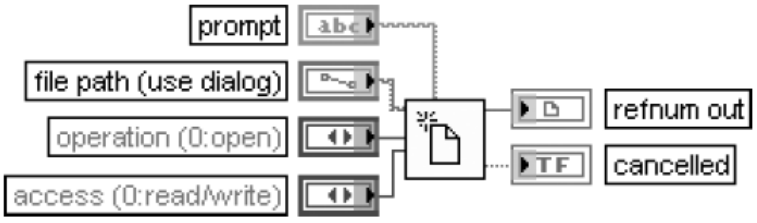


Рис.7.2. – Открытие, создание, и замена файла

- **Read Text File** - Функция считывает определенное число символов или строк из файла, представляющего поток байтов. По умолчанию эта функция считывает все символы из текстового файла. Считывание заданного количества символов, начиная с первого, производится с помощью входа подсчет (**count**). При установке отметки в строке **Read Lines** (читать строки) контекстного меню функции вход подсчет определяет количество считываемых строк. Установка значения –1 на этом входе определяет считывание всех символов в строке или всех строк текстового файла. Выход текст (**text**) содержит текст, считанный из файла. По умолчанию это строка, содержащая символы первой строки файла. Если подключить вход «подсчет», то на этот выход будет выводиться массив строк, считанных из файла. При удалении отметки «читать строки» на этот выход будут выводиться все символы, считываемые из файла. Функция преобразует все зависящие от платформы символы конца строки в аналогичные символы LabVIEW независимо от состояния строки Convert EOL контекстного меню функции. Структурная схема показана на рис.7.3.

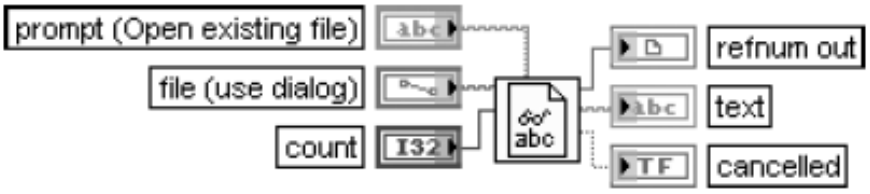


Рис.7.3. – Чтение файла

- **Write Text File** - Функция записывает строку символов (**string of characters**) или массив строк в файл. Если ко входу файл (**file**) подключен путь (**path**), то перед записью функция открывает или создает файл и заменяет его предыдущее содержимое. Если к этому входу подключена ссылка, то запись начинается с текущей позиции файла. Для того чтобы произвести дозапись к существующему файлу, необходимо установить позицию файла в его конец с помощью функции **Set File Position** (Установить позицию файла). Эту же функцию можно использовать для организации произвольного доступа. Вход **(prompt (Choose or enter file path))** (подсказка (выбрать или ввести путь к файлу)) представляет сообщение, которое появляется в заголовке файлового диалогового окна. По умолчанию это пустая строка. Вход **(file (use dialog))** (файл (использовать диалог)) может содержать ссылку или абсолютный путь к файлу. Если на вход файл подается путь, то функция открывает файл, заданный этим путем. Структурная схема показана на рис.7.4.

Рис.7.4. – Запись файла

- **Close File** - Функция закрывает открытый файл, определяемый (**refnum**) (ссылкой), и возвращает (path) (путь) к файлу, соответствующий ссылке. Ошибка ввода/вывода формируется только в этой функции, которая закрывает файл, несмотря на ошибки в предыдущих операциях. Это гарантирует корректное закрытие файлов. Структурная схема показана на рис.7.5.

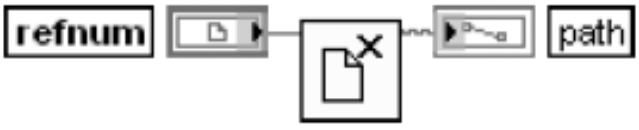


Рис.7.5. – Закрытие файла

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

Для обработки ошибок используются подпрограммы, например, **Simple Error Handler VI** (ВП Простой обработчик ошибок), расположенный в палитре **Functions → Time & Dialog.**

7.3 Сохранение данных в файле

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

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

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

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

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

7.4 Форматирование строк таблицы символов

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

ными платформами следует использовать константу **end of line constant**, расположенную в палитре **Functions → Programming → String**. Константа осуществляет перевод строки.

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

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

7.5 Функции файлового ввода/вывода высокого уровня

Функции файлового предназначены для выполнения основных операций по вводу /выводу данных. Таких как:

- **Write to Spreadsheet File** - преобразует 2D или 1D массив числовых данных одинарной точности в текстовую строку и записывает строку в новый или добавляет в уже существующий файл. Этот ВП используется для создания текстовых файлов, читаемых большинством текстовых редакторов. Структурная схема показана на рис.7.6.

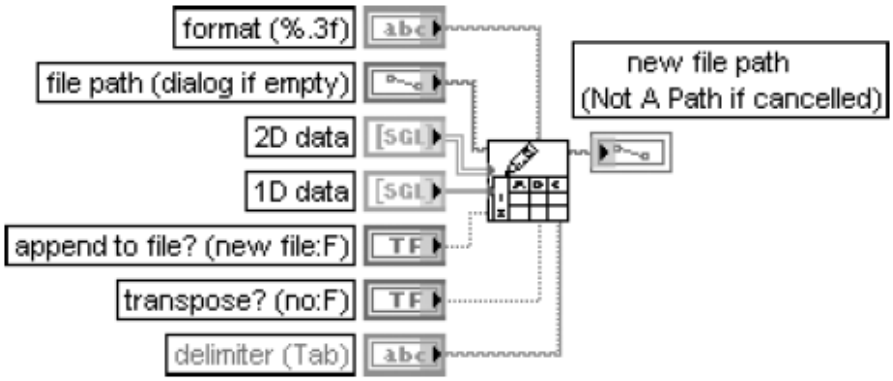


Рис.7.6. – Запись файла в табличном формате

- **Read From Spreadsheet File** - считывает определенное число строк от начального смещения start of read offset и преобразует данные в 2D массив числовых данных одинарной точности. Этот ВП можно использовать для чтения таблицы символов, сохраненной в текстовом формате. Структурная схема показана на рис.7.7.

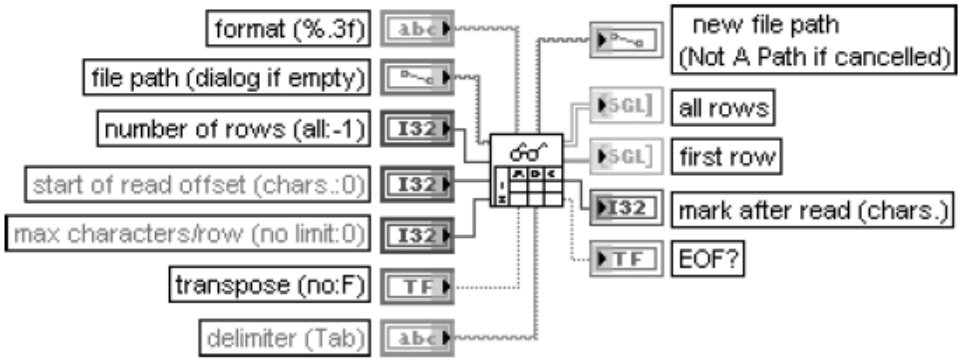


Рис.7.7. – Чтение файла табличного формата

8 ВНЕШНИЙ ВИД ПРОГРАММЫ

8.1 Окно редактирования внешнего вида элементов лицевой панели

Если по каким-либо причинам стандартные элементы управления и индикации не устраивают, их внешний вид можно изменить. **LabVIEW** позволяет. практически до неузнаваемости изменить внешний вид элементов на передней панели приложения при условии, что сохраняется их общая функциональность.

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

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

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

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

Рассмотрим режим настройки подробнее, так как именно он позволят радикально менять внешний вид объектов лицевой панели. При пере-

ходе в этот режим элемент управления или индикации разбивается на составные части и их можно редактировать отдельно.

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

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

8.2 Режим настройки

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

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

• **Copy to Clipboard** - копирует отображаемую в данный момент картинку в буфер обмена. Этот пункт можно использовать, чтобы копировать изображение отдельных частей из одного элемента управления или индикации в другой.

• **Import Picture** - заменяет текущую картинку на ту, что находится в буфере обмена.

• **Import Picture at Same Size** - то же самое, что и в предыдущем пункте, но размер картинки из буфера автоматически погоняется под размер текущей картинки.

• **Revert** - все картинки заменяются на те, что были до того, как вы воспользовались пунктом контекстного меню **Advanced → Customize** на лицевой панели ВП (если вы попали в окно редактирования другим способом, этот пункт меню будет неактивен).

• **Original Size** - восстанавливает изначальные размеры изображения. Используйте этот пункт, если вы растягивали или сжимали рисунок.

8.3 Режим редактирования

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

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

8.4 Определение типа

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

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

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

Диалоговое окно **VI Library Manager** необходимо для того, чтобы вы могли из объединенных общей функциональностью файлов создать один файл - библиотеку. Оно вызывается выбором пункта меню **Tools → VI Library Manager**.

В этом диалоговом окне вы можете создавать библиотеки ВП (файлы lib), работать с файлами в уже существующих библиотеках, копировать, переименовывать, удалять. Окно похоже на классический файловый менеджер с двумя полями, в которых показан список директорий и файлов. При этом файлы lib воспринимаются как директории, в которые можно заходить и работать с файлами в них. После того, как файл выбран, активизируются кнопки, которые позволяют совершать следующие действия:

• **Copy** — копирует выбранный элемент в выбранную директорию второго поля;

• **Rename** — переименовывает выбранный файл или директорию;

• **Delete** — удаляет выбранный элемент из списка. У вас не получится удалить директорию, если она не является пустой;

• **New** — создает новую директорию или файл lib;

• **Convert LLBs to Dirs** — преобразует выделенную библиотеку lib в директорию. Если выбранным элементом является директория, то при нажатии на эту кнопку в директории находятся все lib файлы. После этого можно выбирать, как и какие lib файлы преобразовывать. Новая директория создается в том же месте, где расположен исходный lib файл;

• **Convert Dirs to LLBs** — преобразует выбранную директорию в файл lib;

• **Check Filenames** — производит в директории или библиотеке ВП поиск имен файлов, содержащих символы (:, \, /, ? , * , , , или). Проверяет, чтобы имена файлов содержали менее 31 символа. Опция **Check Filenames** проверяет файлы, в том числе и внутри библиотек lib. Файлы с именами, не содержащими указанных символов и содержащими менее 31 символа, могут быть перенесены на другие платформы. Этот инструмент позволяет обнаружить потенциальные проблемы при перемещении файлов из библиотек ВП.

•**Done** — выходит из диалогового окна **VI Library Manager.**

9 СОЗДАНИЕ АВТОНОМНОГО ПРИЛОЖЕНИЯ В СРЕДЕ LABVIEW

Автономное приложение создается при помощи инструмента Application Builder.

9.1 Подготовка программы к переводу в автономное приложение

Порядок подготовки программы к переводу в автономное приложение:

1) Запускается среда разработки Labview и открывается программа, переводимая в автономное приложение

2) Программа обязательно должна содержать кнопку выхода. В случае отсутствия, ее необходимо сделать.

3) Для настройки выбирается **File/VI Properties**. В категории **General** рисуется иконка.

4) Выбирается категория **Window Appearance** (см. рис.9.1). Чтобы наше приложение, было похоже на что то отдельное от Labview, выбирается пункт (**Custom**) и нажимается кнопка (**Customize**).

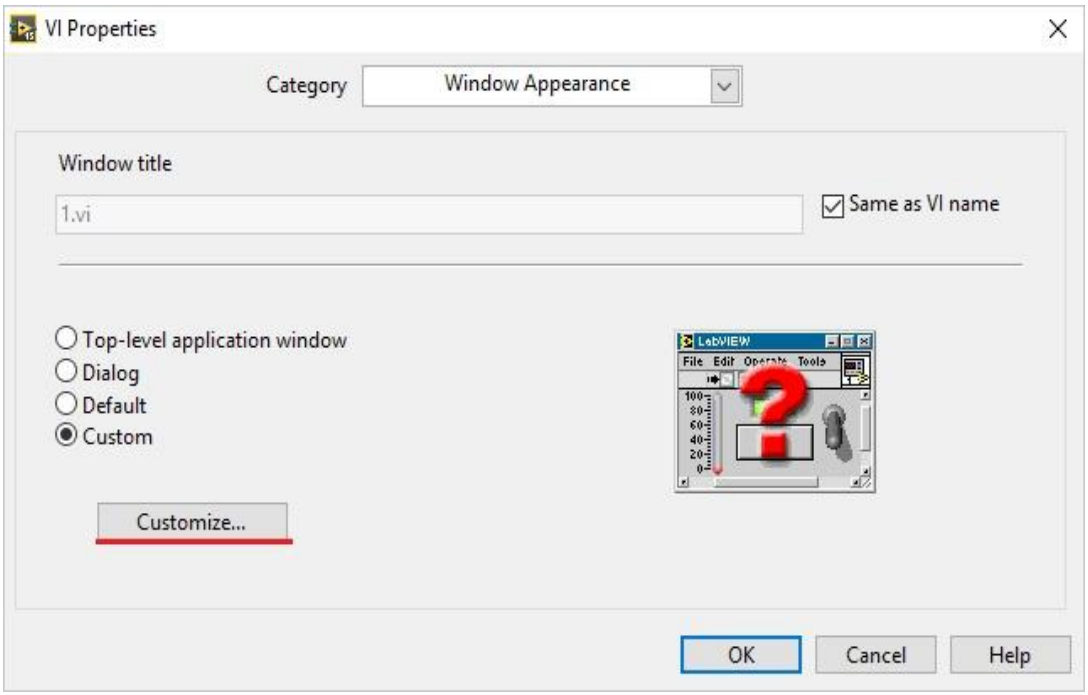


Рис.9.1.

Далее в открывшемся окне (Customize), убираются ненужные галочки с тулбаров.

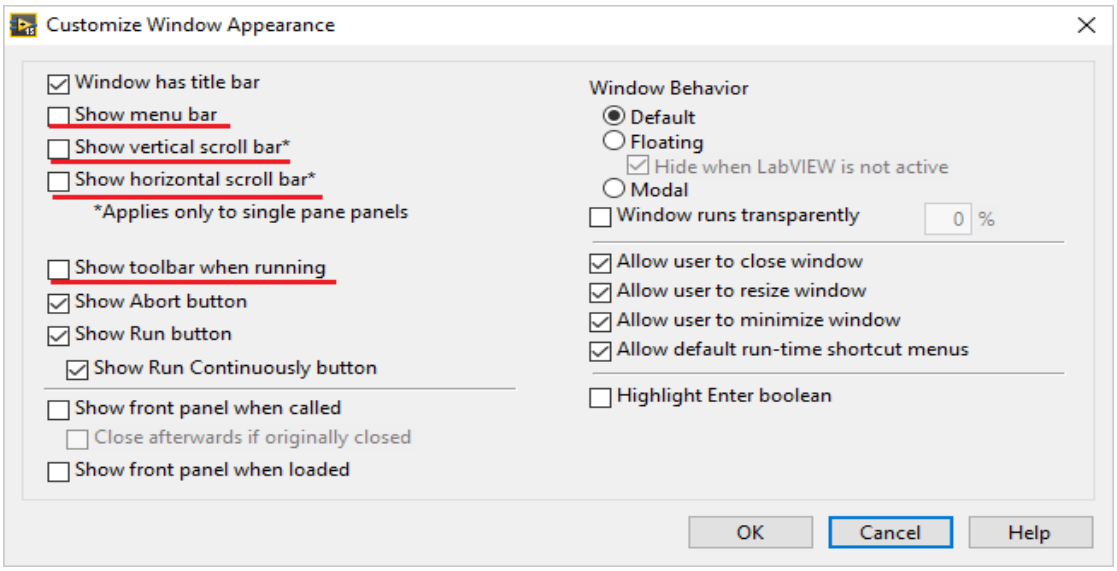


Рис.9.2.

5) Выбирается категория (**Execution**), ставится галочка справа, чтобы будующее приложение запускалось автоматически при открытии.

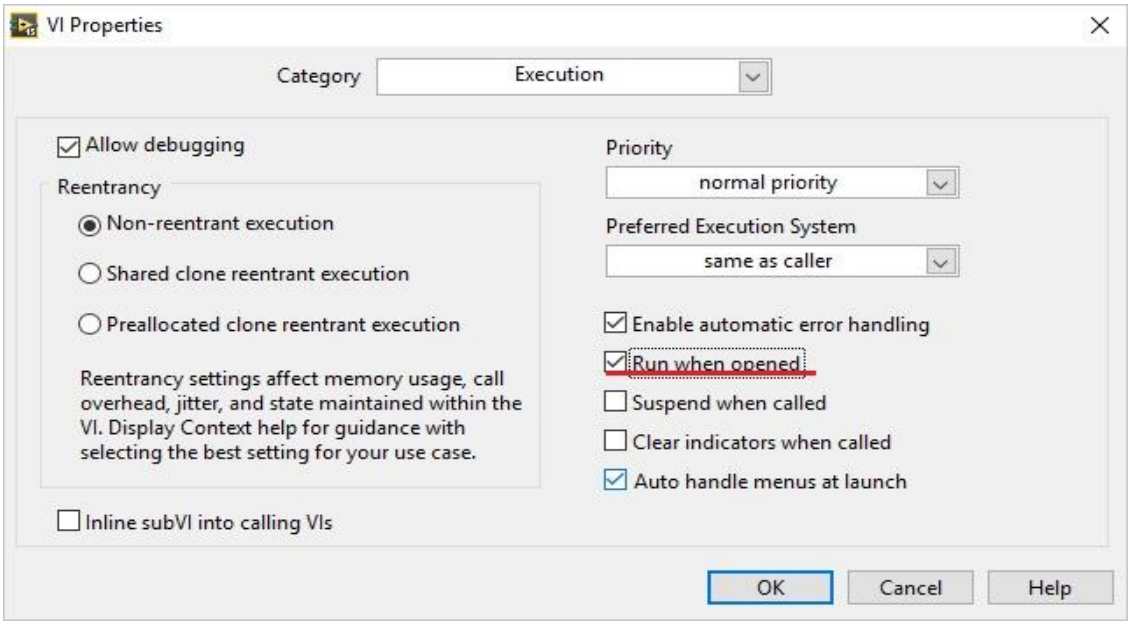


Рис.9.3.

9.2 Знакомство с Application builder

1) Запускаем среду разработки Labview (см. рис.9.4).

Рис.9.4.

2) Создаем новый проект, нажимая кнопку **blank project. О**ткрывается панель (см. рис.9.5).

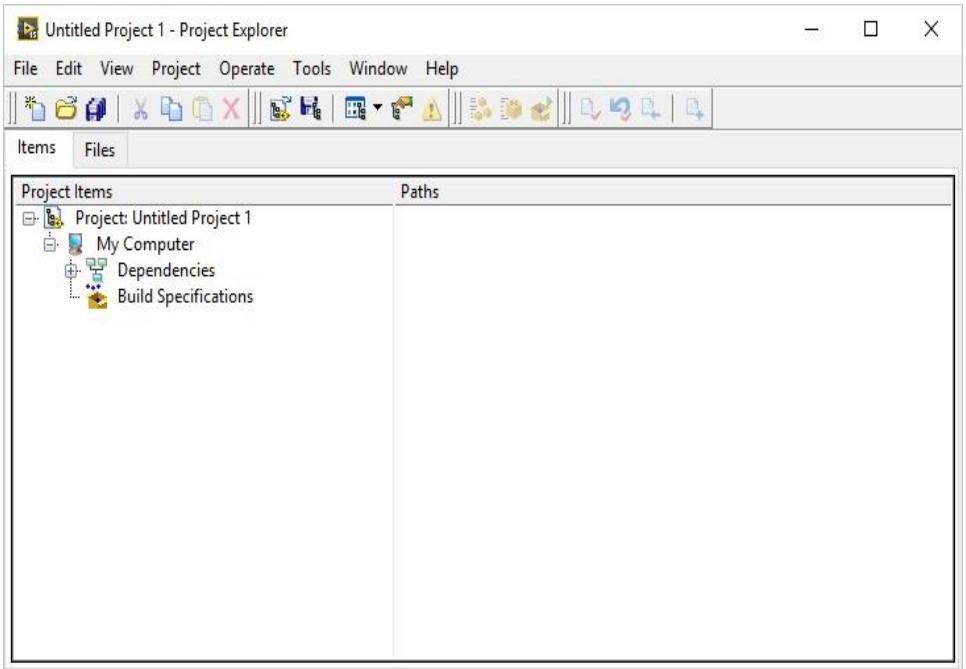


Рис.9.5.

3) Запускаем программу для преобразования в автономное приложение (см. рис. 9.6).

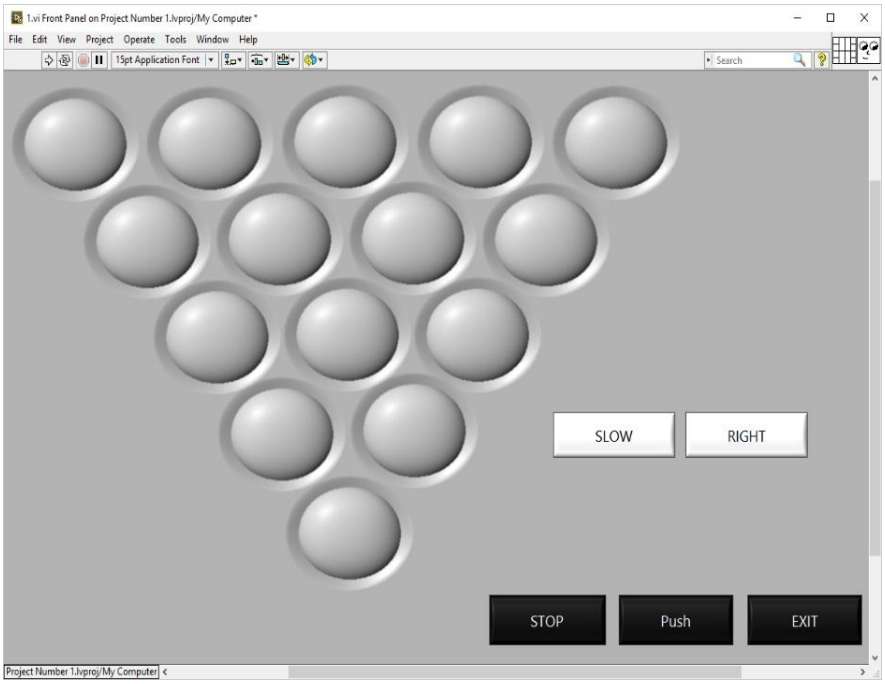


Рис.9.6.

4) Выбираем вкладку **tools** на верхней панели. После чего появляется окно, где выбираем **Build Application (EXE) from VI**. На панели с рис. 9.5 появилась запущенная программа .

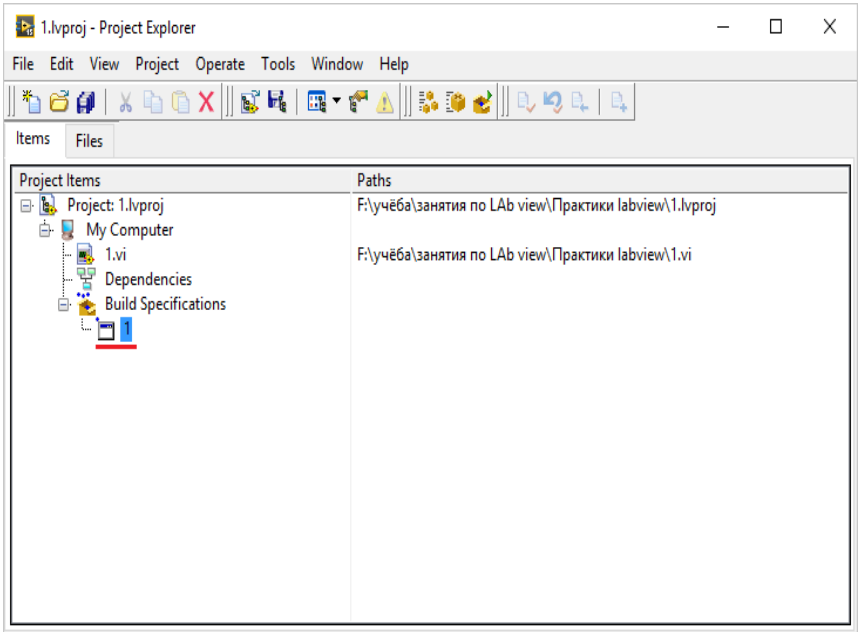


Рис. 9.7.

Если вы случайно закрыли окно (**Application builder**) можно всегда его открыть, не создавая кучу не нужных копий.

9.3 Работа с Application builder

Работа с **Application builder** состоит в задании свойств (**Properties**) будущего автономного приложения (см. рис.9.8).

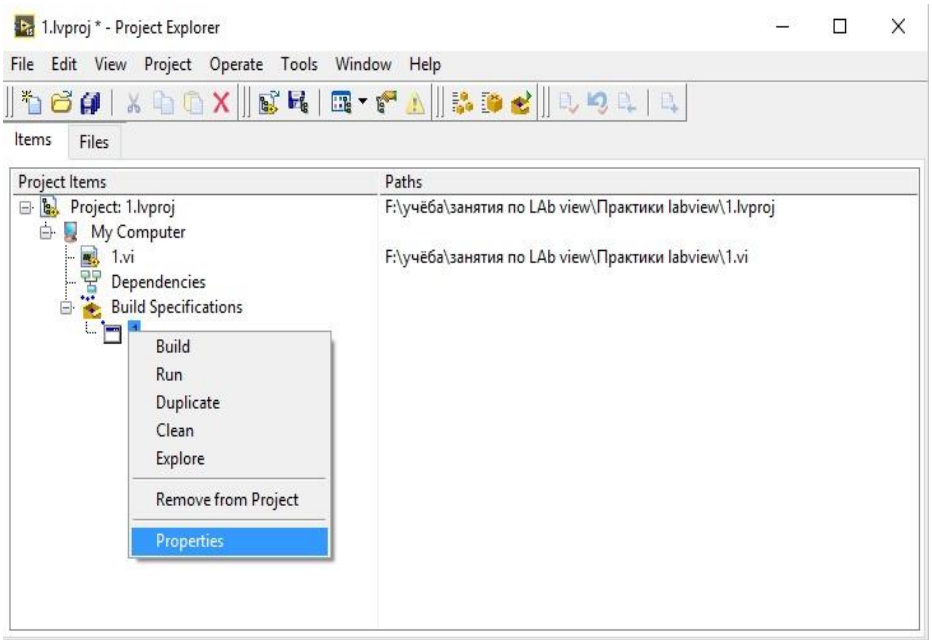


Рис.9.8.

Уникальная иконка будущего приложения задается в свойстве Icon (см. рис. 9.9).

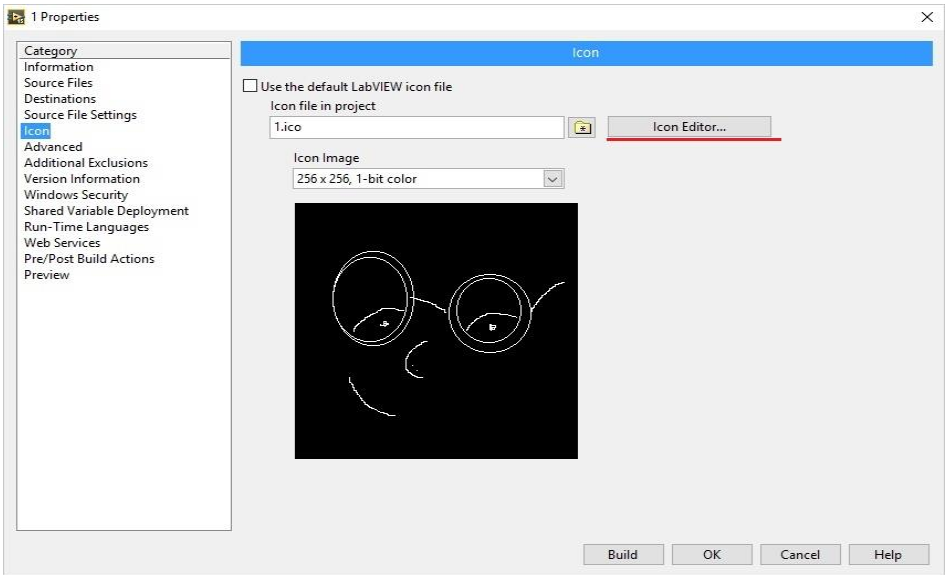


Рис.9.9.

Перечень свойств создаваемого автономного приложения представлен на рис. 9.10÷9.17.

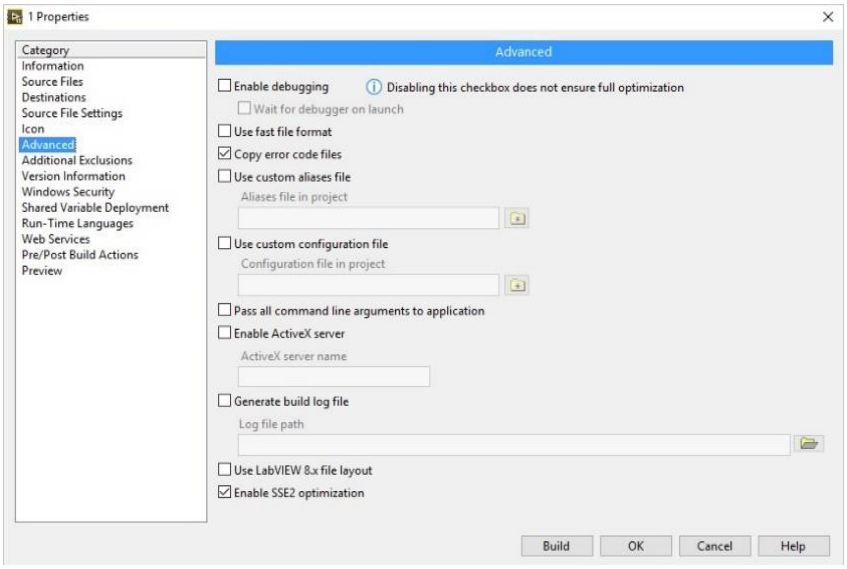


Рис.9.10.

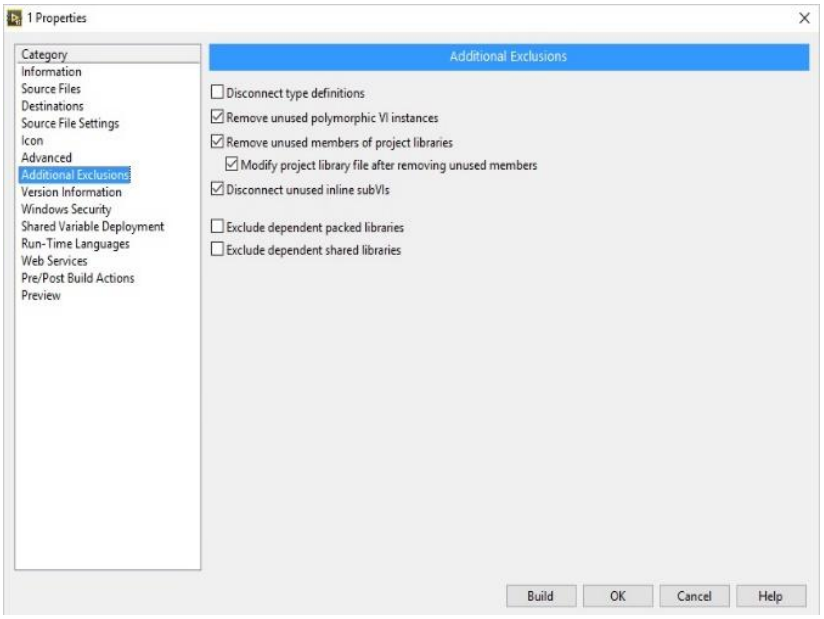


Рис.9.11.

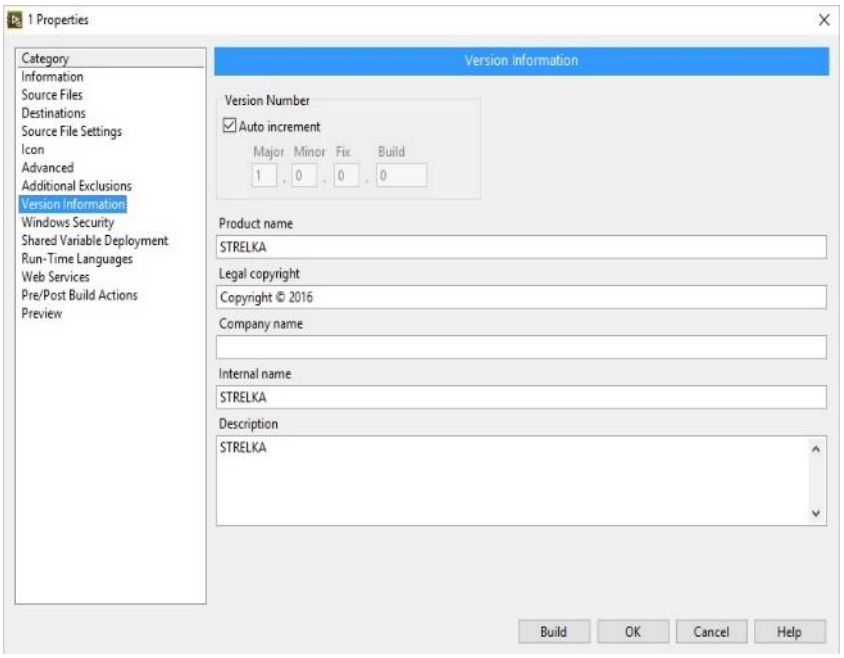


Рис.10.12.

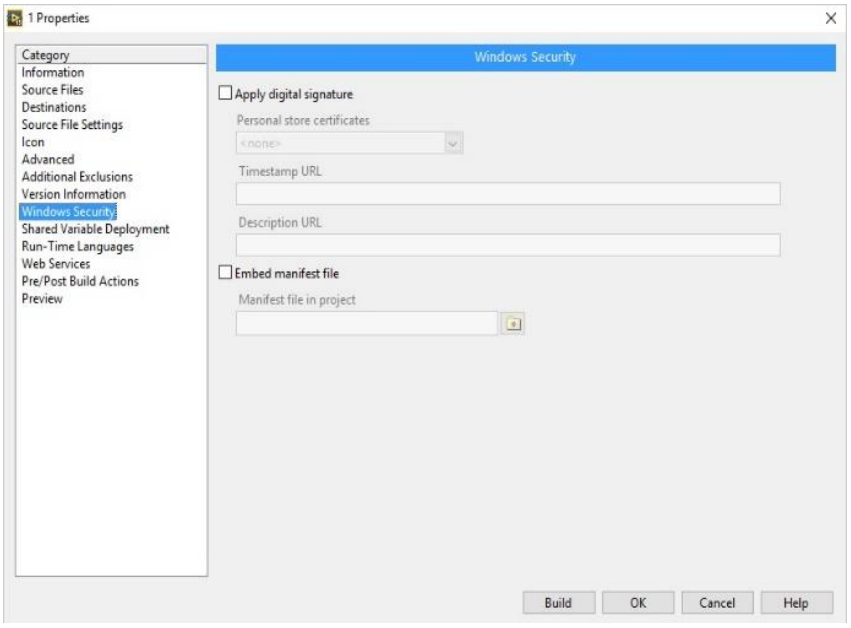


Рис.9.13.

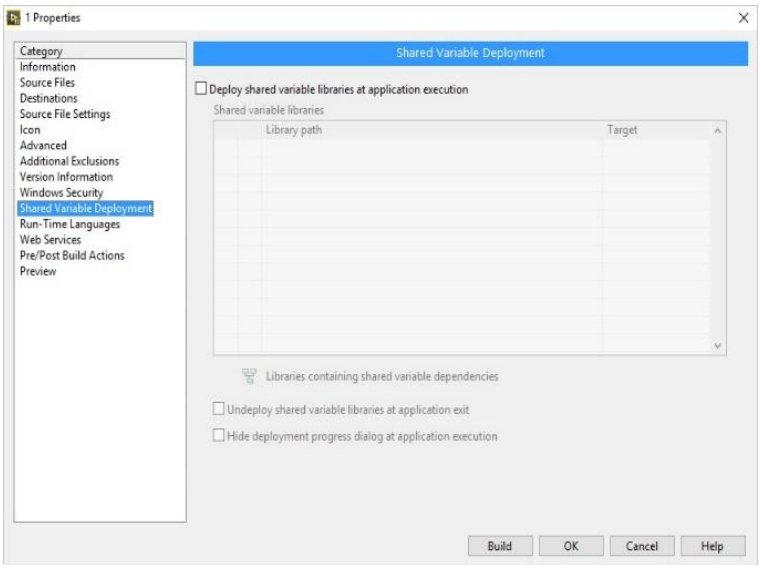


Рис.9.14.

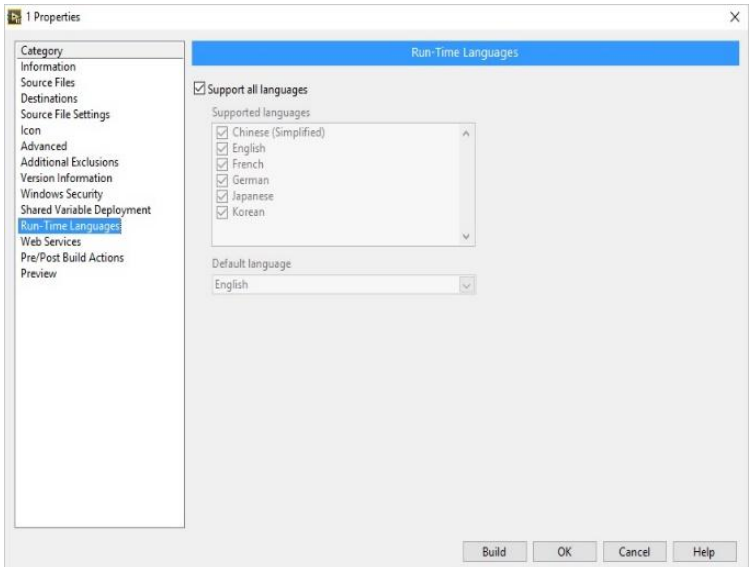


Рис.9.15.

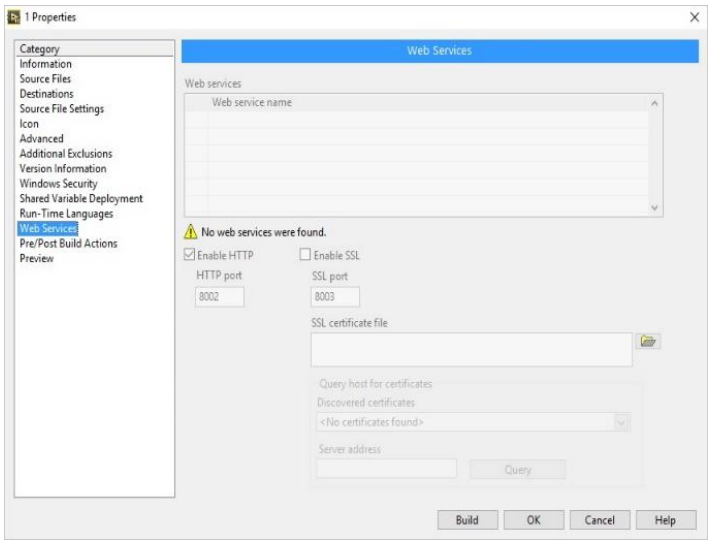


Рис.9.16.

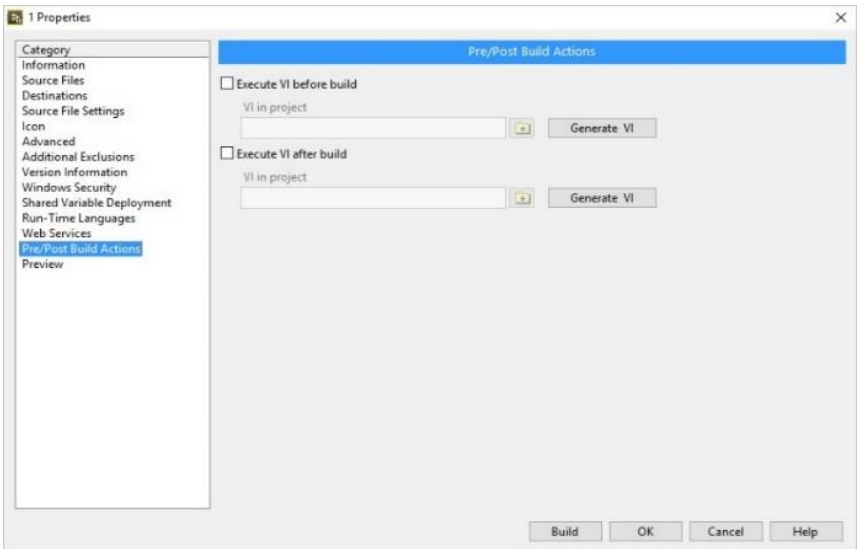


Рис.9.17.

Процесс создания приложения заканчивается нажатием кнопки (**build**), расположенную внизу окна настроек (см. рис. 9.18).

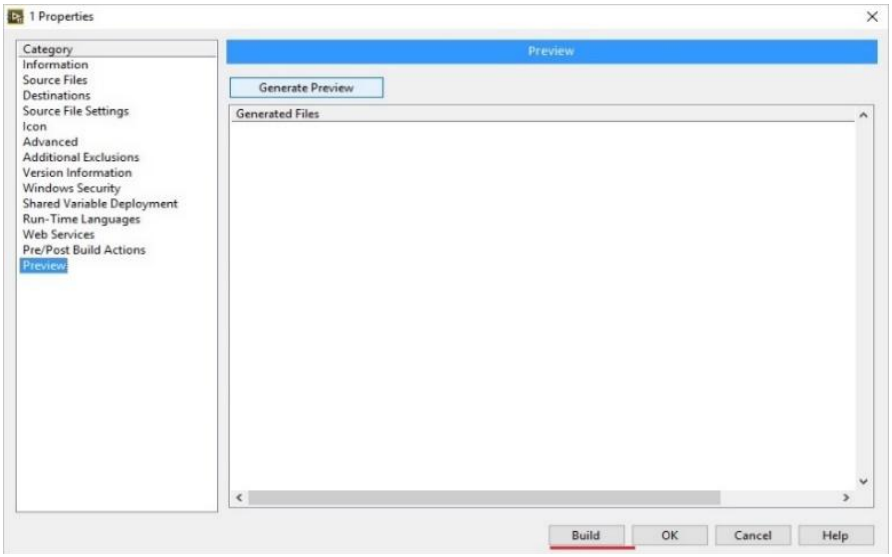


Рис.9.18.

После завершения процесса (**build**), в папке проекта должно появиться самостоятельное ни от чего не зависящие приложение(см. рис. 9.19 и 9.20), которое при открытии сразу же запускается, а при нажатии кнопки **Exit** закрывается.

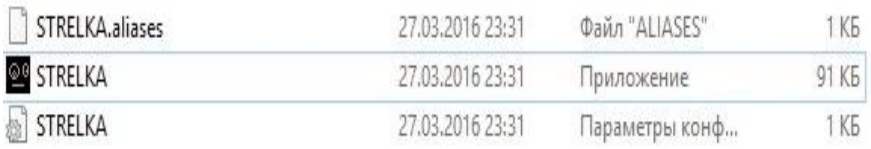


Рис.9.19.

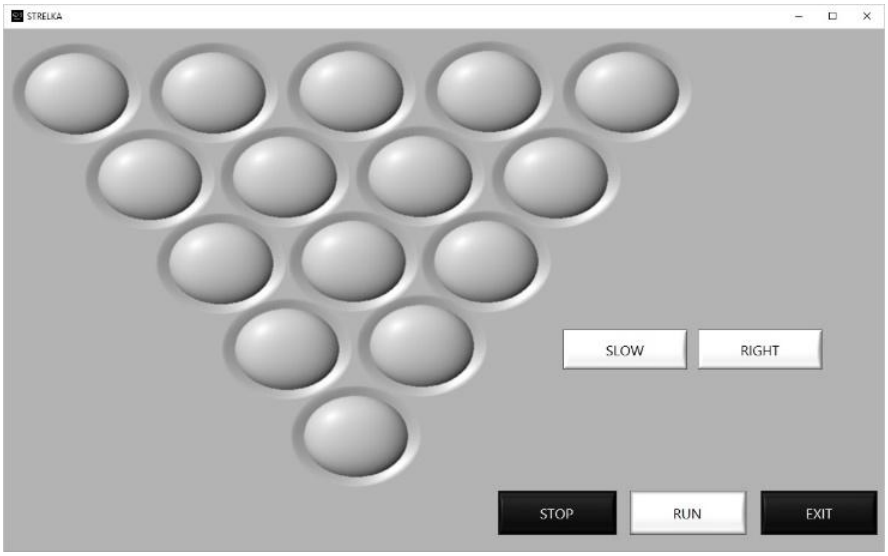


Рис.9.20.

ЛИТЕРАТУРА

1. Автоматизация физических исследований и эксперимен-та:компьютерные измерения и виртуальные приборы на основе LabVIEW 7 : (30 лекций) : учеб. пособие для вузов. - М. : ДМК Пресс, 2005. - 264с.
2. Батоврин В.К. LabVIEW:практикум по электронике и микропроцессорной технике. - М. : ДМК Пресс, 2005. – 180
3. Сирота А.А. Компьютерное моделирование и оценка эффективности сложных систем : учеб. пособие для вузов. - М. : Техносфера, 2006. - 279с.
4. Колесов Ю.Б. Моделирование систем : практикум по компьютерному моделированию : учеб. пособие для вузов. - СПб. : БХВ-Петербург, 2007. - 338с. : ил. + CD-ROM. - (Учебное пособие). - Библиогр.:с.337-338.
5. Колесов Ю.Б. Моделирование систем.Объектно-ориентированный подход : учеб. пособие для вузов. - СПб. : БХВ-Петербург, 2006. - 185с.
6. Суранов А.Я. LabVIEW 7:справочник по функциям. - М. : ДМК Пресс, 2005. – 510
7. Загидуллин Р.Ш. LabVIEW в исследованиях и разработках. - М. : Горячая линия-Телеком, 2005. – 350
8. Тревис Джеффри. LabVIEW для всех / Пер.Клушина Н.А.;Под ред.Шаркова В.В.,Гурьева В.А. - М. : ПриборКомплект:ДМК Пресс, 2005. - 537с.
9. Пейч Л.И. LabVIEW для новичков и специалистов. - М. : Горячая линия-Телеком, 2004. – 383
10. Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе Lab VIEW 7/ Под. ред. Бутырина П. А. -М.: ДМК Пресс, 2005. 264 с.: ил.

Учебное издание

Основы моделирования в LabVIEW

Учебное пособие

Авторы-составители:
Жукова Ирина Николаевна
Проторгуев Сергей Александрович